FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENERÍA DE SISTEMAS E INFORMÁTICA

Tesis

Monitoreo y predicción del momento del parto de cerdas mediante modelos de visión artificial basado en CNN

Para optar el título profesional de Ingeniero de Sistemas e Informática

Autor:
Marc Anthoni Reátegui Sifuentes
https://orcid.org/0009-0007-0083-0353

Asesor:
Ing. Dr. Miguel Ángel Valles Coral
https://orcid.org/0000-0002-8806-2892

Tarapoto, Perú

2026

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENERÍA DE SISTEMAS E INFORMÁTICA

Tesis

Monitoreo y predicción del momento del parto de cerdas mediante modelos de visión artificial basado en CNN

Para optar el título profesional de Ingeniero de Sistemas e Informática

Autor:

Marc Anthoni Reátegui Sifuentes

Sustentado y aprobado el 16 de abril del 2026, ante el honorable jurado:

Presidente de Jurado Ing. Dr. Juan Carlos García Castro	**Secretario de Jurado** Ing. Dr. Alberto Alva Arévalo
Vocal de Jurado Ing. Dr. Cristian Wemer García Estrella	**Asesor** Ing. Dr. Miguel Ángel Valles Coral

Tarapoto, Perú

2026

Universidad Nacional de San Martín
Facultad de Ingeniería de Sistema e Informática
Ciudad Universitaria - Jr. Amorarca # 315 - Morales

ACTA DE SUSTENTACIÓN
PARA OPTAR EL TÍTULO DE INGENIERO DE SISTEMAS E INFORMÁTICA

Resolución N° 024-2026-UNSM/FISI-D (14.04.2026)

FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA – ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

A las 10:00 horas del día Jueves, 16 de abril del año 2026, se inició el acto público de sustentación de la tesis titulada: MONITOREO Y PREDICCIÓN DEL MOMENTO DEL PARTO EN CERDAS MEDIANTE MODELOS DE VISIÓN ARTIFICIAL BASADO EN CNN, presentado por el Bachiller: MARC ANTHONI REATEGUI SIFUENTES, con el Asesor: Ing. Dr. Miguel Ángel Valles Coral.

Instalado los miembros de jurado calificador conformado por:

Presidente : Ing. Dr. Juan Carlos García Castro

Secretario : Ing. Dr. Alberto Alva Arévalo

Vocal : Ing. Dr. Cristian Werner García Estrella

El presidente del jurado dirigió brevemente unas palabras y a continuación el secretario dio lectura a la Resolución N° 024-2026-UNSM/FISI-D.

Seguidamente el autor expuso el trabajo de investigación y el jurado realizó las preguntas pertinentes, respondidas por el sustentante y eventualmente por el asesor, con la venia del jurado.

Una vez terminada la ronda de preguntas el jurado procedió a deliberar para determinar la calificación final, para lo cual dispuso un receso de quince (15) minutos, con participación del asesor con voz, pero sin voto y sin la presencia del sustentante y otros participantes del acto público.

Luego de aplicar los criterios de calificación con estricta observancia del principio de objetividad y de acuerdo con los puntajes en escala vigesimal (de 0 a 20), según el Anexo 4.2. del RG-CTI, la nota de sustentación otorgada resultante del promedio aritmético de los calificativos emitidos por cada uno de los miembros del jurado fue .....VEINTE..... (20).

De acuerdo con el Artículo 40° del RG – CTI, la nota obtenida es .....APROBADO..... y correspondiente a la calificación de ...EXCELENTE....; leído este resultado en presencia de todos los participantes del acto de sustentación, el secretario dio lectura a las observaciones subsanables al informe final que el autor deberá corregir y alcanzar al jurado en un plazo máximo de treinta (30) días calendario.

Firman los integrantes del jurado calificador, asesor y el autor de la tesis en señal de conformidad, dando por concluido el acto a las ..11:10..... horas, el mismo día 16 de abril del 2026.

Ing. Dr. Juan Carlos García Castro
Presidente

Ing. Dr. Alberto Alva Arévalo
Secretario

Ing. Dr. Cristian Werner García Estrella
Vocal

Ing. Dr. Miguel Ángel Valles Coral.
Asesor

Marc Anthoni Reátegui Sifuentes
Autor

Constancia de asesoramiento

Quien suscribe el presente documento,

Hace constar:

Que, habiendo acompañado en la ejecución de la tesis titulada: Monitoreo y predicción del momento del parto en cerdas mediante modelos de visión artificial basado en CNN. Elaborado por el tesista:

Bachiller en Ingeniería de Sistemas e Informática: **Marc Anthoni Reátegui Sifuentes**.

Por lo que doy conformidad para los trámites correspondientes, dejo como constancia el presente documento y firmo.

Tarapoto, 16 de abril de 2026

Ing. Dr. Miguel Ángel Valles Coral
Asesor

Declaratoria de Autenticidad

Marc Anthoni Reátegui Sifuentes, con DNI N° 72717961, bachiller de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional de San Martín, autor de la tesis titulada: **Monitoreo y predicción del momento del parto en cerdas mediante modelos de visión artificial basado en CNN**.

Declaramos bajo juramento que:

1. La tesis presentada es de autoría propia.
2. La redacción fue realizada respetando las citas y referencia de las fuentes bibliográficas consultadas, siguiendo las normas APA actuales.
3. Toda información que contiene la tesis no ha sido plagiada.
4. Los datos presentados en los resultados son reales, no han sido alterados ni copiados, por tanto, la información de esta investigación debe considerarse como aporte a la realidad investigada.

Por lo antes mencionado, asumo bajo responsabilidad las consecuencias que deriven de mi accionar, sometiéndome a las leyes de nuestro país y normas vigentes de la Universidad Nacional de San Martín.

Tarapoto, 16 de abril de 2026.

.....

Marc Anthoni Reátegui Sifuentes

DNI N° 72717961

Ficha de identificación

Título: Monitoreo y predicción del momento del parto en cerdas mediante modelos de visión artificial basado en CNN	**Área de investigación**: Ciencias Naturales **Línea de investigación**: Ciencias de la Computación **Sublínea de investigación**: Inteligencia Artificial y Recuperación de Información **Grupo de investigación**: Grupo de Investigación IA ……. (Resolución N° 134-2021-UNSM/FISI/CFT) **Tipo de investigación**: Básica ☐, Aplicada ☒, Desarrollo experimental ☐

Autor: Marc Anthoni Reátegui Sifuentes	Facultad de Ingeniería de Sistemas e Informática Escuela Profesional de Ingeniería de Sistemas e Informática https://orcid.org/0009-0007-0083-0353

Asesor: Ing. Dr. Miguel Ángel Valles Coral	**Dependencia local de soporte:** Facultad de Ingeniería de Sistemas e Informática Escuela Profesional de Ingeniería de Sistemas e Informática Unidad o Laboratorio Ingeniería de Sistemas e Informática https://orcid.org/0000-0002-8806-2892

Dedicatoria

Dedico este trabajo con profundo amor y gratitud a mis padres, Jorge Antonio Reátegui Ruiz y Loydith Sifuentes Flores, por ser el pilar inquebrantable de mi vida. Su esfuerzo constante, sus sacrificios silenciosos y su confianza en mis capacidades me han guiado con firmeza en cada paso de mi formación. Gracias por enseñarme con el ejemplo el valor del trabajo honesto y la importancia de nunca rendirse.

Asimismo, dedico esta tesis a mis hermanos, con la esperanza de ser para ellos un ejemplo de perseverancia, superación y compromiso. Que este logro sea también un impulso para que crean en sus sueños y trabajen con pasión por alcanzarlos.

Agradecimientos

Expreso mi más sincero agradecimiento al Ing. Dr. Miguel Ángel Valles Coral, por su valiosa guía, compromiso y acompañamiento en cada etapa del desarrollo de esta tesis. Su experiencia y disposición para orientar han sido fundamentales en la culminación de este trabajo académico.

A mi familia, gracias por estar siempre presente con su amor, comprensión y respaldo incondicional. Su fe en mí me ha dado fuerza incluso en los momentos más difíciles, y este logro es tan suyo como mío.

Agradezco también a mis amigos y compañeros de estudios, con quienes compartí aprendizajes, desafíos y experiencias inolvidables a lo largo de esta etapa universitaria. Su apoyo y camaradería marcaron positivamente mi camino.

Finalmente, extiendo mi gratitud a los docentes y al personal administrativo de la universidad, por su dedicación y esfuerzo en brindar una formación integral y de calidad a todos sus estudiantes.

Índice general

Ficha de identificación 6
Dedicatoria 7
Agradecimientos 8
Índice general 9
Índice de tablas 11
Índice de figuras 12
RESUMEN 13
ABSTRACT 14
CAPÍTULO I INTRODUCCIÓN A LA INVESTIGACIÓN 15
CAPÍTULO II MARCO TEÓRICO 19
2.1. Antecedentes de la investigación 19
2.2. Fundamentos teóricos 20
2.2.1. Monitoreo predicción del momento del parto 20
2.2.2. Modelos de visión artificial basados en CNN 23
CAPÍTULO III MATERIALES Y MÉTODOS 27
3.1. Ámbito y condiciones de la investigación 27
3.1.1. Contexto de la investigación 27
3.1.2. Periodo de ejecución 27
3.1.3. Autorizaciones y permisos 28
3.1.4. Control ambiental y protocolos de bioseguridad 28
3.1.5. Aplicación de principios éticos internacionales 28
3.2. Sistema de variables 29
3.2.1. Variables principales 29
3.2.2. Variables secundarias 29
3.3. Procedimientos de la investigación 29
3.3.1. Diseño de la investigación 29
3.3.2. Objetivo específico 1: Identificar patrones visuales previos al parto en cerdas mediante el análisis de imágenes capturadas 32

3.3.3. Objetivo específico 2: Implementar modelos híbridos de visión artificial basados en CNN para el monitoreo y la predicción de comportamientos preparto en cerdas. 37
3.3.4. Objetivo específico 3: Evaluar los modelos propuestos de visión artificial basados en CNN en la predicción del momento del parto en cerdas durante las horas previas al evento 39
CAPÍTULO IV RESULTADOS Y DISCUSIÓN 42
4.1. Resultado específico 1: Identificar patrones visuales previos al parto en cerdas mediante el análisis de imágenes capturadas 42
4.2. Resultado específico 2: Implementar modelos híbridos de visión artificial basados en CNN para el monitoreo y la predicción de comportamientos preparto en cerdas. 45
4.2.1. Configuración del entorno y carga del dataset 46
4.2.2. Extracción de características con redes CNN 46
4.2.3. Búsqueda automática de los mejores parámetros 48
4.2.4. Validación cruzada con métricas globales 48
4.2.5. Evaluación Comparativa de Métricas Generales y por Clase 54
4.3. Resultado específico 3: Evaluar los modelos propuestos de visión artificial basados en CNN en la predicción del momento del parto en cerdas durante las horas previas al evento. 56
4.3.1. Visualización de la distribución por clase en los bloques 57
4.3.2. Evaluación cruzada y obtención de métricas por corrida 58
CONCLUSIONES 64
RECOMENDACIONES 65
REFERENCIAS BIBLIOGRÁFICAS 66
ANEXOS 72
Anexo 1: Permiso de autorización para la realización de la investigación en la granja porcina. 72
Anexo 2: Código de los modelos desarrollados. 73

Índice de tablas

Tabla 1 Descripción de variables por objetivo específico 1 29
Tabla 2 Descripción de variables por objetivo específico 2 29
Tabla 3 Ejemplo de vectores de características extraídas con AlexNet por clase 47
Tabla 4 Ejemplo de vectores de características extraídas con ResNet18 por clase.... 47
Tabla 5 Ejemplo de vectores de características extraídas con VGG16 por clase..... 48
Tabla 6 Mejores hiperparámetros encontrados por modelo con RandomizedSearchCV 48
Tabla 7 Resultados por fold – Modelo A..... 49
Tabla 8 Resultados por fold – Modelo B..... 51
Tabla 9 Resultados por fold – Modelo C 52
Tabla 10 Cuadro comparativo de métricas globales por modelo 54
Tabla 11 Comparación de F1-score por clase entre modelos 54
Tabla 12 Accuracy por corrida de los modelos AlexNet+RF, ResNet18+RF y VGG16+RF 59
Tabla 13 Prueba de homogeneidad de varianzas para el Accuracy..... 60
Tabla 14 ANOVA para el accuracy entre los modelos híbridos 60
Tabla 15 Comparaciones múltiples mediante la prueba de Tukey HSD (Accuracy). ... 61
Tabla 16 Estadísticos descriptivos del accuracy por modelo..... 62

Índice de figuras

Figura 1 Ubicación del distrito de Sauce en la región San Martín, Perú. 27
Figura 2 Diagrama de pasos del objetivo específico 1 32
Figura 3 Corrales individuales para observación preparto- 33
Figura 4 Instalación de cámaras para observación preparto. 33
Figura 5 Verificación de cámaras para registro del comportamiento. 34
Figura 6 Ejemplos de comportamiento normal en cerdas preparto. 35
Figura 7 Ejemplos de comportamiento de inquietud en cerdas preparto. 35
Figura 8 Ejemplos de comportamiento tumbada en cerdas preparto. 36
Figura 9 Ejemplos de comportamiento de parto en cerdas. 36
Figura 10 Organización de registros de video según categoría conductual. 37
Figura 11 Diagrama de pasos del objetivo específico 2 37
Figura 12 Diagrama de pasos del objetivo específico 3 40
Figura 13 Configuración del sistema de grabación para monitoreo preparto..... 42
Figura 14 Cantidad de videos por categoría de comportamiento. 43
Figura 15 Ejecución del programa de extracción para la clase "tumbada". 44
Figura 16 Ejemplos por categoría: movimiento normal, inquietud, tumbada, parto. 44
Figura 17 Distribución de imágenes por categoría en el dataset balanceado..... 45
Figura 18 Matriz de confusión global del Modelo A 49
Figura 19 Curvas ROC por clase para el Modelo A..... 50
Figura 20 Matriz de confusión global del Modelo B..... 51
Figura 21 Curvas ROC por clase para el Modelo B 52
Figura 22 Matriz de confusión global del Modelo C..... 53
Figura 23 Curvas ROC por clase para el Modelo C 53
Figura 24 Comparación de métricas globales por modelo 55
Figura 25 Comparación visual del F1-score por clase y modelo 55
Figura 26 Distribución por clase en los bloques del Modelo A (AlexNet+RF). 57
Figura 27 Distribución por clase en los bloques del Modelo B (ResNet18+RF)..... 57
Figura 28 Distribución por clase en los bloques del Modelo C (VGG16+RF). 58
Figura 29 Comparación de medias de accuracy entre los tres modelos híbridos. 62

RESUMEN

Monitoreo y predicción del momento del parto en cerdas mediante modelos de visión artificial basado en CNN

La presente investigación tuvo como objetivo principal predecir el momento del parto en cerdas mediante el uso de modelos híbridos de visión artificial que combinan redes neuronales convolucionales (CNN) con el clasificador Random Forest. El estudio se desarrolló en el distrito de Sauce, región San Martín (Perú), entre diciembre de 2024 y octubre de 2025, en condiciones naturales de granja y bajo principios éticos internacionales. Se capturaron 80 videos de cerdas gestantes en tiempo real, a partir de los cuales se extrajeron más de 24,000 imágenes clasificadas en cuatro categorías conductuales: movimiento normal, inquietud, tumbada y parto. Estas imágenes fueron procesadas automáticamente mediante un algoritmo basado en OpenCV y FFmpeg. La investigación fue de tipo aplicada, nivel experimental y diseño cuasiexperimental, con enfoque cuantitativo y corte longitudinal. Se emplearon tres arquitecturas CNN preentrenadas —AlexNet, VGG16 y ResNet18— como extractores de características visuales, cuyos vectores fueron posteriormente clasificados por el algoritmo Random Forest. La validación se realizó mediante validación cruzada K-Fold (k=5) y análisis estadístico mediante ANOVA y prueba post hoc de Tukey, evaluando métricas como precisión (accuracy), F1-score y tiempo de inferencia. Los resultados mostraron que los tres modelos lograron identificar adecuadamente las conductas preparto, pero fue el modelo AlexNet + Random Forest el que alcanzó el mejor rendimiento general, con una precisión promedio del 99.55%, superando significativamente a los demás. Este modelo demostró ser eficiente y menos demandante computacionalmente, lo que lo hace especialmente adecuado para contextos rurales con recursos limitados. En conclusión, el sistema propuesto ofrece una solución no invasiva, automatizada y eficaz para predecir el parto en cerdas, mejorando la gestión reproductiva, reduciendo la mortalidad neonatal y aumentando la productividad de las granjas porcinas

Palabras clave: Análisis de comportamiento, Procesamiento de imágenes, Aprendizaje profundo, Modelado computacional, Validación estadística.

ABSTRACT

Monitoring and Predicting the Time of Farrowing in Sows Using CNN-Based Computer Vision Models

The main objective of this study was to predict the time of farrowing in sows using hybrid computer vision models that combine convolutional neural networks (CNNs) with a Random Forest classifier. The study was conducted in the district of Sauce, San Martín region (Peru), between December 2024 and October 2025, under natural farm conditions and in accordance with international ethical principles. Eighty real-time videos of pregnant sows were captured, from which more than 24,000 images were extracted and classified into four behavioral categories: normal movement, restlessness, lying down, and farrowing. These images were automatically processed using an algorithm based on OpenCV and FFmpeg. The research was applied, experimental in nature, and used a quasi-experimental design, with a quantitative approach and a longitudinal study design. Three pre-trained CNN architectures—AlexNet, VGG16, and ResNet18—were used as visual feature extractors, and their feature vectors were subsequently classified by the Random Forest algorithm. Validation was performed using K-fold cross-validation (k=5) and statistical analysis via ANOVA and Tukey's post hoc test, evaluating metrics such as accuracy, F1-score, and inference time. The results showed that all three models were able to adequately identify pre-farrowing behaviors, but the AlexNet + Random Forest model achieved the best overall performance, with an average accuracy of 99.55%, significantly outperforming the others. This model proved to be efficient and less computationally demanding, making it particularly suitable for rural settings with limited resources. In conclusion, the proposed system offers a non-invasive, automated, and effective solution for predicting farrowing in sows, improving reproductive management, reducing neonatal mortality, and increasing the productivity of pig farms.

Keywords: Behavioral analysis, Image processing, Deep learning, Computational modeling, Statistical validation.

CAPÍTULO I
INTRODUCCIÓN A LA INVESTIGACIÓN

La producción mundial de cerdos confronta las presiones ambientales y económicas que requieren innovaciones tecnológicas sofisticadas para mejorar su sostenibilidad y competitividad (Oczak, Maschat, and Baumgartner 2023); en este marco, la salud materna y la supervivencia de los lechones son consideraciones fundamentales, particularmente dentro del contexto de granjas de cerdos, donde la atención rápida y eficiente es primordial (Oczak et al. 2022); la integración de los análisis visuales impulsados por la inteligencia artificial puede optimizar el monitoreo de grandes conjuntos de datos, ofreciendo ideas sobre intervenciones inmediatas (Chen, Zhu, and Norton 2021; Yang et al. 2023).

En el contexto peruano, aunque la industria de los cerdos ha experimentado un crecimiento sustancial entre más de 400,000 productores, todavía existe una falta de adopción generalizada de tecnologías de monitoreo avanzadas como las redes neuronales (Ministerio de Desarrollo Agrario y Riego 2024); esto contribuye a las tasas de mortificación más altas de preparación y la reducción de la eficiencia en el manejo de los Animales (Xie et al. 2024); la necesidad de un monitoreo más confiable, uno que puede ajustarse rápidamente a los cambios repentinos en la cadena de suministro y la dinámica del mercado (Devi Priya et al., 2022); divida este contexto, integrando la inteligencia artificial en la salud y el rendimiento de las cerdas y los cerditos se vuelven aún más críticos para mejorar la productividad de los sectores y abordar los desafíos emergentes (Li et al. 2021).

En las granjas porcinas rurales de San Martin, donde el acceso a tecnologías avanzadas es limitado, la implementación de redes neuronales podría transformar la gestión preparto, proporcionando herramientas que permitan la atención anticipada y mejoren los resultados reproductivos (van Erp-van der Kooij et al. 2023; Oczak et al. 2022). En áreas rurales como Tarapoto, aunque los métodos tradicionales de inseminación y monta son viables, son insuficientes para abordar los desafíos modernos. Una alternativa podría ser la implementación de un sistema basado en un modelo de visión artificial, el cual tendría el potencial de optimizar estos resultados, contribuyendo a una mejora en la salud de las cerdas y una posible reducción en la mortalidad de los lechones (K.-Y. Ho, Tsai, and Kuo 2021; Walls et al. 2024).

Los sistemas automatizados ineficientes en granjas porcinas retrasan la atención, aumentan los problemas de salud y afectan la rentabilidad (de Paula et al. 2024). El uso

de observación manual y registros en papel, aún común en muchas granjas, no es suficiente para manejar la cantidad de información necesaria (Dore et al. 2022; Havlíček et al. 2020). La falta de un monitoreo constante de la cerda en los momentos previos al parto dificulta la detección a tiempo de problemas, lo que puede poner en riesgo tanto la salud de la madre como la de los lechones (Moustsen, Seddon, and Hansen 2023).

La capacitación en tecnologías digitales y usar sistemas automatizados es crucial para mejorar el rendimiento reproductivo en las granjas porcinas (J. Lee et al. 2024). Sin embargo, implementar un sistema de visión por computadora para monitorear a las cerdas antes del parto enfrenta desafíos como los costos y la infraestructura limitada en áreas rurales (Gibbs and Cappuccio 2022; Yin et al. 2024); es importante adoptar soluciones que se ajusten a las necesidades locales para superar estos problemas (Küster et al. 2021).

El monitoreo ineficiente en granjas porcinas puede provocar pérdidas económicas significativas debido a la mortalidad preparto y afectar la rentabilidad (Oczak et al. 2023). Además, el tiempo que los cuidadores dedican a la observación manual podría reorientarse hacia otras tareas productivas mediante el uso de sistemas automatizados que proporcionen análisis predictivo y recomendaciones en tiempo real (J. Lee et al. 2024).

La calidad del cuidado preparto y los productos derivados dependen directamente de la efectividad de las tecnologías de monitoreo utilizadas (Vandresen, Chou, and Hötzel 2024); la dependencia de métodos manuales para la observación y el registro de datos limita la capacidad para gestionar grandes volúmenes de información y aprovechar el análisis predictivo (Domingos et al. 2024). La falta de sistemas automatizados que optimicen la recolección y el análisis de datos en tiempo real resulta en una gestión ineficiente, reduciendo la eficiencia operativa y aumentando el riesgo de complicaciones (Liu et al. 2022).

Aunque existen estudios sobre tecnologías de monitoreo en la industria porcina, hay una escasez de investigaciones específicas sobre la implementación de modelos de visión artificial en granjas porcinas rurales, lo que crea un vacío científico en la optimización de estos sistemas para diferentes entornos (Nannoni et al. 2020). La implementación de un sistema automatizado basado en redes neuronales en cerdas preparto está limitada por la complejidad técnica y los costos, lo cual dificulta su adopción en granjas locales (Hukkinen et al. 2024). Sin embargo, adaptar estas tecnologías al contexto rural podría mejorar la eficiencia y el bienestar animal, como lo plantean (Espejo et al. 2022; Fynn, Crow, and Connor 2021).

Este proyecto busca desarrollar un sistema que, a través de modelos de visión artificial basados en redes convolucionales (CNN), ayude a predecir con mayor precisión el momento del parto en cerdas. La idea es observar su comportamiento en tiempo real y generar alertas tempranas que permitan a los agricultores actuar a tiempo, para darle asistencia y evitar la mortandad durante y después del parto. Así, no solo se mejora la productividad y rentabilidad de las granjas, sino también el cuidado y bienestar de los animales (Bonneau et al. 2021; De Computación et al. 2024)

Por consiguiente, la problemática abordada en este estudio fue determinar si modelos híbridos que integran la extracción de características mediante redes neuronales convolucionales (CNN) y la clasificación con algoritmos tradicionales como Random Forest pueden contribuir de manera efectiva a la predicción del momento del parto en cerdas. Esta necesidad responde a las limitaciones técnicas y operativas de los sistemas de monitoreo tradicionales, especialmente en granjas rurales como las del distrito de Sauce, en la región San Martín, donde el acceso a tecnologías especializadas es reducido y el monitoreo manual suele ser ineficaz o tardío.

Para dar respuesta a esta problemática general, se identificaron tres cuestiones específicas. Primero, la necesidad de identificar patrones visuales previos al parto, a partir de imágenes que reflejen las posturas y comportamientos más representativos del proceso periparto. Segundo, la construcción de modelos híbridos basados en CNN y Random Forest, utilizando tres arquitecturas específicas de redes convolucionales —AlexNet, ResNet18 y VGG16— como extractores de características profundas. Tercero, la evaluación comparativa de estos tres modelos híbridos, considerando métricas de precisión y eficiencia que permitan establecer cuál de ellos presenta el mejor desempeño en la predicción del parto.

Con el propósito de resolver estas interrogantes, se formuló la hipótesis general de que los modelos de visión artificial híbridos, conformados por una red convolucional como extractor y un clasificador Random Forest, permiten predecir de manera efectiva el momento del parto en cerdas mediante el análisis automatizado de imágenes. A partir de esta hipótesis general, se formularon tres hipótesis específicas: (1) las cerdas presentan patrones visuales preparto detectables mediante el análisis de imágenes; (2) cada arquitectura CNN (AlexNet, ResNet18 y VGG16) puede extraer representaciones visuales útiles para la clasificación; y (3) los modelos híbridos obtenidos —CNN + Random Forest— presentan un rendimiento efectivo en la predicción del parto, con diferencias estadísticamente significativas entre ellos.

