Universidad del Desarrollo
Facultad de Ingeniería

SISTEMA DE DETECCIÓN AUTOMÁTICA DE NUDOS QUIRÚRGICOS EN REALIDAD VIRTUAL MEDIANTE APRENDIZAJE PROFUNDO

Desarrollo e integración de un modelo de visión por computadora en el simulador SECMA

POR: PABLO MADARIAGA ORELLANA

Proyecto de grado presentado a la Facultad de Ingeniería de la Universidad del Desarrollo para optar al grado académico de Magíster en Data Science

PROFESORES GUÍA:

Dr. JOSÉ GUZMÁN MONTOTO,

Dr. MAURICIO HERRERA MARÍN

Diciembre 2025

SANTIAGO

Para mis hijos, Antonia y Facundo.

AGRADECIMIENTO

Durante el desarrollo del proyecto fue necesario capturar múltiples sesiones en el simulador SECMA y realizar ajustes progresivos al pipeline frente a desafíos no previstos. Cada iteración permitió mejorar la calidad del dataset y fortalecer la metodología aplicada. Se agradece especialmente al profesor José por su constante apoyo en la captura de imágenes, sin el cual no habría sido posible consolidar el dataset ni avanzar de manera consistente en el desarrollo del proyecto, logrando cumplir los objetivos propuestos.

TABLA DE CONTENIDO

TABLA FIGURAS

TABLAS

Resumen

El entrenamiento en sutura y anudado quirúrgico constituye una habilidad esencial en la formación médico-quirúrgica, pero los métodos tradicionales presentan limitaciones importantes en repetibilidad, estandarización y retroalimentación objetiva. La realidad virtual (RV) ha emergido como una alternativa eficaz para superar estas restricciones, permitiendo la práctica en entornos inmersivos, controlados y libres de riesgo. Dentro de este contexto, el simulador SECMA, desarrollado por la Universidad del Desarrollo, incluye un módulo de sutura que, hasta ahora, carecía de un mecanismo autónomo capaz de evaluar en tiempo real la formación del nudo quirúrgico.

El presente proyecto desarrolla e implementa un sistema de clasificación automática de los estados del nudo (NO_NUDO, UN_NUDO, DOS_NUDO) a partir de imágenes generadas durante la interacción del usuario en RV. Para ello se construyó un pipeline integral que abarca la captura de datos en SECMA, la depuración exhaustiva de las imágenes, la eliminación de duplicados perceptuales, la conversión homogénea de formatos, el balanceo de clases y la selección de un conjunto final robusto (Dataset 4). Este dataset consolidado permitió entrenar modelos de visión por computador mediante aprendizaje profundo, empleando arquitecturas preentrenadas y técnicas de fine-tuning progresivo.

El modelo final, basado en EfficientNet-B0, alcanzó un desempeño sobresaliente (F1-macro = 0.9822), con estabilidad entre los conjuntos de validación y prueba, y una separación precisa entre las clases incluso en escenarios visualmente similares. Posteriormente, el modelo fue exportado a formato ONNX y su equivalencia con PyTorch fue verificada, garantizando compatibilidad con Barracuda, motor de inferencia de Unity, para su ejecución en tiempo real dentro del simulador. Las pruebas operacionales en Meta Quest 3 confirmaron un funcionamiento fluido, con inferencias continuas sin afectar la tasa de refresco (>50 FPS).

Se demuestra que es posible integrar un sistema de visión por computadora en un entorno de RV inmersiva para el reconocimiento de los estados de anudado y brindar una retroalimentación automática y objetiva durante el proceso. Este trabajo constituye un avance relevante hacia simuladores quirúrgicos inteligentes capaces de apoyar la enseñanza autónoma, estandarizar la evaluación del desempeño y escalar hacia aplicaciones en simulación física y escenarios del mundo real.

1. Introducción

El entrenamiento en suturas y anudado quirúrgico constituye una competencia esencial en la educación médico-quirúrgica, dado su impacto directo en la estabilidad del cierre tisular, la prevención de complicaciones postoperatorias y el éxito global de los procedimientos operatorios. Los métodos tradicionales de enseñanza —basados en modelos físicos, práctica animal o tutorías presenciales— presentan limitaciones relevantes en términos de repetibilidad, estandarización y disponibilidad docente. En este contexto, la realidad virtual (RV) ha emergido como una alternativa pedagógica de alto valor, al permitir la práctica en entornos inmersivos, controlados y libres de riesgo, favoreciendo una adquisición más eficiente y consistente de habilidades técnicas.

Entre 2023 y 2025, diversas investigaciones han demostrado que los simuladores basados en RV mejoran de forma significativa la precisión, estabilidad y coherencia de las técnicas de sutura, especialmente cuando se incorporan mecanismos de retroalimentación háptica y analítica automatizada. La literatura también evidencia que estos sistemas permiten evaluar variables críticas del anudado —como la fuerza aplicada, el comportamiento del hilo bajo tensión o la secuencia de ejecución— entregando retroalimentación inmediata que supera las capacidades de la observación tradicional [1]. Paralelamente, se han desarrollado enfoques de diseño centrados en el usuario, como el *bodystorming* híbrido, con el objetivo de asegurar que las experiencias simuladas reproduzcan fielmente los pasos técnicos de la práctica clínica real [2] [3].

A nivel aplicado, plataformas comerciales como Osso VR, FundamentalVR y VirtaMed LaparoS han elevado el estándar del entrenamiento quirúrgico mediante el uso de motores gráficos avanzados, dispositivos hápticos y sistemas de evaluación objetiva del desempeño, demostrando mejoras en tiempos operatorios y reducciones en la tasa de errores [6]. Asimismo, frameworks abiertos como iMSTK han permitido simular con alta fidelidad la dinámica del hilo quirúrgico mediante técnicas de *Position-Based Dynamics*, democratizando el desarrollo de simuladores complejos y de alto realismo [9]. Estas capacidades se integran habitualmente con motores como Unity y Unreal Engine, ampliamente utilizados en simulación médica por su capacidad de combinar realidad virtual, físicas avanzadas y análisis automatizado [7] [8]. Más recientemente, estudios que incorporan inteligencia artificial en entornos de realidad virtual han reportado incrementos de hasta un 42 % en la precisión procedimental y reducciones cercanas al 45 % en errores técnicos; sin embargo, estos resultados se circunscriben principalmente a sistemas de retroalimentación háptica basados en actuadores de fuerza y dispositivos especializados, como interfaces tipo Phantom, y no a sistemas de evaluación visual automática del estado del nudo quirúrgico [10].

En este contexto de avances tecnológicos se enmarca el presente proyecto, desarrollado sobre el simulador **SECMA** (Sistema portátil de entrenamiento para cirugías de mínimo acceso), una plataforma de realidad virtual orientada a la enseñanza de técnicas quirúrgicas y ejecutada en dispositivos Meta Quest, lo que posibilita un entrenamiento

portátil, accesible y altamente inmersivo. **Este simulador, para el ejercicio de sutura, necesita un sistema de detección de nudos que identifique el proceso de anudado, la formación de la primer lazo y segundo lazo**, ya que actualmente no cuenta con un mecanismo autónomo capaz de reconocer y evaluar dichas etapas durante la ejecución del procedimiento.

A futuro, resulta además relevante avanzar hacia la identificación automática del cumplimiento correcto de las distintas fases del proceso de sutura y la incorporación de métricas objetivas tales como el tiempo total de ejecución y la cantidad de errores cometidos. Si bien existen sistemas de detección de nudos, calidad y fases de la sutura basados en visión por computador, estos se han desarrollado principalmente para simulaciones con maquetas físicas. En el caso de simuladores completamente virtuales, predominan enfoques numéricos basados en la geometría, tensión y topología del hilo, los cuales dependen fuertemente de la complejidad de la simulación física y pueden afectar el desempeño general del sistema.

Para abordar esta limitación, en este trabajo se implementó un sistema completo basado en visión por computadora y aprendizaje profundo que permitió detectar y clasificar automáticamente los estados del nudo quirúrgico (**NO_NUDO**, **UN_NUDO** y **DOS_NUDO**) a partir de imágenes generadas durante la práctica en realidad virtual. Para ello, se desarrolló un pipeline integral que incluyó la creación y curación exhaustiva de un dataset especializado capturado directamente desde SECMA, la eliminación de redundancias mediante perceptual hashing, el entrenamiento de modelos convolucionales optimizados y su validación mediante métricas estándar de desempeño. Posteriormente, el modelo seleccionado fue exportado a formato ONNX e integrado en Unity mediante el motor Barracuda, permitiendo realizar inferencia en tiempo real dentro del propio simulador en Meta Quest.

En este contexto, el aporte del proyecto no se limita a la implementación puntual de un sistema de clasificación, sino que se concreta en la definición de una metodología reproducible para la detección automática de nudos quirúrgicos en simulaciones de realidad virtual, aplicada al entorno SECMA. Dicha metodología articula criterios para la captura de imágenes durante la simulación, estrategias sistemáticas de trabajo con los datos (incluyendo curación, control de redundancia y validación) y un esquema de integración del modelo en el entorno de realidad virtual que define claramente el flujo de información desde la adquisición de la imagen hasta la visualización del resultado.

De este modo, el trabajo establece un marco técnico que permite evaluar objetivamente el desempeño del aprendiz y habilitar retroalimentación automática en tiempo real, aportando una base estructurada y escalable para el desarrollo de simuladores quirúrgicos inteligentes, particularmente en etapas del entrenamiento quirúrgico asociadas al análisis del movimiento de manos e instrumentos, cuya modelación mediante enfoques tradicionales (basados en reglas, conteo de interacciones, cálculos espaciales o detección

explícita de colisiones) resulta compleja y puede afectar el desempeño en tiempo real del simulador.

El documento se organiza de la siguiente manera. En la **Sección 2** se presenta el Trabajo Relacionado, donde se revisan los fundamentos teóricos y el estado del arte en simulación quirúrgica, realidad virtual e inteligencia artificial aplicada al entrenamiento en sutura. La **Sección 3** expone la hipótesis de investigación y los objetivos general y específicos que guían el desarrollo del proyecto. En la **Sección 4** se describen los datos utilizados y la metodología empleada y su integración en el simulador. La **Sección 5** presenta la captura, curación del dataset, entrenamiento de los modelos, los resultados obtenidos y su análisis. Finalmente, la **Sección 6** resume las conclusiones del trabajo y discute posibles líneas de desarrollo futuro.

2. Trabajo Relacionado

El trabajo relacionado se organiza en dos ejes complementarios. El **marco teórico** describe los fundamentos esenciales sobre realidad virtual, visión por computador y aprendizaje profundo aplicados al entrenamiento quirúrgico. El **estado del arte** aborda las principales soluciones contemporáneas (académicas y comerciales) en simuladores inteligentes, destacando la tendencia hacia sistemas capaces de evaluar de forma automática y objetiva el desempeño del usuario. Luego, estas perspectivas sitúan el presente proyecto dentro de la evolución reciente de la simulación médica asistida por tecnologías inmersivas y algoritmos de inteligencia artificial.

2.1 Marco Teórico

2.1.1 Realidad Virtual en la Formación Quirúrgica

La realidad virtual (RV) se define, de manera general, como una simulación generada por computador que emplea gráficos tridimensionales en tiempo real, junto con dispositivos de interacción, para inducir la sensación de inmersión del usuario y permitir su interacción activa con un entorno artificial. Distintas aproximaciones teóricas coinciden en estos elementos centrales: inmersión, interacción y simulación computacional.

En el ámbito de la simulación para entrenamiento médico, la RV constituye actualmente una de las áreas de mayor desarrollo, debido a su capacidad para reproducir procedimientos complejos en entornos controlados, repetibles y libres de riesgo para el paciente. A diferencia de los métodos tradicionales basados en modelos físicos o en la práctica clínica supervisada, la RV permite estandarizar escenarios, ajustar progresivamente el nivel de dificultad y ofrecer retroalimentación inmediata, favoreciendo un aprendizaje estructurado y autónomo.

