Keystroke Analysis in Digital Test Security: AI Approaches for Copy-Typing Detection and Cheating Ring Identification

Chenhao Niu*, **Yong-Siang Shih***, **Manqian Liao, Ruidong Liu, Angel Ortmann Lee**

Duolingo, Inc.
{chenhao,yongsiang,mancy,ruidong,angel.ortmannlee}@duolingo.com

Abstract

In high-stakes remote exams like the Duolingo English Test, sophisticated cheating methods such as *copy-typing* from external sources and the operation of *organized cheating rings* pose significant security threats. We introduce a two-pronged, content-agnostic approach that analyzes behavioral data to address these issues. For **copy-typing detection**, we present a sequence model that combines 1D convolutional layers with a Transformer encoder, which captures both local and long-range dependencies in keystroke dynamics to distinguish between organic and transcribed typing, outperforming tabular and other sequential baselines. For **cheating ring identification**, we propose a deep metric learning framework that generates a unique behavioral embedding for each test session using both keystroke and mouse dynamics. The model outperforms a traditional statistical baseline in linking test sessions completed by the same individual. Together, these AI-driven methods provide a powerful and scalable toolkit for safeguarding the integrity of remote assessments.

1 Introduction

Duolingo English Test (DET) (Naismith et al., 2025) is a remotely administered, high-stakes English proficiency exam. Ensuring the integrity of online assessments like the DET presents persistent challenges, especially as new forms of digital test fraud emerge (Belzak et al., 2025a). Two key threats are *copy-typing* and *organized cheating rings*.

Copy-typing refers to the act of transcribing text from an external source, rather than composing it organically. In the context of high-stakes testing, this form of misconduct often involves test takers using pre-written answers, receiving live assistance, or employing large language models (LLMs) to generate responses. Because security measures like disabling copy-paste functionality are applied, these illicitly obtained answers have to be manually typed into the response box. This behavior creates subtle but detectable deviations from natural typing patterns. Our work introduces a content-agnostic model that analyzes these behavioral signals in keystroke dynamics to effectively detect copy-typing.

Organized cheating rings are groups or commercial services that assist multiple test takers in cheating. They pose a significant threat to the integrity of the test due to their large-scale operations. Although such services may use various methods, we focus on a prevalent cheating ring scenario in which a human helper effectively completes tasks for multiple test takers using remote-control software or external peripherals unobservable in webcam footage. Because the same helper tends to assist many test takers, an effective strategy is to *link* test sessions completed by the same helper. We propose to fingerprint behavior using keystrokes and mouse dynamics, then compare test sessions through learned embeddings. Identified clusters of linked test sessions can be surfaced for human review (Shih et al., 2024)[1]. To this end, we employ a deep metric learning framework that embeds each test session into a high-dimensional representation, enabling efficient retrieval of similar test sessions via approximate nearest neighbor search.

Contributions.

- A content-agnostic keystroke sequence model for **copy-typing detection** that combines 1D convolutional networks with a Transformer encoder to effectively capture both local and long-range dependencies in keystroke dynamics.

*Equal contribution.

[1]An initial high-level description of the cheating ring identification system appeared in Shih et al. (2024). Here we contribute full technical details and new experiments.

Proceedings of the Artificial Intelligence in Measurement and Education Conference (AIME-Con) – Volume 2: Works in Progress, pages 107–116
October 27-29, 2025 ©2025 National Council on Measurement in Education (NCME)

- A deep metric learning framework for **cheating ring identification**, including a modified n-pair training objective with hard negatives controlling for hardware/region confounds.

