Integrantes:

● Sebastian Hidalgo

● Ricardo Medina

● Alejandro Osejo

# **Proyecto de Sistema de Anotación de Vídeo**

## **Abstract**

Este proyecto presenta un sistema de anotación de vídeo que reconoce actividades humanas como caminar, girar, sentarse y permanecer de pie. Se utilizaron técnicas de reducción de dimensionalidad (PCA) y modelos avanzados (KNN, Random Forest, SVM, y MLPClassifier) ajustados con GridSearchCV, logrando altos niveles de precisión. El sistema final permite anotaciones automáticas en tiempo real, con aplicaciones en investigación, rehabilitación y diseño de interfaces hombre-máquina.

## **Introduction**

Este proyecto tiene como objetivo desarrollar un software especializado en la anotación de vídeos que capture diversas actividades humanas, como caminar, girar, sentarse y estar de pie. Para lograr esto, se implementará un sistema de análisis de vídeo en tiempo real, lo que permitirá la detección y seguimiento preciso de los movimientos articulares y de la postura de los individuos en las grabaciones. La solución se enfocará en proporcionar herramientas efectivas para la clasificación y análisis de estas actividades, contribuyendo así a campos como la investigación en comportamiento humano, la rehabilitación física y la interacción hombre-máquina.

## **Theory**

1. Preprocesamiento y Escalado de Datos: Pandas y NumPy son fundamentales para cargar, limpiar y manipular datos en estructuras eficientes. El escalado mediante MinMaxScaler y StandardScaler estandariza las características, lo cual es esencial en modelos sensibles a magnitudes, como KNN y redes neuronales.
2. División de Datos: Dividir los datos en conjuntos de entrenamiento y prueba con train\_test\_split permite evaluar si el modelo generaliza bien en datos no vistos.
3. Modelos de Clasificación: Se utilizan cuatro modelos:

○ KNeighborsClassifier (KNN): Clasificación basada en la proximidad.

○ RandomForestClassifier: Modelo de ensamble robusto frente al sobreajuste.

○ SVC: Clasificación mediante hiperplanos en espacios de alta dimensionalidad.

○ MLPClassifier: Red neuronal para patrones complejos y no lineales.

1. Optimización de Parámetros: GridSearchCV ayuda a encontrar la mejor combinación de parámetros para el modelo, utilizando validación cruzada.
2. Métricas de Evaluación: classification\_report y accuracy\_score brindan medidas detalladas como precisión y exhaustividad para evaluar el rendimiento del modelo en cada clase.
3. Visualización de Datos: Seaborn y Matplotlib permiten visualizar patrones y relaciones en los datos, ayudando en el análisis y selección de modelos.
4. Reducción de Dimensionalidad: Se utilizó el análisis de componentes principales (PCA) para reducir la dimensionalidad de los datos. Esto permitió:

● Mejorar la eficiencia computacional al disminuir el número de características utilizadas.

● Preservar el 95% de la varianza explicada en los datos originales, utilizando 11 componentes principales.

● Minimizar el impacto del ruido en las características y evitar posibles redundancias entre ellas.

## **Methodology**

En la recolección y preprocesamiento de datos, recopilamos videos que muestran actividades humanas clave como caminar, girar, sentarse e inclinarse. Las anotaciones iniciales se realizaron de manera manual, asegurando precisión en la identificación de las actividades. Posteriormente, los datos fueron limpiados y procesados utilizando Pandas y NumPy. Como parte del preprocesamiento, se implementó una técnica de normalización centrada en la posición de la cadera, lo que permitió estandarizar las coordenadas en relación con un punto de referencia común, eliminando así variaciones innecesarias en las posiciones absolutas. Adicionalmente, las características fueron escaladas mediante MinMaxScaler y StandardScaler para garantizar que todas las dimensiones tuvieran escalas comparables, optimizando el rendimiento de los modelos de clasificación.