Revisiones recientes reportan mejoras consistentes en la adquisición de habilidades psicomotoras en cirugía, particularmente en sutura laparoscópica y técnicas de anudado intracorpóreo, destacando la utilidad de la RV para el entrenamiento progresivo y estructurado de destrezas complejas [1]. Asimismo, se ha observado que la incorporación de retroalimentación háptica puede aumentar el realismo del procedimiento y mejorar la precisión del aprendiz, especialmente en tareas que involucran manipulación fina de instrumentos [1]. En conjunto, la RV se posiciona como una plataforma educativa que favorece la práctica deliberada, la corrección inmediata y el aprendizaje autónomo.

2.1.2 Aprendizaje de Suturas y Nudos Quirúrgicos

La técnica de sutura constituye una habilidad fundamental en múltiples especialidades quirúrgicas. La correcta ejecución del anudado determina la estabilidad mecánica del

cierre tisular, la resistencia a la tensión y la prevención de complicaciones postoperatorias. El aprendizaje de esta técnica requiere no solo destreza manual, sino también el dominio de una secuencia precisa de movimientos que garantice la formación de nudos seguros y consistentes.

La literatura evidencia que los simuladores basados en RV orientados a sutura permiten acelerar la curva de aprendizaje mediante la repetición de patrones estandarizados, la corrección de errores técnicos y la evaluación cuantitativa del progreso [1]. Desde una perspectiva computacional, un nudo quirúrgico puede analizarse como una configuración geométrica compleja del hilo, caracterizada por cruces, orientación relativa de los extremos, tensión aplicada y secuencialidad temporal. Por esta razón, los sistemas modernos integran visión por computador, simulación física y algoritmos especializados para detectar si un nudo ha sido correctamente completado [1].

2.1.3 Visión por Computador y Aprendizaje Profundo en Procedimientos Quirúrgicos

La visión por computador ha adquirido un rol central en la automatización de tareas quirúrgicas y en la evaluación objetiva del desempeño técnico. En el contexto del entrenamiento en sutura, los modelos de aprendizaje profundo permiten analizar imágenes o secuencias visuales generadas en el simulador para clasificar estados del procedimiento y detectar errores de ejecución en tiempo real.

Las redes neuronales convolucionales (CNN) destacan por su capacidad de extraer patrones visuales complejos, siempre que se disponga de un pipeline robusto que incluya curación del dataset, normalización de imágenes, entrenamiento supervisado y evaluación mediante métricas estándar. La literatura señala que la convergencia entre RV y modelos de IA habilita plataformas adaptativas capaces de personalizar la retroalimentación y ajustar dinámicamente la dificultad según el desempeño del aprendiz [10].

2.1.4 Motores de Simulación y Plataformas XR en Cirugía

El desarrollo de simuladores quirúrgicos depende en gran medida de motores gráficos capaces de representar interacciones complejas en tiempo real. Entre ellos, Unity y Unreal Engine se han consolidado como las plataformas más utilizadas tanto en investigación académica como en desarrollos comerciales.

Unity se ha convertido en un estándar de facto para simulación quirúrgica debido a su soporte nativo para RV y XR (Realidad Extendida), su ecosistema extensible y su compatibilidad con modelos ONNX mediante el framework Barracuda. Estudios recientes demuestran su eficacia en simuladores laparoscópicos de bajo costo con niveles avanzados de realismo e interacción [7]. Por su parte, Unreal Engine se emplea cuando la prioridad es el ultra-realismo gráfico y la simulación detallada de tejidos, como en plataformas

comerciales que han reportado mejoras sustantivas en la adquisición de habilidades operatorias [8].

2.1.5 Simulación Física del Hilo Quirúrgico

La simulación del hilo quirúrgico constituye uno de los desafíos computacionales más complejos en sutura, debido a su flexibilidad, auto-colisión y comportamiento dependiente de tensión y fricción. Frameworks abiertos como el Interactive Medical Simulation Toolkit (iMSTK) han abordado este problema mediante modelos basados en Position-Based Dynamics (PBD), permitiendo simular en tiempo real la interacción aguja-tejido, la penetración tisular, la tensión del hilo y las colisiones continuas [9].

Sobre esta base, se han propuesto métodos algorítmicos para la detección automática de nudos que analizan cruces entre segmentos o partículas del hilo y eventos de auto-colisión, aprovechando el acceso directo al estado interno de la simulación. Estos enfoques han sido aplicados principalmente en simuladores de realidad virtual con modelado físico explícito del hilo, y difieren de los métodos utilizados en entrenamiento sobre maquetas físicas, donde no existe un modelo computacional del hilo y la evaluación debe basarse en observaciones externas.

2.1.6 Evaluación Objetiva y Analítica del Desempeño Quirúrgico

La evaluación tradicional de la sutura ha dependido históricamente de la observación subjetiva del instructor. En contraste, los simuladores modernos integran analíticas automáticas que cuantifican variables como la trayectoria instrumental, el tiempo por nudo, la tensión aplicada y la presencia de errores críticos.

Plataformas comerciales y académicas han demostrado que la integración de métricas objetivas mejora la consistencia de la evaluación y reduce la carga docente [3]. Estudios recientes indican que la combinación de RV con analítica basada en IA y retroalimentación háptica puede mejorar significativamente la precisión procedimental y reducir tasas de error; sin embargo, estos resultados se asocian principalmente a simuladores con actuadores de fuerza y hardware especializado, lo que limita su generalización a entornos más livianos [10].

2.1.7 Integración de Modelos ONNX en Motores de Realidad Virtual

Para ejecutar modelos de visión por computador en tiempo real dentro de un simulador, es necesario utilizar formatos interoperables. ONNX permite transferir modelos entrenados en PyTorch (en nuestro caso) hacia entornos optimizados para ejecución gráfica. En Unity, el motor Barracuda interpreta modelos ONNX, permitiendo realizar inferencia continua y activar retroalimentación visual o háptica dentro del entorno

inmersivo. Esta integración facilita el desarrollo de simuladores quirúrgicos inteligentes capaces de evaluar la ejecución del nudo mientras el estudiante realiza la tarea.

2.2 Estado del Arte

La evolución reciente de la simulación quirúrgica se caracteriza por la integración progresiva de tecnologías inmersivas, modelos físicos de alta fidelidad y algoritmos de inteligencia artificial. Entre 2023 y 2025, el área experimentó un crecimiento acelerado, con avances significativos en tres frentes: investigación académica, plataformas comerciales y herramientas tecnológicas abiertas para simulación.

2.2.1 Avances Académicos Recientes en Simulación de Sutura en RV

La literatura reciente confirma la efectividad de la RV en la enseñanza de sutura y anudado quirúrgico, con mejoras consistentes en psicomotricidad fina y precisión técnica [1]. Asimismo, enfoques de diseño centrados en el usuario, como el bodystorming híbrido, que es una simulación o representación física de situaciones reales (movimientos, gestos, interacciones espaciales), mientras interactúan simultáneamente con soportes tecnológicos como prototipos digitales, realidad virtual/aumentada. Este ha sido utilizado para garantizar que las experiencias XR reproduzcan fielmente la ejecución clínica y resulten pedagógicamente efectivas [2]. Un aspecto clave del avance académico ha sido la transición desde simuladores puramente visuales hacia sistemas capaces de evaluar de forma objetiva la técnica del usuario.

2.2.2 Plataformas Comerciales y Simuladores Clínicos de Última Generación

Plataformas como Osso VR, FundamentalVR y VirtaMed LaparoS representan el estándar industrial en simulación de sutura y procedimientos relacionados.

- **Osso VR** ha demostrado mejoras en tiempos operatorios y reducciones en errores mediante evaluación automática basada en métricas objetivas [3].
- **FundamentalVR** incorpora dispositivos hápticos de grado quirúrgico que replican la resistencia tisular con alta precisión [4].
- **LaparoS (VirtaMed)** combina RV con instrumentos físicos y analíticas avanzadas sobre gemelos digitales de tejidos [5][6].

Estas plataformas constituyen referencias tecnológicas relevantes para el desarrollo de simuladores accesibles y de menor costo como SECMA [11].

2.2.3 Simulación de Bajo Costo y Prototipos Académicos con Unity y Unreal

El uso de Unity y Unreal Engine ha permitido desarrollar simuladores anatómicos de bajo presupuesto con capacidades avanzadas de interacción. Investigaciones recientes han demostrado que es posible integrar sutura, háptica y análisis automatizado sin hardware especializado, abriendo el acceso a instituciones con recursos limitados [7].

2.2.4 Detección Algorítmica de Nudos basada en la Geometría del Hilo

Además de los enfoques basados en visión por computador y aprendizaje profundo, la literatura reporta métodos específicos para la detección automática de nudos quirúrgicos que se apoyan directamente en la geometría interna del hilo simulado. Estos enfoques asumen un entorno de realidad virtual con modelado físico explícito del hilo, lo que permite acceder al estado interno de la simulación y analizar relaciones geométricas entre segmentos o partículas.

Un trabajo pionero en esta línea es el de Sankaranarayanan et al [12],quienes propusieron un método para la detección y congelamiento en tiempo real de nudos quirúrgicos dentro de un simulador de sutura en realidad virtual. El algoritmo identifica la formación de un nudo mediante el análisis de auto-colisiones entre segmentos del hilo, detectando configuraciones cerradas estables que corresponden a nudos válidos. Una vez identificado, el nudo puede ser congelado para su evaluación, permitiendo una interacción consistente incluso bajo restricciones de tiempo real. Este enfoque no utiliza información visual, sino que depende completamente del acceso al modelo físico interno del hilo y a los eventos de colisión generados durante la simulación.

Trabajos posteriores, como el de Qi et al [13], han refinado esta idea mediante algoritmos geométricos más eficientes, capaces de detectar nudos a partir de intersecciones proyectadas entre segmentos del hilo, diseñados específicamente para operar a tasas compatibles con retroalimentación háptica. Estos métodos priorizan la eficiencia computacional, dado que la detección de múltiples cruces y colisiones simultáneas puede convertirse en un cuello de botella durante la simulación interactiva.

Es importante destacar que estos enfoques, aunque efectivos en simuladores de realidad virtual que cuentan con un modelo físico detallado del hilo, no pueden aplicarse directamente en escenarios donde dicho modelo interno no está disponible. Esto ocurre, por ejemplo, en entrenamientos realizados sobre maquetas físicas (box trainers) o en simuladores livianos que no incorporan simulación avanzada de la dinámica del hilo. En estos casos, el sistema no tiene acceso a información geométrica interna como cruces entre segmentos, colisiones o tensiones, por lo que la evaluación automática debe basarse en observaciones externas, típicamente mediante el análisis de imágenes o video.

2.2.5 Frameworks de Simulación de Hilos Quirúrgicos y Tejidos Blandos

Frameworks abiertos como iMSTK han permitido simular con alta fidelidad el comportamiento del hilo quirúrgico y los tejidos blandos, utilizando modelos físicos avanzados que capturan colisiones, fricción y tensión en tiempo real [9]. Sobre estas bases, se han desarrollado métodos de detección de nudos basados en el análisis geométrico de cruces del hilo y eventos de auto-colisión, aplicables principalmente en simuladores de RV con acceso al estado interno de la simulación. Estos enfoques contrastan con los utilizados en entrenamiento sobre maquetas físicas, donde la evaluación automática suele basarse en visión por computador aplicada a imágenes externas.

2.2.6 Inteligencia Artificial y Evaluación Automática

La incorporación de IA en simuladores quirúrgicos constituye una tendencia creciente. Modelos de visión por computador permiten reconocer errores técnicos, clasificar estados del nudo y analizar secuencias completas de movimientos. Ensayos recientes demuestran incrementos significativos de precisión y disminuciones en tasas de error cuando se integran modelos adaptativos dentro de la experiencia de entrenamiento [10].

2.2.7 Evaluación Objetiva del Nudo Quirúrgico

Los sistemas modernos de evaluación objetiva del nudo quirúrgico incorporan métricas automáticas que cuantifican completitud, tensión, secuencia técnica y errores críticos, reduciendo la dependencia de evaluadores humanos [3]. No obstante, la mayoría de estas soluciones se apoyan en sensores físicos o en el estado interno del simulador. En este contexto, los enfoques basados en visión por computador integrados directamente en entornos de RV representan una alternativa prometedora para simuladores de bajo costo, como SECMA, permitiendo clasificar estados del nudo a partir de información visual sin requerir hardware especializado.