2 Data Source and Context

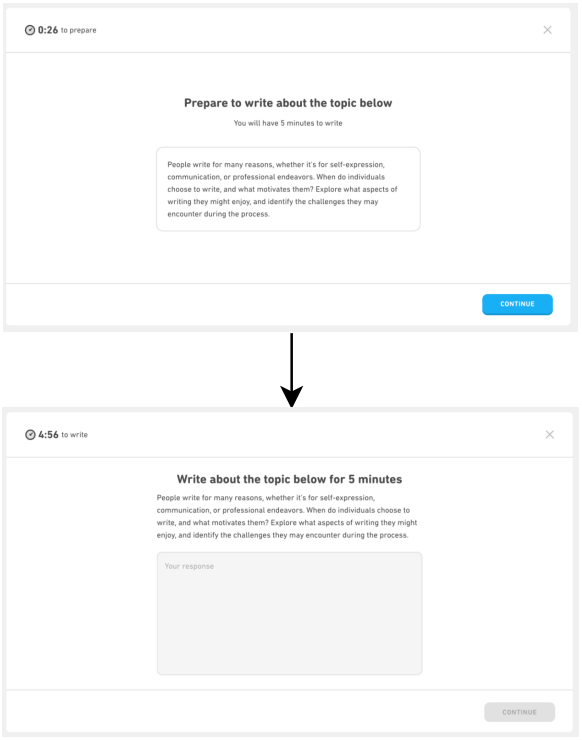

Figure 1: An example of the writing item in DET. The test taker has 30 seconds to prepare for the topic, and 5 minutes to write about it by typing on their keyboard.

Context. Our data originates from the Duolingo English Test (DET), a high-stakes, remotely proctored assessment of English proficiency (Naismith et al., 2025). The test's security protocol involves multiple layers, such as video recording, screen sharing, and input monitoring. Upon completion, every test session undergoes a rigorous review process that combines AI-driven analysis with human proctor oversight to identify any integrity violations (Belzak et al., 2025a). In our copy-typing detection research, we focus on an open-ended writing task in the DET, where test takers have 30 seconds to read a question given by text, and 5 minutes to type their response on a computer (See Figure 1 for an example). Comparatively, in our cheating ring identification research, we utilize the mouse movement patterns from the entire test session in addition to the keystroke patterns from writing tasks to identify the individual who completed the test.

Raw data collection. All DET test sessions record keystroke and mouse activity, including the timestamp of each key press and release, mouse movements and clicks, cursor position, and other contextual information (See Table 1 and Table 2 for details). We extract features from the raw log files with application-specific feature engineering methods.

Labels. To train and evaluate the models, the collected data is labeled based on human and AI-assisted proctoring decisions. The specific labeling criteria for both experiments are detailed in §4.

3 Methodology

Our methodology leverages machine learning to analyze behavioral data for two security tasks, based on the same data source of raw keystroke data.

3.1 Copy-Typing Detection

In this section, we introduce our method for copy-typing detection. At a high level, we treat the problem as a binary classification problem, and we train a machine learning model with labeled data, using a combination of tabular and sequential features.

3.1.1 Feature Engineering

For feature engineering, we process the raw features (§2) of each keystroke event to extract both sequential features and aggregated tabular features for each sample.

Key code categorization. To ensure the model is content-agnostic, we replace the exact `key-code` with a categorical `action-type`, which takes values among { INPUT, DELETE, MOVE, OTHER }, providing a coarser categorization. In addition, we include a binary `is-punctuation` indicator for punctuation keys.

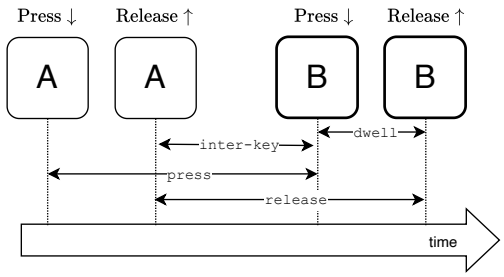

Figure 2: An example of extracting temporal features for the keystroke of the key "B". We consider four types of durations: `dwell`, `inter-key`, `press`, and `release`.

Raw Feature	Description
`key-code`	The exact key being pressed, such as "a", "shift", "delete", etc.
`timestamp`	The timestamp of the key being pressed, in milliseconds.
`dwell-duration`	The duration the key remains pressed, in milliseconds.
`text-length`	The number of characters in the text box at the time of the key being pressed.
`caret-position`	The current position of the caret in the text box, counted by the number of characters from the beginning of the text box.

Table 1: Raw keystroke features and their descriptions.

Raw Feature	Description
`position`	The current mouse cursor position on the screen.
`timestamp`	The timestamp when the data is recorded, in milliseconds.
`is-clicked`	Whether it's a click event or just a record of the current mouse position.
`dwell-duration`	The duration the mouse button remains pressed, in milliseconds.

Table 2: Raw mouse movement features and their descriptions.