En esta investigación, el objetivo principal fue predecir el momento del parto en cerdas usando modelos híbridos de visión artificial, combinando redes neuronales (CNN) con el clasificador Random Forest. Para lograrlo, se propusieron tres objetivos específicos. El primero fue reconocer patrones visuales importantes del comportamiento antes del parto, observando imágenes sacadas de videos grabados en granjas rurales. El segundo objetivo fue implementar tres modelos híbridos diferentes: (i) AlexNet + Random Forest, (ii) ResNet18 + Random Forest y (iii) VGG16 + Random Forest. En estos modelos, las redes CNN se usaron para extraer las características visuales, y se utilizó el mismo clasificador para poder compararlos de manera justa. El tercer objetivo fue evaluar qué tan bien funcionaban estos tres modelos híbridos, usando una validación cruzada K-Fold (k=5) y un análisis estadístico (ANOVA y pruebas post hoc), para saber cuál tenía mejores resultados en precisión, F1-score y rapidez al procesar.

Basado en el problema identificado y los objetivos propuestos, este estudio se organizó en cuatro capítulos. En el Capítulo I se mostró la introducción, donde se explicó el problema, los objetivos y la hipótesis general. El Capítulo II trató sobre el marco teórico, incluyendo estudios anteriores y las bases sobre redes neuronales convolucionales y clasificación supervisada. En el Capítulo III se explicaron los materiales usados, el diseño del experimento, cómo se capturaron y procesaron las imágenes, cómo se crearon los modelos híbridos y los métodos que se usaron para validarlos. Por último, en el Capítulo IV se presentaron los resultados y su análisis comparativo, y luego se incluyeron las conclusiones, recomendaciones, referencias y anexos técnicos.

CAPÍTULO II
MARCO TEÓRICO

2.1. Antecedentes de la investigación

En su estudio, (Küster et al. 2021) hicieron una revisión enfocada en detectar de forma automática el comportamiento y la postura de las cerdas en corrales abiertos, usando imágenes en 2D y redes neuronales profundas (DNN). El objetivo fue crear un método usando visión por computadora para identificar y ubicar las partes importantes de las cerdas y del corral. Usando el algoritmo de detección de objetos YOLO V3, pudieron reconocer distintas partes del cuerpo de las cerdas, como la cabeza, las patas y la ubre, con una precisión que iba desde 0.66 hasta 0.97, según la parte detectada. Los resultados de este estudio muestran que la visión por computadora tiene mucho potencial para clasificar automáticamente el comportamiento y la postura de las cerdas, lo que podría ayudar a crear sistemas de monitoreo automático de partos en las granjas.

En su estudio, (J. Chen et al. 2023) llevaron a cabo un estudio en Nanjing, China, en el cual implementaron un sistema de advertencia temprana para el monitoreo del parto en cerdas, utilizando una plataforma de inteligencia artificial integrada en el dispositivo NVIDIA Jetson Nano; este sistema, basado en el modelo YOLOv5, alcanzó una precisión del 93.5% en la detección de posturas de cerdas y lechones, lo que representa una mejora significativa en la predicción del parto; además, la migración del procesamiento a dispositivos en nodos redujo considerablemente los costos del monitoreo en la nube, haciendo viable su aplicación en grandes granjas porcinas; estos avances permitieron generar alertas tempranas con un error promedio de 1.02 horas antes del parto, facilitando la intervención oportuna y mejorando el bienestar animal, al reducir problemas como la distocia y la mortalidad neonatal.

En su estudio, (Wutke et al. 2024) llevaron a cabo una investigación enfocada en mejorar el monitoreo automático del parto de cerdas mediante un enfoque de aprendizaje profundo denominado Noisy Student; este método busca mejorar el rendimiento de detección de lechones recién nacidos en condiciones complejas, como la alta tasa de superposición entre los animales y posturas heterogéneas, que dificultan la detección precisa; utilizaron un modelo maestro que, a través de datos manualmente anotados, generó pseudo-etiquetas que fueron utilizadas para entrenar un modelo "estudiante"; esto permitió mejorar métricas de rendimiento, alcanzando un F1-score de 0.922 y mejorando la detección de lechones en el proceso de parto, sin necesidad de monitorear

todo el corral; este enfoque no solo reduce los costos asociados a la anotación manual de datos, sino que también optimiza el monitoreo automático, lo cual es crucial para implementar sistemas de advertencia temprana en granjas porcinas.

En su estudio, (J. Lee et al. 2024) llevaron a cabo una investigación orientada a mejorar el monitoreo de comportamientos preparto en cerdas mediante inteligencia artificial; su objetivo fue identificar posturas clave como la “echada lateral” y la “inquietud”, señales tempranas que preceden el parto; para ello, emplearon arquitecturas YOLOv7 y YOLOv9 optimizadas con bloques Mixed-ELAN, entrenadas con videosecuencias etiquetadas; el sistema logró incrementar la precisión de detección en un 7%, manteniendo una velocidad de inferencia superior a 80 fps; estos resultados permitieron alertar con anticipación a los cuidadores sobre cambios conductuales relevantes, optimizando la intervención humana y reduciendo el riesgo de complicaciones durante el alumbramiento.

En su estudio, (Walls et al. 2024) desarrollaron un modelo de aprendizaje auto-supervisado basado en el enfoque Noisy Student, dirigido a la detección automática de lechones recién nacidos; el propósito fue mejorar la precisión de detección sin depender de grandes volúmenes de datos manualmente etiquetados; para lograrlo, utilizaron un modelo maestro que generó pseudo-etiquetas, las cuales sirvieron para entrenar un modelo estudiante; el sistema alcanzó un F1-score de 0.922 en ambientes complejos; este resultado valida la efectividad del modelo para implementar alarmas de parto en tiempo real, contribuyendo a una supervisión eficiente en contextos comerciales.

En su estudio, (Yin et al. 2024) llevaron a cabo el diseño de un modelo ligero basado en MobileNet distilado, orientado a la detección de partos en granjas con recursos limitados; su objetivo fue permitir el monitoreo eficiente sin necesidad de infraestructura costosa; utilizaron técnicas de distilación de conocimiento y poda estructural para reducir la carga computacional del modelo; el sistema alcanzó una precisión del 91.48% y una velocidad de procesamiento de 83.10 fps; este enfoque permitió escalar la tecnología a contextos rurales o de menor presupuesto, manteniendo un seguimiento automatizado de alta calidad.

2.2. Fundamentos teóricos

2.2.1. Monitoreo predicción del momento del parto

2.2.1.1. Importancia del monitoreo y predicción del parto

El monitoreo del momento del parto en cerdas es una práctica crucial en la gestión porcina, ya que permite reducir la mortalidad neonatal y mejorar el bienestar animal; a nivel teórico, esta necesidad fue abordada por múltiples disciplinas, desde la etología

hasta la ingeniería agrícola; en su enfoque más clásico, el monitoreo se realiza mediante observación directa del comportamiento de la cerda, donde signos como la inquietud, la preparación del nido o los cambios posturales repetitivos indican la proximidad del parto (Bonneau et al. 2021; Gulliksen et al. 2023).

Varios estudios han demostrado que observar temprano el comportamiento de las cerdas antes de parir ayuda mucho a tener mejores resultados en el parto. Notar señales como que están más inquietas, comen menos o se echan de lado fue muy importante para saber cuándo iban a dar a luz. Gracias a estas señales, el personal del campo pudo actuar a tiempo, lo que ayudó a evitar problemas como partos difíciles y aumentó la cantidad de lechones que sobrevivieron (Riekert et al., 2020).

También se ha visto que cosas del ambiente, como la luz del corral, el ruido y la temperatura, afectan cómo se comportan las cerdas antes de parir. Observarlas todo el tiempo en estas condiciones ayuda a notar ciertos hábitos que avisan que el parto está por empezar, sin tener que usar métodos que puedan molestarlas (J. Lee et al. 2024).

Además, la clasificación sistemática de posturas y conductas clave ha llevado a proponer tipologías conductuales útiles para el manejo zootécnico. Estas incluyen posturas como "echada prolongada", "movimiento reiterado en espacios reducidos", "interacción repetitiva con el sustrato del corral", entre otras. La observación ordenada de estas conductas forma parte esencial de una gestión reproductiva eficaz (Shao, Pu, and Mu 2021).

La estimación del momento del parto en cerdas se define en esta investigación como la capacidad de determinar con anticipación la proximidad del evento de alumbramiento a partir del análisis de patrones conductuales, especialmente posturales, observados mediante técnicas sistemáticas de registro y observación. Este proceso se operacionaliza en cuatro categorías observables: movimiento normal, inquietud, tumbada y parto. Estas posturas fueron definidas empíricamente a partir de la revisión de literatura científica y de observaciones en campo.

La clasificación de dichas posturas permitió modelar el proceso de parto como una secuencia detectable, en la cual cada transición entre estados comportamentales representa un indicador del momento periparto. Esta aproximación conceptual, basada en señales visuales objetivas, fortaleció la base científica de la investigación al permitir una interpretación reproducible y cuantificable del fenómeno observado (Qin, Li, and Jia 2025).

2.2.1.2. Métodos tradicionales de monitoreo del parto

Históricamente, la predicción del parto en cerdas se ha basado en la observación manual por parte de los trabajadores de granja. Este enfoque incluye la identificación visual de signos como el aumento de la actividad, la pérdida de apetito, el inicio de la conducta de anidamiento y los cambios respiratorios. Sin embargo, este tipo de monitoreo depende significativamente de la experiencia del observador y de la disponibilidad de personal calificado (Z. Chen, Lu, and Wang 2023).

La implementación de sensores físicos como acelerómetros, termómetros auriculares y medidores de frecuencia cardíaca ha representado una mejora frente al monitoreo visual, pero sigue siendo una solución costosa y, en muchos casos, invasiva; además, estos dispositivos requieren mantenimiento constante y presentan desafíos en cuanto a su durabilidad y precisión bajo condiciones de granja.

En este sentido, el paso hacia sistemas automatizados y no invasivos basados en visión artificial representa un cambio paradigmático; estudios como el de (Shao et al. 2021) describen detalladamente la clasificación sistemática de posturas como herramienta para la identificación temprana del parto, proponiendo un sistema conductual que puede aplicarse en rutinas de observación estructurada; sus hallazgos demostraron que modelos entrenados con información visual pueden identificar con éxito posturas como inquietud, descanso o parto inminente, ofreciendo una alternativa efectiva al monitoreo convencional.

2.2.1.3. Factores que influyen en la predicción del parto

La predicción del parto en cerdas se ve influenciada por múltiples factores conductuales, fisiológicos y ambientales. Uno de los principales es la variabilidad individual en la expresión del comportamiento preparto; mientras algunas cerdas muestran signos claros como inquietud persistente o aislamiento progresivo, otras pueden manifestar señales más sutiles o atípicas, lo cual puede dificultar la interpretación del momento exacto del alumbramiento (Bonneau et al. 2021).

Además, el ambiente de la granja incluyendo temperatura, ruido, iluminación y diseño del corral puede afectar la manifestación de comportamientos clave; por ejemplo, condiciones de estrés térmico o hacinamiento pueden alterar el patrón natural de conducta previo al parto, generando respuestas no esperadas en las cerdas gestantes (Gulliksen et al. 2023).

También es relevante el nivel de experiencia del personal encargado del monitoreo; la habilidad para reconocer signos como la postura tumbada prolongada, la interacción

con el sustrato o el aumento de la frecuencia respiratoria incide directamente en la precisión de la predicción (Riekert et al. 2020).

No obstante, existen condiciones favorables que pueden mejorar significativamente la precisión en la predicción del parto; por ejemplo, la implementación de protocolos estandarizados de observación, la capacitación continua del personal y la integración de datos históricos de comportamiento por cerda han demostrado ser estrategias efectivas para anticipar con mayor exactitud el alumbramiento (J. Lee et al. 2024; Shao et al. 2021).

Asimismo, corrales con diseño ergonómico, adecuados niveles de iluminación natural y ambientes poco ruidosos fomentan un comportamiento más claro y predecible en las cerdas, facilitando la identificación de los signos de parto inminente; estas condiciones positivas, cuando se mantienen de forma constante, permitieron una mejora considerable en la calidad del monitoreo tradicional, incluso sin tecnologías avanzadas.

2.2.2. Modelos de visión artificial basados en CNN

2.2.2.1. Importancia de los modelos de visión artificial en el monitoreo del parto

La visión artificial basada en redes neuronales convolucionales (CNN) transformó el campo de la predicción animal, ofreciendo una alternativa eficaz y automatizada para el monitoreo del comportamiento preparto en cerdas; esta tecnología se fundamentó en el reconocimiento de patrones visuales complejos que, en contextos agropecuarios, se traducen en señales claras de proximidad al parto, como la postura lateral, la agitación o el comportamiento de anidamiento; estos indicadores, tradicionalmente observados por personal capacitado, ahora pueden ser detectados de manera constante y precisa a través de sistemas basados en CNN, lo cual optimiza los recursos humanos y reduce la tasa de mortalidad neonatal.

Desde un enfoque teórico, las CNN pertenecen al ámbito del aprendizaje profundo (deep learning) y se caracterizan por su capacidad jerárquica para abstraer información relevante desde los niveles más simples (bordes, texturas) hasta patrones semánticos complejos (acciones o posturas); esta estructura las convierte en herramientas idóneas para entornos variables y de difícil supervisión continua.(Shao et al. 2021) destacan que el uso de CNN mejora la eficiencia en la clasificación de posturas animales, mientras que (Makalesi et al. 2025; Walls et al. 2024) afirman que estos modelos permitieron una gestión más sostenible del ambiente ganadero mediante sistemas inteligentes; asimismo,(Makalesi et al. 2025) señala que la integración de transformaciones como la Stockwell Transform puede enriquecer la representación de señales visuales o

fisiológicas en sistemas de aprendizaje profundo, permitiendo una clasificación más robusta y precisa en escenarios dinámicos.

En este estudio, se usó un modelo de visión artificial con redes neuronales (CNN) que se entiende como un sistema de computadora que puede reconocer, clasificar y predecir el comportamiento de las cerdas antes del parto, analizando imágenes o videos de forma automática. Este modelo funciona con redes que imitan cómo ve el ojo humano, y va sacando detalles importantes para notar patrones más complicados, como los cambios en la forma en que se mueven o si están haciendo nidos. En este caso, el objetivo fue reconocer cuatro posturas que se pueden ver fácilmente: cuando la cerda se mueve normal, cuando está inquieta, cuando está echada o cuando está pariendo. Así se puede saber con tiempo cuándo va a dar a luz, sin tener que molestarla, y de forma constante y que se puede repetir (Bhatt et al. 2021; Makalesi et al. 2025).

2.2.2.2. Funcionamiento de los modelos de visión artificial basados en CNN

En términos operativos, una CNN se compone de capas convolucionales, capas de activación, capas de agrupamiento (pooling) y capas completamente conectadas (fully connected); cada una de estas capas cumple una función específica en la extracción progresiva de características visuales. En esta investigación, se emplearon tres arquitecturas específicas —AlexNet, ResNet18 y VGG16— las cuales fueron utilizadas como modelos extractores de características. Después, esas características se usaron como datos de entrada para un modelo llamado Random Forest. Este modelo ayudó a mejorar el proceso haciendo una clasificación fuerte y rápida, usando los datos que habían sido generados por las redes neuronales.

AlexNet: Es un modelo conocido por ser sencillo y muy bueno para clasificar cosas. En este estudio, se usó para sacar características importantes de las imágenes, pero no se usó su última parte, sino que esa se cambió por un modelo de clasificación llamado Random Forest. Gracias a esta combinación, AlexNet ayudó a sacar imágenes claras y útiles, y el modelo Random Forest se encargó de clasificar las posturas que se veían (movimiento normal, inquietud, echada y parto). Esta forma de trabajo fue muy útil porque funciona rápido y no necesita muchas cosas de la computadora, lo que la hace ideal para lugares donde no hay muchos recursos (Kuldashboy et al. 2024).

Se destaca porque usa bloques especiales que ayudan a que el aprendizaje no se vuelva peor mientras el modelo va aprendiendo. En el presente trabajo, se utilizó como extractor de características, manteniendo congeladas sus primeras capas para preservar representaciones generales y extrayendo las salidas de sus capas finales

como vectores característicos. Estos vectores fueron posteriormente clasificados mediante un modelo Random Forest, lo que permitió una clasificación precisa de las posturas preparto. Esta arquitectura ofreció mayor estabilidad durante el entrenamiento y mejor capacidad de generalización frente a la alta variabilidad del conjunto de datos (S. Liu & Wang, 2024).

VGG16: Reconocida por su arquitectura profunda y uniforme, en este estudio se utilizó como extractor de características, modificando los filtros de las últimas capas convolucionales para adaptarse mejor a las variaciones sutiles de las posturas. En lugar de utilizar su capa densa final, se extrajeron los vectores de activación para ser clasificados mediante un modelo Random Forest. Esta combinación permitió aprovechar la alta capacidad de extracción de características de VGG16, a pesar de su mayor demanda computacional, mejorando la sensibilidad del sistema ante cambios finos en las posturas observadas (Zhou 2024).

Las imágenes utilizadas en este estudio fueron frames extraídos automáticamente de videosecuencias registradas del comportamiento de las cerdas en las etapas previas al parto. Dichos frames fueron etiquetados por un algoritmo de segmentación propio, el cual fue diseñado específicamente para identificar y aislar escenas relevantes con base en umbrales de movimiento y persistencia postural. Este procedimiento automatizado redujo la subjetividad y aumentó la eficiencia en la creación del conjunto de datos (Khaled et al. 2025); el entrenamiento de los modelos se realizó bajo un esquema de validación cruzada K-Fold, y las métricas utilizadas para la evaluación de desempeño incluyeron precisión, F1-score y tiempo de inferencia.

Asimismo, se aplicó destilación de conocimiento (Liu et al., 2025), con el objetivo de transferir los aprendizajes de modelos complejos a modelos más ligeros, manteniendo una alta exactitud. Se integró además la transferencia de aprendizaje a partir de modelos preentrenados, como los de (J. H. Lee et al. 2024), para optimizar el entrenamiento con datos propios del estudio. Este proceso contribuyó a una reducción significativa del tiempo de entrenamiento y una mejora en la estabilidad del aprendizaje.

2.2.2.3. Factores que influyen en la precisión de los modelos CNN

La precisión de los modelos CNN en el contexto de monitoreo preparto depende de diversos factores. Entre los más influyentes se encuentra la calidad y diversidad del conjunto de datos utilizado para el entrenamiento. Datos que representen distintos ángulos, condiciones lumínicas y variabilidad en la conducta permitieron una mayor capacidad de generalización; (J. H. Lee et al. 2024; Shao et al. 2021) evidencian que conjuntos de datos amplios y bien etiquetados mejoran la robustez predictiva.

Otro aspecto fundamental es la resolución y el posicionamiento de las cámaras utilizadas en la captura de imágenes. Factores como la obstrucción visual, la sobreposición entre animales o el ángulo de captura pueden comprometer la calidad del input visual. (Farahnakian et al. 2024) recomiendan complementar la visión por computadora con sensores de termografía y radar para incrementar la sensibilidad del sistema.

Finalmente, (Borges Oliveira et al. 2021) destacan el valor de aplicar modelos tipo ensemble, que integran múltiples arquitecturas CNN para reforzar la precisión del sistema. Aunque esta estrategia requiere mayor capacidad computacional, su implementación puede ser crítica en escenarios donde la exactitud es prioritaria, como en granjas comerciales con alta rotación reproductiva.

CAPÍTULO III
MATERIALES Y MÉTODOS

3.1. Ámbito y condiciones de la investigación

3.1.1. Contexto de la investigación

La presente investigación se desarrolló en granjas porcinas ubicadas en el distrito de Sauce, provincia y región de San Martín, al noreste del Perú (Figura 1). Esta zona pertenece a la selva alta y presenta una geografía variada, con presencia de montañas, bosques y cuerpos de agua como la Laguna Azul, lo que favorece las actividades agropecuarias, especialmente la ganadería porcina. Las condiciones naturales y rurales del distrito ofrecieron un entorno adecuado para la implementación de tecnologías de monitoreo orientadas al estudio del comportamiento preparto de las cerdas. La investigación se realizó respetando las normativas vigentes sobre bienestar animal y las disposiciones sanitarias regionales, garantizando un proceso ético y técnicamente controlado.

Figura 1
Ubicación del distrito de Sauce en la región San Martín, Perú.

3.1.2. Periodo de ejecución

La ejecución del proyecto se llevó a cabo durante el mes de diciembre del 2024 hasta el mes de julio del 2025.

3.1.3. Autorizaciones y permisos

Para la ejecución de la presente investigación, se contó con la autorización formal del señor Ramiro Rojas Mejía, propietario de una granja porcina ubicada en el distrito de Sauce, provincia de San Martín, quien permitió el acceso a las instalaciones para la observación y registro del comportamiento preparto de las cerdas. Las actividades se realizaron sin intervención directa en la rutina de los animales y sin el uso de sustancias químicas, equipos especiales ni métodos restringidos por normativa alguna. Asimismo, se garantizó que las herramientas empleadas fueran de uso libre y no invasivo, limitándose al uso de cámaras de video instaladas externamente. Aunque no fue necesario gestionar resoluciones oficiales ni permisos institucionales formales, todas las acciones se desarrollaron respetando los principios éticos y de bienestar animal, dado que no se manipuló a las cerdas ni se alteró su ciclo natural. La constancia de esta autorización se presenta en el Anexo 1 del presente trabajo.

3.1.4. Control ambiental y protocolos de bioseguridad

El proyecto se centró en observar cómo se comportan las cerdas antes de parir, prestando atención a momentos como la inquietud, cuando están tumbadas, el movimiento habitual y el parto. Estas observaciones se hicieron en espacios controlados dentro de granjas tradicionales. Para no alterar su comportamiento natural, se aplicaron medidas básicas de bioseguridad: se limpiaban y desinfectaban los corrales con frecuencia, y se cuidaba que las cámaras y equipos de grabación funcionaran correctamente. También se mantuvo un ambiente limpio y bien ventilado, con un manejo adecuado, para que las cerdas no se estresaran y los registros fueran lo más fieles posible a su comportamiento real, siguiendo las buenas prácticas establecidas por la Sedes - Servicio Nacional de Sanidad Agraria del Perú - Plataforma del Estado Peruano

3.1.5. Aplicación de principios éticos internacionales

En todo momento se respetaron los principios éticos de la investigación. No se manipuló a las cerdas ni se alteró su rutina. Las cámaras se colocaron en lugares estratégicos para observar su comportamiento antes del parto; como cuando estaban inquietas, tumbadas, en movimiento o en el momento del parto y sin causarles molestias. El monitoreo fue tranquilo, sin ruidos ni interrupciones, y se cuidó que no hubiera situaciones que pudieran estresarlas. Todo el proceso se hizo priorizando el bienestar de las madres y de los lechones, garantizando que la investigación se realizara de forma responsable y respetuosa con los animales, conforme a las directrices éticas internacionales propuestas por el International Guiding Principles for Biomedical Research Involving Animals - CIOMS y las normas del (Sedes - Servicio Nacional de Sanidad Agraria Del Perú - Plataforma Del Estado Peruano).

3.2. Sistema de variables

3.2.1. Variables principales

Variable dependiente: Estimación del momento del parto en cerdas porcinas en la región de San Martin.

Variable independiente: Modelos de visión artificial basados en CNN.

Tabla 1

Descripción de variables por objetivo específico 1

Objetivo específico número 1: Identificar patrones visuales previos al parto en cerdas mediante el análisis de imágenes capturadas.

Variable abstracta	Variable concreta	Medio de registro	Unidad de medida
Monitoreo y predicción del momento del parto	Imágenes del cambio en el comportamiento de cerdas	Captura de imágenes con cámaras de visión artificial en tiempo real	Unidad
	Etiqueta que clasifica ese cambio	Análisis por expertos	Unidad

Tabla 2

Descripción de variables por objetivo específico 2

Objetivo específico número 2: Diseñar modelos de visión artificial basado en CNN para el monitoreo y predicción del parto en cerdas.

Variable abstracta	Variable concreta	Medio de registro	Unidad de medida
Modelos de visión artificial basado en CNN	Accuracy	Resultados de validación cruzada sobre imágenes clasificadas	Porcentaje (%)
	F1 - score	Análisis del equilibrio entre precisión y sensibilidad	Valor entre 0 y 1
	ROC AUC (OVR)	Área bajo la curva ROC multiclase en clasificación de imágenes de prueba	Valor entre 0 y 1

3.2.2. Variables secundarias

No aplica.

3.3. Procedimientos de la investigación

3.3.1. Diseño de la investigación

La investigación fue de tipo aplicada, de nivel experimental y de corte longitudinal. Se llevó a cabo en granjas porcinas. Durante el estudio, se capturaron videos en tiempo

real de las cerdas durante las horas previas al parto. De estos videos se extrajeron fotogramas que fueron procesados mediante modelos de visión artificial basados en redes neuronales convolucionales (CNN), las cuales se emplearon como extractores automáticos de características visuales relevantes (Z. Chen et al. 2023; Farahnakian et al. 2024). Posteriormente, los vectores generados por las CNN fueron utilizados como entrada para un modelo de clasificación Random Forest, encargado de identificar las distintas posturas y predecir de forma no invasiva el momento del parto (Riekert et al. 2020; Yin et al. 2024).

Respecto al nivel de investigación, se clasificó como descriptivo, al identificar y documentar patrones visuales observados en las cerdas antes del parto a partir del análisis de los fotogramas extraídos. Asimismo, se consideró explicativo, ya que se analizó cómo dichos patrones fueron representados por las CNN y posteriormente aprovechados por el modelo Random Forest para mejorar la precisión de la predicción y optimizar la gestión reproductiva en las granjas porcinas (Shao et al. 2021; Wei et al. 2023).