2.3 Síntesis del capitulo

La revisión del marco teórico y del estado del arte muestra que la realidad virtual se ha consolidado como una herramienta eficaz para el entrenamiento quirúrgico, y que la incorporación de técnicas de aprendizaje profundo ha permitido avanzar hacia simuladores capaces de analizar automáticamente el desempeño del aprendiz. Sin embargo, gran parte de las soluciones existentes dependen de modelos físicos detallados, información interna del simulador o instrumentación adicional, lo que limita su implementación en entornos de realidad virtual livianos y reduce su escalabilidad.

En particular, la evaluación automática del anudado quirúrgico continúa siendo un desafío abierto cuando solo se dispone de información visual generada por el motor gráfico. Hasta donde alcanza la bibliografía revisada, no se han identificado trabajos que aborden de manera explícita la detección automática de estados del nudo quirúrgico mediante técnicas

de visión por computador integradas directamente en un entorno de realidad virtual operativo. Esta brecha justifica la necesidad de investigar enfoques basados exclusivamente en visión por computador y aprendizaje profundo, capaces de operar en tiempo real e integrarse directamente en simuladores de RV. En este contexto, la presente investigación propone el desarrollo de un sistema automático de detección de estados del nudo quirúrgico en el simulador SECMA, contribuyendo a una evaluación objetiva, reproducible y autónoma del entrenamiento en sutura.

3. Hipótesis y Objetivos

3.1 Hipótesis

Se plantea como hipótesis que la integración de modelos de visión por computador basados en aprendizaje profundo dentro de un entorno de realidad virtual es capaz de identificar de manera automática y en tiempo real el estado del nudo quirúrgico, específicamente, las clases **NO_NUDO**, **UN_NUDO** y **DOS_NUDO**, a partir de imágenes capturadas por el simulador SECMA. Esta capacidad permitiría entregar retroalimentación objetiva, inmediata y consistente durante el proceso formativo, fortaleciendo la adquisición de habilidades de anudado.

De esta hipótesis central se desprende un supuesto metodológico complementario: el desempeño del sistema depende de forma crítica de la calidad del dataset utilizado para el entrenamiento. Dado que las imágenes provienen de un entorno RV que tiende a generar alta redundancia temporal, variabilidad visual limitada y artefactos propios del motor gráfico (como la presencia de canales alfa en formato RGBA), se postula que un pipeline riguroso de curación de datos, que incluya eliminación de duplicados, normalización de formatos y control de sesgos, es indispensable para obtener modelos robustos y estables. Solo bajo estas condiciones es posible garantizar que el modelo mantenga un rendimiento coherente al ser exportado a ONNX y ejecutado en tiempo real mediante Unity Barracuda.

3.2 Objetivo General

Desarrollar un sistema automático de clasificación del estado del nudo quirúrgico para el simulador SECMA en realidad virtual, utilizando técnicas de aprendizaje profundo y un pipeline de procesamiento diseñado específicamente para datos generados en entornos RV. El sistema debe ser capaz de mantener coherencia entre su desempeño offline y su ejecución en tiempo real dentro de Unity, con el fin de apoyar la enseñanza y evaluación del proceso de anudado mediante retroalimentación inmediata.

3.3 Objetivos Específicos

Los objetivos específicos definen las etapas necesarias para desarrollar e integrar el sistema propuesto, desde la obtención de datos en realidad virtual hasta la ejecución del modelo de clasificación en tiempo real dentro del simulador. A continuación, se presentan los objetivos específicos del proyecto:

1. **Caracterizar** las imágenes del proceso de anudado quirúrgico generadas en el entorno de realidad virtual SECMA, considerando su variabilidad visual, estructura, redundancia y distribución de clases.

2. **Evaluar** modelos de clasificación basados en aprendizaje profundo para la identificación de los estados **NO_NUDO, UN_NUDO y DOS_NUDO**, utilizando métricas estándar de desempeño.

3. **Integrar** el modelo seleccionado en el simulador de realidad virtual, con el fin de habilitar inferencias en tiempo real y retroalimentación automática durante la práctica de sutura.

4. Datos y Metodología

4.1 Procedimiento General de Obtención y Construcción de los Datasets

El desarrollo del sistema de clasificación de imágenes del simulador SECMA requirió una construcción progresiva de varios datasets, cada uno con un propósito específico dentro del pipeline general. El proceso completo abarcó capturas iniciales, depuración, corrección de problemas técnicos y consolidación final de un conjunto amplio, limpio y representativo. La evolución de los datos reflejó aprendizajes continuos, tanto sobre la captura como sobre el preprocesamiento necesario para asegurar un rendimiento adecuado de los modelos.

Dataset 1- Dataset inicial

El primer dataset se construyó con el objetivo de validar el pipeline completo de entrenamiento, validación y limpieza utilizando únicamente dos clases: **NO_NUDO** y **UN_NUDO**. La captura se realizó en cuatro sesiones independientes, manteniendo condiciones similares de cámara (ángulo fijo de 90°) y una alta tasa de adquisición de imágenes. Esta configuración generó un volumen considerable de imágenes redundantes, además de presentar problemas de encuadre, con la cuerda frecuentemente descentrada.

Durante el tratamiento inicial se aplicaron por primera vez técnicas de detección de duplicados mediante pHash. Asimismo, se identificó que las imágenes estaban en formato RGBA, lo que provocaba la aparición de fondos negros al cargarlas en PyTorch. El análisis reveló que cerca del 97% de los píxeles correspondían a canales de transparencia, por lo que se estableció como norma la conversión a formato RGB. Este dataset permitió recorrer de extremo a extremo el pipeline de preprocesamiento y entrenamiento, entregando un primer diagnóstico sobre la calidad real de las capturas y las limitaciones inherentes al flujo de trabajo inicial.

Dataset 2 - Ampliación a tres clases

El Dataset 2 constituyó un paso intermedio relevante al incorporar por primera vez una clasificación triclase (**NO_NUDO**, **UN_NUDO**, **DOS_NUDO**). Para su construcción se realizaron nuevas sesiones de captura en SECMA, reduciendo explícitamente la tasa de cuadros por segundo con el fin de disminuir la redundancia entre imágenes consecutivas. Esta decisión permitió obtener un conjunto con menor cantidad de duplicados evidentes, manteniendo condiciones de cámara y entorno controladas.

En este dataset no fue necesario aplicar una deduplicación intensiva mediante pHash. La curación se centró principalmente en la eliminación de imágenes casi vacías o sin información relevante, utilizando un criterio automático basado en el tamaño del archivo

(menor a 4 KB). Sin embargo, este conjunto introdujo una dificultad visual relevante: por indicación de diseño, la cuerda utilizada era de color negro y prácticamente uniforme. Esta elección redujo el contraste con el fondo y limitó la variabilidad cromática disponible, disminuyendo la cantidad de información visual discriminativa y obligando al modelo a basarse casi exclusivamente en rasgos geométricos, lo que dificultó la separación entre las clases UN_NUDO y DOS_NUDO.

Aunque el Dataset 2 permitió validar la clasificación triclase y comprobar la factibilidad del flujo de exportación ONNX–Barracuda hacia Unity, sus limitaciones visuales y el desbalance entre clases impidieron su uso como base definitiva para entrenamientos más exigentes.

Dataset 3 – Dataset de diagnóstico

El Dataset 3 se diseñó con el objetivo de incrementar la diversidad visual mediante nuevas sesiones de captura, incorporando mayor variabilidad en la posición del hilo y en las configuraciones del nudo. Este conjunto incluyó tres sesiones, aunque una de ellas no se completó debido a inconsistencias durante la captura. A diferencia de los datasets previos, este conjunto presentó una redundancia interna extremadamente elevada, con grandes secuencias de imágenes casi idénticas.

Dado que el uso exclusivo de pHash resultó insuficiente para caracterizar esta redundancia, se implementó de manera excepcional un análisis basado en *embeddings* profundos para evaluar la similitud semántica entre imágenes. Los resultados evidenciaron que la variabilidad real del dataset era considerablemente menor a la esperada, reduciendo drásticamente el número de imágenes verdaderamente informativas una vez eliminadas las redundancias profundas.

Estos hallazgos justificaron la decisión metodológica de descartar el Dataset 3 como fuente principal de entrenamiento, conservando únicamente un subconjunto reducido con valor informativo. En este sentido, el Dataset 3 cumplió un rol principalmente diagnóstico, permitiendo identificar limitaciones estructurales y motivando la construcción de un conjunto final más robusto.

Dataset 4 - Consolidación del pipeline

El Dataset 4 corresponde a la versión final y más madura del conjunto de datos utilizado en el proyecto. Fue construido incorporando las lecciones aprendidas durante el análisis del Dataset 3 y corrigiendo los problemas de redundancia y baja variabilidad detectados previamente. Para ello se realizaron once nuevas sesiones de captura bajo condiciones controladas y se integraron dos sesiones rescatadas del Dataset 3 que cumplían con los criterios de calidad establecidos, conformando un total de trece sesiones.

En esta etapa, el pipeline de limpieza se encontraba completamente consolidado. Todas las imágenes se generaron directamente en formato RGB, se aplicó de manera estandarizada la eliminación de duplicados perceptuales mediante pHash y se incorporaron controles adicionales para garantizar consistencia visual y variedad intra-clase. Asimismo, se realizó una normalización uniforme de etiquetas, estructura de carpetas y divisiones de entrenamiento, validación y prueba.

El Dataset 4 presentó una diversidad sustancialmente mayor en términos de posicionamiento del hilo, progresión del nudo, tensiones y trayectorias, lo que permitió entrenar modelos con mayor capacidad de generalización. Debido a su volumen, calidad y representatividad, este conjunto se utilizó para los entrenamientos finales y para la posterior integración del modelo en Unity.

4.2 Comparación de los Datasets del Proyecto

A continuación, se presenta una comparación global de los distintos datasets utilizados durante el desarrollo del proyecto.

Aspecto	**Dataset 1 (D1)**	**Dataset 2 (D2)**	**Dataset 3 (D3)**	**Dataset 4 (D4)**
Cantidad de sesiones	4 sesiones	3 sesiones	3 sesiones (1 incompleta)	13 sesiones (11 nuevas + 2 rescatadas de D3)
Clases incluidas	NO_NUDO, UN_NUDO	NO_NUDO, UN_NUDO, DOS_NUDO	NO_NUDO, UN_NUDO, DOS_NUDO	NO_NUDO, UN_NUDO, DOS_NUDO
Formato original	RGBA (≈97% de transparencia)	RGBA	RGBA	RGB
Redundancia interna	Alta por FPS	Moderada por FPS	Extremadamente alta	Moderada y controlada
Variabilidad visual	Muy baja	Baja	Muy baja	Alta (tensión, ejecución distinta para cada sesión)
Control de duplicados aplicado	pHash	Eliminación de imágenes casi vacías (tamaño < 4 KB).	pHash + Embeddings (solo diagnóstico)	pHash

Curación semántica	No	No	Sí (únicamente para descartar D3)	No (pHash suficiente por calidad del dataset)
Balance entre clases	Desbalance moderado	Desbalance fuerte (poca DOS_NUDO)	Desbalance severo	Mejor balanceado
Objetivo del dataset	Validar pipeline binario	Probar clasificación triclase + exportación	Diagnóstico y descarte	Dataset final para entrenamientos
Utilidad final	Apta para pruebas de pipeline	Apta para demostrar factibilidad ONNX-Unity	No apto	Apta para entrenamientos finales y despliegue
Problemas principales	RGBA, duplicados, baja variabilidad	RGBA, desbalance, poca DOS_NUDO, Uso de cuerda negra con bajo contraste	Redundancia extrema	Ninguno crítico, pipeline maduro
Estado final	Útil pero limitado	Útil parcialmente	Descartado parcialmente	Principal dataset de entrenamiento

Tabla 1:Comparativa entre distintos datasets

4.2.1 Análisis comparativo

La evolución de los cuatro datasets evidencia un proceso iterativo de depuración y mejora metodológica. El Dataset 1 permitió identificar problemas fundamentales relacionados con redundancia, formato de imagen y baja variabilidad, cumpliendo un rol exploratorio. El Dataset 2 amplió la clasificación a tres clases y validó el flujo de exportación a Unity, aunque mantuvo limitaciones visuales y de balance. El Dataset 3 profundizó la búsqueda de diversidad, pero reveló deficiencias críticas que llevaron a su descarte tras un análisis semántico detallado. Finalmente, el Dataset 4 integró los aprendizajes acumulados y proporcionó un conjunto de datos robusto, consistente y adecuado para entrenamiento y despliegue operativo.

