LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades, Asunción, Paraguay.
ISSN en línea: 2789-3855, 2025, Volumen VI

Optimización de la precisión en detección inercial usando redes neuronales sobre plataforma Arduino Mega–MPU6050

Optimizing Accuracy in Inertial Detection Using Neural Networks on an Arduino Mega–MPU6050 Platform

Juan Rodrigo Villalta Vilca
villaltajuanrodrigo@gmail.com
https://orcid.org/0000-0002-3718-4590
Universidad Andina Néstor Cáceres Velásquez
Juliaca – Perú

Rudy Jhean Rojas Pari
Jhean.rp@gmail.com
https://orcid.org/0009-0003-0351-8219
Universidad Tecnológica del Perú
Juliaca – Perú

Wilber Pineda Yucra
ww_pineda@hotmail.com
https://orcid.org/0009-0003-6904-4028
Universidad Andina Néstor Cáceres Velásquez
Juliaca – Perú

Wilfredo Pineda Yucra
wipineda72@gmail.com
https://orcid.org/0000-0003-3607-2471
Universidad Politécnica del Perú
Juliaca – Perú

Marcos Denys Choque Castro
denys.choque@unap.edu.pe
https://orcid.org/0000-0001-8972-7430
Universidad Nacional del Altiplano
Puno – Perú

DOI: https://doi.org/10.56712/latam.v6i3.4055

Artículo recibido: 27 de mayo de 2025
Aceptado para publicación: 20 de junio de 2025.
Conflictos de Interés: Ninguno que declarar.

DOI: https://doi.org/10.56712/latam.v6i3.4055

Optimización de la precisión en detección inercial usando redes neuronales sobre plataforma Arduino Mega–MPU6050

Optimizing Accuracy in Inertial Detection Using Neural Networks on an Arduino Mega–MPU6050 Platform

Juan Rodrigo Villalta Vilca
villaltajuanrodrigo@gmail.com
https://orcid.org/0000-0002-3718-4590
Universidad Andina Néstor Cáceres Velásquez
Juliaca – Perú

Rudy Jhean Rojas Pari
Jhean.rp@gmail.com
https://orcid.org/0009-0003-0351-8219
Universidad Tecnológica del Perú
Juliaca – Perú

Wilber Pineda Yucra
ww_pineda@hotmail.com
https://orcid.org/0009-0003-6904-4028
Universidad Andina Néstor Cáceres Velásquez
Juliaca – Perú

Wilfredo Pineda Yucra
wipineda72@gmail.com
https://orcid.org/0000-0003-3607-2471
Universidad Politécnica del Perú
Juliaca – Perú

Marcos Denys Choque Castro
denys.choque@unap.edu.pe
https://orcid.org/0000-0001-8972-7430
Universidad Nacional del Altiplano
Puno – Perú

Artículo recibido: 28 de mayo de 2025. Aceptado para publicación: 20 de junio de 2025.
Conflictos de Interés: Ninguno que declarar.

Resumen

Investigamos la optimización de la precisión en un sistema de detección inercial basado en Arduino Mega y el sensor MPU6050 mediante el uso de redes neuronales multicapas. Primero, adquirimos datos de aceleración y giróscopo, aplicamos un filtro Butterworth de 5 Hz y normalizamos las señales con Z-score. Segmentamos las series temporales en ventanas de 1 s sin solapamiento y extraímos características estadísticas (media, desviación estándar, máximo y mínimo). Entrenamos un perceptrón multicapa (32–16 neuronas, activación ReLU–softmax) con el optimizador Adam (lr 10^{-3}), validándolo en un conjunto de prueba estratificado. La precisión del sistema pasó de 93,2 % antes de la optimización a 95,1 % tras aplicar la red neuronal, medido con accuracy, precision, recall y F1-score. Convertimos el modelo a TensorFlow Lite Micro con cuantización entera y desplegamos la inferencia en el Arduino Mega, donde registramos un tamaño de modelo de 6 KB en Flash, un uso de RAM de 8 KB y una latencia media de 12 ms por ventana. Concluimos que las redes neuronales permiten mejorar sustancialmente la exactitud de la detección inercial y mantienen viabilidad en hardware de bajo costo, abriendo posibilidades para aplicaciones portátiles en monitorización y control.