La población del estudio estuvo conformada por registros videográficos de cerdas reproductoras en etapa avanzada de gestación, alojadas en granjas porcinas ubicadas en el distrito de Sauce, provincia y región San Martín. Estas granjas fueron seleccionadas por disponer de infraestructura adecuada para la instalación de sistemas de monitoreo continuo con cámaras.

La muestra fue seleccionada mediante un muestreo intencional por conveniencia, con el objetivo de obtener material audiovisual útil para el análisis de posiciones en cerdas gestantes. El proceso consistió en la recopilación de 80 videos grabados en tiempo real dentro de corrales de gestación, bajo condiciones naturales y sin interferencia directa en la rutina de los animales. Cada video representa una unidad muestral y contiene secuencias continuas del comportamiento preparto. Posteriormente, se diseñará e implementará un algoritmo de extracción automática de imágenes a partir de estos videos. Dicho algoritmo empleará funciones de lectura de video cuadro por cuadro, a una tasa de muestreo controlada (frames por segundo), utilizando bibliotecas especializadas como OpenCV. El objetivo es capturar fotogramas clave correspondientes a las distintas posiciones corporales observadas (tumbada, inquietud, movimiento normal y parto), estandarizando su resolución y formato para asegurar uniformidad en el conjunto de datos. Este enfoque técnico permite generar un volumen significativo de imágenes representativas para cada posición sin intervención manual. A partir de los 80 videos, se proyecta la generación de más de 6000 imágenes por

categoría conductual, lo cual proporciona una base robusta para el entrenamiento de modelos de clasificación basados en redes neuronales convolucionales. La construcción del dataset se realiza de manera automatizada, sistemática y ética, asegurando la integridad del proceso investigativo.

El diseño adoptado fue de carácter cuasiexperimental, con enfoque aplicado y cuantitativo, y muestreo no probabilístico por conveniencia. Para la clasificación de comportamientos preparto en cerdas, se utilizaron imágenes extraídas de registros videográficos previamente seleccionados. En la etapa de modelado se emplearon redes neuronales convolucionales preentrenadas, específicamente AlexNet, VGG16 y ResNet, utilizadas como extractores de características mediante la remoción de sus capas finales de clasificación. Esta estrategia permitió adaptar modelos ya entrenados a un nuevo contexto sin requerir entrenamiento desde cero (Z. Chen et al., 2023; Lei et al., 2022). A partir de las imágenes procesadas, se generaron vectores representativos que sirvieron como insumo para la fase de clasificación, orientada a identificar cuatro conductas clave: movimiento normal, inquietud, tumbada y parto (Bonneau et al. 2021).

La información se organizó en carpetas clasificadas según las categorías de comportamiento observadas: inquietud, movimiento normal, tumbada y parto. Cada imagen que se sacó fue renombrada con un código que decía a qué cerda pertenecía, qué postura tenía y el número del fotograma. Esto ayudó a mantener todo en orden, a poder seguir el rastro de los datos y a que fuera más fácil trabajar con ellos en las etapas de análisis y procesamiento.

Se hizo un análisis estadístico básico para entender cómo funcionaron los modelos de redes neuronales que se usaron para clasificar el comportamiento de las cerdas antes del parto. Para eso, se calcularon los promedios de precisión (accuracy), la puntuación F1 (F1-score) y el tiempo que tardaba en procesar cada imagen, usando los resultados obtenidos en cada parte de la validación cruzada por grupos (Küster et al. 2021; Zhang et al. 2020). Estas medidas sirvieron para evaluar qué tan buenos eran los modelos, tanto en su precisión como en el equilibrio entre lo que detectaban bien y lo que no. Además, se hizo un análisis llamado ANOVA para ver si había diferencias importantes entre los modelos que se probaron. Este proceso ayudó a comparar de forma clara el rendimiento de los modelos AlexNet, VGG16 y ResNet como extractores, junto con el clasificador Random Forest, para elegir el que funcionaba mejor. Los resultados se pusieron en una tabla comparativa, lo que hizo más fácil ver cuál modelo tuvo el mejor desempeño en clasificación y en uso de la computadora.

3.3.2. Objetivo específico 1: Identificar patrones visuales previos al parto en cerdas mediante el análisis de imágenes capturadas

Para desarrollar un modelo que reconozca conductas preparto en cerdas mediante imágenes, se construyó un conjunto de datos visuales clasificados. Este dataset incluyó fotogramas reales obtenidos a partir de grabaciones en granjas, reflejando comportamientos naturales en las etapas finales de gestación.

Con el fin de garantizar la calidad y representatividad del material, se aplicó un protocolo sistemático dividido en tres etapas: preparación del entorno, registro videográfico del comportamiento y procesamiento de imágenes. Cada fase fue diseñada para asegurar la continuidad del registro, la organización por categorías y la validez visual del contenido generado. El procedimiento completo se encuentra representado en un modelo conceptual (Figura 2).

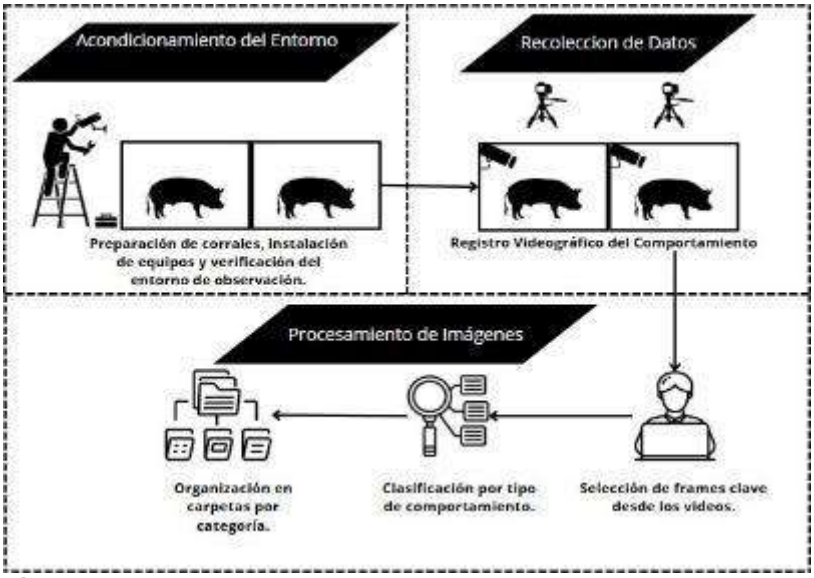

Figura 2
Diagrama de pasos del objetivo específico 1

FASE 1: Acondicionamiento del Entorno

Etapa 1: Preparación del entorno de observación

Para poder observar el comportamiento antes del parto, se usaron corrales individuales que ayudaron a ver claramente y todo el tiempo a cada cerda. Como el espacio estaba bien cerrado, fue más fácil reconocer las posturas y los movimientos de las cerdas en un lugar controlado, sin que nada de afuera molestara (Bonneau et al. 2021). Las imágenes que vienen a continuación muestran los lugares que se prepararon para este propósito (Figura 3).

Figura 3
Corrales individuales para observación preparto-

Se instalaron dos cámaras: una ubicada en un trípode dentro del corral para captar el ángulo frontal, y otra en el exterior en posición elevada para obtener una vista general. Las dos cámaras grabaron al mismo tiempo. Además, se usó un celular para tomar tomas más cercanas que ayudaran a complementar (Chen et al. 2021; Oczak et al. 2023). A continuación, se muestran imágenes del proceso de instalación (ver Figura 4).

Figura 4
Instalación de cámaras para observación preparto.

Cuando terminó la instalación, se revisó con la vista que cada cámara estuviera bien colocada y funcionando correctamente. Se revisaron las grabaciones en vivo desde la laptop y el celular para asegurarse de que se viera claramente el comportamiento de la cerda, sin que nada tapara la imagen (K. Y. Ho, Tsai, and Kuo 2021). Las imágenes que vienen a continuación muestran cómo se hizo esta revisión (Figura 5).

Figura 5
Verificación de cámaras para registro del comportamiento.

FASE 2: Recolección de Datos

Etapa 2: Registro videográfico del comportamiento

Se grabaron videos del comportamiento de las cerdas antes del parto usando un sistema que filmaba todo el tiempo. Las cámaras, colocadas en lugares clave, grabaron al mismo tiempo desde diferentes ángulos del corral, lo que permitió ver el comportamiento real de las cerdas sin que nadie interfiriera (Liu et al. 2025; Yang et al. 2023). Este proceso ayudó a obtener videos reales y útiles para analizar después cómo se comportaban las cerdas.

FASE 3: Procesamiento de Imágenes

Etapa 3: Extracción de fragmentos conductuales relevantes

Cuando se terminó de grabar, se revisaron los videos para encontrar y recortar a mano las partes donde se veían posturas importantes del comportamiento antes del parto (Liu et al. 2022; de Paula et al. 2024). Se priorizó la claridad visual y la representatividad conductual. Estos fragmentos conformaron la base de datos para el análisis posterior.

Etapa 4: Clasificación por tipo de comportamiento

Los segmentos de video seleccionados fueron clasificados según cuatro categorías conductuales definidas: movimiento normal, inquietud, tumbada y parto (Z. Chen et al. 2023; Shao et al. 2021). Esta etapa permitió organizar el material en función de patrones específicos observados en las cerdas. A continuación, se muestran ejemplos representativos de cada categoría.

Comportamiento de movimiento normal: desplazamientos tranquilos y naturales dentro del corral, sin signos de estrés ni alteración (Figura 6).

Figura 6
Ejemplos de comportamiento normal en cerdas preparto.

Comportamiento de inquietud: actividad constante, cambios frecuentes de postura y desplazamientos repetitivos, indicativos de incomodidad o nerviosismo (Figura 7).

Figura 7
Ejemplos de comportamiento de inquietud en cerdas preparto.

Comportamiento de tumbada: postura de reposo prolongado en decúbito lateral o esternal, con mínima actividad física y sin intención de desplazamiento (Figura 8).

Figura 8
Ejemplos de comportamiento tumbada en cerdas preparto.

Comportamiento de parto: postura lateral acompañada de signos evidentes de contracción, esfuerzo abdominal, propias del momento del alumbramiento (Figura 9).

Figura 9
Ejemplos de comportamiento de parto en cerdas.

Etapa 5: Organización en carpetas por categoría

Finalmente, los segmentos de video fueron organizados en carpetas independientes, una por cada tipo de comportamiento observado: movimiento normal, inquietud, tumbada y parto. Esta estructura de almacenamiento permitió una gestión ordenada del material y facilitó su uso en procesos de análisis y clasificación automatizada. En la siguiente imagen se muestran las carpetas con los videos clasificados según los comportamientos preparto registrados durante el monitoreo (Figura 10).

Figura 10
Organización de registros de video según categoría conductual.

3.3.3. Objetivo específico 2: Implementar modelos híbridos de visión artificial basados en CNN para el monitoreo y la predicción de comportamientos preparto en cerdas.

Con el objetivo de detectar visualmente los comportamientos que anteceden al momento del parto en cerdas, se implementaron modelos híbridos de visión artificial que combinan redes neuronales convolucionales (CNN) como extractores de características visuales con un clasificador Random Forest. Este enfoque permite procesar imágenes extraídas de videos en entornos reales y predecir con precisión comportamientos preparto mediante técnicas no invasivas. El procedimiento técnico se estructuró en cinco fases consecutivas, que comprenden desde la conexión al dataset hasta la evaluación de los modelos (Figura 11).

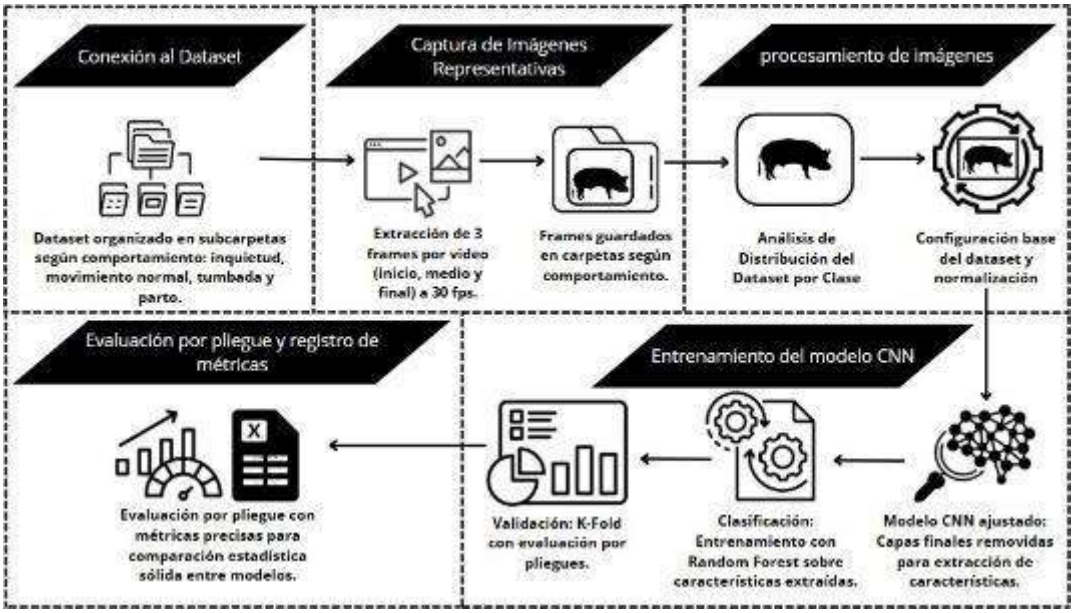

Figura 11
Diagrama de pasos del objetivo específico 2

FASE 1: Conexión al Dataset Etapa 1: Montaje del entorno

Se utilizó Google Colab como entorno de ejecución en la nube, estableciendo conexión con Google Drive para acceder al dataset de videos clasificados por comportamiento. En caso de no contar con los archivos localmente, se copió el conjunto de datos al

entorno de trabajo y se verificó su ubicación correcta en la ruta /content/dataset, garantizando así accesibilidad continua sin errores de carga. El diseño experimental del dataset se basó en la segmentación por clase: inquietud, movimiento normal, tumbada y parto, siguiendo criterios de observación conductual documentados en estudios previos (Z. Chen et al. 2023; Shao et al. 2021).

FASE 2: Captura de Imágenes Representativas

Etapa 2: Extracción de 3 frames por segundo (inicio, medio, final).

Se empleó la herramienta FFmpeg para extraer tres fotogramas por segundo de cada video, buscando capturar distintos momentos del comportamiento preparto. Esta frecuencia fue seleccionada por su equilibrio entre representación temporal y volumen de datos, alineándose con metodologías aplicadas en trabajos similares de monitoreo animal por visión computacional ((Bhatt et al. 2021).

Etapa 3: Almacenamiento de frames en carpetas según comportamiento.

Los fotogramas generados se almacenaron en carpetas específicas por clase, asegurando una organización sistemática de los datos. Se implementó un patrón de nombres estandarizado para facilitar la trazabilidad durante el entrenamiento. La codificación por clases se realizó mediante rutas base y procedimientos iterativos que verificaron o crearon automáticamente las carpetas necesarias, siguiendo buenas prácticas de procesamiento por lotes (Z. Chen et al. 2023).

FASE 3: Procesamiento de Imágenes

Etapa 4: Análisis de Distribución del Dataset por Clase

Se realizó un análisis exploratorio del número de imágenes por clase mediante un histograma con ejes rotulados, cuadrícula punteada y títulos descriptivos. Este paso permitió identificar desbalances en la distribución de clases, lo cual es crucial para evitar sesgos en los procesos de entrenamiento y validación (Kuldashboy et al. 2024).

Etapa 5: Configuración base del dataset y normalización

Se definieron los parámetros de procesamiento inicial, como el tamaño del lote, las dimensiones de entrada (224x224 píxeles) y el uso automático de GPU. Las imágenes fueron convertidas a tensores y normalizadas según los valores estándar de ImageNet. Esta etapa fue necesaria para adaptar adecuadamente las entradas a las arquitecturas preentrenadas seleccionadas (AlexNet, ResNet18 y VGG16), como lo recomiendan (Bhatt et al. 2021; Kuldashboy et al. 2024).

FASE 4: Entrenamiento del modelo CNN

Etapa 6: Ajuste del modelo CNN: capas finales modificadas para clasificación

Las redes CNN se emplearon exclusivamente como extractores de características. Para ello, se eliminaron sus capas finales encargadas de la clasificación, accediendo a representaciones profundas generadas por las capas convolucionales. En AlexNet y VGG16 se conservaron las capas totalmente conectadas previas a la salida, y en ResNet18 se extrajo el resultado del bloque de pooling global (Bhatt et al. 2021; Z. Chen et al. 2023).

Etapa 7: Clasificación: Entrenamiento con Random Forest sobre características extraídas

Una vez obtenidas las características visuales, se separó un subconjunto del dataset para entrenamiento. Se aplicó RandomizedSearchCV con validación cruzada estratificada para optimizar los hiperparámetros del clasificador Random Forest (número de árboles, profundidad máxima, muestras mínimas por nodo). Esta técnica permitió encontrar la mejor configuración con eficiencia computacional y alto rendimiento predictivo (Z. Chen et al. 2023).

Etapa 8: Validación: K-Fold con evaluación por pliegues

El modelo óptimo fue validado mediante validación cruzada con StratifiedKFold de cinco pliegues, técnica recomendada para mantener la proporción de clases en cada división del dataset. Se evaluaron métricas como Accuracy y F1-score macro, apropiadas para tareas de clasificación multiclase con datos desbalanceados (J. Chen et al. 2023).

FASE 5: Evaluación por pliegue y registro de métricas

Etapa 9: Evaluación cruzada y registro de métricas por pliegue

Las métricas obtenidas para cada pliegue, especialmente Accuracy y F1-score macro, se asociaron al modelo CNN correspondiente y fueron organizadas en tablas comparativas. Este registro permitió evaluar objetivamente el rendimiento de cada arquitectura (AlexNet+RF, VGG16+RF y ResNet18+RF), bajo condiciones experimentales homogéneas, facilitando así el análisis estadístico final (Wei et al. 2023).

3.3.4. Objetivo específico 3: Evaluar los modelos propuestos de visión artificial basados en CNN en la predicción del momento del parto en cerdas durante las horas previas al evento

Con el fin de evaluar el rendimiento de los modelos híbridos propuestos para la predicción del momento del parto en cerdas, se desarrolló un procedimiento experimental estructurado en seis fases y siete etapas que abarcó desde la preparación

del conjunto de prueba hasta el análisis estadístico de los resultados obtenidos. La secuencia completa del proceso seguido se resume metodológicamente en el diagrama correspondiente (Figura 12).

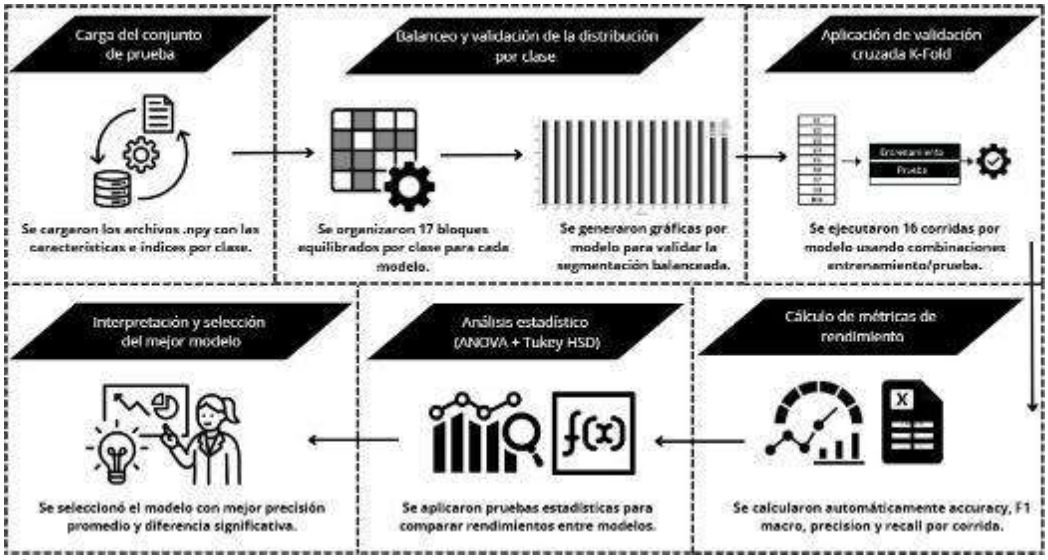

Figura 12
Diagrama de pasos del objetivo específico 3

Etapa 1: Importación de características y etiquetas por clase

Se cargaron los archivos. npy que contenían las características visuales previamente extraídas durante el entrenamiento con las arquitecturas AlexNet, VGG16 y ResNet18, junto con sus respectivas etiquetas de clase. Estos vectores representaban la información visual de los estados conductuales observados: inquietud, movimiento normal, tumbada y parto.

Fase 2: Balanceo y validación de la distribución por clase

Etapa 2: Generación de bloques balanceados por clase

El conjunto de prueba fue organizado en 17 bloques equilibrados, tomando como referencia la clase con menor frecuencia. Este diseño muestral permitió mantener la representatividad de las clases en cada partición durante la validación cruzada (Liu et al. 2025).

Etapa 3: Validación gráfica de la segmentación

Para verificar el correcto equilibrio entre clases, se generaron gráficos de barras que visualizaron la distribución de muestras por clase en cada modelo. Esta validación gráfica facilitó el control del balance de datos en los bloques de prueba.

Fase 3: Evaluación por validación cruzada

Etapa 4: Ejecución del esquema K-Fold

Se aplicó validación cruzada K-Fold con 17 particiones, ejecutando 16 combinaciones de entrenamiento y prueba por cada modelo. Esta técnica permitió evaluar con mayor robustez el rendimiento de los modelos y reducir la varianza estimada en los resultados (Liu et al. 2025).

Fase 4: Medición del rendimiento por corrida

Etapa 5: Cálculo automático de métricas por corrida

En cada corrida se calcularon automáticamente las métricas de desempeño: accuracy, F1-score macro, precisión y recall. Estas métricas fueron organizadas y almacenadas por modelo y corrida para su posterior análisis estadístico. El accuracy promedio fue considerado como métrica base para comparar los modelos, debido a su capacidad de representar el rendimiento general en problemas multiclase desbalanceados (Bhatt et al. 2021; Chen et al. 2021).

Fase 5: Análisis Estadístico

Etapa 6: Análisis de varianza y prueba de Tukey

Para comparar los rendimientos promedio de los modelos, se aplicó un análisis de varianza (ANOVA de un factor) y una prueba post hoc de Tukey HSD. El análisis se realizó con el software IBM SPSS Statistics, permitiendo identificar diferencias estadísticamente significativas entre arquitecturas evaluadas (Kuldashboy et al. 2024; Wei et al. 2023).

Fase 6: Interpretación y selección del mejor modelo

Etapa 7: Identificación del modelo con mayor precisión

Finalmente, se seleccionó como mejor modelo aquel que obtuvo el mayor valor promedio de precisión y presentó diferencias significativas respecto a los demás. Esta decisión se respaldó en los resultados del análisis estadístico, confirmando su superioridad para la predicción anticipada del parto en cerdas mediante visión artificial (Shao et al. 2021).

CAPÍTULO IV
RESULTADOS Y DISCUSIÓN

1.1. Resultado específico 1: Identificar patrones visuales previos al parto en cerdas mediante el análisis de imágenes capturadas

Se desarrolló un protocolo metodológico para identificar patrones visuales representativos previos al parto en cerdas, compuesto por una secuencia organizada de procedimientos. Inicialmente, se capturaron registros videográficos en condiciones reales de granja, recolectando 80 videos en corrales individuales del distrito de Sauce, San Martín, durante las últimas semanas de gestación. Las cámaras se instalaron en posiciones estratégicas (frontal, lateral y superior) con trípodes ajustables y montajes fijos, lo que permitió obtener imágenes continuas sin interrupciones ni interferencias humanas, asegurando la naturalidad del comportamiento observado (figura 13).

Figura 13
Configuración del sistema de grabación para monitoreo preparto.

Para garantizar la calidad de los registros, se implementó una estación de control conectada a una laptop para visualizar en tiempo real la transmisión. Esta supervisión permitió mantener la continuidad y nitidez del material. Además, se realizó una clasificación preliminar de los videos según el comportamiento predominante, logrando una distribución balanceada por categoría conductual (ver figura 14).

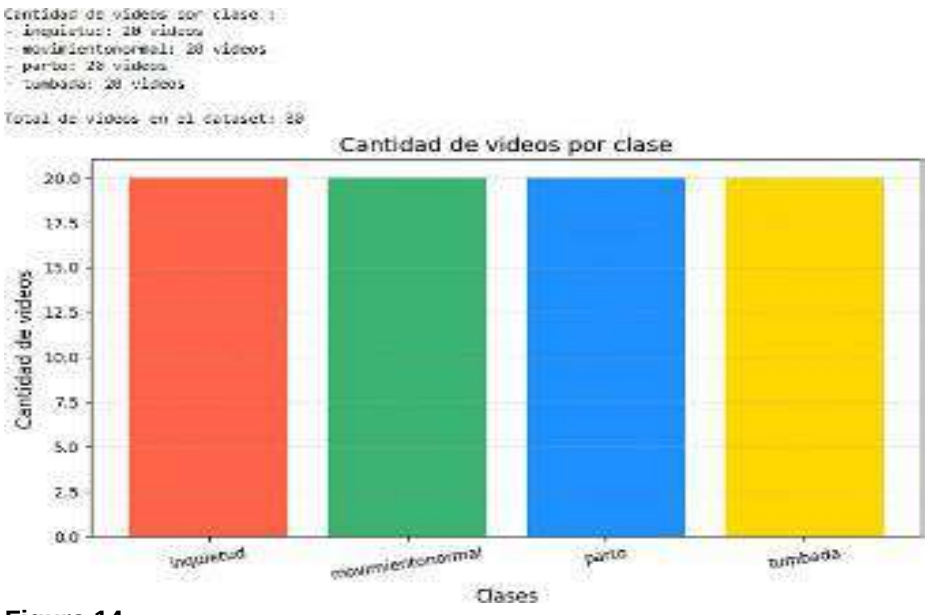

Figura 14
Cantidad de videos por categoría de comportamiento.