4.2.2 Conclusión de la comparación

En síntesis:

- El **Dataset 1** permitió un aprendizaje metodológico inicial.
- El **Dataset 2** demostró la factibilidad de la clasificación triclase y la integración ONNX–Unity.
- El **Dataset 3** evidenció limitaciones estructurales profundas y orientó el rediseño del pipeline.
- El **Dataset 4** consolidó la metodología y se estableció como el conjunto definitivo para el entrenamiento y despliegue del sistema.

Esta progresión refleja un desarrollo riguroso e iterativo, en el cual cada dataset aportó información clave para fortalecer el pipeline y alcanzar un sistema de clasificación estable y confiable.

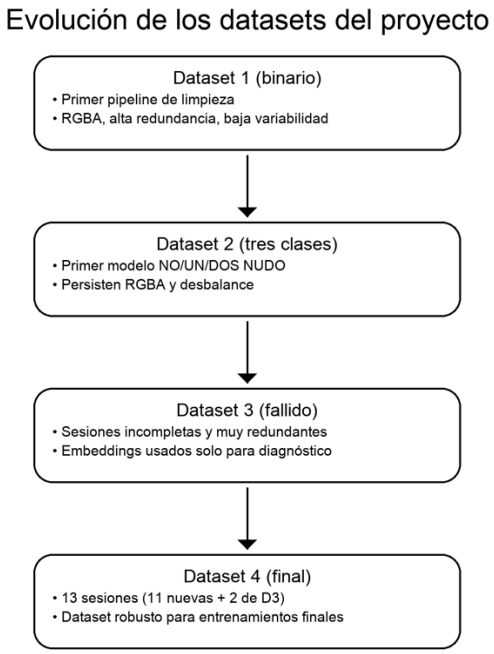

Figura 1:Evolución de los distintos datasets

4.3 Metodología

4.3.1 Metodología General

La metodología del proyecto siguió un flujo secuencial orientado al desarrollo de un sistema robusto de clasificación del estado del nudo quirúrgico a partir de imágenes generadas en el simulador SECMA. El proceso integró: (i) captura de datos en realidad virtual, (ii) curación y normalización del dataset, (iii) entrenamiento y selección de modelos, (iv) evaluación cuantitativa del desempeño y (v) despliegue del modelo en Unity mediante Barracuda para inferencia en tiempo real. Cada fase se diseñó de forma modular, permitiendo ajustes controlados según los problemas detectados en los distintos ciclos de construcción de datasets.

En primer lugar, las imágenes se obtuvieron directamente desde sesiones de simulación en SECMA mediante una cámara virtual en posición fija, asegurando condiciones visuales reproducibles y capturando diferentes estados del proceso de anudado. Posteriormente, las capturas se sometieron a un pipeline de curación que incluyó normalización del formato de color (RGB), estandarización del tamaño de entrada (224×224), eliminación de redundancias mediante técnicas perceptuales y control del desbalance entre clases. Cuando correspondió, se aplicaron técnicas de aumento de datos para robustecer la generalización del modelo.

El entrenamiento se realizó en PyTorch utilizando modelos de clasificación de imágenes preentrenados en ImageNet, con hiperparámetros definidos para favorecer reproducibilidad. El desempeño se evaluó mediante métricas estándar de clasificación multiclase (accuracy, precision, recall y F1-score), complementadas con matrices de confusión para identificar patrones de error, en particular entre clases visualmente similares (UN_NUDO y DOS_NUDO). Esta evaluación permitió seleccionar el modelo con mejor equilibrio entre desempeño predictivo y eficiencia computacional.

Finalmente, el modelo seleccionado fue exportado a formato ONNX e integrado en Unity utilizando el motor Barracuda, incorporando verificaciones de equivalencia entre PyTorch y ONNX y pruebas de inferencia para asegurar consistencia de salidas (logits y predicciones) y compatibilidad con ejecución en tiempo real en dispositivos Meta Quest.

4.3.2 Evolución progresiva del pipeline de limpieza y control de duplicados

La metodología siguió un proceso iterativo en el que cada dataset permitió identificar limitaciones y refinar el pipeline de forma incremental. En las fases iniciales, las técnicas de limpieza se enfocaron en resolver problemas inmediatos derivados de la redundancia temporal y de inconsistencias de formato en las capturas del simulador.

En el Dataset 1, el control de duplicados se basó en pHash, adecuado para detectar imágenes idénticas o casi idénticas. Esta elección fue coherente con el propósito

exploratorio de dicha etapa: validar el flujo de preprocesamiento y detectar problemas críticos como la presencia de canal alfa en imágenes RGBA.

Posteriormente, y de forma excepcional, en el análisis del Dataset 3 se aplicó un método basado en embeddings profundos para cuantificar redundancia semántica y estimar la variabilidad efectiva del conjunto. Este procedimiento no se incorporó al pipeline estándar, sino que se utilizó con propósito diagnóstico para determinar la viabilidad del dataset. Los resultados evidenciaron una redundancia sustantiva y una diversidad insuficiente, lo que justificó descartar el conjunto como base principal y motivó la construcción del Dataset 4. En consecuencia, el pipeline final quedó consolidado como un proceso robusto y reproducible basado principalmente en normalización RGB y deduplicación perceptual, adecuado para capturas de mejor calidad y mayor variabilidad.

4.3.3 Resumen de la metodología

Etapa	Descripción del trabajo realizado
Captura de datos	Imágenes obtenidas en múltiples sesiones en SECMA mediante cámara virtual. Los conjuntos iniciales presentaron RGBA; el conjunto final se estandarizó en RGB y se incrementó la diversidad entre sesiones (progresión del nudo, tensión del hilo y trayectorias).
Preprocesamiento y curación	Conversión RGBA a RGB cuando correspondía, eliminación de duplicados con pHash, control de imágenes vacías y reorganización por clases. En el Dataset 3 se aplicó embeddings de forma extraordinaria para diagnóstico de redundancia.
Entrenamiento	Entrenamiento en PyTorch con modelos preentrenados, función de pérdida CrossEntropy, optimización Adam y regularización con weight decay, evitando sobreajuste mediante un número controlado de épocas.
Evaluación	Métricas de clasificación (accuracy, precision, recall, F1-score) y matrices de confusión para analizar errores sistemáticos entre clases.
Exportación e integración	Exportación a ONNX, verificación de consistencia entre PyTorch con ONNX y despliegue en Unity mediante Barracuda para inferencia en tiempo real.
Retroalimentación	Visualización de la clase predicha en el simulador y diseño del flujo para retroalimentación inmediata basada en el estado del nudo.

Tabla 2: Resumen de la metodología aplicada.

4.4 Captura de datos en el simulador

La aplicación de sutura realiza la simulación de la interacción de componentes para la simulación de la piel (una malla deformable), la aguja y la cuerda de sutura. Presenta además instrumentos utilizados en la realización de la actividad

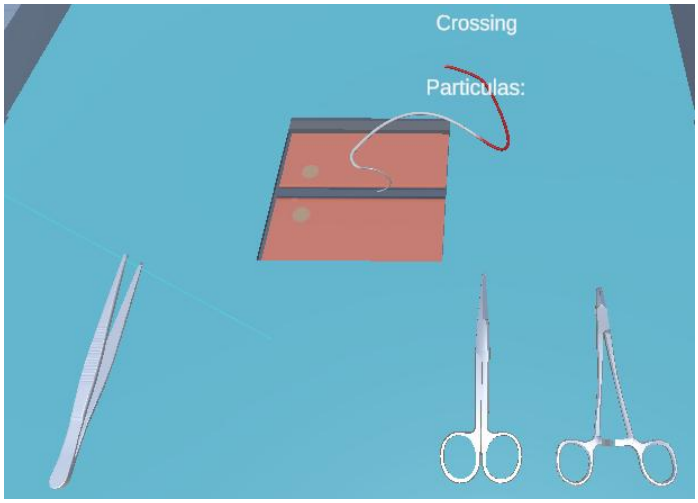

Figura 2: Escena para la actividad de sutura virtual

Para obtener una observación del estado de la cuerda, se colocó una cámara virtual invisible con una rotación de 45 grados respecto al eje X (Pitch), cerca de la zona destinada a la interacción cuerda, la malla deformable.

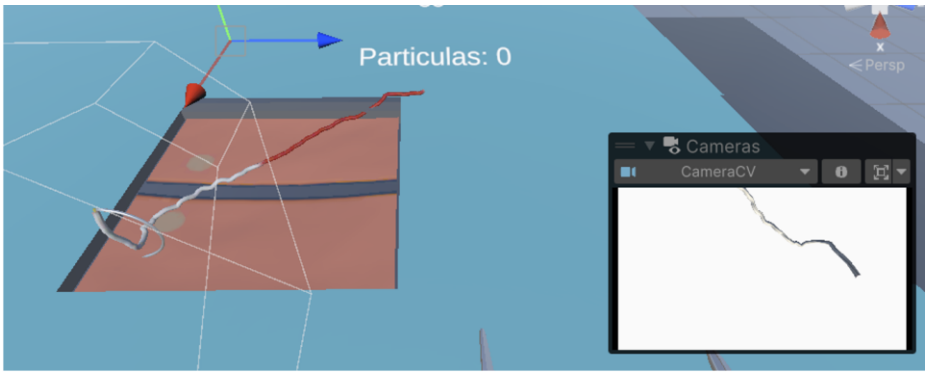

Figura 3: Escena y cámara virtual observadora de la cuerda

Esta cámara (CAMCV), que se incorpora como un gameobject a la escena tiene su salida hacia una textura (recurso *Render Texture*), para que en ella se guarde la imagen capturada. A continuación, se muestra el componente de unity con la salida.

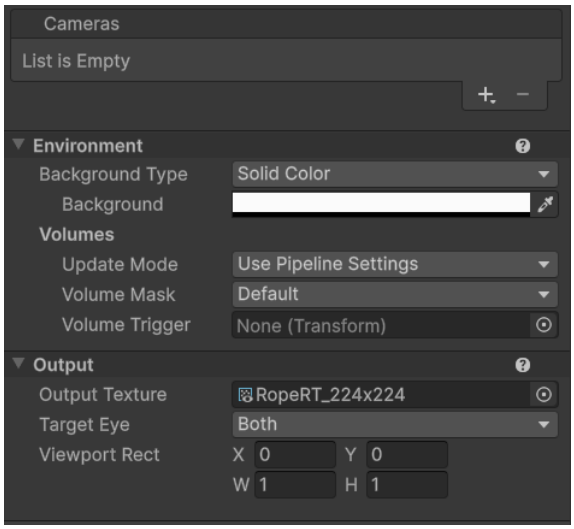

Figura 4: Componente de la cámara con salida a textura

El recuso Render Texture (RT) creado para almacenar la imagen obtenida por la cámara se preparó con un tamaño de 224 x 224 y con el formato de color recomendado por la literatura R8G8B8A8_UNNORM (cuatro canales de 8 bits en el rango de 0 a 255). Se observa la configuración en la siguiente figura.