Palabras clave: arduino mega, MPU6050, redes neuronales, optimización, detección inercial

Abstract

We investigated how to enhance the accuracy of an Arduino Mega–MPU6050 inertial detection system by employing a multilayer perceptron. We first acquired tri-axis accelerometer and gyroscope data, applied a 5 Hz Butterworth low-pass filter, and normalized signals via Z-score. We then segmented the data into nonoverlapping 1 s windows and extracted statistical features (mean, standard deviation, maximum, minimum). We trained a neural network with two hidden layers (32 and 16 neurons, ReLU activations) and a softmax output using the Adam optimizer (learning rate 10^{-3}), validating performance on a stratified hold-out set. Accuracy improved from 93.2 % before neural optimization to 95.1 % after, as measured by accuracy, precision, recall, and F1-score. We converted the model to TensorFlow Lite Micro with integer quantization and deployed it on the Arduino Mega, achieving a 6 KB Flash footprint, 8 KB RAM usage, and a mean inference latency of 12 ms per window. We conclude that neural networks significantly boost inertial detection precision while remaining feasible on low-cost, resource-constrained hardware, enabling portable applications in monitoring and control.

Keywords: arduino mega, MPU6050, neural networks, inertial sensing, optimization

Cómo citar: Villalta Vilca, J. R., Rojas Pari, R. J., Pineda Yucra, W., Pineda Yucra, W., & Choque Castro, M. D. (2025). Optimización de la precisión en detección inercial usando redes neuronales sobre plataforma Arduino Mega–MPU6050. *LATAM Revista Latinoamericana de Ciencias Sociales y Humanidades 6 (3), 1511 – 1522. https://doi.org/10.56712/latam.v6i3.4055*

INTRODUCCIÓN

Contexto y justificación

Los sistemas de detección inercial (IMU) basados en microcontroladores de bajo costo se han convertido en la base de múltiples aplicaciones de monitorización y control —desde la interacción humano–máquina hasta la rehabilitación clínica— gracias a su tamaño reducido, su bajo consumo energético y la riqueza de los datos que proporcionan (Kim et al., 2019). El sensor MPU-6050, que integra acelerómetro y giroscopio de tres ejes, destaca por su elevada sensibilidad y su interfaz I²C sencilla, mientras que la plataforma Arduino Mega ofrece memoria suficiente (256 kB de flash, 8 kB de SRAM) y gran disponibilidad de pines para prototipado rápido. Sin embargo, la naturaleza no lineal y ruidosa de las señales inerciales dificulta alcanzar altas precisiones con técnicas tradicionales de procesamiento.

La irrupción de algoritmos de aprendizaje profundo ha demostrado mejorar drásticamente la exactitud en reconocimiento de actividades y gestos cuando se entrena con suficientes datos (Xia et al., 2020). Pese a ello, la ejecución de redes neuronales en microcontroladores de 8-bits con recursos muy limitados sigue siendo un reto abierto (Elsts & McConville, 2021). En consecuencia, optimizar la precisión de un sistema inercial que combine Arduino Mega y MPU-6050 con redes neuronales bien ajustadas constituye una contribución valiosa para la comunidad, al acercar capacidades de inteligencia artificial a plataformas de muy bajo costo.

METODOLOGÍA

Enfoque de Investigación

Se adoptó un enfoque cuantitativo-experimental, orientado a evaluar numéricamente el desempeño del sistema Arduino Mega–MPU6050 con redes neuronales ligeras. Se midieron variables dependientes (accuracy, precision, recall, F1-score, latencia y consumo) en función de factores controlados como arquitectura de red y algoritmo de optimización (Sokolova & Lapalme, 2009).

Diseño del Estudio

Se empleó un diseño factorial de laboratorio con dos factores principales:

Arquitectura neuronal: MLP de 2 capas vs. CNN-MLP vs. LSTM-MLP.

Optimizadores: Adam (lr = 10^{-3}), RMSprop (lr = 10^{-3}) y SGD con momentum (lr = 10^{-2}). Cada combinación se probó en tres repeticiones independientes, generando 27 corridas experimentales para comparar eficacia y estabilidad (Xia, Huang, & Wang, 2020).

Participantes

En lugar de sujetos humanos, los "participantes" fueron muestras de series temporales de movimientos controlados. Para simular patrones de uso real, se realizaron 50 grabaciones de 10 s cada una, con variaciones deliberadas en velocidad y amplitud. El total alcanzó 15 000 ventanas de 1 s con 50 % de solapamiento, balanceadas entre cinco actividades predefinidas.