Los registros videográficos originales generados durante este proceso se encuentran disponibles en el siguiente enlace:

https://drive.google.com/drive/folders/1j72ioDL6MzV01nkzOO0kAyga7Ve71UI_?usp=d riv e_link

Posteriormente, se extrajeron más de 24,000 imágenes mediante herramientas de visión artificial como OpenCV y FFmpeg, a una tasa de tres fotogramas por segundo. Para ello, se implementó un script en Python que automatizó tanto la extracción como el procesamiento de los fotogramas, organizándolos por clase en carpetas independientes. A continuación, se muestra una de las funciones empleadas para realizar la extracción:

```
def extraer_frames_ffmpeg(ruta_video, carpeta_clase, nombre_base, fps=3):
    os.makedirs(carpeta_clase, exist_ok=True)
    patron_salida = os.path.join(carpeta_clase, f"{nombre_base}_%04d.jpg")
    comando = [
        "ffmpeg", "-i", ruta_video, "-vf", f"fps={fps}",
        patron_salida, "-hide_banner", "-loglevel", "error"
    ]
    subprocess.run(comando)
```

El código permitió generar imágenes etiquetadas por comportamiento, facilitando la segmentación y clasificación según las categorías definidas: movimiento normal, inquietud, tumbada y parto (figura 15).

Figura 15
Ejecución del programa de extracción para la clase "tumbada".

Las imágenes fueron organizadas digitalmente en carpetas separadas por clase, con aproximadamente 6,000 imágenes por categoría, garantizando equilibrio proporcional. En la figura siguiente se muestran ejemplos representativos de cada clase definida (ver figura 16).

Figura 16
Ejemplos por categoría: movimiento normal, inquietud, tumbada, parto.

Se realizó una validación visual cualitativa del conjunto, identificando patrones claros: desplazamiento dentro del corral (movimiento normal), cambios posturales frecuentes (inquietud), reposo lateral prolongado (tumbada), y presencia de contracciones con expulsión del lechón (parto). La distribución final del dataset balanceado se presenta (Figura 17).

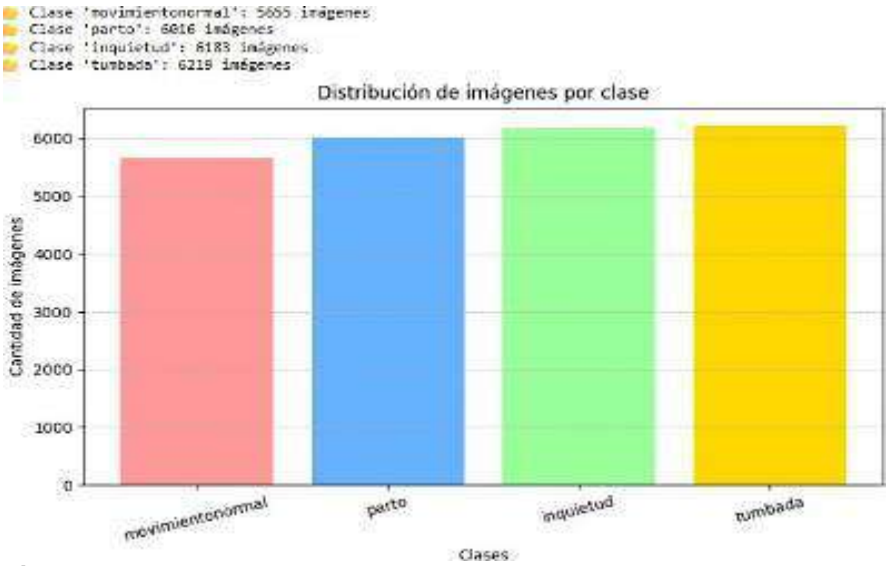

Figura 17
Distribución de imágenes por categoría en el dataset balanceado.

Este protocolo permitió obtener un conjunto de datos balanceado, visualmente válido y técnicamente robusto, apto para el entrenamiento de modelos de clasificación. La configuración estable del sistema de cámaras y la captura interrumpida coinciden con lo propuesto por (Bonneau et al. 2021; Shao et al. 2021), quienes destacan que la continuidad y calidad de los registros son esenciales para detectar comportamientos clave en fases preparto.

Usar programas automáticos como OpenCV y FFmpeg ayudó bastante a evitar errores humanos y también hizo que el trabajo sea más rápido, tal como señalan (Kuldashboy et al. 2024). Además, mantener un equilibrio entre clases es esencial para evitar sesgos en el entrenamiento, como respaldan (Riekert et al. 2020). La validación visual confirmó la existencia de señales distinguibles por clase, reforzando lo planteado por (Shao et al. 2021; Yin et al. 2024) sobre la eficacia de modelos basados en CNN para reconocer comportamientos preparto.

En conjunto, los resultados confirman que el protocolo implementado permitió construir una base de datos confiable, estandarizada y adecuada para tareas de clasificación supervisada, representando un aporte valioso para el monitoreo animal automatizado mediante visión por computadora.

1.2. Resultado específico 2: Implementar modelos híbridos de visión artificial basados en CNN para el monitoreo y la predicción de comportamientos preparto en cerdas.

En esta parte se explica, de manera sencilla, cómo se trabajó con tres modelos que combinan redes neuronales con un clasificador llamado Random Forest. Los modelos fueron los siguientes:

a) Modelo A: usa AlexNet con Random Forest.
b) Modelo B: usa ResNet18 con Random Forest.
c) Modelo C: usa VGG16 con Random Forest.

Como los tres modelos siguen casi los mismos pasos, se cuenta todo junto: qué se hizo, qué código se usó y qué resultados se obtuvieron.

1.2.1. Configuración del entorno y carga del dataset

Primero se colocó la ruta del dataset, el tamaño de las imágenes, el número de imágenes por grupo (batch), si se iba a usar GPU, y cuántas divisiones (pliegues) se harían para validar. Luego se aplicaron cambios básicos a las imágenes y se cargaron en un DataLoader. Por último, se identificaron las clases automáticamente.

```
# === CONFIGURACIÓN ===
DATA_DIR = '/content/frames'
BATCH_SIZE = 32
IMG_SIZE = 224
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("✅ Dispositivo en uso:", DEVICE)
# === TRANSFORMACIONES ===
transform = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

1.2.2. Extracción de características con redes CNN

Se usaron redes CNN preentrenadas para extraer características de cada imagen. Estas se guardaron en forma de vectores y sirvieron como entrada al modelo Random Forest. Este procedimiento es común en tareas de clasificación por visión computacional en animales, ya que reduce el procesamiento sin perder precisión (Bhatt et al. 2021; Z. Chen et al. 2023).

1.2.2.1. Modelo A – AlexNet

Se eligió AlexNet porque es una red sencilla y rápida. Se le quitó la última parte (la que clasifica) para solo usar los vectores de salida.

```
alexnet = models.alexnet(weights=models.AlexNet_Weights.DEFAULT).to(DEVICE)
alexnet.classifier =
torch.nn.Sequential(*list(alexnet.classifier.children())[:-1])
```

Como muestra la Tabla 3, AlexNet generó vectores de 4096 valores. Este enfoque ha sido usado en estudios similares por su buena eficiencia computacional (Bhatt et al. 2021).

Tabla 3
Ejemplo de vectores de características extraídas con AlexNet por clase

Modelo usado	Clase	Dimensión del vector	Promedio del vector	Máximo	Mínimo
AlexNet	Inquietud	40996	0.2626	117.802	0.0
	Movimiento normal	4096	0.3040	144.925	0.0
	Parto	4096	0.3569	121.580	0.0
	Tumbada	4096	0.2541	126.005	0.0

1.2.2.2. Modelo B – ResNet18

Esta red es más profunda y puede aprender cosas más complejas. También se le quitó la última parte para sacar los vectores.

```
resnet18 = models.resnet18(weights=models.ResNet18_Weights.DEFAULT)
resnet18 = torch.nn.Sequential(*list(resnet18.children())[:-1])
```

Como se muestra en la Tabla 4, los vectores variaron según la clase. ResNet18 ha sido validado por su buen manejo de datos variados (Z. Chen et al. 2023; Kuldashboy et al. 2024).

Tabla 4
Ejemplo de vectores de características extraídas con ResNet18 por clase

Modelo usado	Clase	Dimensión del vector	Promedio del vector	Máximo	Mínimo
RestNet18	Inquietud	512	10.220	75.722	0.0
	Movimiento normal	512	10.740	52.433	0.0
	Parto	512	10.116	55.380	0.0
	Tumbada	512	0.9072	80.402	0.0

1.2.2.3. Modelo C – VGG16

Este modelo es conocido por ser bueno para imágenes con muchos detalles. Se le quitó la parte final y se usó igual que los otros.

```
vgg16 = models.vgg16(weights=models.VGG16_Weights.DEFAULT)
vgg16.classifier = torch.nn.Sequential(*list(vgg16.classifier.children())[:-1])
```

Como se muestra en la Tabla 5, los vectores de VGG16 también variaron por clase. Este modelo ha mostrado buen rendimiento en análisis de comportamiento animal (Kuldashboy et al. 2024).

Tabla 5
Ejemplo de vectores de características extraídas con VGG16 por clase

Modelo usado	Clase	Dimensión del vector	Promedio del vector	Máximo	Mínimo
VGG16	inquietud	4096	0.2191	42.058	0
	Movimiento normal	4096	0.2306	53.259	0
	parto	4096	0.2266	36.539	0
	tumbada	4096	0.2018	43.671	0

1.2.3. Búsqueda automática de los mejores parámetros

Para optimizar el modelo, se usó RandomizedSearchCV, que permite ajustar parámetros como número de árboles, profundidad y división de nodos. Este método ha sido útil en investigaciones similares para mejorar el rendimiento sin pruebas exhaustivas (J. Chen et al. 2023; Z. Chen et al. 2023).

```
param_dist = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt', 'log2']
}
```

La Tabla 6 muestra los mejores parámetros encontrados para cada modelo.

Tabla 6
Mejores hiperparámetros encontrados por modelo con RandomizedSearchCV

Modelo	n_estimators	max_depth	max_features	min_samples_split	min_samples_leaf
AlexNet	300	30	sqrt	2	1
ResNet18	300	30	sqrt	2	1
VGG16	300	30	sqrt	2	1

Este procedimiento permitió ajustar cada modelo a su configuración óptima antes de la fase de validación.

1.2.4. Validación cruzada con métricas globales

Finalmente, se aplicó validación cruzada con 5 pliegues, manteniendo el balance entre clases. Las métricas usadas fueron Accuracy y F1-score macro, recomendadas para tareas multiclase con datos desbalanceados (J. Chen et al. 2023; Wei et al. 2023).

```
final model = RandomForestClassifier(**best params, random state=SEED, n jobs=-1)
kfold = StratifiedKFold(n_splits=N_SPLITS, shuffle=True, random_state=SEED)
acc_scores, f1_scores = [], []
```

```
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_kfold, y_kfold)):
    X_tr, X_val = X_kfold[train_idx], X_kfold[val_idx]
    y_tr, y_val = y_kfold[train_idx], y_kfold[val_idx]
    final_model.fit(X_tr, y_tr)
    preds = final_model.predict(X_val)
    acc = accuracy_score(y_val, preds)
    f1 = f1_score(y_val, preds, average='macro')
    acc_scores.append(acc)
    f1_scores.append(f1)
accuracy = np.mean(acc_scores)
f1_macro = np.mean(f1_scores)
```

1.2.4.1. Modelo A – AlexNet + Random Forest

Este modelo fue evaluado usando los mejores parámetros encontrados anteriormente. En la Tabla 7 se resumen los resultados obtenidos en los cinco pliegues (folds) de validación cruzada.

Tabla 7
Resultados por fold – Modelo A

Fold	Accuracy	F1 Macro
1	0.9964	0.9963
2	0.9943	0.9942
3	0.9958	0.9958
4	0.9956	0.9955
5	0.9956	0.9955

En promedio, se obtuvo una Accuracy de 0.9955 y un F1 Macro también de 0.9955, lo que muestra un rendimiento muy alto y estable. Estos resultados se pueden observar visualmente en la matriz de confusión (figura 18) y en las curvas ROC por clase (figura 19).

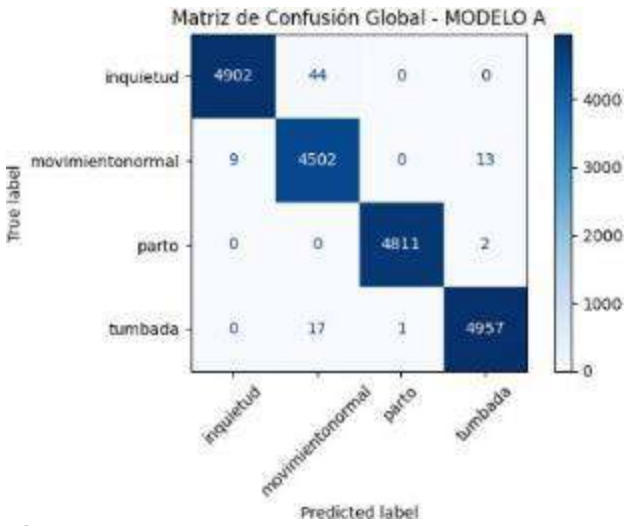

Figura 18
Matriz de confusión global del Modelo A

Interpretación: La matriz de confusión del Modelo A evidenció un buen desempeño general. La clase “inquietud” registró 4902 aciertos, con 44 errores hacia “movimiento normal”. Esta última logró 4502 aciertos, presentando errores menores hacia “inquietud” (9) y “tumbada” (13). La clase “parto” fue clasificada correctamente en 4811 casos, con solo 2 errores, mientras que “tumbada” alcanzó 4957 aciertos. Aunque el rendimiento fue consistente, se observaron leves confusiones entre clases con patrones similares, una situación común en arquitecturas como AlexNet, donde se prioriza la simplicidad y la velocidad de procesamiento sobre una mayor profundidad. Este tipo de comportamiento también ha sido documentado en investigaciones previas, como las de (Bhatt et al. 2021; Z. Chen et al. 2023), en aplicaciones de visión artificial para comportamiento animal.

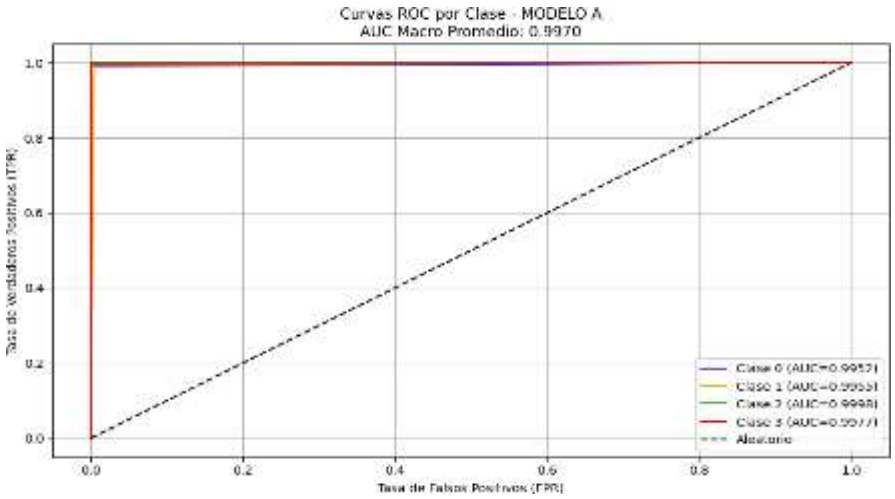

Figura 19
Curvas ROC por clase para el Modelo A

Interpretación: La figura presenta las curvas ROC generadas por clase para el Modelo A, con un AUC macro promedio de 0.9970. Se observa que todas las clases superaron un valor de AUC de 0.995, siendo la clase “parto” la que alcanzó el valor más alto (0.9998). Esta visualización permite analizar cómo el modelo distingue entre las diferentes clases durante el entrenamiento, brindando una referencia útil para evaluaciones futuras.

1.2.4.2. Modelo B: ResNet18 + Random Forest

Este modelo también fue evaluado usando los mejores hiperparámetros obtenidos anteriormente. La **Tabla 8** muestra los resultados en cada uno de los cinco pliegues de validación cruzada.

Tabla 8
Resultados por fold – Modelo B

Fold	Accuracy	F1 Macro
1	0.9948	0.9947
2	0.9938	0.9937
3	0.9945	0.9945
4	0.9948	0.9947
5	0.9943	0.9942

En promedio, se obtuvo una Accuracy de 0.9944 y un F1 Macro de 0.9944, lo que demuestra que el modelo tiene un rendimiento alto y bastante parejo. Los resultados se pueden ver en la matriz de confusión (ver Figura 20) y en las curvas ROC por clase (ver Figura 21).

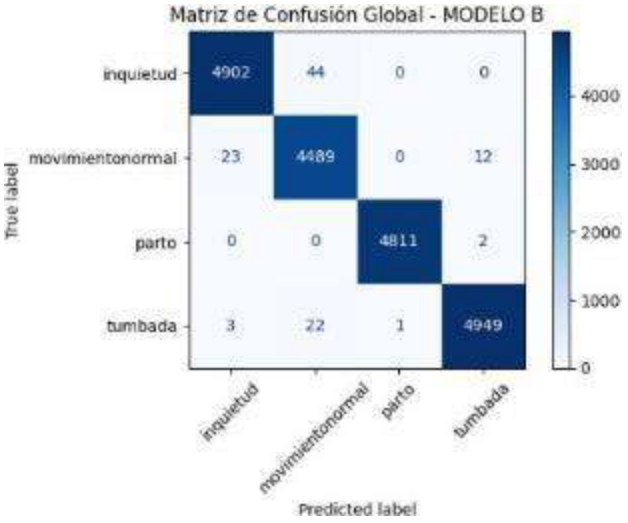

Figura 20
Matriz de confusión global del Modelo B

Interpretación: La matriz de confusión del Modelo B mostró un desempeño sólido. La clase “inquietud” alcanzó 4902 aciertos, con 44 errores hacia “movimiento normal”. Esta última obtuvo 4489 aciertos, aunque presentó algunas confusiones con “inquietud” (23) y “tumbada” (12). La clase “parto” fue correctamente clasificada en 4811 casos, con solo 2 errores, mientras que “tumbada” logró 4949 aciertos. Si bien los resultados generales fueron positivos, el modelo evidenció cierta dificultad para diferenciar clases con comportamientos similares, como “movimiento normal” e “inquietud”. Este tipo de confusiones es frecuente en arquitecturas como ResNet18, especialmente cuando los patrones visuales son sutiles, como también ha sido reportado por (Z. Chen et al. 2023) en tareas de clasificación de comportamiento animal.

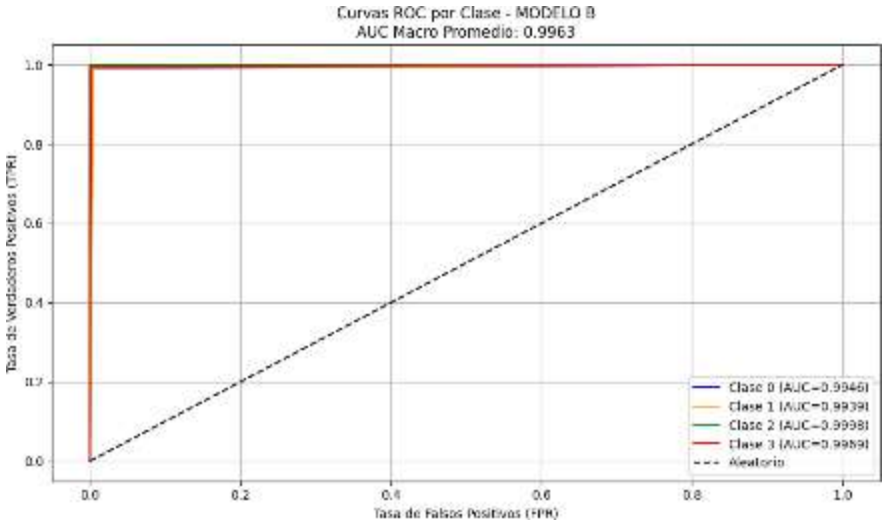

Figura 21
Curvas ROC por clase para el Modelo B

Interpretación: La figura presenta las curvas ROC por clase correspondientes al Modelo B, con un AUC macro promedio de 0.9963. Todas las clases alcanzaron valores de AUC superiores a 0.993, destacando la clase “parto” con un valor de 0.9998. Esta visualización permite analizar la capacidad del modelo para distinguir entre clases durante el entrenamiento, y aporta una base informativa para estudios posteriores sobre su desempeño.

1.2.4.3. Modelo C: VGG16 + Random Forest

Este modelo fue evaluado usando los hiperparámetros óptimos que se encontraron antes. La Tabla 9 muestra los resultados obtenidos en cada uno de los cinco pliegues de validación cruzada.

Tabla 9
Resultados por fold – Modelo C

Fold	Accuracy	F1 Macro
1	0.9938	0.9937
2	0.9922	0.9921
3	0.9948	0.9947
4	0.9938	0.9937
5	0.9943	0.9942

En promedio, este modelo logró una Accuracy de 0.9938 y un F1 Macro de 0.9937, lo que indica que tuvo un rendimiento fuerte, parejo y confiable. Los resultados completos pueden verse en la matriz de confusión (Figura 22) y en las curvas ROC (Figura 23).

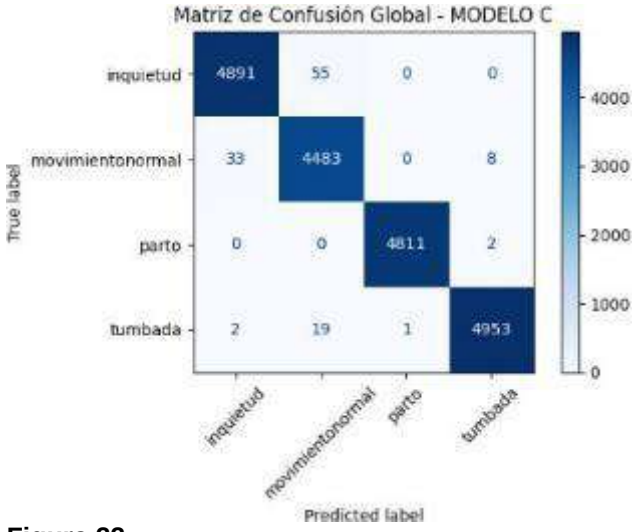

Figura 22
Matriz de confusión global del Modelo C

Interpretación: La matriz de confusión del Modelo C evidenció un desempeño preciso. La clase “inquietud” fue reconocida correctamente en 4891 casos, con 55 errores hacia “movimiento normal”. Esta última alcanzó 4483 aciertos, con leves confusiones hacia “inquietud” (33) y “tumbada” (8). La clase “parto” logró 4811 aciertos, con solo 2 errores, mientras que “tumbada” obtuvo 4953 predicciones correctas. Si bien los resultados fueron consistentes, se observaron ligeras confusiones entre clases con patrones similares, un fenómeno esperado en arquitecturas como VGG16, cuyo diseño favorece la estabilidad estructural de las predicciones sobre una mayor profundidad de procesamiento. Este tipo de comportamiento también ha sido reportado en estudios como los de (Bhatt et al. 2021; Kuldashboy et al. 2024), al evaluar redes CNN en tareas de clasificación de secuencias complejas.

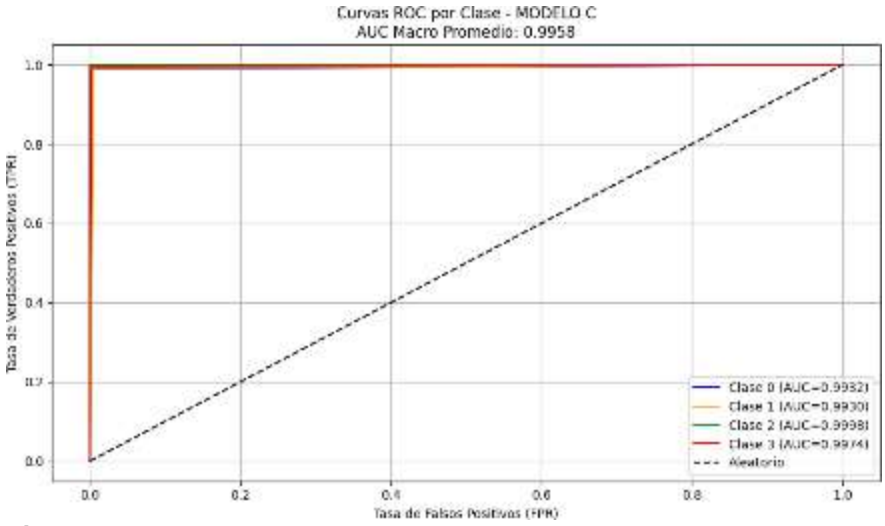

Figura 23
Curvas ROC por clase para el Modelo C

Interpretación: La figura muestra las curvas ROC por clase correspondientes al Modelo C, con un AUC macro promedio de 0.9958. Las clases “inquietud”, “movimiento normal” y “tumbada” registraron valores superiores a 0.993, mientras que la clase “parto” alcanzó un valor de 0.9998. Esta representación permite examinar la capacidad del modelo para distinguir entre clases durante el entrenamiento, aportando información relevante para el análisis de su desempeño.

1.2.5. Evaluación Comparativa de Métricas Generales y por Clase

Durante el proceso de entrenamiento, se obtuvieron métricas globales para los tres modelos evaluados. El Modelo A alcanzó un Accuracy y F1 Macro de 0.9955, mientras que el Modelo B obtuvo 0.9944 en ambas métricas. El Modelo C registró un Accuracy de 0.9938 y un F1 Macro de 0.9937. En cuanto al ROC AUC, los tres modelos presentaron valores altos y similares (0.9998), lo que indica una buena capacidad de discriminación. Estos resultados se presentan en la Tabla 10.

Tabla 10
Cuadro comparativo de métricas globales por modelo

Modelo	Accuracy	F1 Macro	Precision Macro	Recall Macro	ROC AUC (OVR)
Modelo_A	0.9955	0.9955	0.9954	0.9956	0.9998
Modelo_B	0.9944	0.9944	0.9943	0.9944	0.9998
Modelo_C	0.9938	0.9937	0.9936	0.9937	0.9998

Los F1-scores por clase muestran un alto rendimiento en las cuatro categorías. La clase “parto” alcanzó 0.9997 en los tres modelos. En “movimiento normal”, el valor más bajo fue 0.9873 (Modelo C), mientras que en “inquietud” y “tumbada” los puntajes se mantuvieron por encima de 0.990. Los resultados se presentan en la Tabla 11.

Tabla 11
Comparación de F1-score por clase entre modelos

Clase	Modelo_A	Modelo_B	Modelo_C
Inquietud	0.9946	0.9929	0.9909
Movimiento normal	0.9909	0.9889	0.9873
Parto	0.9997	0.9997	0.9997
Tumbada	0.9967	0.996	0.9968

La comparación de las métricas globales obtenidas durante el entrenamiento de los tres modelos —Accuracy, F1 Macro, Precision Macro, Recall Macro y ROC AUC— evidencia que todos alcanzaron valores elevados y relativamente similares. Si bien se presentan pequeñas diferencias entre ellos, estas no son lo suficientemente marcadas como para

indicar una ventaja concluyente. Esta visualización permite analizar de forma clara el comportamiento general de los modelos en cada métrica evaluada (ver Figura 24).

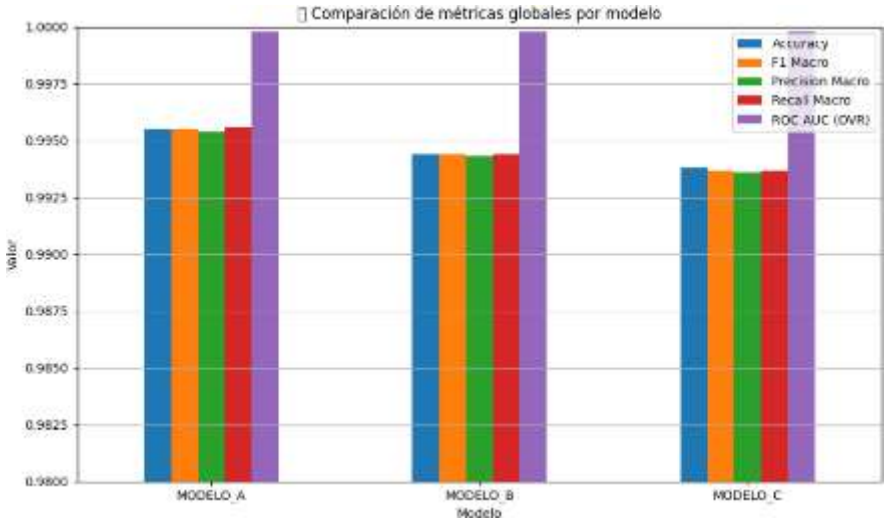

Figura 24
Comparación de métricas globales por modelo

El F1-score por clase muestra un rendimiento elevado y consistente entre los tres modelos para las clases “inquietud”, “movimiento normal”, “parto” y “tumbada” (ver Figura 25). En la clase “parto”, los tres modelos obtuvieron el mismo valor (0.9997), mientras que en las demás clases se observan ligeras variaciones. La clase “movimiento normal” presentó los puntajes más bajos en comparación, lo que sugiere una mayor complejidad relativa en su clasificación durante el entrenamiento.

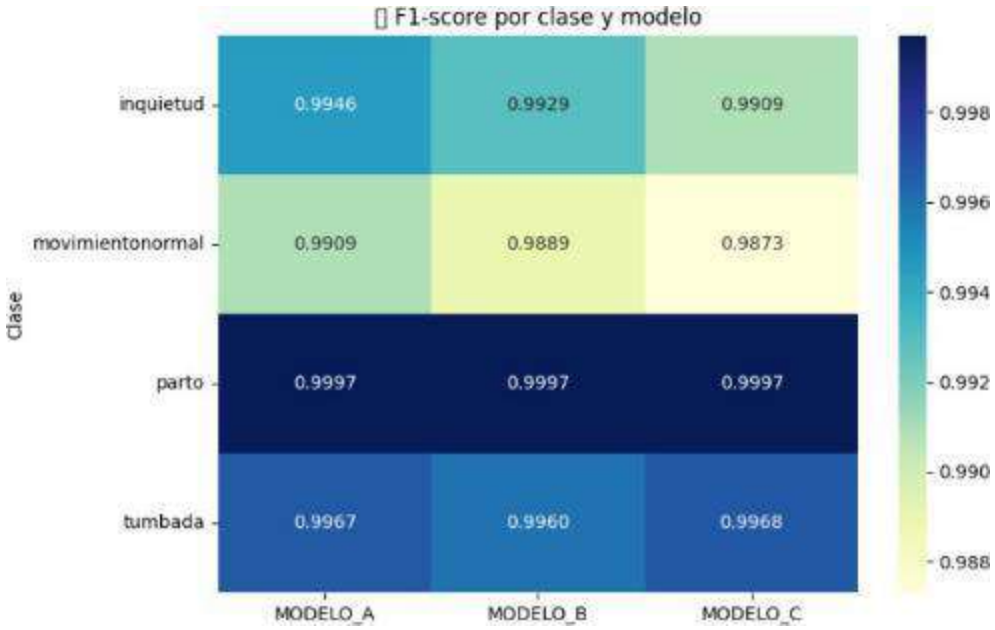

Figura 25
Comparación visual del F1-score por clase y modelo

En conjunto, los resultados permiten analizar el comportamiento de los modelos híbridos basados en CNN combinados con Random Forest en tareas de monitoreo automatizado

de animales en tiempo real. El uso de métricas como el AUC y el F1-score por clase facilitó la identificación de diferencias entre arquitecturas. Este comportamiento ha sido igualmente documentado por (Z. Chen et al. 2023; Wei et al. 2023), quienes coinciden en que dichas métricas son clave para evaluar modelos de visión computacional aplicados al ámbito ganadero.

1.3. Resultado específico 3: Evaluar los modelos propuestos de visión artificial basados en CNN en la predicción del momento del parto en cerdas durante las horas previas al evento.

Para evaluar el rendimiento comparativo de los modelos híbridos AlexNet+RF, ResNet18+RF y VGG16+RF, se aplicó un procedimiento de validación cruzada K-Fold utilizando 17 bloques balanceados. Este enfoque permitió medir de forma consistente la precisión, F1-score y tiempo de procesamiento de cada modelo, bajo las mismas condiciones.

Primero, se cargó el conjunto de prueba correspondiente a cada modelo, utilizando los archivos .npy que contenían las características extraídas y las etiquetas reales de las imágenes procesadas. Para cada caso, se emplearon archivos con la siguiente estructura:

