APLICACIÓN DE PRUEBAS DE CONOCIMIENTO CERO (ZKP) EN INTELIGENCIA ARTIFICIAL PARA LA MEJORA DE LA SEGURIDAD Y PRIVACIDAD DE LOS DATOS

Francisco G. Borgogno, Felipe Cañas, Paolo E. Cetti, Maximo Lucero Ruiz, Santiago N. Quesada, Fernando Boiero

Facultad de Ingeniería, Universidad Católica de Córdoba,
Av. Armada Argentina 3555, Ciudad de Córdoba, Argentina
maximolr.itr@gmail.com

Resumen

Este proyecto aborda la implementación de Pruebas de Conocimiento Cero (ZKP) en sistemas de inteligencia artificial para mejorar la seguridad y privacidad en la autenticación de usuarios. El objetivo principal es utilizar la dinámica de tecleo (Keystroke Dynamics) para crear un modelo de autenticación biométrica que reconozca a los usuarios basándose en sus patrones únicos de escritura, sin necesidad de revelar información sensible.

Para lograr esto, se recopiló un conjunto de datos a través de un keylogger que capturó las pulsaciones de teclas, el tiempo de presión y el tiempo de vuelo entre teclas de los usuarios. Estos datos fueron utilizados para entrenar una red neuronal profunda (DNN) que aprendió a identificar los patrones de escritura de cada usuario. El modelo fue optimizado mediante un algoritmo de descenso de gradiente y se ajustaron los hiperparámetros para mejorar su precisión y capacidad de generalización.

Los resultados indican que el modelo desarrollado puede distinguir eficazmente entre usuarios con patrones de escritura claramente diferenciados, mostrando una alta precisión en estos casos. Sin embargo, se encontró que la similitud en los perfiles de escritura entre usuarios con características similares dificulta la diferenciación, lo que sugiere la necesidad de incrementar la diversidad en los datos de entrenamiento. En conclusión, la aplicación de ZKP combinada con Keystroke Dynamics ofrece un enfoque prometedor para mejorar la seguridad en la autenticación, aunque se requieren ajustes adicionales para maximizar su efectividad en escenarios con usuarios similares.

Palabras Clave: Pruebas de Conocimiento Cero, Inteligencia Artificial, Privacidad y Seguridad de los Datos, Autenticación Biométrica, Deep Learning.

Introducción

El objetivo de este proyecto es lograr utilizar el concepto de Zero Knowledge Proof (ZKP) en el campo de la contraseña mediante la creación y posterior validación de un nuevo dato biométrico del usuario obtenido a través de la forma de escribir del mismo. Esto se puede lograr partiendo de las bases teóricas de la noción de mecanografía conocida como Keystroke Dynamics, la cual nos brindará las primeras dimensiones a tener en cuenta para alimentar y entrenar a nuestro modelo de Inteligencia Artificial utilizando Deep Learning con el fin de conocer los patrones de escritura de cada usuario y así poder identificarlos.

La importancia del proyecto radica en su capacidad para ofrecer una alternativa más segura y conveniente a los métodos tradicionales de autenticación, abordando problemas comunes como el robo de contraseñas. Además, este enfoque puede mejorar la experiencia del usuario al eliminar la necesidad de recordar y gestionar múltiples contraseñas. Desde una perspectiva económica y tecnológica, esta implementación utiliza el hardware ya existente sin la necesidad de implementar sensores biométricos adicionales, lo que elimina los costos y facilita su adopción en diversas plataformas y dispositivos existentes.

Materiales y métodos

Para la creación del Training Dataset (Conjunto de datos de entrenamiento), dos de los integrantes tipearon un texto de alrededor de 1500 caracteres. La idea es que cada usuario tenga un modelo propio entrenado con sus datos de escritura; un modelo especializado en identificar a dicho usuario. Con esto se espera que el algoritmo pueda validar y otorgarle acceso al usuario tras haber ingresado una frase aleatoria luego de ingresar su nombre de usuario.

El tamaño del Dataset fue de 3000 caracteres, y se utilizó una Deep Neural Network (DNN).

Utilizamos un Keylogger para capturar la información del teclado al momento de la escritura, esto nos permitió identificar 3 dimensiones: *Pressed Key* (Tecla Presionada), *Hold Time* (Tiempo de Presión) y *Flight Time* (Tiempo de vuelo entre teclas). Estas dimensiones son determinantes al momento de generar el patrón de escritura del usuario que luego se utilizó para entrenar el modelo de Deep Learning, aprendiendo a reconocer las características individuales de la escritura de cada usuario.