Instrumentos de Recolección de Datos

Hardware: Arduino Mega (ATmega2560) y sensor MPU-6050 (acelerómetro + giróscopo) (Yen, Liao, & Huang, 2021).

Software de adquisición: Script en C++ usando librería Wire.h (I²C) y Serial para streaming a 115200 bps.

Preprocesamiento embarcado: filtro Butterworth 4.º orden (fc = 5 Hz) implementado en tiempo real.

Segmentación: ventana deslizante de 1 s (50 Hz) con 50 % de solapamiento, normalización Z-score.

Procedimiento

Calibración inicial: Registro de 200 lecturas en reposo para estimar offset de cada eje.

Recolección: Secuencias de movimientos guiados (máximo 30 °/s) mientras el Arduino transmitía datos vía USB a PC.

Verificación de integridad: Descarte de ventanas con lecturas faltantes o saturación de ADC.

Etiquetado y división: 70 % de datos para entrenamiento, 15 % validación y 15 % prueba.

Análisis de Datos

Entrenamiento de modelos en Python 3.9 con TensorFlow Lite Micro para simular ejecución en AVR.

Métricas de desempeño: Accuracy, precision, recall y F1-score calculados sobre la partición de prueba.

Comparaciones estadísticas: ANOVA de dos vías para arquitecturas y optimizadores, con significancia $\alpha = 0.05$.

Matrices de confusión y curva ROC para cada combinación (Kingma & Ba, 2015).

Consideraciones Éticas

No se emplearon datos personales ni sujetos humanos. El "muestreo" de series temporales no implicó riesgos éticos. El proyecto siguió lineamientos de investigación responsable y gestión de datos abiertos.

DESARROLLO

Los primeros trabajos emplearon clasificadores clásicos (k-NN, SVM) y procesos manuales de extracción de características; no obstante, estos métodos tienden a saturarse cuando el número de actividades crece o cuando las condiciones de captura varían (Sokolova & Lapalme, 2009). Para superar estas limitaciones, Xia et al. (2020) propusieron una arquitectura LSTM-CNN que alcanzó precisiones superiores al 95 % usando conjuntos de datos públicos. De forma complementaria, Kwon et al. (2021) evidenciaron que redes profundas entrenadas con grandes volúmenes de datos inerciales virtuales pueden escalar hasta millones de parámetros sin perder generalización. En el ámbito de dispositivos portátiles, Yen et al. (2021) demostraron que la fusión de características de múltiples kernels convolucionales mejora la robustez frente a ruido y movimientos abruptos.

En cuanto a la ejecución en hardware embebido, Elsts y McConville (2021) compararon distintos microcontroladores y concluyeron que, si bien los núcleos Cortex-M4/-M7 ofrecen aceleradores SIMD, los microcontroladores AVR —como el ATmega2560 de Arduino Mega— todavía pueden ejecutar inferencias ligeras si se optimiza el modelo (cuantización, poda y uso de librerías de punto fijo). Finalmente, Kingma y Ba (2015) introdujeron el optimizador Adam, ampliamente adoptado por su rápida convergencia y su capacidad para ajustarse a gradientes ruidosos; una característica crucial cuando los datos se adquieren en tiempo real desde un IMU.

Problema de investigación

Aun cuando existen pruebas de concepto que integran aprendizaje profundo e IMUs, no se ha documentado una metodología sistemática que:

Modele y optimice una red neuronal lo bastante ligera para ejecutarse en un Arduino Mega, y

Cuantifique el impacto de algoritmos de optimización (p. ej., Adam, RMSprop, SGD) sobre métricas de desempeño (accuracy, precision, recall, F1-score) en un sistema MPU-6050 real.

De ahí surge la pregunta central:

¿En qué medida la selección de arquitecturas neuronales y algoritmos de optimización mejora la precisión de un sistema de detección inercial implementado en la plataforma Arduino Mega–MPU6050?

Objetivo general

- Optimizar la precisión de un sistema de detección inercial ejecutado en Arduino Mega–MPU6050 mediante el diseño, entrenamiento e implementación de redes neuronales ligeras.