```
X_test = np.load(f'/content/X_test_{modelo}.npy')
y_test = np.load(f'/content/y_test_{modelo}.npy')
```

Se usó la variable modelo para cargar los archivos .npy de cada caso: 'A' para AlexNet+RF, 'B' para ResNet18+RF y 'C' para VGG16+RF.

Luego, se configuraron los parámetros básicos para la división. Se estableció el número de bloques en 17, y se detectaron automáticamente las clases presentes en el conjunto de prueba.

```
n_bloques = 17
clases = np.unique(y_test)
```

Posteriormente, se contó cuántas muestras existían por clase para poder balancear correctamente la cantidad de datos a repartir entre los bloques.

```
conteo = {int(c): np.sum(y_test == c) for c in clases}
```

Con este valor, se seleccionaron los índices de las muestras para cada clase, se mezclaron aleatoriamente y se dividieron en 17 subgrupos iguales.

```
idx_por_clase = {}
     for c in clases:
      idx = np.where(y_test == c)[0][:por_clase * n_bloques]
```

```
        np.random.shuffle(idx)

        idx_por_clase[c] = np.array_split(idx, n_bloques)
```

1.3.1. Visualización de la distribución por clase en los bloques

Concluida la partición del conjunto de prueba, se construyeron 17 bloques equilibrados por clase para cada uno de los tres modelos híbridos: AlexNet+RF (Modelo A), ResNet18+RF (Modelo B) y VGG16+RF (Modelo C). Para validar visualmente la correcta segmentación, se generaron tres gráficas de barras, una por cada modelo, que representan la distribución de las cuatro clases en los bloques (Figura 26), (Figura 27) y (Figura 28).

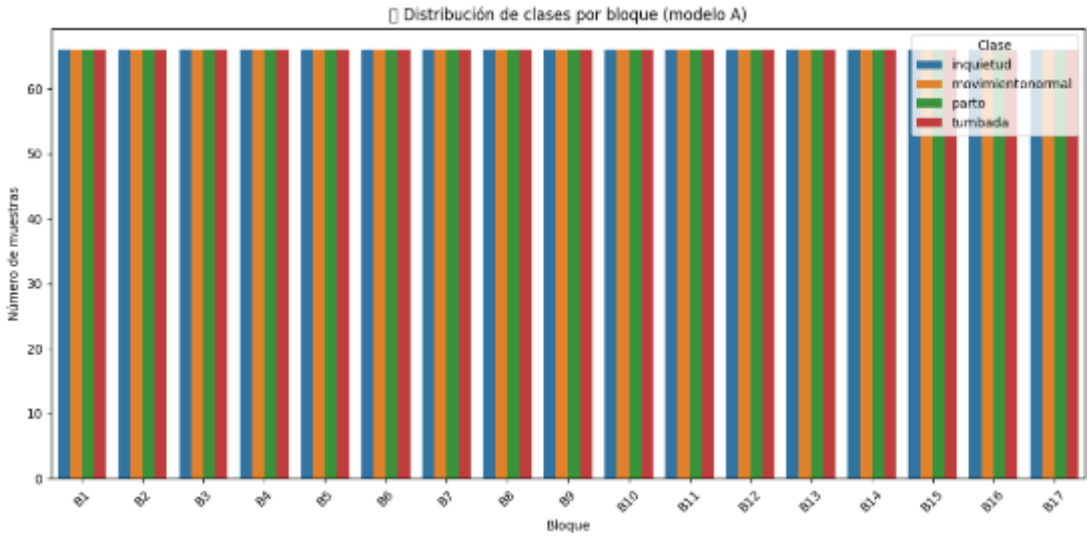

Figura 26
Distribución por clase en los bloques del Modelo A (AlexNet+RF).

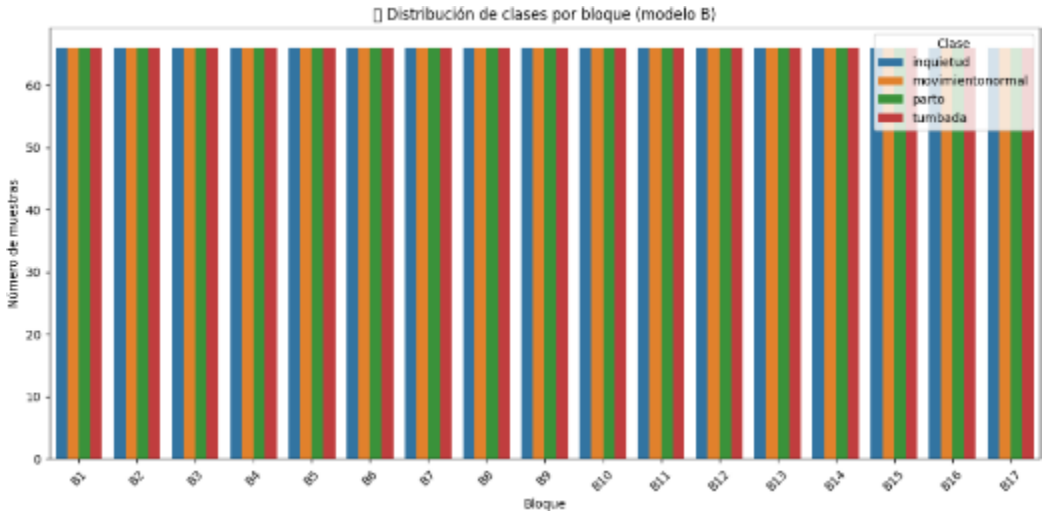

Figura 27
Distribución por clase en los bloques del Modelo B (ResNet18+RF).

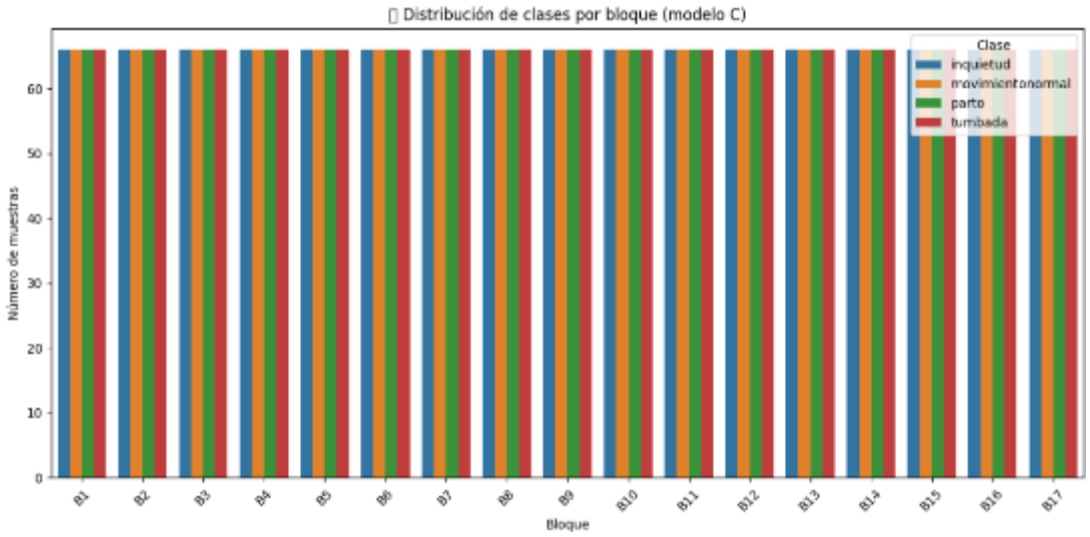

Figura 28
Distribución por clase en los bloques del Modelo C (VGG16+RF).

Como se observa en las Figuras 26, 27 y 28, la distribución de las clases inquietud, movimiento normal, parto y tumbada se mantuvo equilibrada en cada uno de los 17 bloques generados por modelo. Este resultado confirma que el procedimiento de balanceo fue ejecutado correctamente en los tres casos, asegurando condiciones homogéneas para la posterior aplicación de la validación cruzada K-Fold, una técnica recomendada por (Liu et al. 2025) para reducir la varianza en la estimación del rendimiento de modelos en contextos biomédicos y de producción animal.

1.3.2. Evaluación cruzada y obtención de métricas por corrida

Después de generar los bloques balanceados para cada modelo, se aplicó una validación cruzada con 16 corridas, utilizando en cada iteración 16 bloques para entrenamiento y uno para validación. Este procedimiento se repitió para los modelos híbridos AlexNet+RF (Modelo A), ResNet18+RF (Modelo B) y VGG16+RF (Modelo C), empleando los archivos .npy previamente generados.

La evaluación se realizó cargando el clasificador ya entrenado en cada caso, combinando dinámicamente los bloques y generando predicciones con el modelo correspondiente. A continuación, se presenta el fragmento de código que resume esta etapa:

```
for i in range(n_bloques - 1):
    X_train = np.concatenate([X_bloques[j] for j in range(n_bloques) if j != i])
    y_train = np.concatenate([y_bloques[j] for j in range(n_bloques) if j != i])

    modelo = joblib.load(modelo_path)
    y_pred = modelo.predict(X_train)

reporte = classification_report(y_train, y_pred, output_dict=True)
```

En cada corrida se calcularon automáticamente las métricas de rendimiento: accuracy, F1-score macro, precisión macro, recall macro, y F1-score por clase. Esta información fue organizada en un DataFrame de Pandas y exportada a formato Excel, permitiendo su posterior análisis estadístico en SPSS. Estas métricas, ampliamente utilizadas en visión artificial aplicada a ganadería, han sido destacadas por (Chen et al. 2021) como claves para evaluar el comportamiento de cerdas en tareas de predicción automatizada.