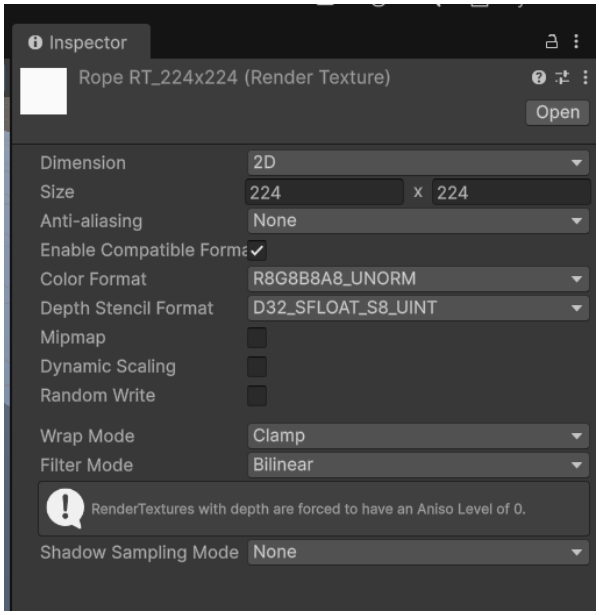

Figura 5: Configuración del recurso Render Texture

Para completar el guardado de las imágenes, se creó un componente tipo script, con el objetivo de leer el recurso de Render Texture. Este script se anexó al objeto CAMCV que observa la realización del nudo. El pseudocódigo se muestra a continuación.

```
Cada Δt segundos:
  Solicitar lectura asíncrona del RenderTexture en formato RGBA32.
  Cuando la lectura es completada:
    Obtener buffer de bytes.
    Determinar etiqueta de clase según evento de nudo.
    Encolar estructura SaveJob{bytes, ancho, alto, etiqueta, timestamp}.
En cada frame de la simulación:
  Mientras existan SaveJobs pendientes (máx. N por frame):
   Reconstruir imagen en memoria (RGBA).
    Codificar como PNG.
    Guardar archivo con nombre que incluye la etiqueta de clase.
```

La lectura de píxeles se realizó mediante la función AsyncGPUReadback, que permite extraer los datos del RenderTexture sin bloquear el hilo principal. Esta técnica evita caídas en la tasa de frames y elimina interferencias con la ejecución de la simulación, condición indispensable para la recopilación continua de datos en tiempo real. Si bien la lectura es GPU a CPU se realiza en formato RGBA, el sistema descarta el canal alfa durante el post-procesamiento. Esto permite generar archivos PNG exclusivamente en RGB, reduciendo el tamaño del dataset sin pérdida de información relevante para los modelos utilizados.

El proceso completo de captura se organizó en dos etapas asíncronas: la adquisición de datos desde la GPU y el procesamiento en la CPU. En la primera etapa, cada cierto intervalo temporal $\Delta t = 1/f$ (donde f es la frecuencia de muestreo), se solicita una lectura asincrónica del RenderTexture. Esta operación devuelve un buffer con los bytes crudos de la imagen. En la segunda etapa, los datos se encolan en una estructura FIFO y se procesan gradualmente en el hilo principal, donde se reconstruye la textura, se codifica a PNG y se escribe en disco.

El sistema incorpora además un mecanismo de etiquetado automático basado en un algoritmo que contemplan los cruces existentes y la presión ejercida en la cuerda. Cada vez que se detecta un nudo durante el ejercicio de sutura, se genera un evento que actualiza el contador global de nudos formados. Dicho valor se incrusta en el nombre del archivo generado, permitiendo organizar el dataset por clase sin requerir un proceso de rotulado manual posterior.

Hay que tener en cuenta que este etiquetado tiene sus características determinadas por el algoritmo y que pueden no coincidir con la apreciación visual, ya que solo considera los nudos que se realizan se encuentran apretados, no un estado previo donde no está apretado. Por ejemplo, al hacer el primer lazo, si no se aprieta sigue considerando NO_NUDO. Este algoritmo resulta util, pero para ser más precisos se reetiqueta manualmente en la etapa de preprocesamiento.

4.5 Integración de la predicción en el flujo de trabajo del simulador (Unity + Barracuda)

Para habilitar inferencia en tiempo real dentro del simulador, se integró el motor Barracuda de Unity, el cual permite ejecutar modelos ONNX en CPU/GPU. El modelo entrenado se exportó a ONNX y se importó como un activo NNModel dentro del proyecto, lo que facilita detectar inconsistencias de exportación mediante advertencias del editor.

El sistema de inferencia se implementó mediante un GameObject dedicado (CVKnotClassifier) y un script responsable de: (i) capturar la imagen desde el RenderTexture (224×224), (ii) convertirla a tensor RGB, (iii) ejecutar el modelo y (iv) publicar la predicción. La ejecución se organizó mediante corutinas para distribuir el costo de inferencia y evitar bloqueos del hilo principal, preservando la fluidez del simulador en Meta Quest.

La salida del modelo corresponde a un vector de logits sobre las clases {NO_NUDO, UN_NUDO, DOS_NUDO}. La clase predicha se obtiene mediante argmax. Dado que pueden existir discrepancias de índices entre el orden de clases en PyTorch y el orden exportado a ONNX, se incorporó un vector de remapeo para alinear correctamente las etiquetas finales. La predicción se visualiza en la interfaz del simulador y el sistema gestiona la liberación de recursos temporales para asegurar estabilidad durante ejecución continua.

A continuación, se muestra en pseudocódigo el algoritmo utilizado.

```
Cada Δt segundos:
    Capturar la imagen 224×224 desde el RenderTexture.
    Convertir la imagen a tensor RGB.
Ejecutar el modelo neuronal de forma asíncrona:
    Procesar las capas del modelo a lo largo de varios frames.
    Esperar a que la ejecución finalice.
Obtener la salida "logits" del modelo.
Seleccionar la clase:
    Aplicar ArgMax sobre logits.
    Remapear el índice según la tabla onnxToPt.
    Obtener la etiqueta final (NO_NUDO, UN_NUDO, DOS_NUDOS).
Actualizar la interfaz con la etiqueta predicha.
Liberar recursos temporales (tensor e imagen).
```

5 Análisis, curación y resultados de los datasets

Una vez definido el procedimiento de captura de imágenes en el simulador SECMA, se estructuró un pipeline integrado de análisis, curación y evaluación orientado a asegurar calidad visual, consistencia de formato, trazabilidad y validez experimental. Este proceso se concibió como un ciclo iterativo, en el que las decisiones metodológicas adoptadas en cada etapa se evaluaron directamente a través de los resultados obtenidos, condicionando el diseño y refinamiento de los conjuntos de datos posteriores.

En términos generales, el pipeline consideró: (i) el control del formato de color de las imágenes (RGB), (ii) la detección de imágenes vacías o carentes de información útil, (iii) la eliminación de duplicados perceptuales mediante pHash, (iv) la reorganización de las imágenes por clases y sesiones, y (v) la construcción de particiones para entrenamiento, validación y prueba. Con el fin de garantizar reproducibilidad y trazabilidad, las imágenes descartadas en cada etapa no se eliminaron permanentemente, sino que se almacenaron en estructuras separadas (por ejemplo, empty y dup), preservando su sesión de origen.

El análisis de cada dataset combinó un estudio exploratorio de las imágenes, necesario para comprender su calidad, redundancia, distribución de clases y características visuales relevantes, con la evaluación del desempeño de los modelos de clasificación entrenados en las distintas iteraciones del pipeline. Esta evaluación se realizó mediante métricas estándar de clasificación, incluyendo accuracy, precision, recall, F1-score y matrices de confusión, permitiendo comparar de manera objetiva el impacto de cada decisión de curación y selección de datos.

Adicionalmente, se llevó a cabo la validación de la equivalencia funcional entre los modelos entrenados en PyTorch y sus versiones exportadas a ONNX, requisito indispensable para asegurar que la integración posterior en Unity y su ejecución en el dispositivo Meta Quest mantuvieran el comportamiento del modelo original.

En las subsecciones siguientes se presentan los resultados obtenidos para cada uno de los datasets construidos a lo largo del proyecto, destacando las dificultades identificadas, las soluciones metodológicas aplicadas, los avances logrados en cada iteración y la evolución progresiva del desempeño del sistema, desde las primeras pruebas exploratorias hasta la versión final integrada en el simulador de realidad virtual.

5.1 Dataset 1

5.1.1 caracterización y curación

El primer dataset fue generado a partir de cuatro sesiones de captura independientes en el simulador SECMA, obtenidas el 14 de octubre de 2025, con los identificadores y número de imágenes por sesión:

- session_20251014_212836 (700)
- session_20251014_213003 (680)
- session_20251014_213140 (976)
- session_20251014_213336 (878)
  Total incial; 3234 imágenes

Las imágenes se separaron manualmente en dos categorías: NO_NUDO y UN_NUDO, con la cuerda en diferentes etapas del proceso de anudado. Este dataset tuvo como propósito inicial validar el pipeline completo de procesamiento, entrenamiento y servir como base inicial del proyecto.

Problemas detectados:
El análisis exhaustivo reveló varios problemas estructurales y técnicos:

4. Formato RGBA en el 100% de la muestra revisada: Todas las imágenes analizadas tenían cuatro canales en lugar de tres. Un muestreo de 100 imágenes confirmó transparencia promedio: 97.39% de los píxeles del canal alfa eran transparentes. Esto generaba fondos negros al cargar las imágenes en PyTorch, afectando la apariencia y el preprocesamiento.
5. Alta redundancia por tasa de captura elevada: Debido a la velocidad del simulador, numerosas imágenes consecutivas mostraban exactamente la misma configuración, generando cientos de duplicados perceptuales.
6. Desbalance significativo entre clases: Tras la limpieza física inicial, la distribución final previa al rebalanceo fue de NO_NUDO: 1115 imágenes y UN_NUDO: 333 imágenes. Esto provocaba una tendencia natural del modelo hacia la clase mayoritaria.
7. Variabilidad visual limitada dado que las imágenes no estaban centradas.

Curación aplicada

El dataset fue sometido a un pipeline de depuración y normalización con los siguientes pasos:

1. Conversión de RGBA a RGB. Se eliminó el canal alfa y se reconstruyó un fondo blanco adecuado para entrenamiento.

2. Eliminación de duplicados perceptuales. Se redujo de forma sustancial el volumen total al separar imágenes idénticas o casi idénticas.
3. Estandarización de tamaño y formato. Todas las imágenes se normalizaron a resolución 224×224 en RGB.
4. División estratificada train–val–test
   - Train: 780 NO_NUDO, 233 UN_NUDO
   - Val: 167 NO_NUDO, 49 UN_NUDO
   - Test: 168 NO_NUDO, 51 UN_NUDO
5. Rebalanceo del conjunto de entrenamiento. Para reducir el sesgo, la clase mayoritaria fue reducida mediante undersampling controlado:
   - NO_NUDO reducida a 349 imágenes
   - UN_NUDO mantenida en 233 imágenes

   Total final de entrenamiento: 582 imágenes

Impacto de la limpieza

La curación aplicada tuvo un efecto directo y positivo en la calidad del dataset y la estabilidad del modelo:

- Se resolvió el problema del fondo negro, lo cual permitió normalizar los canales de entrada.
- Se redujo drásticamente la redundancia, pasando de 3234 imágenes brutas a 1448 imágenes limpias.
- El rebalanceo mejoró la capacidad discriminativa del modelo entre NO_NUDO y UN_NUDO, evitando que priorizara la clase mayoritaria.
- El tamaño final del dataset quedó más alineado con un escenario experimental controlado, ideal para validar el pipeline antes de avanzar hacia datasets más complejos.
- El entrenamiento se volvió más estable, ya que el modelo dejó de aprender patrones triviales derivados de duplicados consecutivos.

Así, este proceso permitió utilizar el Dataset 1 como una base sólida para validar todas las etapas del pipeline (carga, limpieza, normalización, entrenamiento y evaluación), antes de escalar hacia datasets más robustos y variados.

5.1.2 Resultados obtenidos y limitaciones

El análisis del Dataset 1 constituyó el primer paso fundamental para validar la solidez del pipeline de preprocesamiento, entrenamiento y evaluación diseñado para la detección automática de nudos quirúrgicos dentro del simulador SECMA. Esta etapa inicial se enfocó exclusivamente en la clasificación binaria lo que permitió evaluar en un entorno controlado la capacidad del modelo para aprender patrones visuales discriminativos básicos antes de incorporar mayor complejidad en datasets posteriores.