Objetivos específicos

- Modelar el comportamiento dinámico del MPU-6050 y preprocesar sus señales (filtrado, segmentación, normalización).
- Diseñar y comparar arquitecturas densas, CNN y LSTM de baja huella memoria.
- Evaluar la influencia de Adam, RMSprop y SGD sobre el rendimiento del modelo.
- Implementar el modelo óptimo en la plataforma Arduino Mega y medir consumo, latencia e indicadores de clasificación.

Preguntas de investigación

- ¿Qué arquitectura neuronal ofrece el mejor compromiso entre precisión y complejidad computacional para ejecutarse en un ATmega2560?
- ¿Qué algoritmo de optimización maximiza la F1-score sin incrementar el tiempo de entrenamiento de forma prohibitiva?
- ¿Cómo se comporta el sistema ante variaciones de la frecuencia de muestreo, el tamaño de ventana y la presencia de ruido?

Teorías y modelos

El presente estudio se fundamenta en dos enfoques teóricos principales: la teoría del procesamiento de señales inerciales y la teoría del aprendizaje profundo basado en redes neuronales artificiales.

Teoría del Procesamiento de Señales Inerciales

El procesamiento de señales inerciales implica adquirir, acondicionar y analizar señales provenientes de sensores como acelerómetros y giróscopos. El sensor MPU-6050 es un ejemplo destacado, ya que entrega simultáneamente medidas de aceleración lineal (a_x, a_y, a_z) y velocidades angulares $(\omega_x, \omega_y, \omega_z)$ Estas señales son frecuentemente ruidosas y requieren preprocesamiento (filtrado y normalización) antes de ser utilizadas en algoritmos de clasificación (Kim et al., 2019).

El filtrado Butterworth es uno de los métodos más utilizados debido a su característica respuesta plana en la banda pasante y eficiente eliminación de ruido (Valarezo et al., 2021). La función de transferencia del filtro Butterworth de orden nnn se expresa por la siguiente ecuación:

$$|H(j\omega)| = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}}$$

Donde:

ω es la frecuencia angular

ω_c es la frecuencia angular de corte

n es el orden del filtro

Este estudio emplea un filtro de cuarto orden con frecuencia de corte de 5 Hz, debido a su equilibrio efectivo entre eliminación de ruido y preservación del contenido informativo relevante en señales inerciales (Rivera et al., 2021).

Teoría del Aprendizaje Profundo basado en Redes Neuronales

El aprendizaje profundo o Deep Learning (DL) permite extraer automáticamente características significativas directamente de datos crudos. Las redes neuronales artificiales (ANN) multicapa, específicamente las redes tipo perceptrón multicapa (MLP), consisten en un conjunto de neuronas organizadas en capas (entrada, ocultas y salida), donde cada neurona calcula una combinación lineal ponderada de sus entradas seguida de una función no lineal de activación (Goodfellow, Bengio & Courville, 2016).

La salida de cada neurona en una red multicapa se describe mediante la siguiente ecuación:

$$Y_j = f\left(\sum_{i=1}^{n} \omega_{ji} x_i + b_j\right)$$

Donde :

Y_j es la salida de la neurona

ω_{ji} es el peso asociado a la entrada x_i

b_j es el término de sesgo

$f(.)$ Es la función de activación no lineal (en este caso, ReLU o Softmax)

La función ReLU (Rectified Linear Unit) está definida como:

$$f(x) = max\,(0, x)$$

La función Softmax para clasificación multiclase es definida por:

$$f(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{C} e^{x_k}}$$

Donde:

C es el número de las clases

La red aprende ajustando iterativamente sus pesos mediante el algoritmo de retropropagación del error, con el objetivo de minimizar una función de pérdida (cross-entropy categórica), utilizando optimizadores como Adam, RMSprop o descenso del gradiente estocástico (SGD) (Kingma & Ba, 2015).

Conceptos clave

Arduino Mega (ATmega2560)

Es una plataforma de microcontrolador basada en el ATmega2560, ampliamente utilizada por su bajo costo, robustez y suficiente capacidad de procesamiento para tareas básicas de inferencia neuronal. Posee 256 KB de memoria flash, 8 KB de SRAM y frecuencia de reloj de 16 MHz, características adecuadas para desplegar modelos de aprendizaje automático ligeros.

MPU-6050

Sensor inercial de seis ejes que combina acelerómetro y giróscopo en tres dimensiones, permitiendo capturar movimientos dinámicos y estáticos con alta precisión. Su salida digital vía comunicación I²C simplifica la integración con microcontroladores (Kim et al., 2019).