El modelo se entrenó utilizando un algoritmo de Gradient Descent, también conocido como Backpropagation (retropropagación) con optimización mediante el optimizador Adam para evitar que el modelo se estanque en puntos de ensilladura con pendiente pequeña retrasando su velocidad de aprendizaje. La función de pérdida utilizada fue la entropía cruzada binaria, adecuada para problemas de clasificación binaria. Las métricas de evaluación incluyeron la precisión y el nivel de confianza del modelo, que fueron monitoreadas durante el entrenamiento para ajustar los Hyperparameters (parámetros que controla el entrenador) y mejorar el rendimiento.

Códigos de programas

El código a continuación muestra la implementación del motor neuronal, clase creada por nosotros, que define la estructura en capas de la red neuronal. Este motor se encarga de entrenar, evaluar y predecir la autenticidad de los usuarios basándose en sus patrones de escritura.

```
class NeuralEngine:
   def __init__(self):
       self.model = Sequential(
           [
               Input((3,)),
               Dense(128, activation=LeakyReLU(negative_slope=0.1)),
               Dense(64, activation=LeakyReLU(negative_slope=0.1)),
               Dense(32, activation=LeakyReLU(negative_slope=0.1)),
               Dense(16, activation=LeakyReLU(negative_slope=0.1)),
               Dense(8, activation=LeakyReLU(negative_slope=0.1)),
               Dense(4, activation=LeakyReLU(negative_slope=0.1)),
               Dense(2, activation=LeakyReLU(negative_slope=0.1)),
               Dense(1, activation="sigmoid"),
           ]
       )
       self.model.compile(
           optimizer="adam",
           loss="binary_crossentropy",
           metrics=["accuracy"],
       )
```

La arquitectura del modelo está compuesta por 9 capas: Input Layer, 7 Hidden Layers con activaciones LeakyReLU[1] con un α (pendiente negativa) de 0.1 y la Output Layer que utiliza Sigmoid[2] como función de activación, finalizando con una capa sigmoide para producir una salida binaria que indica si el usuario es auténtico o no.

$$LeakyReLU(x) = max(0, x) + \alpha \cdot min(0, x) \quad (1)$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

Resultados y discusiones

Inicialmente, los niveles de precisión y confianza del modelo fueron bajos, esto debido al entrelazamiento de los datos de entrenamiento como podemos ver en la Fig. 1. Sin embargo, estos niveles mejoraron significativamente a medida que fuimos ajustando los parámetros del modelo y aumentando la cantidad de datos de entrenamiento, alcanzando los niveles que se muestran en la Fig. 2. Durante las pruebas, encontramos que el modelo era efectivo pero no excelente para distinguir a los dos integrantes. Indagando en las dinámicas de escritura comprendimos que esto se debía a que ambos tienen perfiles muy similares, siendo ambos programadores y estudiantes de ingeniería de la misma edad; por lo que sus patrones de escritura son muy similares y complejos de distinguir para la red. No obstante, cuando las pruebas se realizaron con personas de perfiles más diversos observamos un entrelazamiento menor (Fig. 3), gracias a esto, el modelo logró mejores niveles de precisión como podemos observar en la Fig. 4.

A continuación, se presentarán los gráficos antes mencionados donde los colores de los puntos permiten identificar a las distintas personas que entrenaron el modelo.