Temporal sequence features. Similar to existing research (Acien et al., 2021; Stragapede et al., 2024), we extract timestamp-related features from `timestamp` and `dwell-duration`, by calculating the following four types of durations for each keystroke. See Table 3 for details, and Figure 2 for a concrete example.

Text length and caret position processing. For `text-length` and `caret-position`, we normalize them by the maximum text length observed within the item, to convert the feature values to the range of $[0, 1]$.

Auxiliary tabular features. In addition to temporal sequence features, we compute several tabular features for each sample. These features enhance the sequential model and enable comparison against tabular baselines. Specifically, the tabular features include: (i) counts by `action-type`, (ii) summary statistics { mean, std, p1, p25, p50, p75, p99, skewness } for each temporal feature, and (iii) pause-related measures: the count, total paused time, and average pause duration for pauses longer than thresholds { 200ms, 500ms, 1,000ms, 2,000ms, 3,000ms }.

All aggregates are normalized using training-set statistics. When combined with the sequence encoder, we include only a minimal, non-duplicative subset.

3.1.2 Model Architecture

Inspired by existing research (Acien et al., 2021; Stragapede et al., 2024), we model copy-typing detection as binary sequence classification over keystroke tokens. Each item is represented as a sequence $\{\mathbf{x}_i\}_{i=1}^{L}$, where $\mathbf{x}_i \in \mathbb{R}^{D_{\text{seq}}+D_{\text{tab}}}$ is the per-keystroke feature vector (§3.1.1), including D_{seq} sequential features concatenated with D_{tab} tabular features. Figure 3 is a diagram of the model architecture.

Overview. The architecture consists of: (i) one-dimensional convolutional networks (1D CNN) (Krizhevsky et al., 2012; Lea et al., 2016) that down-samples the sequence; (ii) a Transformer encoder (Vaswani et al., 2017) that models long-range dependencies; and (iii) a classifier head that produces an item-level logit. The design prioritizes robustness and efficiency while preserving discriminative temporal patterns characteristic of copy-typing.

Down-sampling with 1D CNN. An element-wise multi-layer perceptron (MLP) projects each input embedding from the width $D_{seq} + D_{tab}$ to a hidden width D_h. We then apply 1D CNN blocks with kernel size 3 and stride-2 max pooling to progressively halve the temporal resolution. With N_{CNN}=3, this yields a sequence of length $L/2^3$ with width $D_h = 32$. Intuitively, the CNN captures subword-scale rhythms while reducing the attention burden downstream.

Temporal Feature	Description
`dwell`	the duration between the press and release of the current key.
`inter-key`	the duration between the release of the previous key and the press of the current key.
`press`	the duration between the press of the previous key and the press of the current key.
`release`	the duration between the release of the previous key and the release of the current key.

Table 3: Definition of temporal sequence features that are extracted from raw keystroke timestamps.

Transformer encoder. We employ a Transformer encoder with $N_{\text{Transformer}} = 4$ layers and multi-head attention. Following a typical approach for Transformer-based classification (Devlin et al., 2019), we add positional embeddings and a trainable `[CLS]` token embedding at the input of the Transformer encoder, where the `[CLS]` token is a special token used for the model to aggregate the information for classification. The `[CLS]` embedding of the last layer is the output of the Transformer encoder.

Classification. At the final classification layer, we concatenate the output of the `[CLS]` token embedding from the Transformer encoder with a linear-transformed vector of the tabular features, and use a multi-layer perceptron (MLP) with sigmoid output to get the final classification output.

Training. Following a common approach in binary classification, we use Binary Cross Entropy (BCE) loss as the training objective with the AdamW optimizer (Loshchilov and Hutter, 2017). For a batch with N samples, the BCE loss is defined as

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)]$$

Where $y_i \in \{0, 1\}$ is the ground truth label for the i-th sample, and $\hat{y}_i \in (0, 1)$ is the prediction for that sample.

3.2 Cheating Ring Identification

In this section, we describe our cheating ring identification method. At a high level, the problem can be framed as a binary classification task: determining whether a pair of test sessions was completed by the same individual or not. Rather than training a direct classifier, we approach this with a deep metric learning framework (Kaya and Bilge, 2019). The approach makes deployment more practical, as the learned representations can be leveraged to efficiently retrieve similar test sessions using approximate nearest neighbor search (Li et al., 2019).