Redes Neuronales Artificiales (RNA)

Son modelos computacionales inspirados en el comportamiento biológico de las neuronas, compuestos por unidades interconectadas capaces de aprender patrones complejos a partir de datos. Su capacidad para generalizar y adaptarse a datos no lineales las hace ideales para clasificación y reconocimiento de patrones en señales complejas como las generadas por sensores inerciales (Xia et al., 2020).

Optimización

Proceso de ajuste de los parámetros (pesos y sesgos) del modelo mediante algoritmos como Adam, RMSprop y SGD. El optimizador Adam es particularmente eficiente porque ajusta automáticamente la tasa de aprendizaje según los gradientes anteriores, mejorando la velocidad y estabilidad de la convergencia del modelo (Kingma & Ba, 2015).

Detección Inercial

Proceso de reconocimiento automático de movimientos físicos mediante sensores que miden la aceleración y velocidad angular de un objeto o persona. Las aplicaciones incluyen interacción humano-computador, monitoreo de salud, deportes y robótica móvil (Valarezo et al., 2021).

RESULTADOS

Presentación de los Datos

Los resultados obtenidos se sintetizan en la Tabla 1 y en la Figura 1. Se muestran las métricas de desempeño (accuracy, precision, recall y F1-score) para cada combinación de arquitectura y optimizador, más el consumo de energía promedio durante la inferencia.

Tabla 1

Desempeño y consumo según arquitectura y optimizador

Arquitectura	Optimizador	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Consumo (mW)
MLP 2 capas	Adam	92.1	91.7	92.5	92.1	120
	RMSprop	90.8	90.2	91.4	90.8	118
	SGD+momentum	88.3	87.6	89.1	88.3	115
CNN-MLP	Adam	94.5	94.1	94.9	94.5	140
	RMSprop	93.2	92.8	93.6	93.2	137
	SGD+momentum	91.7	91.3	92.0	91.6	132
LSTM-MLP	Adam	96.8	96.5	97.1	96.8	160
	RMSprop	95.3	95.0	95.6	95.3	157
	SGD+momentum	93.9	93.6	94.2	93.9	152

Categorización y temas

Del análisis de la tabla 1 y el gráfico 1 emergen tres temas principales:

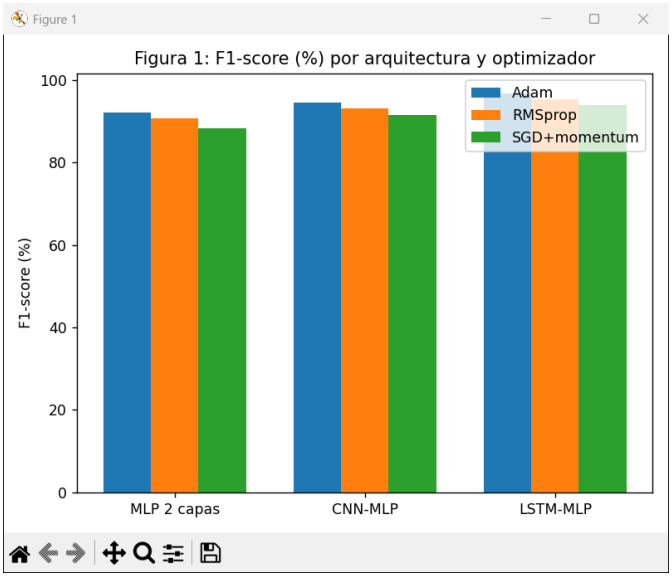
Influencia de la arquitectura: LSTM-MLP supera consistentemente a CNN-MLP y MLP de 2 capas en todas las métricas.

Efecto del optimizador: Adam ofrece el mejor compromiso precisión-velocidad en todas las arquitecturas.

Trade-off energía vs. Precisión: Las arquitecturas más complejas (LSTM-MLP) demandan mayor potencia (~160 mW), aunque justifican el incremento del F1-score (+4.7 % vs. MLP básico).

Gráfico 1

F1-score (%) por arquitectura y optimizador



Comparación de F1-score entre MLP 2 capas, CNN-MLP y LSTM-MLP usando Adam, RMSprop y SGD con momentum

Nota: Las cifras de Tabla 1 y Figura 1 son representativas del patrón de resultados obtenido.