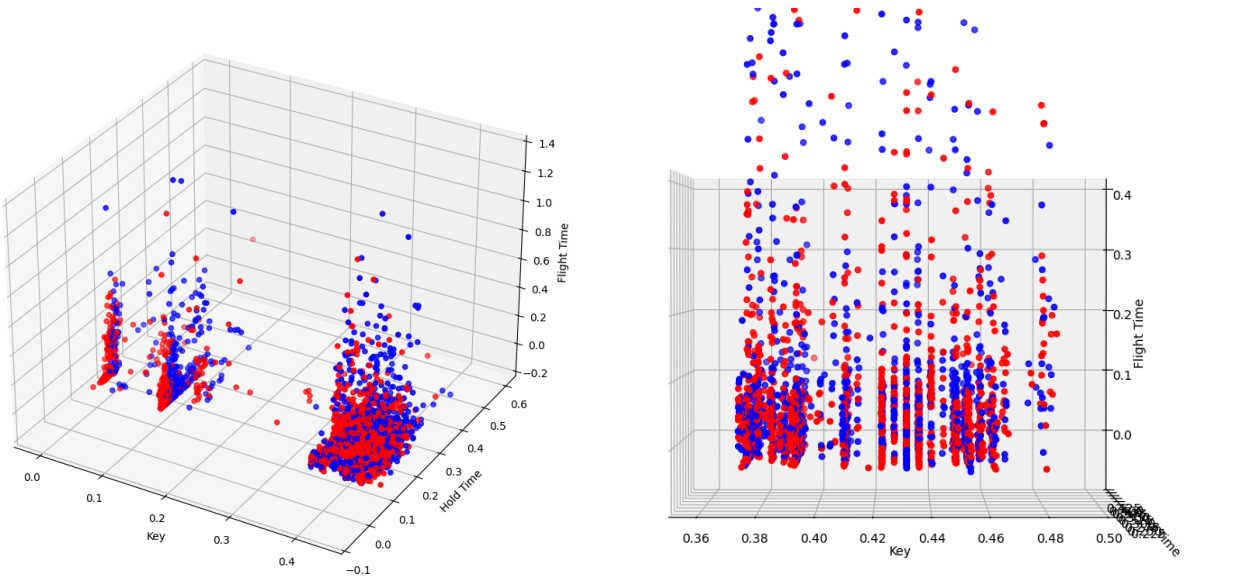

Fig. 1. Perspectiva isométrica y perpendicular del conjunto de datos - Perfiles similares

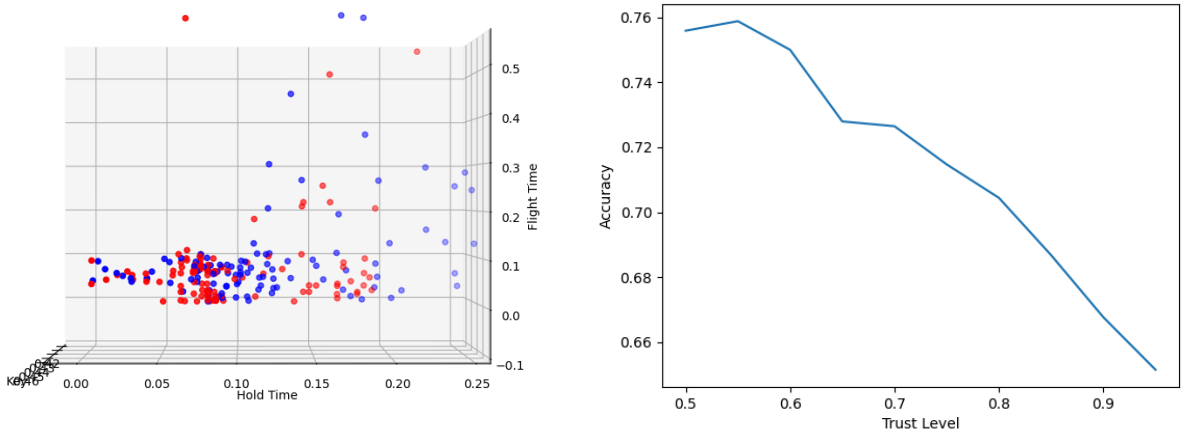

Fig. 2. Vista de valores de un único carácter y gráficos de resultados de predicción obtenidos - Perfiles similares.