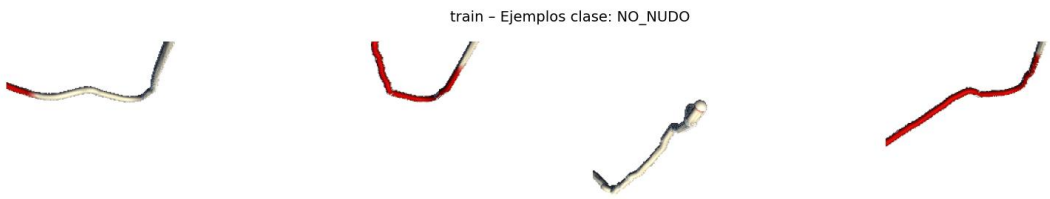

Figura 6: Muestra aleatoria de la clase NO_NUDO

Figura 7: Muestra aleatoria de la clase UN_NUDO

El proceso comenzó con un análisis exploratorio detallado de la estructura y distribución de las imágenes. El dataset se encontraba organizado en tres particiones independientes (train, validation y test), con una distribución claramente desbalanceada hacia la clase NO_NUDO: 780 imágenes en entrenamiento frente a 233 para UN_NUDO; 167 frente a 49 en validación; y 168 frente a 51 en el conjunto de prueba. Este desbalance inicial evidenciaba la necesidad de aplicar una estrategia de rebalanceo para evitar que el modelo desarrollara un sesgo hacia la clase mayoritaria. Además, la inspección visual permitió constatar que algunas imágenes se encontraban en formato RGBA, lo que resultaba en fondos negros tras la aplicación de los transforms de PyTorch. Esto requirió una conversión explícita a RGB, junto con una composición sobre fondo blanco para mantener coherencia estética y evitar introducir patrones artificiales en los datos.

Como parte de la curación del dataset, se aplicaron métodos para identificar y eliminar duplicados, utilizando técnicas perceptuales como pHash, lo cual permitió reducir redundancias y asegurar que cada imagen aportara información única al proceso de aprendizaje. Con la estructura limpia, se procedió al rebalanceo del conjunto de entrenamiento, ajustando la proporción de clases mediante muestreo aleatorio para mantener una relación más apropiada para un modelo binario. Este proceso redujo la clase NO_NUDO a 349 ejemplos, manteniendo todos los 233 casos de UN_NUDO, y generó

un conjunto equilibrado de 582 imágenes finales que fortalecieron la capacidad del modelo para aprender patrones de ambas clases.

Una vez completada la curación, se entrenó una arquitectura ResNet18, elegida por su estabilidad y capacidad para tareas de clasificación de bajo y mediano nivel de complejidad. Se utilizaron parámetros consistentes con buenas prácticas del aprendizaje supervisado: diez épocas de entrenamiento, *learning rate* de 0.0003, *weight decay* de 0.0001, y un *batch size* de 32. Los resultados del entrenamiento fueron especialmente alentadores. La pérdida del modelo mostró una disminución progresiva y estable, pasando de 0.2235 en la primera época a 0.0240 en la última, mientras que la pérdida del conjunto de validación acompañó esta tendencia, manteniéndose siempre en niveles bajos. Esta convergencia armónica sugiere que el modelo logró aprender patrones relevantes sin incurrir en sobreajuste, lo cual quedó reforzado por la similitud en el comportamiento entre entrenamiento y validación.

Un aspecto crítico en el pipeline fue la determinación del umbral óptimo de decisión para la clasificación. En lugar de emplear directamente la predicción basada en *argmax*, se analizó la curva Precision–Recall del conjunto de validación para encontrar un umbral que maximizara el F1-score en la clase minoritaria. Este procedimiento arrojó un valor óptimo de threshold = 0.7708, con métricas perfectas en validación (precisión, recall y F1-score iguales a 1.0000). Este resultado evidencia una separación clara entre ambas clases en el espacio latente del modelo, lo que refuerza la eficacia del preprocesamiento y la calidad de los datos tras la limpieza.

La evaluación final en el conjunto de prueba confirmó el buen desempeño del modelo. Con un total de 219 imágenes evaluadas, el sistema logró:

$$TP = 50, FP = 1, TN = 167 \text{ y } FN = 1,$$

lo que se tradujo en una exactitud del 0.9909, una precisión y un recall de 0.9804, y un F1-score también de 0.9804. Estos resultados revelan una alta capacidad de generalización y una estabilidad notable entre las métricas de validación y prueba. Las dos únicas equivocaciones se concentraron en ejemplos en los que la clase real era UN_NUDO, pero fueron predichos como NO_NUDO. Este tipo de error es coherente con la variabilidad intrínseca de la clase UN_NUDO, que tiende a presentar configuraciones más heterogéneas y transiciones visuales menos marcadas.

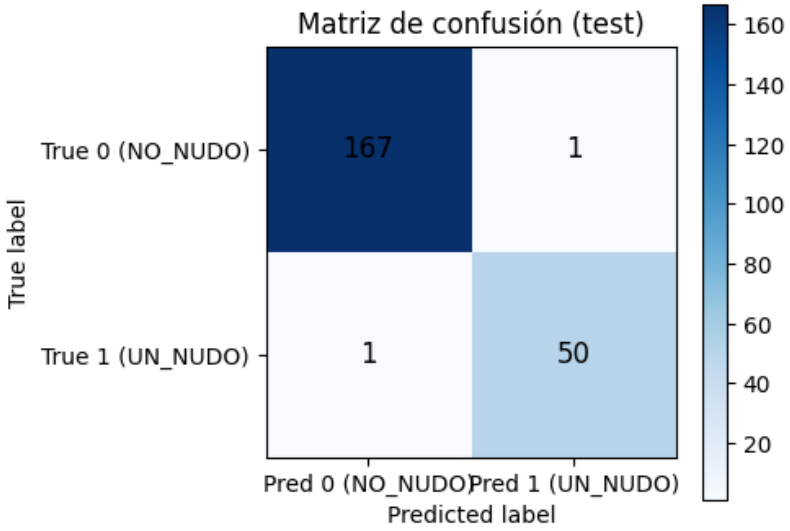

Figura 8: Matriz de confusión del modelo Resnet-18

Para complementar el análisis cuantitativo, se realizó también una evaluación cualitativa de los aciertos y fallos mediante la inspección visual de muestras aleatorias presentadas en formato de grilla. Las imágenes correctamente clasificadas mostraban patrones geométricos claros y consistentes, mientras que los ejemplos mal predichos correspondían a casos limítrofes, como nudos parcialmente formados. Este análisis refuerza la interpretación de que los errores no representan fallas sistemáticas del modelo, sino casos complejos cercanos a la frontera de decisión.

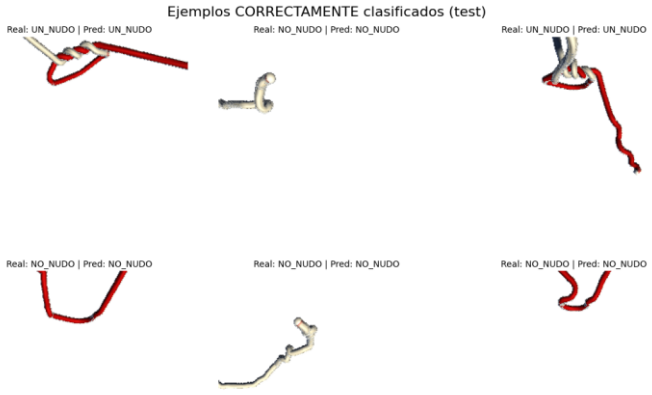

Figura 9:Muestra aleatoria de ejemplos clasificados correctamente por el modelo.

Figura 10: Muestra aleatoria de ejemplos clasificados correctamente por el modelo.

Los resultados del Dataset 1 demostraron que este conjunto fue plenamente adecuado para validar la integridad del pipeline completo: la lectura y normalización de imágenes, la eliminación de ruido, el rebalanceo, el entrenamiento, la búsqueda del umbral óptimo y la evaluación cuantitativa y cualitativa. Aunque el dataset presenta una variabilidad visual limitada y no está diseñado para su uso en producción, cumplió su propósito metodológico con creces. Permite establecer una base sólida y confiable para abordar datasets posteriores más complejos y representativos, asegurando que el pipeline es capaz de aprender, generalizar y discriminar patrones relevantes en la detección automática de nudos quirúrgicos dentro del entorno RV de SECMA.

5.2 Dataset 2

5.2.1 caracterización y curación

El análisis inicial del Dataset 2 tomado el 26 de octubre estaba compuesto por tres sesiones completas distribuidas de la siguiente manera:

- session_20251026_225833 (876)
- session_20251026_230016 (796)
- session_20251026_230155 (697)
  Total inicial: 2369 imágenes.

A diferencia del Dataset 1, el Dataset 2 ya venía prácticamente libre de duplicados perceptuales, debido a la reducción explícita de la tasa de cuadros por segundo en la captura original, razón por la cual no se aplicó un proceso intensivo de deduplicación basado en pHash. La curación se centró en eliminar exclusivamente las imágenes casi vacías, utilizando un script que descartaba automáticamente todos los archivos con un peso inferior a 4 KB. Tras esta limpieza, el dataset quedó estructurado en tres carpetas principales (dataset_005, dataset_006, dataset_007), con un total de 1908 imágenes válidas**,** distribuidas como sigue:

- dataset_005: 717
- dataset_006: 669
- dataset_007: 522

Estos valores reflejan que la gran mayoría de las imágenes contenían información útil y que la reducción del dataset se debió principalmente a la eliminación de capturas sin contenido relevante, más que a la eliminación de duplicados.

Luego se construyó un dataset unificado mediante una estrategia de merge por sesiones. En particular, se adoptó el siguiente particionado: las carpetas dataset_005 y dataset_006 se utilizaron como fuentes exclusivas para el conjunto de entrenamiento, mientras que dataset_007 se reservó íntegramente para los conjuntos de validación y prueba. Esta decisión buscó garantizar que el modelo fuese evaluado sobre imágenes provenientes de una sesión distinta a las usadas para entrenar, reduciendo así el riesgo de sobreajuste a patrones específicos de una sesión y permitiendo una evaluación más realista de su capacidad de generalización.

5.2.2 Resultados de entrenamiento y evaluación

El proceso de entrenamiento del Dataset 2 tuvo como objetivo evaluar por primera vez la clasificación triclase (DOS_NUDO**,** NO_NUDO**,** UN_NUDO) utilizando un conjunto de datos con menor redundancia y mayor limpieza que el Dataset 1. Esta fase representó un avance significativo respecto al enfoque binario inicial, pues permitió estudiar cómo se comportaba el pipeline al incorporar una clase nueva (DOS_NUDO), cuyo patrón visual es naturalmente más ambiguo y difícil de discriminar. La partición utilizada siguió la estrategia basada separar los dataset_005 *y* dataset_006 para entrenamiento, mientras que dataset_007 se reservó íntegramente para validación y prueba. Esta decisión metodológica redujo de forma sustantiva la dependencia del modelo a patrones específicos de captura y garantizó una evaluación sobre datos completamente inéditos, condición esencial para estimar su capacidad real de generalización.

Tras la limpieza básica y la organización final del dataset, el conjunto de entrenamiento quedó compuesto por 1386 imágenes, distribuidas de manera relativamente equilibrada entre las tres clases: 560 imágenes de NO_NUDO**,** 415 de UN_NUDO **y** 411 de DOS_NUDO. Tanto el conjunto de validación como el de prueba contenían 522 imágenes cada uno, permitiendo una evaluación estadísticamente estable. Como primer paso, se realizó un análisis exploratorio visual que consistió en generar muestras aleatorias de cada clase. Este análisis permitió confirmar que el dataset no contenía imágenes corruptas ni problemas derivados del canal alfa, pero también puso en evidencia una limitación central: la cuerda negra utilizada en la captura dificultaba distinguir visualmente la diferencia entre UN_NUDO y DOS_NUDO, especialmente en vistas donde los cruces de hilo quedaban parcialmente ocultos o comprimidos. Mientras NO_NUDO presentaba patrones visuales

claramente distinguibles, la frontera entre un nudo simple y uno doble era sutil en muchos casos, lo que anticipaba potenciales dificultades para el modelo.