Los datos se dividieron en conjuntos de entrenamiento y prueba utilizando train\_test\_split, asignando un 80% de los datos al entrenamiento y un 20% al conjunto de prueba. Este paso permitió evaluar el desempeño de los modelos en datos no vistos. Asimismo, se aseguró una distribución equilibrada de las etiquetas en ambos conjuntos, evitando sesgos que pudieran influir en la clasificación de las actividades humanas.

Con el objetivo de mejorar la eficiencia computacional y reducir la redundancia en los datos, se aplicó el análisis de componentes principales (PCA) como técnica de reducción de dimensionalidad. Este proceso redujo las características originales a 11 componentes principales, los cuales preservaron el 95% de la varianza total explicada en los datos. Esto no solo disminuyó el impacto de características ruidosas, sino que también facilitó el entrenamiento de los modelos al reducir la complejidad del espacio de características.

Se implementaron y reentrenaron cuatro modelos principales para la clasificación: KNeighborsClassifier, RandomForestClassifier, SVC y MLPClassifier. Estos modelos fueron seleccionados por su capacidad para manejar diferentes tipos de relaciones y patrones en los datos. Cada modelo fue entrenado utilizando tanto los datos originales como los transformados mediante PCA, con el fin de evaluar el impacto de la reducción de dimensionalidad en su rendimiento.

Para optimizar el desempeño de los modelos, se utilizó GridSearchCV, una técnica de búsqueda en rejilla que permitió ajustar los hiperparámetros clave de cada modelo mediante validación cruzada. La evaluación del rendimiento incluyó métricas detalladas como precisión, exhaustividad y f1-score, generadas con classification\_report y accuracy\_score. Adicionalmente, se emplearon visualizaciones con Seaborn y Matplotlib para identificar patrones en los datos y analizar los efectos del PCA.

Finalmente, se desarrolló una interfaz para realizar anotaciones automáticas en tiempo real, empleando el modelo mejor ajustado después de la reducción de dimensionalidad. Esta implementación permitió detectar y clasificar actividades humanas de manera eficiente en los videos procesados, mostrando el impacto directo de las técnicas aplicadas en el sistema final.

## **Results**

El conjunto de datos fue normalizado y estandarizado, centrando las coordenadas en la cadera para proporcionar un marco de referencia consistente para el reconocimiento de actividades. Los datos se dividieron en entrenamiento (80%) y prueba (20%) para evaluar el rendimiento de los modelos.

1. K-Nearest Neighbors(KNN)

* Mejores hiperparámetros: n\_neighbors = 3
* Mejor precisión en entrenamiento: 0.9889639688575048

1. Random Forest

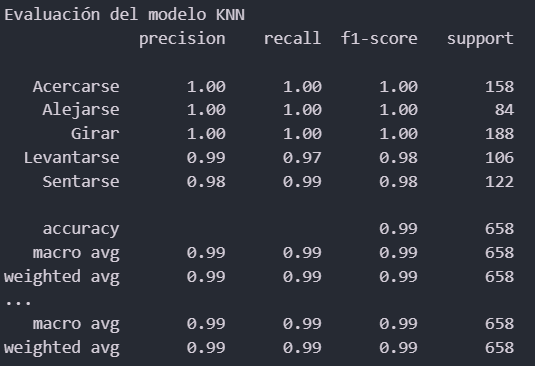
* Mejores hiperparámetros: max\_depth = 20, n\_estimators = 150
* Mejor precisión en entrenamiento: 0.9961933731667573

1. Support Vector Machine(SVM)

* Mejores hiperparámetros: C = 10, kernel = rbf'
* Mejor precisión en entrenamiento: 0.9821104472207134

1. Red Neuronal(MLPClassifier)

* Mejores hiperparámetros: alpha = 0.0001, hidden\_layer\_sizes = (100,)
* Mejor precisión en entrenamiento: 0.9965750497917798



## **Results Analysis**

K-Nearest Neighbors(KNN): Con un valor óptimo de n\_neighbors = 3, KNN alcanzó una precisión de entrenamiento de 98.89%, mostrando un buen rendimiento en la identificación de actividades basadas en la proximidad de las instancias.