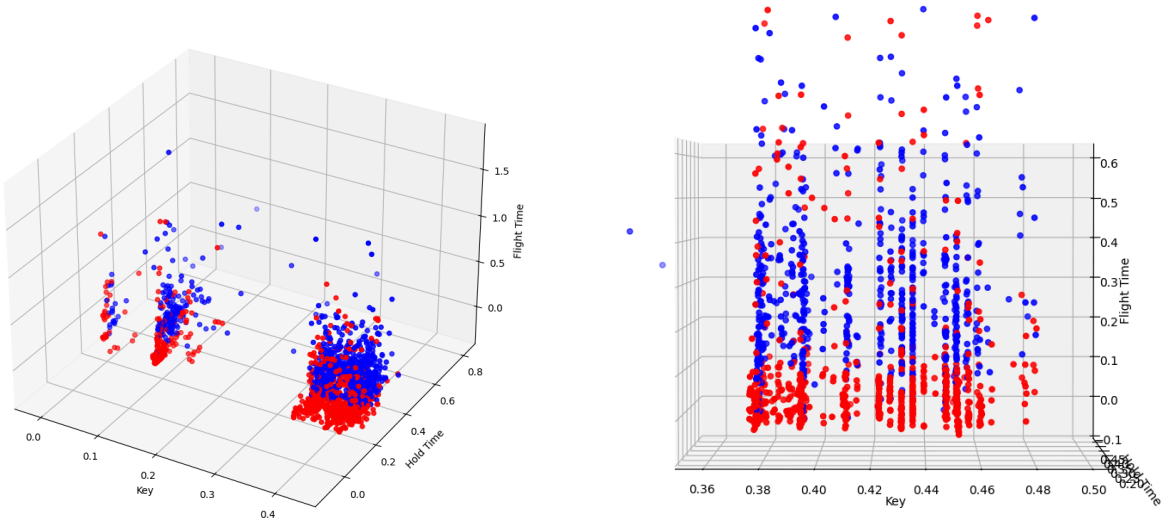

Fig. 3. Perspectiva isométrica y perpendicular del conjunto de datos - Perfiles diferentes.

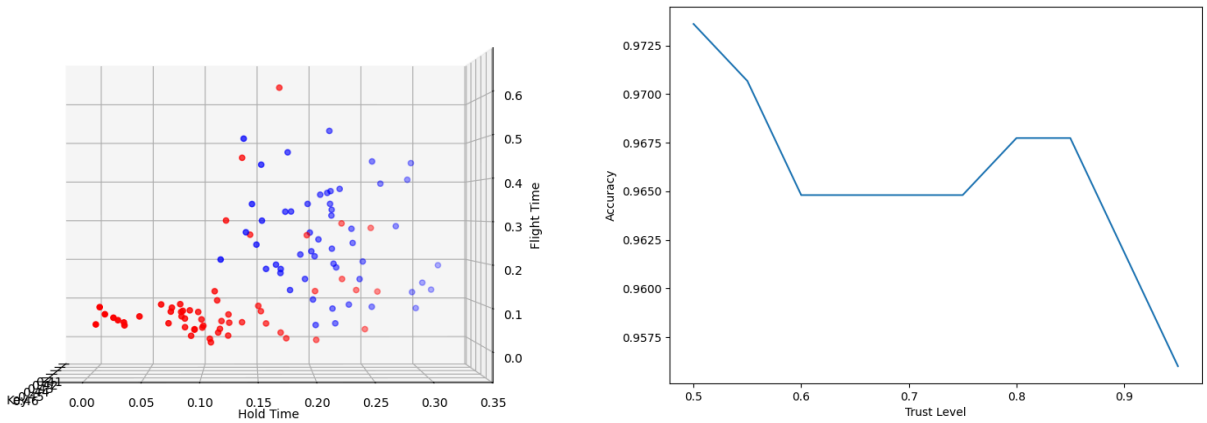

Fig. 4. Vista de valores de un único carácter y gráfico de resultados de predicción obtenidos - Perfiles diferentes.

El análisis de los resultados mostró que el modelo podía distinguir eficazmente entre los usuarios cuando sus perfiles de escritura eran significativamente diferentes. En cambio, en usuarios con perfiles similares, el modelo presentó dificultades para diferenciarlos correctamente, lo que reduce la efectividad del mismo. Esto sugiere que la diferenciabilidad en los datos de entrenamiento es crucial para mejorar el rendimiento del modelo en escenarios reales.

Además, se observó que la precisión del modelo aumentaba con el número de iteraciones de entrenamiento, probando la convergencia del problema e indicando que el modelo era capaz de aprender y generalizar mejor con más datos y entrenamiento prolongado.

Los ajustes en los Hyperparameters (hiper parámetros), como el Learning Rate (Tasa de aprendizaje) y el Batch Size (Tamaño del lote), también tuvieron un impacto significativo en el rendimiento del modelo.

Conclusiones

El modelo desarrollado para la identificación de usuarios mediante Keystroke Dynamics muestra una efectividad prometedora, especialmente cuando se aplica a usuarios con perfiles diversos. Sin embargo, la precisión actual del modelo, con las configuraciones empleadas, no es suficiente para sustituir por completo el uso de contraseñas tradicionales. La similitud en los patrones de tipeo de individuos con características similares puede afectar la precisión del modelo, lo que destaca la importancia de considerar una mayor variabilidad en los datos de entrenamiento.

Asimismo, estamos seguros de que para alcanzar una precisión adecuada y cumplir con el objetivo planteado inicialmente, es necesario aumentar la dimensionalidad del problema. Esto implica buscar nuevas direcciones de diferenciación que hagan que el conjunto de datos sea más diferenciable, habiendo diferencias destacables entre los datos relevados de distintos usuarios. Además, es importante continuar investigando técnicas de preprocesamiento y normalización de datos que puedan ayudar a resaltar las diferencias individuales en los patrones de escritura.