3.2.1 Feature Engineering

Keystroke features. Following existing research (Young et al., 2019), we compute summary statistics – mean, standard deviation, and sample count – for (i) the `dwell` duration of each key and (ii) the `transition` duration[2] between key pairs within each test session. These summary statistics constitute the keystroke features.

Mouse features. For mouse patterns, we compute several mouse movement metrics adapted from Zheng et al. (2011). The histograms of these metrics are used as mouse features:

- For any three consecutive recorded mouse cursor positions, A, B, and C, where the test taker clicked the mouse at C, we compute the following metrics:
  1. Direction: the angle between the horizontal line and $\overrightarrow{AB}$.
  2. Angle of Curvature: the angle between $\overrightarrow{BA}$ and $\overrightarrow{BC}$.
  3. Curvature Distance Ratio: the ratio between (1) the distance from B to $\overrightarrow{AC}$ and (2) the distance from A to C.
- For any two mouse clicks, we compute the time duration between the two clicks.

3.2.2 Model Architecture

Overview. A visual overview of our metric learning model is shown in Figure 4. We use an MLP to encode the input features to an embedding that represents the test session. The embeddings from two test sessions are used to compute a dissimilarity score, which is used in both training and inference.

Note that the same procedure is used to train two MLPs based on keystroke features and mouse fea-

[2] In our implementation, the `inter-key` interval in Figure 2 is used as the `transition` duration if the previous key is released before the next key is pressed; otherwise, the `press` interval is used.

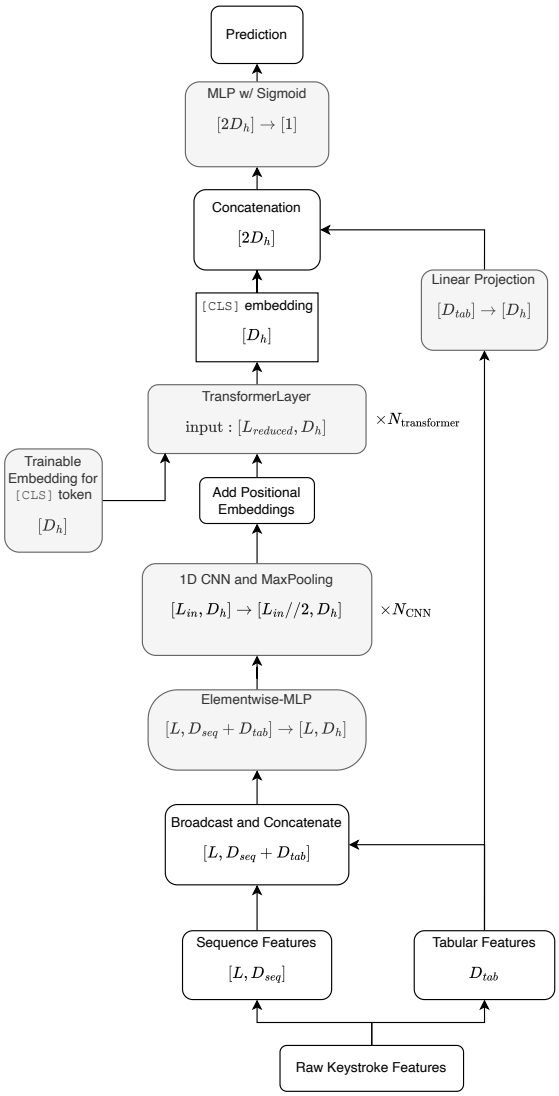

Figure 3: Copy-typing detection model architecture.

tures, and in the following discussions, we will not distinguish whether the input features are keystroke features or mouse features. During inference, we aggregate the dissimilarity scores by computing a weighted sum of the two. The weights are selected by fitting a Logistic Regression model (Cox, 1958) on the training dataset using the dissimilarity scores as inputs to predict whether a pair of test sessions is positive or negative.

Training. We use a modified multi-class n-pair loss (Sohn, 2016) to train the MLP encoder, replacing cosine with L_2 distance $d(\cdot)$ and augmenting batches with **hard negatives** matched on device type or region to reduce confounding. Intuitively, positives are pulled together while negatives are pushed apart.