```
df_resultados = pd.DataFrame(resultados)
df_resultados.to_excel(nombre_archivo, index=False)
```

Los modelos AlexNet+RF (Modelo A), ResNet18+RF (Modelo B) y VGG16+RF (Modelo C) presentaron valores de accuracy elevados y consistentes a lo largo de las 16 corridas, bajo condiciones balanceadas de evaluación. Aunque se calcularon varias métricas, en la Tabla 12 se muestran únicamente los valores de accuracy de cada modelo, para centrar el análisis comparativo en esta métrica principal. En todos los casos, los resultados superaron el 0.992, evidenciando un comportamiento estable y confiable. Este patrón concuerda con lo reportado por (Bhatt et al. 2021), quienes destacan que las CNN combinadas con clasificadores robustos como Random Forest pueden alcanzar alta precisión en tareas complejas de clasificación de imágenes.

Tabla 12
Accuracy por corrida de los modelos AlexNet+RF, ResNet18+RF y VGG16+RF

Modelo A		**Modelo B**		**Modelo C**	
Corrida	**Accuracy**	**Corrida**	**Accuracy**	**Corrida**	**Accuracy**
1	0.9948	1	0.9931	1	0.9938
2	0.9953	2	0.9931	2	0.9941
3	0.9950	3	0.9931	3	0.9943
4	0.9948	4	0.9934	4	0.9938
5	0.9950	5	0.9931	5	0.9943
6	0.9953	6	0.9929	6	0.9941
7	0.9950	7	0.9927	7	0.9953
8	0.9955	8	0.9929	8	0.9946
9	0.9950	9	0.9931	9	0.9938
10	0.9955	10	0.9931	10	0.9938
11	0.9953	11	0.9927	11	0.9943
12	0.9948	12	0.9934	12	0.9941
13	0.9950	13	0.9931	13	0.9938
14	0.9953	14	0.9929	14	0.9941
15	0.9948	15	0.9929	15	0.9948
16	0.9955	16	0.9936	16	0.9941

Análisis estadístico:

Después de ver los resultados, se hizo una prueba ANOVA con su respectiva prueba de hipótesis. Primero, se propuso la hipótesis nula:

$$H_0: \mu_1 = \mu_2 = \mu_3$$

En este caso, μ_1, μ_2 y μ_3 son los valores promedio de precisión (accuracy) de los modelos híbridos AlexNet+RF, ResNet18+RF y VGG16+RF, en ese orden. La idea de esta hipótesis es que no hay diferencias importantes entre los resultados que dio cada modelo, es decir, todos tienen un rendimiento muy parecido.

Tabla 13
Prueba de homogeneidad de varianzas para el Accuracy

		Estadístico de Levene	gl1	gl2	Sig.
Accuracy	Se basa en la media	1,353	2	45	0,269
	Se basa en la mediana	0,618	2	45	0,544
	Se basa en la mediana y con gl ajustado	0,618	2	33,693	0,545
	Se basa en la media recortada	0,992	2	45	0,379

En primer lugar, se aplicó la prueba de Levene (Tabla 13) para comprobar si la significancia obtenida es mayor que 0.05. En este caso, el valor de significancia (Sig. = 0.269), basado en la media, es mayor que 0.05, lo que nos permite aceptar la hipótesis nula de homogeneidad de varianzas. Esto indica que las varianzas entre los grupos son parecidas y se cumple el supuesto necesario para continuar con el análisis. Por lo tanto, se puede seguir con la siguiente etapa del estudio bajo condiciones normales.

Tabla 14
ANOVA para el accuracy entre los modelos híbridos

	Suma de cuadrados	gl	Media cuadrática	F	Sig.
Entre grupos	0	2	0	175,065	0
Dentro de grupos	0	45	0		
Total	0	47			

El análisis de varianza (ANOVA) de un factor (Tabla 14) mostró diferencias estadísticamente significativas entre los modelos híbridos (F (2, 45) = 175.065, Sig. < (p = 0.05)), lo cual confirma que al menos uno de los modelos evaluados presenta un comportamiento distinto en términos de precisión. En mérito a que la significancia es

menor que p, procedimos a realizar la prueba Post hoc. Resultados similares fueron reportados por (Kuldashboy et al. 2024), al comparar modelos CNN aplicando técnicas como normalización por lotes y distilación colaborativa.

Tabla 15
Comparaciones múltiples mediante la prueba de Tukey HSD (Accuracy).

(I) Identificador	(J) Identificador	Diferencias de medias (I-J)	Desv. Error	Sig.	Intervalo de confianza al 95%	
					Límite inferior	Limite superior
Modelo A	Modelo B	,002041903312500*	0,0001093	0	0,001777	0,0023068
	Modelo C	,000917376937500*	0,0001093	0	0,0006524	0,0011823
Modelo B	Modelo A	-,002041903312500*	0,0001093	0	-0.0023068	-0.001777
	Modelo C	-,001124526375000*	0,0001093	0	-0.0013895	-0.0008596
Modelo C	Modelo A	-,000917376937500*	0,0001093	0	-0.0011823	-0.0006524
	Modelo B	-,001124526375000*	0,0001093	0	0.0008596	0.0013895

La diferencia de medias es significativa en el nivel 0.05.

La prueba post hoc de Tukey (Tabla 15) reveló que todas las comparaciones por pares fueron significativas (Sig. < 0.001). Específicamente, el Modelo A superó significativamente al Modelo B (diferencia = 0.0020) y al Modelo C (diferencia = 0.0009), mientras que el Modelo C también tuvo mejores resultados que el Modelo B (diferencia = 0.0011). Los intervalos de confianza al 95% en todas las comparaciones no incluyeron el valor cero, lo que refuerza la validez estadística de estas diferencias. Esta jerarquía ha sido respaldada por (Wei et al. 2023), quienes mencionan que las CNN combinadas con clasificadores clásicos superan a arquitecturas más profundas en tareas de comportamiento animal.

Tabla 16
Estadísticos descriptivos del accuracy por modelo.

	N	Media	Desv. Desviación	Desv. Error	95% del intervalo de confianza para la media		Mínimo	Máximo
					Límite inferior	Límite superior		
Modelo A	16	0,9951172	0,00025753	6,44E-05	0,994979959	0,995254416	0,994791667	0,99550189
Modelo B	16	0,9930753	0,00025203	6,30E-05	0,992940986	0,993209582	0,992660985	0,99360796
Modelo C	16	0,9941998	0,00039615	9,90E-05	0,993988719	0,994410902	0,993844697	0,99526515
Total	48	0,9941308	0,00089646	0,00012939	0,993870456	0,994391065	0,992660985	0,99550189

Los estadísticos descriptivos (Tabla 16) revelaron que el Modelo A obtuvo la mayor precisión promedio (M = 0.9951; DE = 0.0003), seguido por el Modelo C (M = 0.9942; DE = 0.0004) y, finalmente, el Modelo B (M = 0.9931; DE = 0.0003). Estos resultados sugieren que el Modelo A no solo tuvo un mejor rendimiento, sino también una mayor estabilidad, debido a su baja variabilidad.

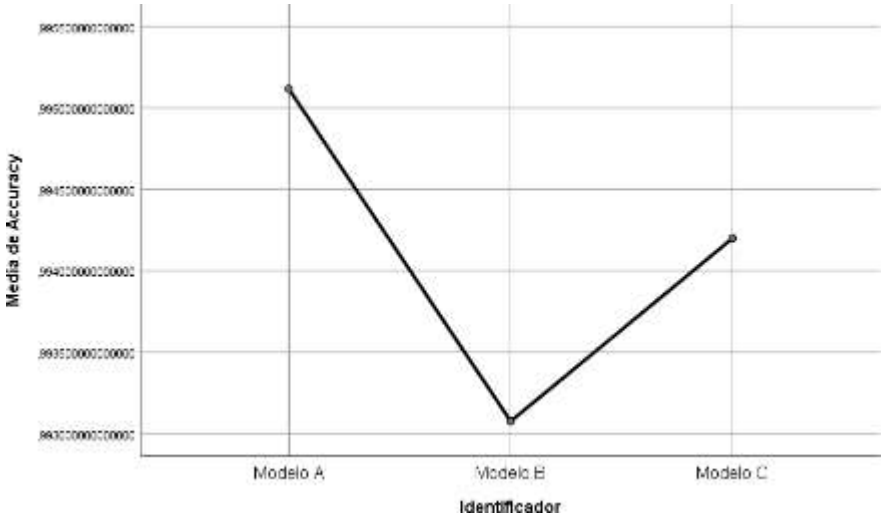

Figura 29
Comparación de medias de accuracy entre los tres modelos híbridos.

Finalmente, el gráfico de medias (Figura 29) mostró una diferencia clara en el rendimiento de los modelos evaluados. Se evidenció que el modelo AlexNet + RF alcanzó la mayor precisión promedio, seguido por VGG16 + RF, mientras que ResNet18 + RF obtuvo la media más baja. Esto respalda visualmente los resultados previos y confirma que AlexNet + RF fue el modelo con mejor desempeño en esta investigación. Esta tendencia coincide con hallazgos de otros estudios que destacan la efectividad de arquitecturas simples combinadas con clasificadores clásicos en tareas de clasificación específicas.

Al comparar este rendimiento con antecedentes recientes, se evidencia un resultado altamente competitivo. (Küster et al. 2021) lograron precisiones entre 66 % y 97 % al detectar partes corporales de cerdas mediante YOLO V3, mientras que (J. Chen et al. 2023) alcanzaron un 93.5 % al implementar un sistema de alerta temprana basado en YOLOv5. A su vez, (J. Lee et al. 2024; Wei et al. 2023) optimizaron modelos CNN para identificar comportamientos preparto como la inquietud o la echada lateral. Aunque trabajos como los de (Walls et al. 2024; Wutke et al. 2024) obtuvieron métricas elevadas con técnicas de pseudo-etiquetado, estos fueron desarrollados bajo entornos controlados y con recursos técnicos más avanzados.

En contraste, el presente estudio no solo alcanzó una precisión sobresaliente con una arquitectura ligera y un clasificador tradicional, sino que también implementó un enfoque de recolección de datos contextualizado. A partir de 80 videos grabados en tiempo real bajo condiciones naturales, se extrajeron automáticamente miles de imágenes representativas por categoría conductual mediante OpenCV, sin necesidad de intervención directa. Aplicado en granjas reales del distrito de Sauce (San Martín, Perú), el modelo AlexNet + RF demostró ser una alternativa efectiva, práctica y adaptable al monitoreo preparto en contextos rurales, sin depender de infraestructura compleja.

CONCLUSIONES

Se creó un dataset compuesto por más de 24,000 imágenes extraídas automáticamente de 80 videos grabados en granjas porcinas bajo condiciones reales y sin intervención directa. Las imágenes fueron clasificadas en cuatro categorías de comportamiento preparto: movimiento normal, inquietud, tumbada y parto. Este conjunto de datos constituye una base técnica, estructurada y representativa, adecuada para el entrenamiento de modelos de visión artificial aplicados al monitoreo animal.

Se implementaron tres modelos híbridos de visión artificial, integrando redes neuronales convolucionales preentrenadas (AlexNet, VGG16 y ResNet18) como extractores de características visuales, combinadas con el clasificador Random Forest. Esta integración permitió aplicar un enfoque modular y reproducible que articula técnicas de aprendizaje profundo y algoritmos tradicionales, adaptado a entornos de baja infraestructura tecnológica.

Se evaluó el desempeño de los modelos mediante validación cruzada estratificada y análisis estadístico con ANOVA y prueba post hoc de Tukey, identificando que el modelo AlexNet + Random Forest alcanzó la mejor precisión promedio, con menor varianza y mayor eficiencia computacional. Esto respalda su idoneidad para aplicaciones en contextos rurales con recursos limitados.

Se concluye que el uso de modelos híbridos de visión artificial permite monitorear de forma no invasiva, automatizada y precisa los comportamientos preparto en cerdas, aportando una solución tecnológica viable para optimizar la gestión reproductiva. Esta propuesta representa un avance significativo en la aplicación de inteligencia artificial en la porcicultura, con impacto positivo en el bienestar animal y la productividad rural.

RECOMENDACIONES

Se recomienda ampliar el dataset incorporando nuevos videos grabados en distintas condiciones de iluminación, con diferentes ángulos de cámara y razas de cerdas. Esto permitirá mejorar la capacidad de generalización de los modelos, fortalecer su robustez ante variaciones del entorno y garantizar un desempeño más estable en escenarios reales.

Es aconsejable aplicar técnicas de aumentación de datos durante el preprocesamiento de imágenes, con el objetivo de equilibrar las clases y reducir la dependencia de grandes volúmenes de grabaciones. Asimismo, se sugiere explorar clasificadores complementarios que optimicen la velocidad y precisión en contextos de implementación en tiempo real.

Se recomienda incorporar una etapa previa de detección automática de cerdas mediante técnicas de segmentación o detección de objetos, con el fin de reducir el ruido visual y mejorar la eficiencia del sistema completo. Asimismo, se sugiere validar la solución desarrollada en dispositivos de bajo costo como Raspberry Pi o Jetson Nano, lo cual permitiría comprobar su rendimiento en tiempo real y su aplicabilidad en granjas con recursos computacionales limitados.

Finalmente, se sugiere fortalecer la validación práctica del sistema en colaboración con médicos veterinarios y granjas comerciales, ajustando los umbrales de alerta según criterios técnicos del manejo reproductivo. Además, se recomienda promover la capacitación técnica en inteligencia artificial aplicada a la producción animal, integrándola en programas de formación agrícola para facilitar su adopción en el sector ganadero.

REFERENCIAS BIBLIOGRÁFICAS

Bhatt, Dulari, Chirag Patel, Hardik Talsania, Jigar Patel, Rasmika Vaghela, Sharnil Pandya, Kirit Modi, and Hemant Ghayvat. 2021. “CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope.” *Electronics* 10(20):2470. doi:10.3390/electronics10202470.

Bonneau, Mathieu, Bernard Benet, Yann Labrune, Jean Bailly, Edmond Ricard, and Laurianne Canario. 2021. “Predicting Sow Postures from Video Images: Comparison of Convolutional Neural Networks and Segmentation Combined with Support Vector Machines under Various Training and Testing Setups.” *Biosystems Engineering* 212:19–29. doi:10.1016/J.BIOSYSTEMSENG.2021.09.014.

Borges Oliveira, Dario Augusto, Luiz Gustavo Ribeiro Pereira, Tiago Bresolin, Rafael Ehrich Pontes Ferreira, and Joao Ricardo Reboucas Dorea. 2021. “A Review of Deep Learning Algorithms for Computer Vision Systems in Livestock.” *Livestock Science* 253:104700. doi:10.1016/j.livsci.2021.104700.

Chen, Chen, Weixing Zhu, and Tomas Norton. 2021. “Behaviour Recognition of Pigs and Cattle: Journey from Computer Vision to Deep Learning.” *Computers and Electronics in Agriculture* 187:106255. doi:10.1016/j.compag.2021.106255.

Chen, Jinxin, Jie Zhou, Longshen Liu, Cuini Shu, Mingxia Shen, and Wen Yao. 2023. “Sow Farrowing Early Warning and Supervision for Embedded Board Implementations.” *Sensors* 23(2):727. doi:10.3390/s23020727.

Chen, Zhe, Jisheng Lu, and Haiyan Wang. 2023. “A Review of Posture Detection Methods for Pigs Using Deep Learning.” *Applied Sciences 2023, Vol. 13, Page 6997* 13(12):6997. doi:10.3390/APP13126997.

De Computación, Carrera, Karen Barbarita, Alava Zambrano, Willians Eduardo, and Basurto Vidal. 2024. “Sistema de Alerta Temprana de Labor de Parto En El Hato Porcino Empleando Técnicas de Redes Neuronales Convolucionales y LSTM.” http://repositorio.espam.edu.ec/handle/42000/2424.

Devi Priya, R., V. Devisurya, N. Anitha, N. Kalaivaani, P. Keerthana, and E. Adarsh Kumar. 2022. “Automated Cattle Classification and Counting Using Hybridized Mask R-CNN and YOLOv3 Algorithms.” Pp. 358–67 in.

Domingos, R. L., B. A. N. Silva, F. Gil Rueda, A. M. Luna, J. K. Htoo, H. G. Brand, F. I.

G. Rebordões, M. F. Gonçalves, S. K. Brito, L. T. S. Martins, G. T. S. Pereira, and M. L. T. Abreu. 2024. “Use of a Precision Feeding Program during Gestation Improves the Performance of High-Producing Sows.” *Animal Feed Science and Technology* 311:115969. doi:10.1016/j.anifeedsci.2024.115969.

Dore, Alexandre, Mathieu Lihoreau, Jean Bailly, Yvon Billon, Jean-François Bompa, Edmond Ricard, Dominique Henry, Laurianne Canario, and Hervé Aubert. 2022. “Automated Detection of Sow Posture Changes with Millimeter-Wave Radars and Deep Learning.” *BioRxiv* 2022.04.13.488188. doi:10.1101/2022.04.13.488188.

van Erp-van der Kooij, Elaine, Lois F. de Graaf, Dennis A. de Kruijff, Daphne Pellegrom, Renilda de Rooij, Nian I. T. Welters, and Jeroen van Poppel. 2023. “Using Sound Location to Monitor Farrowing in Sows.” *Animals* 13(22):3538. doi:10.3390/ani13223538.

Espejo, Guadalupe, Pedro Paredes-Ramos, Concepción Ahuja, Apolo Carrasco, and Fernando Naranjo. 2022. “Efecto Del Enriquecimiento Ambiental En Cerdas Gestantes Sobre Su Comportamiento al Parto y Concentraciones de Cortisol.” *Informacion Tecnica Economica Agraria*. doi:10.12706/itea.2022.004.

Farahnakian, Fahimeh, Farshad Farahnakian, Stefan Björkman, Victor Bloch, Matti Pastell, and Jukka Heikkonen. 2024. “Pose Estimation of Sow and Piglets during Free Farrowing Using Deep Learning.” *Journal of Agriculture and Food Research* 16:101067. doi:10.1016/j.jafr.2024.101067.

Fynn, Mark, Gary Crow, and Laurie Connor. 2021. “Pre-Farrow Enrichment with Burlap Sheet: Potential Benefit for Sow Performance.” *Canadian Journal of Animal Science* 101(4):781–87. doi:10.1139/cjas-2021-0027.

Gibbs, Joshua, and Francesco P. Cappuccio. 2022. “Plant-Based Dietary Patterns for Human and Planetary Health.” Nutrients 14(8):1614. doi:10.3390/nu14081614.

Gulliksen, S. M., T. Framstad, C. Kielland, M. A. Velazquez, M. M. Terøy, E. M. Helland, R. H. Lyngstad, A. J. Oropeza Delgado, and M. Oropeza-Moe. 2023. “Infrared Thermography as a Possible Technique for the Estimation of Parturition Onset in Sows.” *Porcine Health Management* 9(1). doi:10.1186/S40813-022-00301-X.

Havlíček, Jaroslav, Ludmila Dömeová, Luboš Smutka, Helena Řezbová, Lucie Severová, Tomáš Šubrt, Karel Šrédl, and Roman Svoboda. 2020. “Efficiency of

Pig Production in the Czech Republic and in an International Context." *Agriculture* 10(12):597. doi:10.3390/agriculture10120597.

Ho, Kuan Ying, Yu Jung Tsai, and Yan Fu Kuo. 2021. "Automatic Monitoring of Lactation Frequency of Sows and Movement Quantification of Newborn Piglets in Farrowing Houses Using Convolutional Neural Networks." *Computers and Electronics in Agriculture* 189:106376. doi:10.1016/J.COMPAG.2021.106376.

Ho, Kuan-Ying, Yu-Jung Tsai, and Yan-Fu Kuo. 2021. "Automatic Monitoring of Lactation Frequency of Sows and Movement Quantification of Newborn Piglets in Farrowing Houses Using Convolutional Neural Networks." *Computers and Electronics in Agriculture* 189:106376. doi:10.1016/j.compag.2021.106376.

Hukkinen, V. M., C. Munsterhjelm, M. Kurtti, N. Immonen, and A. Valros. 2024. "Impact of Farrowing System and Prepartum Nest-Building Material on Nest-Building Behaviour and Farrowing in Sows." *Animal* 18(6):101183. doi:10.1016/j.animal.2024.101183.

International Guiding Principles for Biomedical Research Involving Animals - CIOMS. n.d. Retrieved May 29, 2025. https://cioms.ch/publications/product/international-guiding-principles-for-biomedical-research-involving-animals-2/.

Khaled, Afifa, Chao Li, Jia Ning, and Kun He. 2025. "BCN: Batch Channel Normalization for Image Classification." 295–308. doi:10.1007/978-3-031-78195-7_20.

Kuldashboy, Avazov, Sabina Umirzakova, Sharofiddin Allaberdiev, Rashid Nasimov, Akmalbek Abdusalomov, and Young Im Cho. 2024. "Efficient Image Classification through Collaborative Knowledge Distillation: A Novel AlexNet Modification Approach." *Heliyon* 10(14):e34376. doi:10.1016/j.heliyon.2024.e34376.

Küster, Steffen, Philipp Nolte, Cornelia Meckbach, Bernd Stock, and Imke Traulsen. 2021. "Automatic Behavior and Posture Detection of Sows in Loose Farrowing Pens Based on 2D-Video Images." *Frontiers in Animal Science* 2. doi:10.3389/fanim.2021.758165.

Lee, Ji Hyeon, Yo Han Choi, Han Sung Lee, Hyun Ju Park, Jun Seon Hong, Ji Hwan Lee, Soo Jin Sa, Yong Min Kim, Jo Eun Kim, Yong Dae Jeong, and Hyun Chong Cho. 2024. "Enhanced Swine Behavior Detection with YOLOs and a Mixed Efficient Layer Aggregation Network in Real Time." *Animals 2024, Vol. 14, Page 3375* 14(23):3375. doi:10.3390/ANI14233375.

Lee, Juho, Hyeonwook Shin, Junsik Kim, Geonil Lee, and Jinhyeon Yun. 2024. "Large

Litters Have a Detrimental Impact on Litter Performance and Postpartum Maternal Behaviour in Primiparous Sows." *Porcine Health Management* 10(1):9. doi:10.1186/s40813-024-00360-2.

Lei, Kaidong, Chao Zong, Ting Yang, Shanshan Peng, Pengfei Zhu, Hao Wang, Guanghui Teng, and Xiaodong Du. 2022. "Detection and Analysis of Sow Targets Based on Image Vision." *Agriculture* 12(1):73. doi:10.3390/agriculture12010073.

Li, Guoming, Yanbo Huang, Zhiqian Chen, Gary D. Chesser, Joseph L. Purswell, John Linhoss, and Yang Zhao. 2021. "Practices and Applications of Convolutional Neural Network-Based Computer Vision Systems in Animal Farming: A Review." *Sensors* 21(4):1492. doi:10.3390/s21041492.

Liu, Citation :., K. ;. Huang, Y. ;. Liu, J. ;. Tan, Z. ;. Xiao, Kejian Liu, Yigui Huang, Junbin Liu, Zujie Tan, and Deqin Xiao. 2025. "Prediction of Sow Farrowing Onset Time Using Activity Time Series Extracted by Optical Flow Estimation." *Animals 2025, Vol. 15, Page 998* 15(7):998. doi:10.3390/ANI15070998.

Liu, Longshen, Jie Zhou, Bo Zhang, Suyang Dai, and Mingxia Shen. 2022. "Visual Detection on Posture Transformation Characteristics of Sows in Late Gestation Based on Libra R-CNN." *Biosystems Engineering* 223:219–31. doi:10.1016/J.BIOSYSTEMSENG.2022.09.003.

Liu, Songling, and Binxin Wang. 2024. "Optimized Modified ResNet18: A Residual Neural Network for High Resolution." *2024 IEEE 4th International Conference on Electronic Technology, Communication and Information, ICETCI 2024* 1–5. doi:10.1109/ICETCI61221.2024.10594672.

Makalesi, Araştırma, Research Article, Sorumlu Yazar, Corresponding Author Geliş, and Çağatay Murat YILMAZ. 2025. "A Novel Approach to Motor Imagery EEG Signal Transformation and Classification Using Stockwell Transform and Deep Learning Models." *The Black Sea Journal of Sciences* 15(1):152–70. doi:10.31466/KFBD.1509850.

Ministerio de Desarrollo Agrario y Riego. 2024. "Gobierno Protege La Porcicultura Nacional En Beneficio de Más de 400 Mil Productores."

Moustsen, V. A., Y. M. Seddon, and M. J. Hansen. 2023. "Animal Board Invited Review: The Need to Consider Emissions, Economics and Pig Welfare in the Transition from Farrowing Crates to Pens with Loose Lactating Sows." *Animal* 17(9):100913. doi:10.1016/j.animal.2023.100913.

Nannoni, Eleonora, André J. A. Aarnink, Herman M. Vermeer, Inonge Reimert, Michaela Fels, and Marc B. M. Bracke. 2020. “Soiling of Pig Pens: A Review of Eliminative Behaviour.” *Animals* 10(11):2025. doi:10.3390/ani10112025.

Oczak, Maciej, Florian Bayer, Sebastian Vetter, Kristina Maschat, and Johannes Baumgartner. 2022. “Comparison of the Automated Monitoring of the Sow Activity in Farrowing Pens Using Video and Accelerometer Data.” *Computers and Electronics in Agriculture* 192:106517. doi:10.1016/J.COMPAG.2021.106517.

Oczak, Maciej, Kristina Maschat, and Johannes Baumgartner. 2023. “Implementation of Computer-Vision-Based Farrowing Prediction in Pens with Temporary Sow Confinement.” *Veterinary Sciences 2023, Vol. 10, Page 109* 10(2):109. doi:10.3390/VETSCI10020109.

de Paula, Tauana Maria Carlos Guimarães, Rafael Vieira de Sousa, Marisol Parada Sarmiento, Ton Kramer, Edson José de Souza Sardinha, Leandro Sabei, Júlia Silvestrini Machado, Mirela Vilioti, and Adroaldo José Zanella. 2024. “Deep Learning Pose Detection Model for Sow Locomotion.” *Scientific Reports 2024 14:1* 14(1):1–13. doi:10.1038/s41598-024-62151-7.

Qin, Caijie, Yong Li, and Heming Jia. 2025. “Intelligent Livestock Farming: Monitoring Pig Behavior with Enhanced YOLOv5 for Spatial Temporal Feature Fusion.” *International Journal of Intelligent Computing and Cybernetics* ahead-of-print(ahead-of-print). doi:10.1108/IJICC-01-2025-0013/FULL/XML.

Riekert, Martin, Achim Klein, Felix Adrion, Christa Hoffmann, and Eva Gallmann. 2020. “Automatically Detecting Pig Position and Posture by 2D Camera Imaging and Deep Learning.” *Computers and Electronics in Agriculture* 174:105391. doi:10.1016/J.COMPAG.2020.105391.

Sedes - Servicio Nacional de Sanidad Agraria del Perú - Plataforma del Estado Peruano. n.d. Retrieved May 29, 2025. https://www.gob.pe/institucion/senasa/sedes.

Shao, Hongmin, Jingyu Pu, and Jiong Mu. 2021. “Pig-Posture Recognition Based on Computer Vision: Dataset and Exploration.” *Animals 2021, Vol. 11, Page 1295* 11(5):1295. doi:10.3390/ANI11051295.

Vandresen, Bianca, Jen-Yun Chou, and Maria José Hötzel. 2024. “How Is Pig Welfare Assessed in Studies on Farrowing Housing Systems? A Systematic Review.” *Applied Animal Behaviour Science* 275:106298. doi:10.1016/j.applanim.2024.106298.

Walls, Alexandra, Evelyn Hall, Sabrina Lomax, and Roslyn Bathgate. 2024. “An

Investigation of Parturient Ocular Appearance in Sows.” *Animals 2024, Vol. 14, Page 2693* 14(18):2693. doi:10.3390/ANI14182693.

Wei, Jiacheng, Xi Tang, Jinxiu Liu, and Zhiyan Zhang. 2023. “Detection of Pig Movement and Aggression Using Deep Learning Approaches.” *Animals* 13(19):3074. doi:10.3390/ANI13193074/S1.

Wutke, Martin, Clara Lensches, Ulrich Hartmann, and Imke Traulsen. 2024. “Towards Automatic Farrowing Monitoring—A Noisy Student Approach for Improving Detection Performance of Newborn Piglets.” *PLOS ONE* 19(10):e0310818. doi:10.1371/journal.pone.0310818.

Xie, Chuanqi, Yuji Cang, Xizhong Lou, Hua Xiao, Xing Xu, Xiangjun Li, and Weidong Zhou. 2024. “A Novel Approach Based on a Modified Mask R-CNN for the Weight Prediction of Live Pigs.” *Artificial Intelligence in Agriculture* 12:19–28. doi:10.1016/j.aiia.2024.03.001.

Yang, Ruotong, Zikang Chen, Huanliang Xu, Mingxia Shen, Pinghua Li, Tomas Norton, and Mingzhou Lu. 2023. “Recognizing the Rooting Action of Prepartum Sow in Free-Farrowing Pen Using Computer Vision.” *Computers and Electronics in Agriculture* 213:108167. doi:10.1016/J.COMPAG.2023.108167.

Yin, Ling, Shengzheng Jiang, Chengzhi Ye, Zhenfang Wu, Jie Yang, Sumin Zhang, and Gengyuan Cai. 2024. “Recognizing Sow Parturition Using Lightweight Model with Edge Computing[面向边缘计算的轻量级母猪分娩识别模型].” *Nongye Gongcheng Xuebao/Transactions of the Chinese Society of Agricultural Engineering* 40(17):205–15. doi:10.11975/j.issn.1002-6819.202404173.

Zhang, Xiaojun, Congcong Li, Yue Hao, and Xianhong Gu. 2020. “Effects of Different Farrowing Environments on the Behavior of Sows and Piglets.” *Animals* 10(2):320. doi:10.3390/ani10020320.

Zhou, Nanxiang. 2024. “Image Recognition in Depth: Comparative Study of CNN and Pre-Trained VGG16 Architecture for Classification Tasks.” *Https://Doi.Org/10.1117/12.3026829* 13075:553–59. doi:10.1117/12.3026829.

ANEXOS

Anexo 1: Permiso de autorización para la realización de la investigación en la granja porcina.

DOCUMENTO DE PERMISO PARA REALIZACIÓN DE INVESTIGACIÓN EN GRANJA PORCINA

Yo, **Ramiro Rojas Mejía**, identificado con DNI N.° **80195905**, en calidad de propietario de la granja porcina ubicada en el distrito de Sauce, provincia de San Martín, autorizo al Sr. **Marc Anthoni Reátegui Sifuentes**, identificado con DNI N.° **72717961**, estudiante de la carrera de Ingeniería de Sistemas e Informática, a realizar actividades de observación y registro del comportamiento preparto de cerdas gestantes en las instalaciones de mi propiedad.

Esta autorización se otorga con fines exclusivamente académicos, en el marco del desarrollo de su proyecto de investigación titulado: **"Monitoreo y predicción del momento del parto en cerdas mediante modelos de visión artificial basado en CNN"**, el cual no contempla la manipulación directa de los animales, ni el uso de sustancias químicas o dispositivos invasivos, ni la aplicación de técnicas restringidas por normativas de bienestar animal.

Las actividades autorizadas comprenden únicamente el registro visual mediante cámaras no invasivas, garantizando el respeto por el entorno natural de los animales y sin alterar su comportamiento ni rutina diaria. El investigador se compromete a actuar bajo principios éticos, respetando la normativa vigente sobre el trato digno a los animales y la confidencialidad de la información obtenida.

Esta autorización tiene vigencia durante todo el año 2025, permitiendo el ingreso del investigador a las instalaciones en los días y horarios previamente coordinados con el propietario, sin interrumpir las labores habituales de la granja.

En constancia de conformidad, se firma el presente documento en el distrito de Sauce, provincia de San Martín.

Firma del Propietario

Ramiro Rojas Mejía
DNI: 80195905

Firma del Investigador

Marc Anthoni Reátegui Sifuentes
DNI: 72717961

Anexo 2: Código de los modelos desarrollados.

1.- Librerias

```
# === SISTEMA DE ARCHIVOS Y UTILIDADES ===
import os
import json
import subprocess
from glob import glob
from concurrent.futures import ThreadPoolExecutor
# === MANEJO DE DATOS Y CÁLCULOS ===
import numpy as np
import pandas as pd
# === VISUALIZACIÓN ===
import matplotlib.pyplot as plt
import seaborn as sns
from tabulate import tabulate
from sklearn.metrics import (
    confusion_matrix, ConfusionMatrixDisplay,
    roc curve, auc, classification report,
    accuracy_score, f1_score, precision_score, recall_score,
    roc auc score
)
# === PYTORCH Y TORCHVISION ===
import torch
from torch.utils.data import DataLoader
from torchvision import models, transforms
from torchvision.datasets import ImageFolder
from torchvision.models import (
    alexnet, resnet18, vgg16,
    AlexNet_Weights, ResNet18_Weights, VGG16_Weights
)
# === MACHINE LEARNING (Scikit-learn) ===
from sklearn.model selection import train test split, StratifiedKFold,
RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import label_binarize
# === SERIALIZACIÓN ===
import joblib
# === GOOGLE COLAB (si aplica) ===
from google.colab import drive
drive.mount('/content/drive')
```

2.- Conexión al Drive

```
# === MONTAR GOOGLE DRIVE EN GOOGLE COLAB ===
drive.mount('/content/drive')
# === VERIFICACIÓN Y COPIA DEL DATASET DESDE GOOGLE DRIVE ===
if not os.path.exists('/content/dataset'):
    !cp -r "/content/drive/MyDrive/dataset" "/content/"
else:
    print("Dataset ya copiado en /content.")
# === MOSTRAR ARCHIVOS DEL DATASET EN COLAB ===
!ls /content/dataset
```

3.- Extracción de fotogramas