Estos hallazgos indican el potencial del uso de Deep Learning en la autenticación basada en Keystroke Dynamics y sugieren áreas para futuras investigaciones y mejoras.

Agradecimientos

Los autores desean expresar su agradecimiento a su tutor, Fernando Boiero, por incentivarlos a realizar esta investigación y por sus valiosos consejos, recomendaciones y su continuo apoyo.

Glosario

Zero Knowledge Proof: Método criptográfico en el cual una parte (el probador) puede demostrar a otra parte (el verificador) que sabe un valor determinado sin revelar ninguna información adicional sobre ese valor.
Biométrico: Relativo a las técnicas de identificación de individuos basadas en características físicas y comportamentales únicas.
Keystroke Dynamics: Técnica de autenticación que analiza los patrones de escritura y pulsaciones de teclas de un usuario para verificar su identidad.
Inteligencia Artificial: Campo de la informática que se enfoca en la creación de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana.
Deep Learning: Subcampo del machine learning que utiliza redes neuronales artificiales con muchas capas (deep neural networks) para modelar y analizar patrones complejos en datos grandes.
Hardware: Componentes físicos que conforman una computadora o un sistema informático.
Entrenamiento (Machine learning): Proceso mediante el cual un modelo de machine learning aprende a realizar una tarea específica a partir de datos de entrenamiento, ajustando sus parámetros internos para mejorar su rendimiento.

Deep Neural Network (DNN): Tipo de red neuronal artificial con múltiples capas ocultas entre la capa de entrada y la capa de salida, utilizada para modelar relaciones complejas en los datos.
Keylogger: Software o dispositivo que registra las pulsaciones de teclas en un teclado.
Gradient Descent (Backpropagation): Algoritmo de optimización utilizado para ajustar los pesos de una red neuronal durante el entrenamiento, minimizando el error al actualizar los pesos en función de la gradiente del error con respecto a cada peso.
Puntos de Ensilladura: Puntos en el espacio de parámetros de una red neuronal donde la gradiente es cero, pero que no son mínimos locales ni globales, dificultando la convergencia durante el entrenamiento.
Velocidad de Aprendizaje (Learning Rate): Parámetro que controla el tamaño de los pasos que toma el algoritmo de optimización al ajustar los pesos de una red neuronal durante el entrenamiento.
Entropía Cruzada Binaria: Función de pérdida utilizada para evaluar la calidad de las predicciones de un modelo en problemas de clasificación binaria, comparando las probabilidades predichas con las etiquetas verdaderas.
Input Layer: Primera capa de una red neuronal que recibe las características de entrada y las pasa a las capas ocultas para su procesamiento.
Hidden Layer: Capa de una red neuronal que se encuentra entre la capa de entrada y la capa de salida, donde se realizan cálculos intermedios para modelar las relaciones complejas en los datos.
LeakyReLU: Variante de la función de activación ReLU (Rectified Linear Unit) que permite un pequeño gradiente cuando la entrada es negativa, ayudando a mitigar el problema de los "neuronal death".
Output Layer: Última capa de una red neuronal que produce las predicciones finales o las salidas del modelo.
Sigmoid: Función de activación utilizada en redes neuronales que transforma la entrada en un valor entre 0 y 1, útil para problemas de clasificación binaria.
Efectividad (Accuracy): Métrica que mide la proporción de predicciones correctas realizadas por un modelo en comparación con el total de predicciones realizadas.
Nivel de Confianza (Trust Level): Medida de la certeza con la que un modelo de machine learning hace sus predicciones, a menudo expresada como una probabilidad.
Hyperparameters: Parámetros configurables de un modelo de machine learning que no se aprenden directamente de los datos, como la tasa de aprendizaje, el tamaño del batch y el número de capas ocultas.
Batch Size: Número de muestras de entrenamiento utilizadas para actualizar los pesos del modelo en cada iteración del algoritmo de optimización.

Referencias

Abhijeet. (s.f.). *User Verification based on Keystroke Dynamics*. Recuperado de https://github.com/abhijeet3922/User-Verification-based-on-Keystroke-Dynamics
CMU. (s.f.). *Keystroke*. Recuperado de http://www.cs.cmu.edu/~keystroke/
Janakiev, N. (s.f.). *Keystroke Biometrics*. Recuperado de https://github.com/njanakiev/keystroke-biometrics/blob/master/keystroke-biometrics.ipynb
KeyStroke-Dynamics. Recuperado de https://github.com/rakshithca/KeyStroke-Dynamics