Let $\{(x_i, x_i^+, x_i^-)\}_{i=1}^N$ be N triplets of input features sampled from the dataset, where each (x_i, x_i^+) is a positive pair corresponding to test sessions completed by the same individual (e.g., the same helper), and each (x_i, x_i^-) is a hard negative pair corresponding to test sessions completed by different individuals but involving the same type of keyboard or mouse, or occurring in the same region. We formulate a modified n-pair loss as shown in Equation 1. [3]

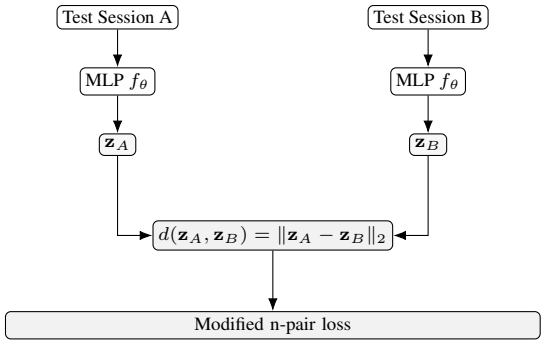

Figure 4: Each test session is encoded by the same MLP to produce an embedding. The embeddings can be compared to compute the dissimilarities between pairs of test sessions. Here $\mathbf{z}_A$ and $\mathbf{z}_B$ represent the output embeddings.

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{\hat{x}\in S}\frac{\delta(x_i,\hat{x})}{c_i}\log\frac{\exp(-d_\theta(x_i,\hat{x}))}{\sum_{x'\in S}\exp(-d_\theta(x_i,x'))}, \tag{1}$$

$$\begin{aligned}
\text{where } c_i &= \sum_{\hat{x}\in S}\delta(x_i,\hat{x}),\\
d_\theta(x_i,\hat{x}) &= d(f_\theta(x_i), f_\theta(\hat{x})),\\
d(\mathbf{z}_A,\mathbf{z}_B) &= \|\mathbf{z}_A - \mathbf{z}_B\|_2,\\
S &= \{x_i^+\}_{i=1}^N \cup \{x_i^-\}_{i=1}^N,\\
\delta(x_i,\hat{x}) &= \begin{cases} 1, & \text{if } (x_i,\hat{x}) \text{ is a positive pair,}\\ 0, & \text{otherwise.}\end{cases}
\end{aligned}$$

Deployment. In our framework, the model learns a distance function between test sessions through their high-dimensional representations. At deployment, these representations could be stored in a vector database, enabling efficient retrieval of similar test sessions using any test session as a query. By surfacing hidden connections between a test

[3] In our setup, there is a small probability that multiple positive pairs exist in Equation 1. The normalization term c_i is introduced to ensure a valid probability distribution for the cross-entropy loss. This is similar to the implementation of `tfa.losses.npairs_loss` in TensorFlow Addons.

session under review and its most similar test sessions, the system helps human proctors more effectively identify organized cheating rings. To combine keystroke-based and mouse-pattern-based dissimilarity scores, we scale each embedding by the inverse square root of its corresponding weight. This normalization allows us to directly leverage approximate nearest neighbor search in standard vector databases.

4 Experiment

4.1 Copy-Typing Detection Experiment

Dataset. As described in §2, we collect samples from DET test sessions. For copy-typing detection specifically, we filter out samples with short responses (less than 100 characters) or with non-standard `key-code` values, and collect 12,000 positive samples and 126,000 negative samples from January 12, 2024 to August 1, 2024, and randomly split them by a ratio of 6:2:2 for training, validation, and testing. We use the training set to train the copy-typing detection model, the validation set for early stopping and hyperparameter selection, and the testing set for evaluation.

Labels. To support robust model development and evaluation, labels are derived from both human and AI-assisted proctoring decisions. Specifically, positive examples are test sessions where proctors confirmed the usage of external resources, such as large language model (LLM) generated responses, with both AI-based detection (Niu et al., 2024) and video-based evidence. Negative examples are drawn from clean certified test sessions, where proctors find no violations in the current test session, and the user has no previous violations.

Settings. To validate the effectiveness of the proposed approach against other settings, in addition to the model architecture described in §3.1.2 (i.e., CNN + Transformer in Table 4), we select the following variations:

- **Tabular models**: To verify the effectiveness of adding sequential features, we adopt two baselines that use only the tabular features, including Logistic Regression (Cox, 1958) and LightGBM (Ke et al., 2017). They are noted as "Logistic Regression" and "LightGBM" in Table 4.
- **Other sequential models**: To verify the advantage of the Transformer-based encoder, we replace the Transformer blocks with other sequential models such as 1D convolutional networks (CNN) and Long Short-Term Memory networks (LSTM) (Hochreiter and Schmidhuber, 1997). They are noted as "CNN only" and "CNN + LSTM" in Table 4.