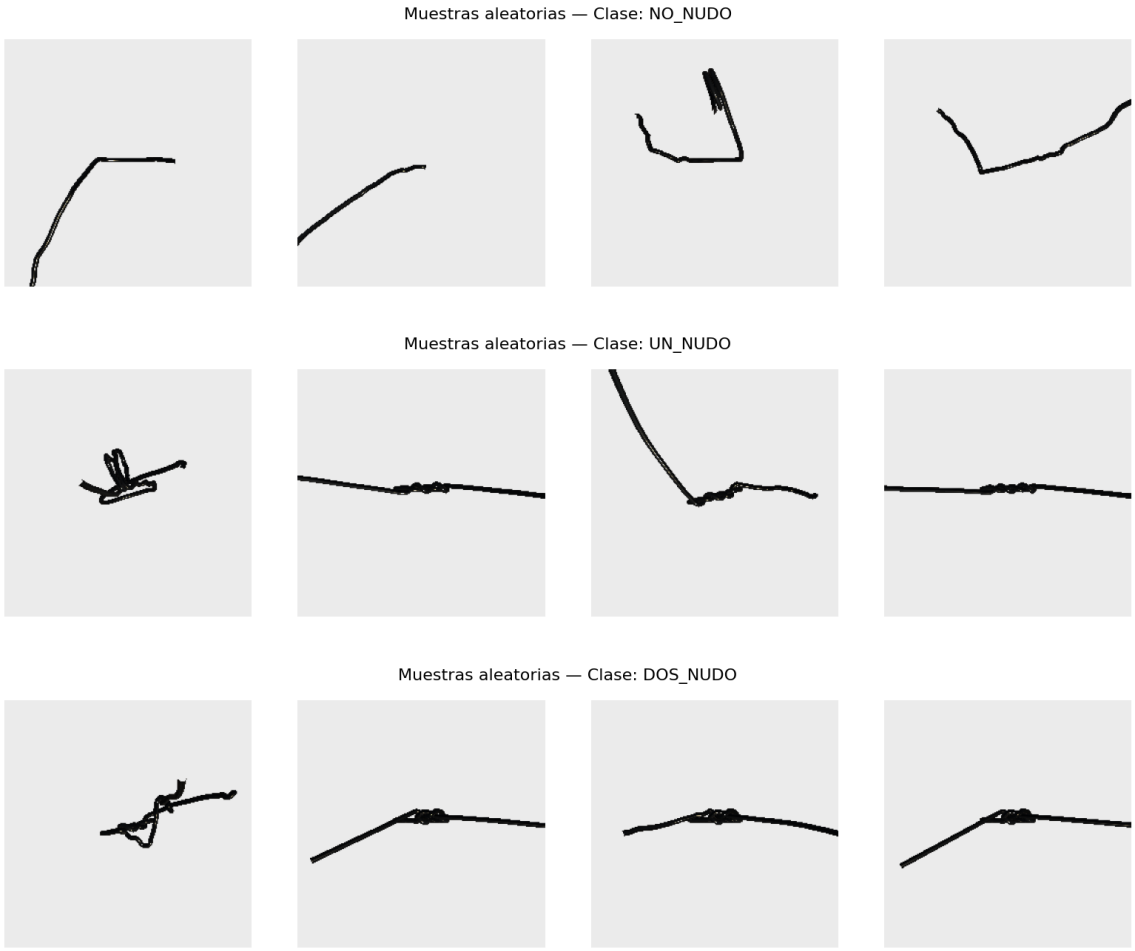

Figura 11: Muestras aleatorias de las tres clases

Para evaluar la capacidad de aprendizaje bajo condiciones uniformes, se entrenaron tres arquitecturas convolucionales preentrenadas en ImageNet: EfficientNet-B0, ResNet18 y MobileNetV3-Small, seleccionadas tanto por su diversidad estructural como por su amplia adopción en tareas de clasificación de imágenes. Con el fin de garantizar una comparación justa entre modelos, las tres arquitecturas fueron entrenadas bajo un marco experimental estrictamente controlado, utilizando exactamente los mismos hiperparámetros, la misma cantidad de épocas (12) y el mismo conjunto de particiones. En los tres casos se empleó la función de pérdida CrossEntropyLoss, el optimizador Adam con una tasa de aprendizaje inicial de 3×10^{-4} y un esquema de regularización mediante weight decay. Adicionalmente, se incorporó el scheduler ReduceLROnPlateau con factor = 0.5 y patience = 3, con el objetivo de ajustar dinámicamente la tasa de aprendizaje cuando la pérdida de validación dejara de mejorar, mitigando así riesgos de estancamiento durante el entrenamiento.

Los resultados mostraron un patrón consistente en las tres arquitecturas: todas alcanzaron pérdidas de entrenamiento extremadamente bajas y precisiones superiores al 99%, lo que indica que el modelo fue capaz de memorizar eficazmente los patrones presentes en el conjunto de entrenamiento. No obstante, la pérdida de validación mostró fluctuaciones marcadas, particularmente en EfficientNet-B0 y MobileNetV3-Small, las cuales alternaron entre valores bajos y picos abruptos según la época. ResNet18 presentó un comportamiento más estable, aunque también evidenció oscilaciones en las primeras épocas antes de converger a valores de validación más bajos. Este desajuste entre entrenamiento y validación refleja un grado de sobreajuste atribuible al tamaño limitado del dataset y a la reducida variabilidad visual entre sesiones, especialmente en la distinción entre UN_NUDO y DOS_NUDO.

Entre los modelos evaluados, ResNet18 obtuvo el mejor rendimiento general en el conjunto de validación, alcanzando una val_acc de 0.9751, seguido muy de cerca por EfficientNet-B0 (0.9732). MobileNetV3-Small, si bien más ligero y rápido, logró un rendimiento claramente inferior (máximo de 0.9387), mostrando dificultades particularmente marcadas para diferenciar UN_NUDO de DOS_NUDO. Esto refuerza la idea de que los modelos más compactos tienden a perder sensibilidad ante transiciones visuales sutiles presentes en este dataset.

La evaluación final en el conjunto de prueba confirmó la robustez del modelo basado en ResNet18. Sobre un total de 522 imágenes, el modelo obtuvo 514 aciertos y solo 8 errores, alcanzando una exactitud del 98%, un F1-score macro de 0.97 y un F1-score ponderado de 0.98. El análisis detallado por clase mostró un desempeño particularmente sólido:

- DOS_NUDO**:** precisión 0.91, recall 1.00, F1 = 0.95.
- NO_NUDO**:** precisión 1.00, recall 1.00, F1 = 1.00.
- UN_NUDO**:** precisión 0.99, recall 0.93, F1 = 0.96.

La matriz de confusión reveló que los errores se concentraron principalmente en la clase UN_NUDO, donde se registraron 12 casos etiquetados incorrectamente como DOS_NUDO, lo que coincide plenamente con la dificultad observada en el análisis visual. La cuerda completamente negra reduce la posibilidad de distinguir detalles morfológicos finos del cruce del hilo, especialmente cuando el nudo se encuentra parcialmente formado. Por el contrario, las clases NO_NUDO y DOS_NUDO se separaron con extrema claridad, lo que confirma que estos dos estados poseen patrones estructurales suficientemente distintivos incluso bajo condiciones visuales homogéneas.

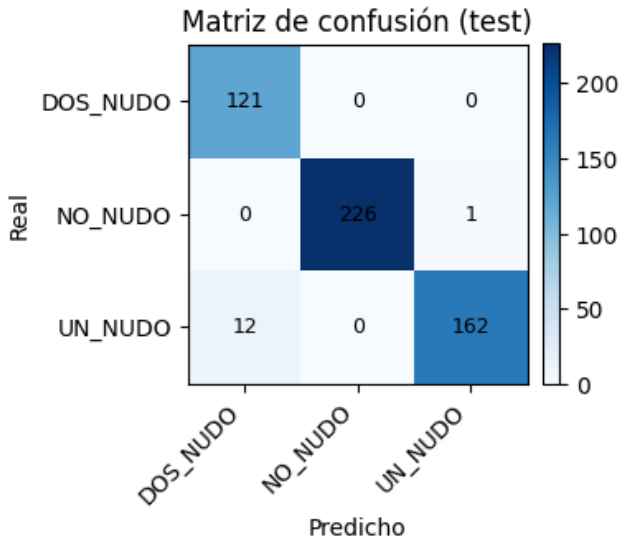

Figura 12: Matriz de confusión del modelo Resnet-18

Además del análisis cuantitativo, se realizó una inspección cualitativa mediante la generación de muestras aleatorias de aciertos y errores. Los aciertos mostraron representaciones nítidas en las que la posición y geometría del hilo definían claramente el estado del nudo. En contraste, los errores correspondían a situaciones limítrofes, en las cuales el hilo presentaba configuraciones intermedias o parcialmente tensadas que dificultaban la correcta diferenciación entre un nudo simple y uno doble. Estas observaciones refuerzan que las equivocaciones no son fallos del modelo, sino consecuencia directa de las limitaciones intrínsecas del dataset.