Random Forest: Con parámetros óptimos (max\_depth=20, n\_estimators=150), obtuvo una precisión de entrenamiento de 99.61%, mostrando robustez frente al sobreajuste gracias a su naturaleza de ensamble.

Support Vector Machine(SVM): utilizando un kernel radial (rbf) y un valor de C=10, obtuvo una precisión de entrenamiento de 98.21%, lo que sugiere que el modelo identifica bien los patrones en el espacio transformado.

Red Neuronal(MLPClassifier): Con parámetros óptimos (alpha=0.0001, hidden\_layer\_sizes=(100,)), MLPClassifier logró una precisión de entrenamiento de 99.65%, destacando en la captura de relaciones complejas y no lineales entre las características.

Random Forest y KNN muestran un buen rendimiento en términos de generalización, pero sería útil realizar un análisis más profundo utilizando métricas adicionales en el conjunto de prueba para confirmar su capacidad de generalizar correctamente a datos no vistos. MLPClassifier, por su parte, destaca por su alta precisión en el conjunto de entrenamiento, lo que indica un buen ajuste, pero también sugiere un posible sobreajuste, lo que requiere una validación más exhaustiva para asegurarse de que no está simplemente memorizando los datos de entrenamiento. En cuanto al SVM, presenta un menor riesgo de sobreajuste debido a su enfoque en maximizar el margen entre clases, aunque su rendimiento debería ser evaluado más a fondo en escenarios con datos ruidosos o características irrelevantes, para garantizar su eficacia en situaciones más complejas.

## **Conclusion and future works**

Se desarrolló un sistema de anotación de vídeos enfocado en actividades humanas como caminar, girar, sentarse y permanecer de pie. El proceso incluyó el preprocesamiento de datos mediante normalización centrada en la posición de la cadera y el uso de técnicas de escalado como MinMax Scaler y Standard Scaler para uniformizar las características. Se aplicó Análisis de Componentes Principales (PCA) para reducir las dimensiones a 10 componentes principales, conservando el 95% de la varianza original. Posteriormente, se entrenaron y optimizaron modelos como KNN, Random Forest, SVM y MLPClassifier utilizando GridSearchCV. Finalmente, se desarrolló una interfaz en tiempo real que integra el modelo más eficiente para automatizar la anotación de videos procesados.

A lo largo del proyecto se destacó la importancia del preprocesamiento y la reducción de dimensionalidad, que mejoran el rendimiento y la eficiencia de los modelos. Modelos como KNN y Random Forest mostraron comportamientos robustos, mientras que SVM y MLP Classifier revelaron sus limitaciones y fortalezas en diferentes escenarios. La herramienta final es funcional y traslada el conocimiento técnico a aplicaciones prácticas. Sin embargo, se identificaron oportunidades de mejora, como aumentar la diversidad de datos, optimizar el rendimiento en tiempo real, explorar modelos avanzados como CNN y mejorar la aplicabilidad de los resultados. El trabajo futuro incluye ampliar el sistema a escenarios multitarea, realizar validaciones en entornos reales y desarrollar versiones multiplataforma, integrando dispositivos IoT para análisis multimodal.

## **Bibliographic References**

* Scikit-learn, “scikit-learn: machine learning in Python,” *Scikit-learn.org*, 2019. https://scikit-learn.org/stable/
* “Matplotlib documentation — Matplotlib 3.5.0 documentation,” *matplotlib.org*. https://matplotlib.org/stable/
* M. Waskom, “seaborn: statistical data visualization — seaborn 0.10.1 documentation,” *seaborn.pydata.org*, 2021. https://seaborn.pydata.org/index.html
* NumPy, “Overview — NumPy v1.19 Manual,” *numpy.org*, 2022. https://numpy.org/doc/stable/
* “pandas documentation — pandas 1.0.3 documentation,” *pandas.pydata.org*. https://pandas.pydata.org/docs/index.html