Metrics. Given that the positive and negative samples are naturally imbalanced, we use the Area Under the ROC Curve (AUROC) as a metric. Besides, since the practical application typically requires a low False Positive Rate (FPR), we also report the True Positive Rate (TPR, a.k.a., recall) at a 1% FPR.

4.2 Cheating Ring Identification Experiment

Dataset. As detailed in §2, our dataset is built from DET test sessions. Specifically, we sample certified test sessions from Q1 2025, excluding any with insufficient keystroke data. The resulting dataset covers approximately 102,000 test takers. We partition these into training, validation, and test sets with a 6:2:2 split. For validation and testing, pairs of test sessions were sampled and the labels are defined below. For training, we sample triplets of test sessions and construct pairs following the methodology described in §3.2.2. We use the validation set for early stopping, while the test set is used for evaluation.

Labels. For this task, we aim to detect when two test sessions were taken by the same human helper. However, the ground-truth data on cheating rings is scarce. Therefore, we construct our dataset using certified test sessions rather than confirmed cheating ring test sessions. The key intuition is that test sessions completed by the same individual can serve as positive pairs, since their behavior patterns originate from the same person, and test sessions completed by different individuals can serve as negative pairs. To reduce potential confounding factors, we require that test sessions in negative pairs match on keyboard/mouse type or region.

Evaluation scenarios. We evaluated our models under two scenarios. In the first scenario, 3051 positive pairs and 3051 negative pairs were sampled for each split, and the models were evaluated based on binary classification. We report the AUROC and the true positive rate at a 1% FPR. In the second scenario, we compare one test session against K different test sessions from other test takers. The K comparisons are treated as a single false positive if any of the K pairs are predicted as positive by the

model. Otherwise, they are treated as a single true negative. The test session is additionally compared with one other test session from the same test taker. The comparison is treated as true positive if the model predicts positive. Otherwise, it is treated as a false negative. 3051 test sessions were sampled from the test split to construct $3051(K+1)$ pairs as described above, and we report the AUROC of different models with different K.[4]

Baseline. We compare our proposed method with an in-house t-test based method that is built upon the work of Young et al. (2019), which utilizes the summary statistics of `dwell` durations and `transition` durations to determine if the two tests are completed by the same individual.

5 Results

5.1 Copy-Typing Detection

The evaluation results for our copy-typing detection models are summarized in Table 4. These results suggest two critical points. First, the superior performance of sequential models over the tabular baselines (Logistic Regression and LightGBM) confirms that the temporal dynamics of keystrokes contain essential signals for detecting copy-typing. Second, the choice of sequence architecture is crucial. While a simple "CNN only" model offers little advantage over a strong tabular baseline, incorporating a sophisticated encoder like a Transformer or LSTM to model long-range dependencies unlocks substantial performance gains. This highlights the necessity of using powerful sequence models to fully leverage the predictive patterns in keystroke data.

5.2 Cheating Ring Identification

The results based on the first evaluation scenario for cheating ring identification are shown in Table 5. As shown in the table, the proposed method outperforms the baseline. We additionally report the performance of our method using different subsets of features. The superior performance of the deep-full model demonstrates that keystroke and mouse dynamics provide complementary signals, and that combining them creates a more robust and accurate behavioral fingerprint.

[4]The second scenario simulates the real-world use case: a test session is compared with multiple test sessions to detect links to the same human helper, and even a single predicted match among them may require proctors' review.

The results based on the second evaluation scenario with K comparisons are shown in Figure 5. In the plot, we can see that our method remains competitive against the baseline as K increases, demonstrating the robustness of our methods.