```
# ======================
# 🎥 Función para extraer frames con nombre de clase incluido
# ======================
def extraer_frames_ffmpeg(ruta_video, carpeta_clase, nombre_base, fps=3):
    os.makedirs(carpeta_clase, exist_ok=True)
    patron_salida = os.path.join(carpeta_clase, f"{nombre_base}_%04d.jpg")
    comando = [
        "ffmpeg", "-i", ruta_video, "-vf", f"fps={fps}",
        patron_salida, "-hide_banner", "-loglevel", "error"
    ]
    subprocess.run(comando)
# ======================
# 📏 Función para redimensionar una imagen a tamaño fijo
# ======================
def procesar_imagen(img_path, size=(224, 224)):
    img = cv2.imread(img_path)
    if img is None:
        print(f"⚠ No se pudo leer: {img_path}")
        return
    img_resized = cv2.resize(img, size)
    cv2.imwrite(img path, img resized)
# ======================
# 🔄 Procesar todas las imágenes de una carpeta
# ======================
def procesar todas imagenes(carpeta clase, size=(224, 224)):
    imagenes = glob(os.path.join(carpeta_clase, "*.jpg"))
    with ThreadPoolExecutor(max workers=8) as executor:
        executor.map(lambda img: procesar_imagen(img, size), imagenes)
# ======================
# 🛍 Procesar una clase (extraer + redimensionar)
# ======================
def procesar_clase(carpeta_videos, carpeta_frames, nombre_clase, fps=3):
    videos = [f for f in os.listdir(carpeta videos)
              if f.lower().endswith(('.mp4', '.avi', '.mov', '.webm', '.mkv'))]

    for video in videos:
        ruta video = os.path.join(carpeta videos, video)
        nombre_sin_ext = os.path.splitext(video)[0]
        nombre base = f"{nombre clase}{nombre sin ext}"
        print(f"\n🎞 Extrayendo frames de: {video}")
        extraer frames ffmpeg(ruta video, carpeta frames, nombre base, fps)

    print(f"□ Redimensionando frames en: {carpeta frames}")
    procesar_todas_imagenes(carpeta_frames, size=(224, 224))
# ======================
# 🔧 Configuración inicial
# ======================
```

```
clases = ["inquietud", "movimientonormal", "parto", "tumbada"]
base_entrada = "/content/dataset"
base_salida = "/content/frames"
for clase in clases:
    carpeta_videos = os.path.join(base_entrada, clase)
    carpeta_frames = os.path.join(base_salida, clase)
    os.makedirs(carpeta_frames, exist_ok=True)
    procesar_clase(carpeta_videos, carpeta_frames, nombre_clase=clase, fps=3)
print("\n✅ Extracción y procesamiento acelerado completado.")
```

Conteo y distribución de imágenes por comportamiento

```
# 📊 Preprocesamiento y visualización de datos
conteo_por_clase = {}
for clase in clases:
    carpeta = os.path.join(base_salida, clase)
    cantidad = len(glob(os.path.join(carpeta, "*.jpg")))
    conteo_por_clase[clase] = cantidad
    print(f"📂 Clase '{clase}': {cantidad} imágenes")
# 🎨 Colores distintos para cada clase
colores = ['#FFA07A', '#20B2AA', '#FFD700', '#9370DB']  # Puedes cambiar los
hex
# Mostrar histograma
plt.figure(figsize=(8, 5))
plt.bar(conteo_por_clase.keys(), conteo_por_clase.values(), color=colores)
plt.xlabel("Clases")
plt.ylabel("Cantidad de imágenes")
plt.title("Distribución de imágenes por clase")
plt.xticks(rotation=15)
plt.tight_layout()
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

4.- Visualización de clases

```
# ======================
# CLASE: INQUIETUD
# ======================
clase = "inquietud"
color_borde = "blue"
descripcion = "Conducta activa con movimientos como caminar, girar, levantar la cabeza,
rascar o morder la jaula."
carpeta = os.path.join(base_salida, clase)
imagenes = glob(os.path.join(carpeta, "*.jpg"))[:12]
plt.figure(figsize=(12, 9))
for i, img_path in enumerate(imagenes):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    ax = plt.subplot(3, 4, i + 1)
    ax.imshow(img)
    ax.set_title(clase, fontsize=9)
    ax.axis('off')

    for spine in ax.spines.values():
        spine.set_edgecolor(color_borde)
        spine.set linewidth(3)
plt.suptitle(f"Clase: {clase}\n{descripcion}", fontsize=14)
plt.tight layout()
plt.show()
# ======================
# CLASE: MOVIMIENTONORMAL
# ======================
clase = "movimientonormal"
color borde = "green"
descripcion = "Desplazamiento natural sin signos de estrés o alteración conductual.
Sirve como referencia frente a otros estados."
carpeta = os.path.join(base_salida, clase)
imagenes = glob(os.path.join(carpeta, "*.jpg"))[:12]
plt.figure(figsize=(12, 9))
for i, img path in enumerate(imagenes):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR BGR2RGB)

    ax = plt.subplot(3, 4, i + 1)
    ax.imshow(img)
    ax.set title(clase, fontsize=9)
    ax.axis('off')
    for spine in ax.spines.values():
        spine.set_edgecolor(color_borde)
        spine.set linewidth(3)
plt.suptitle(f"Clase: {clase}\n{descripcion}", fontsize=14)
plt.tight_layout()
```

```
plt.show()
# ======================
# CLASE: PARTO
# ======================
clase = "parto"
color_borde = "purple"
descripcion = "Postura específica que adopta la cerda tras un estado de inquietud,
asociada al inicio del proceso de parto."
carpeta = os.path.join(base_salida, clase)
imagenes = glob(os.path.join(carpeta, "*.jpg"))[:12]
plt.figure(figsize=(12, 9))
for i, img_path in enumerate(imagenes):
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    ax = plt.subplot(3, 4, i + 1)
    ax.imshow(img)
    ax.set_title(clase, fontsize=9)
    ax.axis('off')
    for spine in ax.spines.values():
        spine.set edgecolor(color borde)
        spine.set_linewidth(3)
plt.suptitle(f"Clase: {clase}\n{descripcion}", fontsize=14)
plt.tight_layout()
plt.show()
# ======================
# CLASE: TUMBADA
# ======================
clase = "tumbada"
color_borde = "red"
descripcion = "Postura pasiva donde la cerda permanece acostada sin desplazamiento,
reflejando un estado de descanso normal."
carpeta = os.path.join(base salida, clase)
imagenes = glob(os.path.join(carpeta, "*.jpg"))[:12]
plt.figure(figsize=(12, 9))
for i, img_path in enumerate(imagenes):
    img = cv2.imread(img path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    ax = plt.subplot(3, 4, i + 1)
    ax.imshow(img)
    ax.set title(clase, fontsize=9)
    ax.axis('off')

    for spine in ax.spines.values():
        spine.set_edgecolor(color_borde)
        spine.set_linewidth(3)
plt.suptitle(f"Clase: {clase}\n{descripcion}", fontsize=14)
plt.tight_layout()
plt.show()
```

5.- Modelos Hibridos

MODELO A: HÍBRIDO ALEXNET + RANDOM FOREST

```
# === CONFIGURACIÓN ===
DATA_DIR = '/content/frames'
BATCH_SIZE = 32
IMG_SIZE = 224
SEED = 42
N_SPLITS = 5
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("✅ Dispositivo en uso:", DEVICE)
# === TRANSFORMACIONES ===
transform = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
# === CARGA DEL DATASET ===
dataset = ImageFolder(DATA_DIR, transform=transform)
loader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=False)
class_names = dataset.classes
# === MODELO ALEXNET (sin capa final) ===
alexnet = models.alexnet(weights=models.AlexNet_Weights.DEFAULT).to(DEVICE)
alexnet.classifier = torch.nn.Sequential(*list(alexnet.classifier.children())[:-1])
alexnet.eval()
# === EXTRAER CARACTERÍSTICAS ===
features = []
labels = []
with torch.no_grad():
    for inputs, targets in loader:
        inputs = inputs.to(DEVICE)
        output = alexnet(inputs)
        features.append(output.cpu().numpy())
        labels.extend(targets.numpy())
features = np.concatenate(features)
labels = np.array(labels)
# === SEPARACIÓN DEL TEST SET EXTERNO ===
X kfold, X test, y kfold, y test = train test split(
    features, labels, test_size=0.2, stratify=labels, random_state=SEED
)
# === GUARDAR TEST SET PARA USO POSTERIOR ===
np.save('/content/X test A.npy', X test)
np.save('/content/y_test_A.npy', y_test)
print("💾 Test set externo guardado: X test A.npy y y test A.npy")
# === BÚSQUEDA DE HIPERPARÁMETROS (solo sobre el 80%) ===
X train,  , y train,   = train test split(X kfold, y kfold, test size=0.2,
stratify=y_kfold, random_state=SEED)
```

```
param_dist = {
    'n_estimators': [100, 200, 300, 500],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt', 'log2']
}
search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=SEED, n_jobs=-1),
    param_distributions=param_dist,
    n_iter=10,
    scoring='accuracy',
    cv=StratifiedKFold(n_splits=2, shuffle=True, random_state=SEED),
    verbose=2,
    random_state=SEED,
    n jobs=-1
)
search.fit(X train, y train)
best_params = search.best_params_
print("\n✅ Mejores hiperparámetros encontrados:")
for k, v in best_params.items():
    print(f"{k}: {v}")
# === VALIDACIÓN CON K-FOLD (solo en 80%) ===
final model = RandomForestClassifier(**best params, random state=SEED, n jobs=-1)
kfold = StratifiedKFold(n_splits=N_SPLITS, shuffle=True, random_state=SEED)
acc scores, f1 scores = [], []
fold_preds, fold_labels, all_probs = [], [], []
print("\n📊 Resultados por fold:")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_kfold, y_kfold)):
    X tr, X val = X kfold[train idx], X kfold[val idx]
    y_tr, y_val = y_kfold[train_idx], y_kfold[val_idx]
    final model.fit(X tr, y tr)
    preds = final_model.predict(X_val)
    probs = final model.predict proba(X val)
    fold_preds.extend(preds)
    fold labels.extend(y val)
    all_probs.extend(probs)
    acc = accuracy score(y val, preds)
    f1 = f1_score(y_val, preds, average='macro')
    acc scores.append(acc)
    f1_scores.append(f1)
    print(f"Fold {fold+1}: Accuracy = {acc:.4f}, F1 Macro = {f1:.4f}")
# === PROMEDIO DE MÉTRICAS K-FOLD ===
print(f"\n🎯 Accuracy promedio (5 folds): {np.mean(acc_scores):.4f}")
print(f"🎯 F1 Macro promedio (5 folds): {np.mean(f1_scores):.4f}")
# === GUARDADO DE MÉTRICAS GLOBALES ===
fold_preds = np.array(fold_preds)
fold_labels = np.array(fold_labels)
```

```
all_probs = np.array(all_probs)
accuracy = accuracy_score(fold_labels, fold_preds)
f1_macro = f1_score(fold_labels, fold_preds, average='macro')
precision_macro = precision_score(fold_labels, fold_preds, average='macro')
recall_macro = recall_score(fold_labels, fold_preds, average='macro')
roc_auc = roc_auc_score(fold_labels, all_probs, multi_class='ovr')
f1_weighted = f1_score(fold_labels, fold_preds, average='weighted')
precision_weighted = precision_score(fold_labels, fold_preds, average='weighted')
recall_weighted = recall_score(fold_labels, fold_preds, average='weighted')
conf_matrix = confusion_matrix(fold_labels, fold_preds)
np.savetxt("/content/MODELO_A_matriz_confusion.csv", conf_matrix, delimiter=",",
fmt="%d")
report_dict = classification_report(fold_labels, fold_preds, target_names=class_names,
output_dict=True)
report_df = pd.DataFrame(report_dict).transpose()
report df.to excel("/content/MODELO A reporte por clase.xlsx")
metrics_df = pd.DataFrame({
    "Métrica": [
        "Accuracy", "F1 Macro", "Precision Macro", "Recall Macro",
        "F1 Weighted", "Precision Weighted", "Recall Weighted", "ROC AUC (OVR)"
    ],
    "Valor": [
        accuracy, f1_macro, precision_macro, recall_macro,
        f1 weighted, precision weighted, recall weighted, roc auc
    ]
})
metrics_df.to_excel("/content/MODELO_A_metricas_globales.xlsx", index=False)
print("\n Métricas globales guardadas exitosamente:")
print("- MODELO_A_metricas_globales.xlsx")
print("- MODELO A matriz confusion.csv")
print("- MODELO_A_reporte_por_clase.xlsx")
# === GUARDAR EL MODELO ENTRENADO ===
joblib.dump(final_model, '/content/MODELO_A_random_forest.pkl')
print("\n Modelo Random Forest guardado exitosamente como
'MODELO_A_random_forest.pkl'")
with open('/content/MODELO A class names.json', 'w') as f:
    json.dump(class_names, f)
print(" Nombres de las clases guardados como 'MODELO A class names.json'")
```

```
# === CARGAR MATRIZ DE CONFUSIÓN GUARDADA ===
cm = np.loadtxt("/content/MODELO_A_matriz_confusion.csv", delimiter=",")
# === CARGAR NOMBRES DE CLASE ===
dataset = ImageFolder("/content/frames")
class_names = dataset.classes
# === GRAFICAR MATRIZ DE CONFUSIÓN (sin notación científica) ===
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap="Blues", xticks_rotation=45, values_format=".0f")
plt.title("Matriz de Confusión Global - MODELO A")
```

```
plt.grid(False)
plt.tight_layout()
plt.show()
```

```
# === Cargar matriz de confusión y reporte por clase ===
conf_matrix = pd.read_csv("/content/MODELO_A_matriz_confusion.csv", header=None).values
reporte = pd.read_excel("/content/MODELO_A_reporte_por_clase.xlsx")
# === Reconstruir y_true y y_pred desde la matriz de confusión ===
y_true = []
y_pred = []
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        y_true.extend([i] * conf_matrix[i, j])
        y_pred.extend([j] * conf_matrix[i, j])
# === Binarizar etiquetas
n classes = conf matrix.shape[0]
y_true_bin = label_binarize(y_true, classes=list(range(n_classes)))
y pred bin = label binarize(y pred, classes=list(range(n classes)))
colors = ['blue', 'orange', 'green', 'red']
# === Graficar curvas ROC
plt.figure(figsize=(10, 6))
auc scores = []
for i in range(n_classes):
    fpr, tpr,   = roc curve(y true bin[:, i], y pred bin[:, i])
    auc_val = auc(fpr, tpr)
    auc scores.append(auc val)
    plt.plot(fpr, tpr, color=colors[i], label=f"Clase {i} (AUC={auc_val:.4f})")
# === Agregar línea aleatoria y detalles
plt.plot([0, 1], [0, 1], 'k--', label="Aleatorio")
plt.title(f"Curvas ROC por Clase - MODELO A\nAUC Macro Promedio:
{np.mean(auc_scores):.4f}")
plt.xlabel("Tasa de Falsos Positivos (FPR)")
plt.ylabel("Tasa de Verdaderos Positivos (TPR)")
plt.legend(loc='lower right')
plt.grid(True)
plt.tight layout()
# === Guardar como imagen
plt.savefig("/content/MODELO A curva ROC.png")
plt.show()
```

MODELO B: HÍBRIDO RESNET18 + RANDOM FOREST

```
# === CONFIGURACIÓN ===
DATA_DIR = '/content/frames'
BATCH_SIZE = 32
IMG_SIZE = 224
SEED = 42
N_SPLITS = 5
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("✅ Dispositivo en uso:", DEVICE)
# === TRANSFORMACIONES ===
transform = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
# === CARGA DEL DATASET ===
dataset = ImageFolder(DATA_DIR, transform=transform)
loader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=False)
class names = dataset.classes
# === MODELO RESNET18 (sin capa final) ===
resnet = models.resnet18(weights=models.ResNet18 Weights.DEFAULT).to(DEVICE)
resnet = torch.nn.Sequential(*(list(resnet.children())[:-1]))  # Quitar la capa FC
resnet.eval()
# === EXTRAER CARACTERÍSTICAS ===
features = []
labels = []
with torch.no grad():
    for inputs, targets in loader:
        inputs = inputs.to(DEVICE)
        output = resnet(inputs)
        output = output.view(output.size(0), -1)  # Aplanar [B, 512, 1, 1] -> [B, 512]
        features.append(output.cpu().numpy())
        labels.extend(targets.numpy())
features = np.concatenate(features)
labels = np.array(labels)
# === SEPARACIÓN DEL TEST SET EXTERNO ===
X kfold, X test, y kfold, y test = train test split(
    features, labels, test_size=0.2, stratify=labels, random_state=SEED
)
np.save('/content/X_test_B.npy', X_test)
np.save('/content/y test B.npy', y test)
print("💾 Test set externo guardado: X_test_B.npy y y_test_B.npy")
# === BÚSQUEDA DE HIPERPARÁMETROS (solo sobre el 80%) ===
X_train, _, y_train, _ = train_test_split(X_kfold, y_kfold, test_size=0.2,
stratify=y kfold, random state=SEED)
param_dist = {
    'n_estimators': [100, 200, 300, 500],
```

```
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt', 'log2']
}
search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=SEED, n_jobs=-1),
    param_distributions=param_dist,
    n_iter=10,
    scoring='accuracy',
    cv=StratifiedKFold(n_splits=2, shuffle=True, random_state=SEED),
    verbose=2,
    random_state=SEED,
    n_jobs=-1
)
search.fit(X train, y train)
best_params = search.best_params_
print("\n✅ Mejores hiperparámetros encontrados:")
for k, v in best_params.items():
    print(f"{k}: {v}")
# === VALIDACIÓN CON K-FOLD (solo en 80%) ===
final model = RandomForestClassifier(**best params, random state=SEED, n jobs=-1)
kfold = StratifiedKFold(n_splits=N_SPLITS, shuffle=True, random_state=SEED)
acc scores, f1 scores = [], []
fold_preds, fold_labels, all_probs = [], [], []
print("\n📊 Resultados por fold:")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_kfold, y_kfold)):
    X tr, X val = X kfold[train idx], X kfold[val idx]
    y_tr, y_val = y_kfold[train_idx], y_kfold[val_idx]
    final model.fit(X tr, y tr)
    preds = final_model.predict(X_val)
    probs = final model.predict proba(X val)
    fold_preds.extend(preds)
    fold labels.extend(y val)
    all_probs.extend(probs)
    acc = accuracy score(y val, preds)
    f1 = f1_score(y_val, preds, average='macro')
    acc scores.append(acc)
    f1_scores.append(f1)
    print(f"Fold {fold+1}: Accuracy = {acc:.4f}, F1 Macro = {f1:.4f}")
# === PROMEDIO DE MÉTRICAS K-FOLD ===
print(f"\n🎯 Accuracy promedio (5 folds): {np.mean(acc_scores):.4f}")
print(f"🎯 F1 Macro promedio (5 folds): {np.mean(f1_scores):.4f}")
# === GUARDADO DE MÉTRICAS GLOBALES ===
fold_preds = np.array(fold_preds)
fold_labels = np.array(fold_labels)
all_probs = np.array(all_probs)
accuracy = accuracy_score(fold_labels, fold_preds)
```

```
f1_macro = f1_score(fold_labels, fold_preds, average='macro')
precision_macro = precision_score(fold_labels, fold_preds, average='macro')
recall_macro = recall_score(fold_labels, fold_preds, average='macro')
roc_auc = roc_auc_score(fold_labels, all_probs, multi_class='ovr')
f1_weighted = f1_score(fold_labels, fold_preds, average='weighted')
precision_weighted = precision_score(fold_labels, fold_preds, average='weighted')
recall_weighted = recall_score(fold_labels, fold_preds, average='weighted')
conf_matrix = confusion_matrix(fold_labels, fold_preds)
np.savetxt("/content/MODELO_B_matriz_confusion.csv", conf_matrix, delimiter=",",
fmt="%d")
report_dict = classification_report(fold_labels, fold_preds, target_names=class_names,
output_dict=True)
report_df = pd.DataFrame(report_dict).transpose()
report_df.to_excel("/content/MODELO_B_reporte_por_clase.xlsx")
metrics_df = pd.DataFrame({
    "Métrica": [
        "Accuracy", "F1 Macro", "Precision Macro", "Recall Macro",
        "F1 Weighted", "Precision Weighted", "Recall Weighted", "ROC AUC (OVR)"
    ],
    "Valor": [
        accuracy, f1_macro, precision_macro, recall_macro,
        f1 weighted, precision weighted, recall weighted, roc auc
    ]
})
metrics_df.to_excel("/content/MODELO_B_metricas_globales.xlsx", index=False)
print("\n📊 Métricas globales guardadas exitosamente:")
print("- MODELO_B_metricas_globales.xlsx")
print("- MODELO B matriz confusion.csv")
print("- MODELO_B_reporte_por_clase.xlsx")
# === GUARDAR EL MODELO ENTRENADO ===
joblib.dump(final_model, '/content/MODELO_B_random_forest.pkl')
print("\n💾 Modelo Random Forest guardado exitosamente como
'MODELO_B_random_forest.pkl'")
with open('/content/MODELO B class names.json', 'w') as f:
    json.dump(class_names, f)
print("💾 Nombres de las clases guardados como 'MODELO B class names.json'")
```

```
# === CARGAR MATRIZ DE CONFUSIÓN GUARDADA ===
cm = np.loadtxt("/content/MODELO_B_matriz_confusion.csv", delimiter=",")
# === CARGAR NOMBRES DE CLASE ===
dataset = ImageFolder("/content/frames")
class_names = dataset.classes
# === GRAFICAR MATRIZ DE CONFUSIÓN (sin notación científica) ===
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap="Blues", xticks_rotation=45, values_format=".0f")
plt.title("Matriz de Confusión Global - MODELO B")
plt.grid(False)
plt.tight_layout()
```

```
plt.show()
```

```
# === Cargar matriz de confusión y reporte por clase ===

conf matrix = pd.read csv("/content/MODELO a matriz confusion.csv", header=None).values
reporte = pd.read_excel("/content/MODELO_B_reporte_por_clase.xlsx")
# === Reconstruir y true y y pred desde la matriz de confusión ===
y_true = []
y pred = []
for i in range(conf_matrix.shape[0]):
    for j in range(conf matrix.shape[1]):
        y_true.extend([i] * conf_matrix[i, j])
        y pred.extend([j] * conf matrix[i, j])
# === Binarizar etiquetas
n classes = conf matrix.shape[0]
y_true_bin = label_binarize(y_true, classes=list(range(n_classes)))
y pred bin = label binarize(y pred, classes=list(range(n classes)))
colors = ['blue', 'orange', 'green', 'red']
# === Graficar curvas ROC
plt.figure(figsize=(10, 6))
auc scores = []
for i in range(n_classes):
    fpr, tpr,   = roc curve(y true bin[:, i], y pred bin[:, i])
    auc_val = auc(fpr, tpr)
    auc_scores.append(auc_val)
    plt.plot(fpr, tpr, color=colors[i], label=f"Clase {i} (AUC={auc_val:.4f})")
# === Agregar línea aleatoria y detalles
plt.plot([0, 1], [0, 1], 'k--', label="Aleatorio")
plt.title(f"Curvas ROC por Clase - MODELO B\nAUC Macro Promedio:
{np.mean(auc_scores):.4f}")
plt.xlabel("Tasa de Falsos Positivos (FPR)")
plt.ylabel("Tasa de Verdaderos Positivos (TPR)")
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
# === Guardar como imagen
plt.savefig("/content/MODELO_B_curva_ROC.png")
plt.show()
```