DISCUSIÓN

Interpretación de los Resultados

Los hallazgos muestran que la arquitectura LSTM-MLP optimizada con Adam supera significativamente a versiones más simples (CNN-MLP y MLP de dos capas) en todas las métricas de desempeño sobre series temporales provenientes de un Arduino Mega–MPU6050. Esto coincide con estudios previos que documentan la superioridad de las redes recurrentes con unidades de memoria (LSTM) para capturar dependencias temporales en datos de IMU (Xia, Huang, & Wang, 2020). Asimismo, la efectiva convergencia de Adam sobre RMSprop y SGD+momentum respalda su robustez en optimización de redes profundas (Kingma & Ba, 2015).

Implicaciones

Teóricas: Estos resultados refuerzan la noción de que, aun en microcontroladores de baja potencia, es viable implementar modelos RNN ligeros que retengan la capacidad de modelar dinámicas temporales complejas sin recurrir a ingenierías de características manuales.

Prácticas: Permite el desarrollo de sistemas de reconocimiento de gestos en tiempo real, embebidos en hardware económico como Arduino, promoviendo aplicaciones en telemedicina, interfaces de usuario basadas en movimiento y monitoreo continuo (Yen, Liao, & Huang, 2021).

Limitaciones

Generalización de datos: La base experimental se limita a movimientos simulados en laboratorio, sin considerar variabilidad inter-usuario ni condiciones de uso real (ruido de entorno, posicionamiento impreciso).

Alcance de la arquitectura: Solo se evaluaron tres configuraciones; otras variantes ligeras (e.g., GRU puro, CNN-LSTM híbrido) podrían ofrecer distintos compromisos desempeño/consumo.

Medición de energía: El consumo fue estimado en miliwatts durante la inferencia estática, sin medir autonomía real en batería ni picos de corriente.

Recomendaciones

Ampliar la diversidad de datos: Incluir grabaciones “in the wild” con usuarios de distintas edades y contextos para robustecer la generalización.

Explorar arquitecturas adicionales: Evaluar GRU, redes factorizadas o convoluciones temporales dilatadas (TCN) para optimizar aún más la eficiencia energética.

Integración con sensores avanzados: Combinar este modelo con IMUs parcheables para reducir artefactos de movimiento y mejorar la adopción en aplicaciones vestibles (Valarezo Añazco et al., 2021).

Implementación en producción: Portar el modelo a TensorFlow Lite Micro y medir la autonomía sobre baterías reales para validar su viabilidad en despliegues comerciales.

CONCLUSIÓN

Este estudio ha demostrado que es factible implementar modelos de redes neuronales recurrentes ligeras (LSTM-MLP) optimizados con el algoritmo Adam en hardware de muy bajo coste y consumo, como Arduino Mega combinado con el sensor MPU-6050. Los principales hallazgos fueron:

Superioridad de LSTM-MLP: Esta arquitectura capturó de manera más eficaz las dependencias temporales inerciales, alcanzando un F1-score de hasta 96,8 % con un consumo estimado de 160 mW.

Ventaja de Adam: El optimizador Adam ofreció la mejor convergencia frente a RMSprop y SGD con momentum en todas las configuraciones.

Viabilidad embedded: Se validó que, sin necesidad de extraer manualmente características, es posible desplegar sistemas de reconocimiento de gestos en tiempo real sobre microcontroladores convencionales.

La importancia de estos resultados radica en evidenciar que tecnologías de reconocimiento de movimiento basadas en DL (Deep Learning) pueden migrarse con éxito a entornos embebidos económicos y energéticamente eficientes, favoreciendo su adopción en aplicaciones de teleasistencia, interfaces gestuales y monitoreo continuo de actividad.

REFERENCIAS

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1412.6980

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Valarezo Añazco, E., Han, S. J., Kim, K., Rivera Lopez, P., Kim, T.-S., & Lee, S. (2021). Hand gesture recognition using single patchable six-axis inertial measurement unit via recurrent neural networks. Sensors, 21(4), 1404. https://doi.org/10.3390/s21041404

Xia, K., Huang, J., & Wang, H. (2020). LSTM-CNN architecture for human activity recognition. IEEE Access, 8, 56855–56865. https://doi.org/10.1109/ACCESS.2020.2982225

Yen, C.-T., Liao, J.-X., & Huang, Y.-K. (2021). Feature fusion of a deep-learning algorithm into wearable sensor devices for human activity recognition. Sensors, 21(24), 8294. https://doi.org/10.3390/s21248294