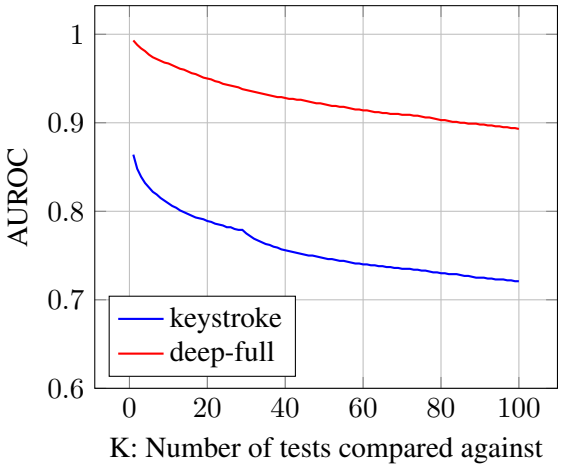

Figure 5: Performance comparison between our deep-full method and the t-test based keystroke baseline.

5.3 Fairness Analysis

Fairness is an important aspect of a Responsible AI system (Burstein et al., 2025). With the notion of *equality of opportunity* (Hardt et al., 2016), we evaluate the True Negative Rate (TNR) across demographic groups for both AI models. Intuitively, a similar TNR across groups means that innocent test takers in each group have a similar possibility of not being falsely flagged by the AI models. For our study, we focus on major geographical sub-regions according to the United Nations geoscheme[5], and evaluate the TNR of the models on clean[6] certified test sessions (for copy-typing detection) and negative pairs (for cheating ring identification) across groups. Table 6 and Table 7 present the proportion of test takers and pairs from each major sub-regions in the dataset, and the evaluation results of group-wise TNR. Note that the demographic distributions are slightly different for the two experiments because the datasets are sampled from different time periods. For both experiments, the results verify that the TNRs are within a small difference across groups, ensuring the equality of honest test takers.

[5]https://unstats.un.org/unsd/methodology/m49/

[6]Filtered to reduce data contamination.

Features	Model	AUROC	TPR@1%FPR
Tabular	Logistic Regression	84.91%	11.12%
Tabular	LightGBM	86.56%	12.64%
Sequential	CNN only	84.96%	15.18%
Sequential	CNN + LSTM	94.76%	39.05%
Sequential	CNN + Transformer	**95.22%**	**41.41%**

Table 4: Copy-typing detection model performance. The best results are in bold.

Method	AUROC	TPR@1%FPR
keystroke	86.44%	69.42%
deep-keystroke	98.56%	72.93%
deep-mouse	93.63%	39.86%
deep-full	**99.28%**	**89.28%**

Table 5: Performance for cheating ring identification methods. **keystroke** is the t-test based method, and **deep-*** are our proposed methods with different input features, where **deep-full** uses both keystrokes and mouse features.

6 Conclusion

We presented a two-pronged, content-agnostic framework for enhancing test security using behavioral data. For **copy-typing detection**, our CNN-Transformer model effectively learns sequential patterns in keystroke dynamics, outperforming both tabular and simpler sequential baselines in identifying copy-typing behavior. For **cheating ring identification**, our deep metric learning system produces robust embeddings from keystroke and mouse features, outperforming a statistical baseline and enabling efficient, large-scale deployment via approximate nearest neighbor search. Together, these AI-driven approaches provide a powerful, scalable toolkit for safeguarding the integrity of remote assessments. However, the deployment of such powerful tools necessitates a commitment to Responsible AI standards (Burstein et al., 2025). To that end, these models are best implemented not as automated judges, but as essential components in a human-in-the-loop proctoring framework that surfaces evidence for human review. This approach ensures that the AI models are governed with human oversight, providing both a secure and accountable system for assessment.

Limitations

For **copy-typing detection**, although the proposed model achieves a meaningful TPR at a low FPR of 1%, there still exist practical challenges when adopting such AI-generated signals in test proctoring process. For instance, since the output from the deep learning model is not interpretable, the proctoring guidelines need to be carefully designed for human proctors to accurately confirm the copy-typing detection result. Belzak et al. (2025b) discuss the findings in the practical usage of copy-typing detection in detail. Besides, although our approach is content-agnostic, prior work (Liang et al., 2023) has shown that some AI-text detectors can disproportionately flag non-native English writing as AI-generated, raising fairness concerns in educational settings. We therefore audit our models for group-wise error differences (§5.3) and maintain human-in-the-loop confirmation before adverse actions.

For **cheating ring identification**, one limitation is that our proxy positives (same-person test sessions) are not equivalent to true cheating ring labels, and may not fully capture the operational complexity of organized cheating rings. For example, the helper may deliberately vary typing/mouse behaviors to evade detection. In addition, the method is designed to be used as a retrieval aid rather than a fully automated proctor; its outputs must be combined with other safeguards – such as additional behavioral rules, corroborating signals, or human proctor judgment – to avoid unfair penalization of innocent test takers.