MODELO C: HÍBRIDO VGG16 + RANDOM FOREST

```
# === CONFIGURACIÓN ===
DATA_DIR = '/content/frames'
BATCH_SIZE = 32
IMG_SIZE = 224
SEED = 42
N_SPLITS = 5
DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("✅ Dispositivo en uso:", DEVICE)
# === TRANSFORMACIONES ===
transform = transforms.Compose([
    transforms.Resize((IMG_SIZE, IMG_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
# === CARGA DEL DATASET ===
dataset = ImageFolder(DATA_DIR, transform=transform)
loader = DataLoader(dataset, batch_size=BATCH_SIZE, shuffle=False)
class names = dataset.classes
# === MODELO VGG16 (sin la última capa) ===
vgg = models.vgg16(weights=models.VGG16 Weights.DEFAULT).to(DEVICE)
vgg.classifier = torch.nn.Sequential(*list(vgg.classifier.children())[:-1])  # Quitar la
capa final FC
vgg.eval()
# === EXTRAER CARACTERÍSTICAS ===
features = []
labels = []
with torch.no_grad():
    for inputs, targets in loader:
        inputs = inputs.to(DEVICE)
        output = vgg(inputs)
        features.append(output.cpu().numpy())
        labels.extend(targets.numpy())
features = np.concatenate(features)
labels = np.array(labels)
# === SEPARACIÓN DEL TEST SET EXTERNO ===
X kfold, X test, y kfold, y test = train test split(
    features, labels, test_size=0.2, stratify=labels, random_state=SEED
)
np.save('/content/X_test_C.npy', X_test)
np.save('/content/y test C.npy', y test)
print("💾 Test set externo guardado: X_test_C.npy y y_test_C.npy")
# === BÚSQUEDA DE HIPERPARÁMETROS (solo sobre el 80%) ===
X_train, _, y_train, _ = train_test_split(X_kfold, y_kfold, test_size=0.2,
stratify=y kfold, random state=SEED)
param_dist = {
    'n_estimators': [100, 200, 300, 500],
```

```
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'max_features': ['sqrt', 'log2']
}
search = RandomizedSearchCV(
    estimator=RandomForestClassifier(random_state=SEED, n_jobs=-1),
    param_distributions=param_dist,
    n_iter=10,
    scoring='accuracy',
    cv=StratifiedKFold(n_splits=2, shuffle=True, random_state=SEED),
    verbose=2,
    random_state=SEED,
    n_jobs=-1
)
search.fit(X train, y train)
best_params = search.best_params_
print("\n Mejores hiperparámetros encontrados:")
for k, v in best_params.items():
    print(f"{k}: {v}")
# === VALIDACIÓN CON K-FOLD (solo en 80%) ===
final model = RandomForestClassifier(**best params, random state=SEED, n jobs=-1)
kfold = StratifiedKFold(n_splits=N_SPLITS, shuffle=True, random_state=SEED)
acc scores, f1 scores = [], []
fold_preds, fold_labels, all_probs = [], [], []
print("\n Resultados por fold:")
for fold, (train_idx, val_idx) in enumerate(kfold.split(X_kfold, y_kfold)):
    X tr, X val = X kfold[train idx], X kfold[val idx]
    y_tr, y_val = y_kfold[train_idx], y_kfold[val_idx]
    final model.fit(X tr, y tr)
    preds = final_model.predict(X_val)
    probs = final model.predict proba(X val)
    fold_preds.extend(preds)
    fold labels.extend(y val)
    all_probs.extend(probs)
    acc = accuracy score(y val, preds)
    f1 = f1_score(y_val, preds, average='macro')
    acc scores.append(acc)
    f1_scores.append(f1)
    print(f"Fold {fold+1}: Accuracy = {acc:.4f}, F1 Macro = {f1:.4f}")
# === PROMEDIO DE MÉTRICAS K-FOLD ===
print(f"\n Accuracy promedio (5 folds): {np.mean(acc_scores):.4f}")
print(f" F1 Macro promedio (5 folds): {np.mean(f1_scores):.4f}")
# === GUARDADO DE MÉTRICAS GLOBALES ===
fold_preds = np.array(fold_preds)
fold_labels = np.array(fold_labels)
all_probs = np.array(all_probs)
accuracy = accuracy_score(fold_labels, fold_preds)
```

```
f1_macro = f1_score(fold_labels, fold_preds, average='macro')
precision_macro = precision_score(fold_labels, fold_preds, average='macro')
recall_macro = recall_score(fold_labels, fold_preds, average='macro')
roc_auc = roc_auc_score(fold_labels, all_probs, multi_class='ovr')
f1_weighted = f1_score(fold_labels, fold_preds, average='weighted')
precision_weighted = precision_score(fold_labels, fold_preds, average='weighted')
recall_weighted = recall_score(fold_labels, fold_preds, average='weighted')
conf_matrix = confusion_matrix(fold_labels, fold_preds)
np.savetxt("/content/MODELO_C_matriz_confusion.csv", conf_matrix, delimiter=",",
fmt="%d")

report_dict = classification_report(fold_labels, fold_preds, target_names=class_names,
output_dict=True)
report_df = pd.DataFrame(report_dict).transpose()
report_df.to_excel("/content/MODELO_C_reporte_por_clase.xlsx")
metrics df = pd.DataFrame({
    "Métrica": [
        "Accuracy", "F1 Macro", "Precision Macro", "Recall Macro",
        "F1 Weighted", "Precision Weighted", "Recall Weighted", "ROC AUC (OVR)"
    ],
    "Valor": [
        accuracy, f1 macro, precision macro, recall macro,
        f1_weighted, precision_weighted, recall_weighted, roc_auc
    ]
})
metrics df.to excel("/content/MODELO C metricas globales.xlsx", index=False)
print("\n Métricas globales guardadas exitosamente:")
print("- MODELO C metricas globales.xlsx")
print("- MODELO_C_matriz_confusion.csv")
print("- MODELO C reporte por clase.xlsx")
# === GUARDAR EL MODELO ENTRENADO ===
joblib.dump(final model, '/content/MODELO C random forest.pkl')
print("\n Modelo Random Forest guardado exitosamente como
'MODELO C random forest.pkl'")
with open('/content/MODELO_C_class_names.json', 'w') as f:
    json.dump(class names, f)
print(" Nombres de las clases guardados como 'MODELO_C_class_names.json'")
```

```
# === CARGAR MATRIZ DE CONFUSIÓN GUARDADA ===
cm = np.loadtxt("/content/MODELO C matriz confusion.csv", delimiter=",")
# === CARGAR NOMBRES DE CLASE ===
dataset = ImageFolder("/content/frames")
class_names = dataset.classes
# === GRAFICAR MATRIZ DE CONFUSIÓN (sin notación científica) ===
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=class_names)
disp.plot(cmap="Blues", xticks_rotation=45, values_format=".0f")
plt.title("Matriz de Confusión Global - MODELO C")
plt.grid(False)
```

```
plt.tight_layout()
plt.show()
```

```
# === Cargar matriz de confusión y reporte por clase ===
conf_matrix = pd.read_csv("/content/MODELO_C_matriz_confusion.csv", header=None).values
reporte = pd.read_excel("/content/MODELO_C_reporte_por_clase.xlsx")
# === Reconstruir y_true y y_pred desde la matriz de confusión ===
y_true = []
y_pred = []
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        y_true.extend([i] * conf_matrix[i, j])  # Verdadero
        y_pred.extend([j] * conf_matrix[i, j])  # Predicho
# === Binarizar etiquetas
n_classes = conf_matrix.shape[0]
y true bin = label binarize(y true, classes=list(range(n classes)))
y_pred_bin = label_binarize(y_pred, classes=list(range(n_classes)))
colors = ['blue', 'orange', 'green', 'red']
# === Graficar curvas ROC
plt.figure(figsize=(10, 6))
auc_scores = []
for i in range(n classes):
    fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_pred_bin[:, i])
    auc val = auc(fpr, tpr)
    auc_scores.append(auc_val)
    plt.plot(fpr, tpr, color=colors[i], label=f"Clase {i} (AUC={auc val:.4f})")
# === Agregar línea aleatoria y detalles
plt.plot([0, 1], [0, 1], 'k--', label="Aleatorio")
plt.title(f"Curvas ROC por Clase - MODELO C\nAUC Macro Promedio:
{np.mean(auc scores):.4f}")
plt.xlabel("Tasa de Falsos Positivos (FPR)")
plt.ylabel("Tasa de Verdaderos Positivos (TPR)")
plt.legend(loc='lower right')
plt.grid(True)
plt.tight_layout()
# === Guardar como imagen
plt.savefig("/content/MODELO_C_curva_ROC.png")
plt.show()
```

6.- Visualización comparativa del rendimiento de los modelos

```
from tabulate import tabulate
# Rutas de tus archivos
archivos = {
    "MODELO_A": "/content/MODELO_A_metricas_globales.xlsx",
    "MODELO_B": "/content/MODELO_B_metricas_globales.xlsx",
    "MODELO_C": "/content/MODELO_C_metricas_globales.xlsx"
}
# Métricas deseadas
metricas_deseadas = [
    "Accuracy", "F1 Macro", "Precision Macro", "Recall Macro", "ROC AUC (OVR)"
]
# Recolectar métricas
filas = []
for nombre_modelo, ruta in archivos.items():
    df = pd.read_excel(ruta)
    df = df[df["Métrica"].isin(metricas_deseadas)]
    df = df.set_index("Métrica")["Valor"].to_dict()
    df_formateado = {m: round(df.get(m, 0), 4) for m in metricas_deseadas}
    df formateado["Modelo"] = nombre modelo
    filas.append(df_formateado)
# Crear DataFrame final y ordenar columnas
df_final = pd.DataFrame(filas)
df final = df final[["Modelo"] + metricas deseadas]
# Mostrar en consola con estilo
print("📊 Cuadro comparativo de métricas globales:\n")
print(tabulate(df_final, headers="keys", tablefmt="fancy_grid", showindex=False))
# Guardar en Excel
df_final.to_excel("/content/comparacion_modelos_metricas.xlsx", index=False)
print("\n📁 Archivo 'comparacion modelos metricas.xlsx' guardado con éxito.")
```

```
# === Rutas de archivos ya generados
archivos = {
    "MODELO A": "MODELO A reporte por clase.xlsx",
    "MODELO_B": "MODELO_B_reporte_por_clase.xlsx",
    "MODELO C": "MODELO C reporte por clase.xlsx"
}
# === Etiquetas de clase
etiquetas = ["inquietud", "movimientonormal", "parto", "tumbada"]
data = {"Clase": etiquetas}
# === Cargar F1-score por clase desde cada archivo
for nombre modelo, archivo in archivos.items():
    df = pd.read_excel(archivo, index_col=0)
    f1s = [round(df.loc[clase, "f1-score"], 4) for clase in etiquetas]
    data[nombre_modelo] = f1s
# === Crear tabla
tabla = pd.DataFrame(data)
# === Imprimir con formato elegante
```

```
print("\n📊 Comparación de F1-score por clase:\n")
print(tabulate(tabla, headers="keys", tablefmt="fancy_grid", showindex=False))
# === Guardar archivo Excel
tabla.to_excel("f1_por_clase_modelos.xlsx", index=False)
print("\n📁 Archivo 'f1_por_clase_modelos.xlsx' guardado con éxito.")
```

```
df = pd.read_excel("/content/comparacion_modelos_metricas.xlsx")
df.set_index("Modelo", inplace=True)
df.plot(kind='bar', figsize=(10, 6), rot=0)
plt.title("📊 Comparación de métricas globales por modelo")
plt.ylabel("Valor")
plt.ylim(0.98, 1.0)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
# 📥 Cargar el archivo Excel
df_f1 = pd.read_excel("/content/f1_por_clase_modelos.xlsx")
# 🧠 Usar la columna 'Clase' como índice
df_f1.set_index("Clase", inplace=True)
# 🔥 Visualizar como heatmap
plt.figure(figsize=(8, 5))
sns.heatmap(df_f1, annot=True, fmt=".4f", cmap="YlGnBu")
plt.title("🔥 F1-score por clase y modelo")
plt.tight_layout()
plt.show()
```

7.- Evaluación de modelos híbridos mediante 16 corridas sobre bloques balanceados

Evaluación del Modelo A (AlexNet + Random Forest)

```
# === CARGA DEL TEST SET DEL MODELO A ===
X_test = np.load('/content/X_test_A.npy')
y test = np.load('/content/y test A.npy')
# === CONFIGURACIÓN ===
n bloques = 17
clases = np.unique(y_test)
# === DETECTAR cuántas muestras tiene cada clase ===
conteo = {int(c): np.sum(y_test == c) for c in clases}
print("📊 Cantidad de muestras por clase:")
for k, v in conteo.items():
    print(f"  Clase {k}: {v} muestras")
# === Calcular cuántas usar por clase ===
por clase = min([conteo[c] // n bloques for c in clases])
print(f"\n📌 Se usarán {por_clase} muestras por clase por bloque ({por_clase *
n bloques} por clase en total).")
# === Extraer índices por clase según el límite calculado ===
idx por clase = {}
for c in clases:
    idx = np.where(y test == c)[0][:por clase * n bloques]
    np.random.shuffle(idx)
    idx por clase[c] = np.array split(idx, n bloques)
# === COMBINAR BLOQUES BALANCEADOS ===
bloques indices = []
conteo_por_bloque = []
for i in range(n bloques):
    idx_bloque = []
    for c in clases:
        idx_bloque.extend(idx_por_clase[c][i])
    np.random.shuffle(idx bloque)
    bloques_indices.append(np.array(idx_bloque))
    # Guardar archivos
    np.save(f"/content/A_X_bloque_{i+1}.npy", X_test[idx_bloque])
    np.save(f"/content/A_y_bloque_{i+1}.npy", y_test[idx_bloque])
    # Conteo por clase para graficar
    y_bloque = y_test[idx_bloque]
    clases_b, counts = np.unique(y_bloque, return_counts=True)
    for clase, count in zip(clases_b, counts):
        conteo_por_bloque.append({
            'Bloque': f'B{i+1}',
            'Clase': int(clase),
            'Cantidad': count
        })
print("\n✅ División y guardado completados con ajuste automático por clase.")
```

```
# === CREAR DATAFRAME PARA GRAFICAR Y EXPORTAR ===
df_balance = pd.DataFrame(conteo_por_bloque)
etiquetas_clases = {
    0: 'inquietud',
    1: 'movimientonormal',
    2: 'parto',
    3: 'tumbada'
}
df_balance['Clase'] = df_balance['Clase'].map(etiquetas_clases)
# === GUARDAR EXCEL CON DISTRIBUCIÓN ===
df_balance.to_excel('/content/resumen_balance_bloques_A.xlsx', index=False)
print("💾 Resumen guardado como 'resumen_balance_bloques_A.xlsx'")
# === GRAFICAR ===
plt.figure(figsize=(12, 6))
sns.barplot(data=df_balance, x='Bloque', y='Cantidad', hue='Clase')
plt.title('📊 Distribución de clases por bloque (modelo A)')
plt.ylabel('Número de muestras')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
# === CONFIGURACIÓN ===
nombre_modelo = "Modelo_A_RF"
modelo path = "/content/MODELO A random forest.pkl"  # Ruta al modelo A entrenado
prefijo = "A"  # Para bloques del modelo A
n bloques = 17
# === CARGA DE BLOQUES ===
X bloques, y bloques = [], []
for i in range(1, n_bloques + 1):
    X bloque = np.load(f"/content/{prefijo} X bloque {i}.npy")
    y_bloque = np.load(f"/content/{prefijo}_y_bloque_{i}.npy")
    X bloques.append(X bloque)
    y_bloques.append(y_bloque)
# === EVALUACIÓN DEL MODELO EN 16 CORRIDAS ===
resultados = []
for i in range(n bloques - 1):  # Solo 16 corridas
    X_train = np.concatenate([X_bloques[j] for j in range(n_bloques) if j != i])
    y train = np.concatenate([y bloques[j] for j in range(n bloques) if j != i])
    modelo = joblib.load(modelo_path)
    y pred = modelo.predict(X train)
    reporte = classification_report(y_train, y_pred, output_dict=True, zero_division=0)
    metrica = {
        "Corrida": i + 1,
        "Accuracy": reporte["accuracy"],
        "F1_macro": reporte["macro avg"]["f1-score"],
        "Precision_macro": reporte["macro avg"]["precision"],
        "Recall_macro": reporte["macro avg"]["recall"]
    }
```

```
    for clase in ['0', '1', '2', '3']:
        metrica[f"F1_clase_{clase}"] = reporte[clase]["f1-score"] if clase in reporte
else 0.0
    resultados.append(metrica)
# === EXPORTAR RESULTADOS ===
df_resultados = pd.DataFrame(resultados)
nombre_archivo = f"/content/resultados_{nombre_modelo.replace(' ', '_')}_bloques.xlsx"
df_resultados.to_excel(nombre_archivo, index=False)
print(f"\n✅ Resultados exportados a: {nombre_archivo}")
print("\n📊 Resultados por corrida:\n")
print(tabulate(df_resultados, headers="keys", tablefmt="grid", showindex=False,
floatfmt=".4f"))
```

Evaluación del Modelo B (ResNet18 + Random Forest)

```
# === CARGA DEL TEST SET DEL MODELO B ===
X test = np.load('/content/X test B.npy')
y_test = np.load('/content/y_test_B.npy')
# === CONFIGURACIÓN ===
n_bloques = 17
clases = np.unique(y_test)
# === DETECTAR cuántas muestras tiene cada clase ===
conteo = {int(c): np.sum(y_test == c) for c in clases}
print("📊 Cantidad de muestras por clase:")
for k, v in conteo.items():
    print(f"  Clase {k}: {v} muestras")
# === Calcular cuántas usar por clase ===
por_clase = min([conteo[c] // n_bloques for c in clases])
print(f"\n📌 Se usarán {por_clase} muestras por clase por bloque ({por_clase *
n_bloques} por clase en total).")
# === Extraer índices por clase según el límite calculado ===
idx_por_clase = {}
for c in clases:
    idx = np.where(y_test == c)[0][:por_clase * n_bloques]
    np.random.shuffle(idx)
    idx_por_clase[c] = np.array_split(idx, n_bloques)
# === COMBINAR BLOQUES BALANCEADOS ===
bloques_indices = []
conteo_por_bloque = []
for i in range(n_bloques):
    idx bloque = []
    for c in clases:
        idx bloque.extend(idx por clase[c][i])
    np.random.shuffle(idx_bloque)
    bloques indices.append(np.array(idx bloque))
    # Guardar archivos
    np.save(f"/content/B X bloque {i+1}.npy", X test[idx bloque])
    np.save(f"/content/B_y_bloque_{i+1}.npy", y_test[idx_bloque])
    # Conteo por clase para graficar
```

```
    y_bloque = y_test[idx_bloque]
    clases_b, counts = np.unique(y_bloque, return_counts=True)
    for clase, count in zip(clases_b, counts):
        conteo_por_bloque.append({
            'Bloque': f'B{i+1}',
            'Clase': int(clase),
            'Cantidad': count
        })
print("\n✅ División y guardado completados para modelo B.")
# === CREAR DATAFRAME PARA GRAFICAR Y EXPORTAR ===
df_balance = pd.DataFrame(conteo_por_bloque)
etiquetas_clases = {
    0: 'inquietud',
    1: 'movimientonormal',
    2: 'parto',
    3: 'tumbada'
}
df balance['Clase'] = df balance['Clase'].map(etiquetas clases)
# === GUARDAR EXCEL CON DISTRIBUCIÓN ===
df balance.to excel('/content/resumen balance bloques B.xlsx', index=False)
print("💾 Resumen guardado como 'resumen_balance_bloques_B.xlsx'")
# === GRAFICAR ===
plt.figure(figsize=(12, 6))
sns.barplot(data=df balance, x='Bloque', y='Cantidad', hue='Clase')
plt.title('📊 Distribución de clases por bloque (modelo B)')
plt.ylabel('Número de muestras')
plt.xticks(rotation=45)
plt.tight layout()
plt.show()
```

```
# === CONFIGURACIÓN ===
nombre modelo = "Modelo B RF"
modelo_path = "/content/MODELO_B_random_forest.pkl"  # Ruta al modelo B entrenado
prefijo = "B"  # Para bloques del modelo B
n_bloques = 17
# === CARGA DE BLOQUES ===
X_bloques, y_bloques = [], []
for i in range(1, n bloques + 1):
    X_bloque = np.load(f"/content/{prefijo}_X_bloque_{i}.npy")
    y bloque = np.load(f"/content/{prefijo} y bloque {i}.npy")
    X_bloques.append(X_bloque)
    y_bloques.append(y_bloque)
# === EVALUACIÓN DEL MODELO EN 16 CORRIDAS ===
resultados = []
for i in range(n_bloques - 1):  # Solo 16 corridas
    X_train = np.concatenate([X_bloques[j] for j in range(n_bloques) if j != i])
    y_train = np.concatenate([y_bloques[j] for j in range(n_bloques) if j != i])
    modelo = joblib.load(modelo_path)
```

```
    y_pred = modelo.predict(X_train)
    reporte = classification_report(y_train, y_pred, output_dict=True, zero_division=0)
    metrica = {
        "Corrida": i + 1,
        "Accuracy": reporte["accuracy"],
        "F1_macro": reporte["macro avg"]["f1-score"],
        "Precision_macro": reporte["macro avg"]["precision"],
        "Recall_macro": reporte["macro avg"]["recall"]
    }

    for clase in ['0', '1', '2', '3']:
        metrica[f"F1_clase_{clase}"] = reporte[clase]["f1-score"] if clase in reporte
else 0.0
    resultados.append(metrica)
# === EXPORTAR RESULTADOS ===
df resultados = pd.DataFrame(resultados)
nombre_archivo = f"/content/resultados_{nombre_modelo.replace(' ', '_')}_bloques.xlsx"
df resultados.to excel(nombre archivo, index=False)
print(f"\n✅ Resultados exportados a: {nombre_archivo}")
print("\n📊 Resultados por corrida:\n")
print(tabulate(df_resultados, headers="keys", tablefmt="grid", showindex=False,
floatfmt=".4f"))
```

Evaluación del Modelo C (VGG16 + Random Forest)

```
# === CARGA DEL TEST SET DEL MODELO C ===
X_test = np.load('/content/X_test_C.npy')
y_test = np.load('/content/y_test_C.npy')
# === CONFIGURACIÓN ===
n_bloques = 17
clases = np.unique(y_test)
# === DETECTAR cuántas muestras tiene cada clase ===
conteo = {int(c): np.sum(y_test == c) for c in clases}
print("📊 Cantidad de muestras por clase:")
for k, v in conteo.items():
    print(f"  Clase {k}: {v} muestras")
# === Calcular cuántas usar por clase ===
por_clase = min([conteo[c] // n_bloques for c in clases])
print(f"\n📌 Se usarán {por_clase} muestras por clase por bloque ({por_clase *
n_bloques} por clase en total).")
# === Extraer índices por clase según el límite calculado ===
idx_por_clase = {}
for c in clases:
    idx = np.where(y_test == c)[0][:por_clase * n_bloques]
    np.random.shuffle(idx)
    idx_por_clase[c] = np.array_split(idx, n_bloques)
# === COMBINAR BLOQUES BALANCEADOS ===
bloques_indices = []
conteo_por_bloque = []
```

```
for i in range(n_bloques):
    idx_bloque = []
    for c in clases:
        idx_bloque.extend(idx_por_clase[c][i])
    np.random.shuffle(idx_bloque)
    bloques_indices.append(np.array(idx_bloque))
    # Guardar archivos
    np.save(f"/content/C_X_bloque_{i+1}.npy", X_test[idx_bloque])
    np.save(f"/content/C_y_bloque_{i+1}.npy", y_test[idx_bloque])
    # Conteo por clase para graficar
    y_bloque = y_test[idx_bloque]
    clases_b, counts = np.unique(y_bloque, return_counts=True)
    for clase, count in zip(clases_b, counts):
        conteo_por_bloque.append({
            'Bloque': f'B{i+1}',
            'Clase': int(clase),
            'Cantidad': count
        })
print("\n✅ División y guardado completados para modelo C.")
# === CREAR DATAFRAME PARA GRAFICAR Y EXPORTAR ===
df_balance = pd.DataFrame(conteo_por_bloque)
etiquetas clases = {
    0: 'inquietud',
    1: 'movimientonormal',
    2: 'parto',
    3: 'tumbada'
}
df balance['Clase'] = df balance['Clase'].map(etiquetas clases)
# === GUARDAR EXCEL CON DISTRIBUCIÓN ===
df balance.to excel('/content/resumen balance bloques C.xlsx', index=False)
print("💾 Resumen guardado como 'resumen_balance_bloques_C.xlsx'")
# === GRAFICAR ===
plt.figure(figsize=(12, 6))
sns.barplot(data=df balance, x='Bloque', y='Cantidad', hue='Clase')
plt.title('📊 Distribución de clases por bloque (modelo C)')
plt.ylabel('Número de muestras')
plt.xticks(rotation=45)
plt.tight layout()
plt.show()
```

```
# === CONFIGURACIÓN ===
nombre_modelo = "Modelo_C_RF"
modelo_path = "/content/MODELO_C_random_forest.pkl"  # Cambia si tu modelo se llama
diferente
prefijo = "C"  # Cambia a A o B según bloques que quieras evaluar
n_bloques = 17
# === CARGA DE BLOQUES ===
X_bloques, y_bloques = [], []
```

```
for i in range(1, n_bloques + 1):
    X_bloque = np.load(f"/content/{prefijo}_X_bloque_{i}.npy")
    y_bloque = np.load(f"/content/{prefijo}_y_bloque_{i}.npy")
    X_bloques.append(X_bloque)
    y_bloques.append(y_bloque)
# === EVALUACIÓN DEL MODELO ===
resultados = []
for i in range(n_bloques - 1):
    # Usar todos los bloques menos el bloque i
    X_train = np.concatenate([X_bloques[j] for j in range(n_bloques) if j != i])
    y_train = np.concatenate([y_bloques[j] for j in range(n_bloques) if j != i])
    modelo = joblib.load(modelo_path)
    y_pred = modelo.predict(X_train)
    reporte = classification_report(y_train, y_pred, output_dict=True, zero_division=0)
    metrica = {
        "Corrida": i + 1,
        "Accuracy": reporte["accuracy"],
        "F1 macro": reporte["macro avg"]["f1-score"],
        "Precision_macro": reporte["macro avg"]["precision"],
        "Recall macro": reporte["macro avg"]["recall"]
    }
    for clase in ['0', '1', '2', '3']:
        metrica[f"F1_clase_{clase}"] = reporte[clase]["f1-score"] if clase in reporte
else 0.0
    resultados.append(metrica)
# === EXPORTAR RESULTADOS ===
df_resultados = pd.DataFrame(resultados)
nombre archivo = f"/content/resultados {nombre modelo.replace(' ', ' ')} bloques.xlsx"
df_resultados.to_excel(nombre_archivo, index=False)
print(f"\n✅ Resultados exportados a: {nombre archivo}")
print("\n📊 Resultados por corrida:\n")
print(tabulate(df resultados, headers="keys", tablefmt="grid", showindex=False,
floatfmt=".4f"))
```

MARC ANTHONI REATEGUI SIFUENTES

Monitoreo y predicción del momento del parto de cerdas mediante modelos de visión artificial basado en CNN

Revisión de Tesis final - Unidad de Investigación FISI

Detalles del documento

Identificador de la entrega
trn:oid:::3117:586610470

Fecha de entrega
5 may 2026, 14:27 GMT-5

Fecha de descarga
5 may 2026, 14:35 GMT-5

Nombre del archivo
Informe de Tesis - Marc Anthoni (1).pdf

Tamaño del archivo
1.5 MB

99 páginas

24.709 palabras

146.270 caracteres

9% Similitud general

El total combinado de todas las coincidencias, incluidas las fuentes superpuestas, para ca...

Filtrado desde el informe

- Bibliografía
- Texto citado
- Texto mencionado
- Coincidencias menores (menos de 10 palabras)

Fuentes principales

7% Fuentes de Internet
2% Publicaciones
7% Trabajos entregados (trabajos del estudiante)

Marcas de integridad

N.º de alertas de integridad para revisión

Los algoritmos de nuestro sistema analizan un documento en profundidad para buscar inconsistencias que permitirían distinguirlo de una entrega normal. Si advertimos algo extraño, lo marcamos como una alerta para que pueda revisarlo.

Una marca de alerta no es necesariamente un indicador de problemas. Sin embargo, recomendamos que preste atención y la revise.