References

Alejandro Acien, Aythami Morales, John V Monaco, Ruben Vera-Rodriguez, and Julian Fierrez. 2021. Typenet: Deep learning keystroke biometrics. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(1):57–70.

William Belzak, Basim Baig, Ramsey Cardwell, Rose

Sub-region	Proportion	TNR@1%FPR (Copy-Typing Detection)
Southern Asia	25.46%	98.51%
Eastern Asia	12.99%	97.21%
Latin America and the Caribbean	12.89%	99.68%
Northern America	9.58%	99.77%
Western Asia	7.10%	99.50%
Sub-Saharan Africa	6.75%	99.51%
South-eastern Asia	6.29%	99.69%

Table 6: Proportion of test takers from major sub-regions (> 5%), and the True Negative Rate (TNR) in each group.

Sub-region	Proportion	TNR@1%FPR (Cheating Ring Identification)
Eastern Asia	35.66%	99.22%
Southern Asia	15.63%	99.05%
Northern America	13.42%	99.40%
Western Asia	12.24%	99.64%
Latin America and the Caribbean	11.34%	98.94%
South-eastern Asia	6.18%	99.06%
Western Europe	6.01%	99.51%

Table 7: Proportion of pairs from major sub-regions (> 5%), and the True Negative Rate (TNR) in each group. Note that since each pair involves two test sessions, which may come from different groups, a single pair can be counted more than once. As a result, the sum of the proportions may exceed 100%.

Hastings, André Kenji Horie, Geoff LaFlair, Manqian Liao, Chenhao Niu, and Yong-Siang Shih. 2025a. Duolingo English Test: Security and score integrity. Duolingo research report, Duolingo.

William Belzak, Chenhao Niu, and Angel Ortmann Lee. 2025b. When machines mislead: Human review of erroneous AI cheating signals. *Artificial intelligence in measurement and education conference*.

Jill Burstein, Geoffrey T. LaFlair, Kathleen Yancey, Alina A. von Davier, and Rotem Dotan. 2025. Responsible ai for test equity and quality: The duolingo english test as a case study. In Earl M. Tucker, Eleanor Armour-Thomas, and Edmund W. Gordon, editors, *Handbook for Assessment in the Service of Learning, Volume I: Foundations for Assessment in the Service of Learning*. University of Massachusetts Amherst Library Press.

David R Cox. 1958. The regression analysis of binary sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mahmut Kaya and Hasan Şakir Bilge. 2019. Deep metric learning: A survey. *Symmetry*, 11(9):1066.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems 30*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. 2016. Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision*, pages 47–54. Springer.

Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488.

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. 2023. Gpt detectors are biased against non-native english writers. *Patterns*, 4(7).

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Ben Naismith, Ramsey Cardwell, Geoffrey T. LaFlair, Steven Nydick, and Masha Kostromitina. 2025. Duolingo English Test: Technical manual. Duolingo research report, Duolingo.

Chenhao Niu, Kevin P. Yancey, Ruidong Liu, Mirza Basim Baig, André Kenji Horie, and James Sharpnack. 2024. Detecting LLM-assisted cheating on open-ended writing tasks on language proficiency tests. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 940–953, Miami, Florida, US. Association for Computational Linguistics.

Yong-Siang Shih, Manqian Liao, Ruidong Liu, and Mirza Basim Baig. 2024. Human-in-the-loop AI for cheating ring detection. *arXiv preprint arXiv:2403.14711*.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29.

Giuseppe Stragapede, Paula Delgado-Santos, Ruben Tolosana, Ruben Vera-Rodriguez, Richard Guest, and Aythami Morales. 2024. Typeformer: Transformers for mobile keystroke biometrics. *Neural Computing and Applications*, 36(29):18531–18545.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jay R Young, Randall S Davies, Jeffrey L Jenkins, and Isaac Pfleger. 2019. Keystroke dynamics: establishing keyprints to verify users in online courses. *Computers in the Schools*, 36(1):48–68.

Nan Zheng, Aaron Paloski, and Haining Wang. 2011. An efficient user verification system via mouse movements. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 139–150.