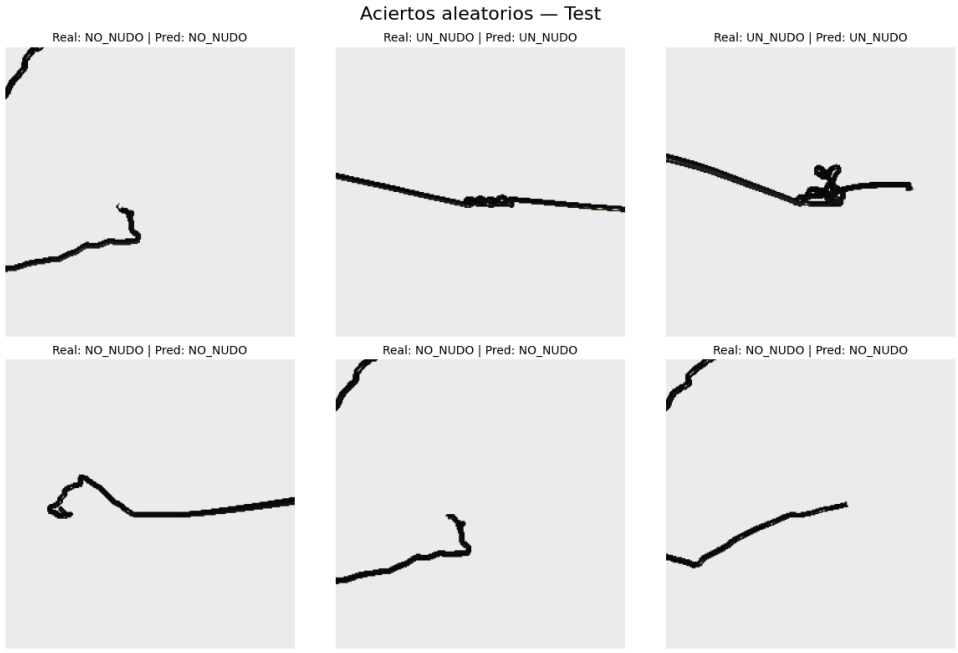

Figura 13: Vista aleatoria de aciertos del modelo Resnet-18

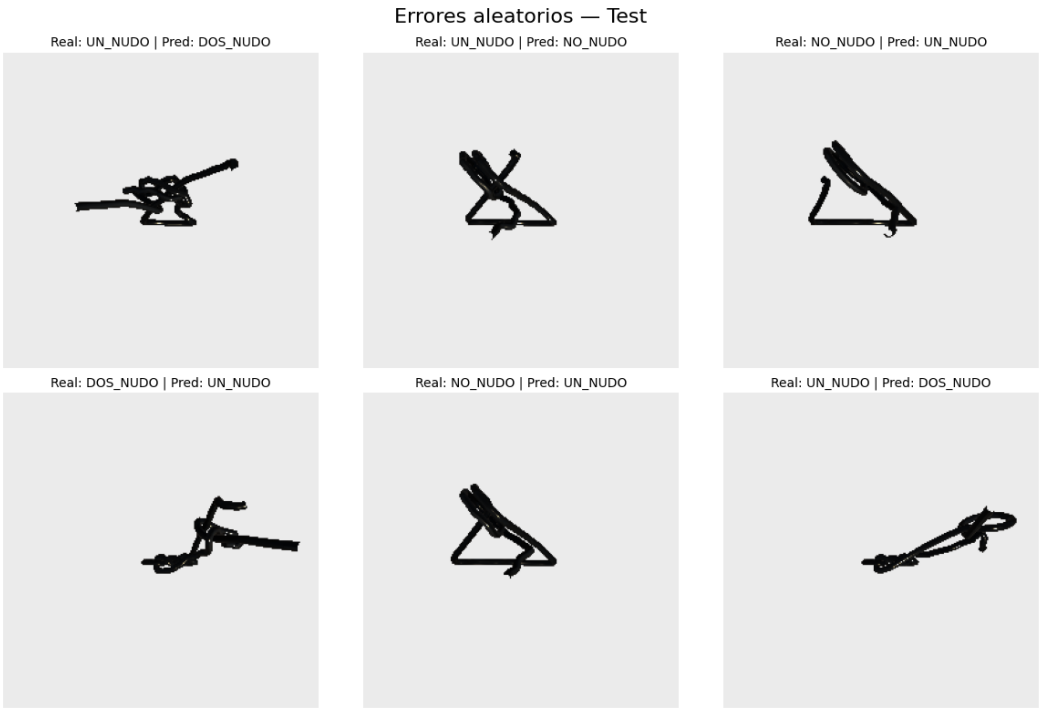

Figura 14: Vista aleatoria de errores del modelo Resnet-18

Aunque el modelo alcanzó un rendimiento sobresaliente, se decidió no integrarlo al simulador SECMA. El Dataset 2 tenía un propósito exclusivamente exploratorio y metodológico; su función principal era validar que la clasificación triclase fuese factible y que el pipeline de entrenamiento estuviera correctamente instrumentado. Como parte de un ejercicio técnico, el modelo final de ResNet18 fue exportado a formato ONNX, demostrando que la arquitectura era compatible con el flujo de despliegue utilizado posteriormente en Unity + Barracuda. Sin embargo, esta exportación se realizó solo como demostración, ya que el modelo no cumplía con los requisitos visuales y de variabilidad necesarios para una integración real en el emulador.

Con todo lo anterior, los resultados obtenidos confirman que el Dataset 2 fue una etapa crucial para validar la factibilidad de la clasificación triclase y que ResNet18 ofrece el mejor compromiso entre capacidad de representación, estabilidad y eficiencia. No obstante, las dificultades persistentes derivadas del uso de una cuerda negra, la limitada variabilidad visual entre sesiones y la confusión inherente entre UN_NUDO y DOS_NUDO reforzaron la necesidad de construir un dataset más diverso y representativo. Estas conclusiones motivaron directamente el desarrollo del Dataset 3 y, posteriormente, del Dataset 4, que constituye la base del modelo final integrado en el simulador SECMA.

5.3 Dataset 3

5.3.1 Caracterización y diagnóstico de redundancia

El Dataset 3 se obtuvo a partir de tres sesiones independientes capturadas el 13 de noviembre. Su distribución interna fue la siguiente:

- session_20251113_122939 (744)
- session_20251113_122522 (272)
- session_20251113_122350 (901)
  Total inicial: 1917 imágenes.

A diferencia de los datasets previos, las sesiones que componen este conjunto no fueron homogéneas ni en extensión ni en calidad. Una de ellas presentó una cantidad significativamente menor de imágenes, reflejo de una captura incompleta. Adicionalmente, la variabilidad visual entre sesiones resultó limitada, y un número considerable de imágenes mostraba únicamente la cuerda sin contexto suficiente para identificar con claridad el estado del nudo o el progreso del procedimiento.

Si bien el volumen inicial de imágenes sugería, en principio, un conjunto adecuado para entrenamientos preliminares, la aplicación de controles iniciales de duplicación mediante pHash evidenció una redundancia superior a la esperada. Dada esta inconsistencia, y considerando además la escasez de información contextual en múltiples segmentos, se

decidió realizar un análisis extraordinario basado en embeddings profundos. Cabe destacar que este análisis no tuvo como objetivo la curación rutinaria del dataset, sino el diagnóstico exhaustivo de su diversidad visual efectiva.

El análisis semántico reveló que una fracción mayoritaria de las imágenes era altamente redundante a nivel de contenido, reduciendo la variabilidad efectiva del conjunto a una porción mínima de su tamaño original. Este resultado evidenció que el Dataset 3 no aportaba información visual suficiente para fortalecer el proceso de entrenamiento, lo que justificó su descarte parcial y motivó la construcción del Dataset 4, incorporando únicamente aquellas sesiones que cumplían con criterios mínimos de calidad visual y variabilidad.

5.3.2 Resultados del diagnóstico

El análisis exhaustivo del Dataset 3 confirmó que este conjunto de datos no era adecuado para ningún tipo de entrenamiento robusto en visión por computador. Aunque inicialmente se esperaba que representara una mejora respecto al Dataset 2, tanto en calidad como en diversidad, los análisis técnicos revelaron múltiples problemas estructurales que comprometían seriamente su utilidad para la clasificación supervisada de nudos quirúrgicos.

El primer hallazgo relevante fue la presencia de 343 imágenes completamente vacías, equivalentes al 17.89 % del total. Estas imágenes, producto de fallos durante la captura en el simulador, correspondían a lienzos uniformes sin presencia de cuerda u otros elementos visuales relevantes, reduciendo de manera significativa la calidad del dataset incluso antes de aplicar procesos de análisis más avanzados.

Posteriormente, se aplicó un proceso de deduplicación perceptual utilizando pHash, que permitió identificar 942 imágenes duplicadas a nivel visual, lo que corresponde al 59.85 % de las imágenes no vacías. Este resultado indica una captura de múltiples frames consecutivos sin variaciones significativas en la geometría o posición del hilo, evidenciando una redundancia temporal severa.

El análisis más concluyente provino de la deduplicación profunda basada en embeddings obtenidos mediante ResNet18, técnica que permite evaluar similitud semántica en un espacio de representación aprendido. Este procedimiento identificó 1373 duplicados semánticos, equivalentes a un 87.2 % de redundancia profunda. Desde la perspectiva del modelo, la gran mayoría de las imágenes correspondía a configuraciones visualmente indistinguibles del mismo estado del nudo, con orientaciones, tensiones y estructuras geométricas prácticamente idénticas.

Como consecuencia directa, el número efectivo de imágenes realmente distintas se redujo a solo 201, una cantidad claramente insuficiente para entrenar un modelo con capacidad

de generalización. Adicionalmente, la ausencia de progresión visual, entendida como transiciones claras entre los distintos estados del nudo, hacía inviable el entrenamiento de un clasificador triclase confiable, ya que el modelo habría tendido a memorizar configuraciones repetidas en lugar de aprender representaciones discriminativas.

Esta falta de variabilidad no solo limita la cobertura del espacio de configuraciones posibles del hilo, sino que también vulnera los supuestos fundamentales de cualquier pipeline de visión por computador. Un conjunto de datos con tan baja diversidad favorece el sobreajuste, invalida la efectividad del data augmentation, dificulta la calibración del modelo y compromete la separación adecuada entre conjuntos de entrenamiento, validación y prueba. En este contexto, incluso técnicas avanzadas de regularización o balanceo no habrían sido suficientes para compensar la carencia de información visual relevante.

En consecuencia, los resultados confirmaron que el Dataset 3 debía considerarse metodológicamente fallido, al no cumplir con los requisitos mínimos de variabilidad necesarios para soportar un modelo de clasificación triclase. Este hallazgo condujo a una reevaluación del protocolo de captura en el simulador SECMA, dando origen al diseño del Dataset 4, construido bajo condiciones más controladas, con ajustes en la progresión del procedimiento y un control más estricto sobre la cadencia de captura. Solo a partir de este rediseño fue posible obtener un conjunto de datos adecuado para entrenamiento, validación e integración posterior en el simulador.

5.4 Dataset 4

5.4.1 Curación final, trazabilidad y balance

El dataset original consistía de múltiples sesiones, capturadas el 25 de noviembre, como se declara a continuación:

- session_20251125_203649 (380)
- session_20251113_122939 (721)
- session_20251125_202319 (398)
- session_20251125_201505 (474)
- session_20251125_204024 (416)
- session_20251125_204227 (351)
- session_20251125_204354 (279)
- session_20251125_205405 (326)
- session_20251125_202012 (423)
- session_20251125_202151 (361)
- session_20251125_200801 (566)
- session_20251125_204840 (314)

- session_20251113_122350 (716)
  Total inicial: 5725 imágenes.

Las imágenes presentaban los siguientes problemas:

1. Imágenes vacías o casi vacías (completamente blancas, negras o con menos del 5 % de información útil).
2. Redundancia temporal, debido a la captura a media tasa de FPS en secuencias muy estáticas.
3. Cuasi-duplicados con diferencias imperceptibles, donde solo variaba pocos píxeles.
4. Duplicación entre sesiones distintas, asociada a grabaciones repetidas o reinicios manuales.

Este conjunto de factores hacía imposible entrenar un modelo robusto sin un proceso riguroso de limpieza.

Eliminación sistemática de imágenes vacías

Se estableció un umbral heurístico basado en proporción de píxeles únicos, desviación estándar de intensidades y detección de frames con información estadística cercana a cero. Una imagen se clasificó como vacía o casi vacía si:

- el 97 % o más de sus píxeles eran idénticos,
- la desviación estándar de los canales RGB era extremadamente baja.

Esto representaba típicamente frames que correspondían a transición entre escenas, errores de captura, inicios o finales abruptos del flujo de video.

Para mantener la trazabilidad del dataset, no se eliminó ninguna imagen**.** En cambio, se creó: dataset_4_empty. Dentro de esta carpeta se almacenaron todas las imágenes vacías, preservando la sesión original.

Una vez removidas las imágenes vacías, se procedió a detectar duplicados. Se utilizó perceptual hashing**,** una técnica robusta ante variaciones mínimas en los píxeles. Esto fue necesario porque muchos duplicados diferían en, un pixel debido al compresor del visor, ruido imperceptible, fluctuaciones pequeñas de iluminación. pHash permitió identificar imágenes perceptualmente idénticas, aun cuando los archivos no fueran binariamente iguales.

Para cada imagen:

1. Se calculó su perceptual hash.
2. Si el hash ya existía en el conjunto de hash únicos, la imagen se marcaba como duplicada.

3. De lo contrario, se añadía al set como imagen única.

Se creó para cada sesión: session_XXXX_DUP/ donde se movieron todos los duplicados detectados, manteniendo la sesión como unidad lógica. Este enfoque permitió mantener intacto el dataset depurado, registrado como: dataset_4_clean/ y al mismo tiempo conservar evidencia del proceso de limpieza.

Resultados cuantitativos del proceso

Luego de eliminar imágenes vacías y mover duplicados, se generó la siguiente tabla final por sesión:

Carpeta	Imágenes restantes
session_20251113_122350	399
session_20251113_122939	421
session_20251125_200801	384
session_20251125_201505	281
session_20251125_202012	272
session_20251125_202151	239
session_20251125_202319	254
session_20251125_203649	286
session_20251125_204024	247
session_20251125_204227	221
session_20251125_204354	177
session_20251125_204840	202
session_20251125_205405	207

Tabla 3:Cantidad de imágenes por sesión luego de la limpieza.

Este resultado mostró una reducción promedio del 25 % al 35 %**,** dependiendo de la sesión. Las sesiones con movimientos más delicados o cambios de cámara produjeron menos duplicados; las sesiones estáticas generaron más reducciones.

Verificación humana posterior y etiquetado

Una vez limpiadas las sesiones:

1. Se revisaron manualmente las carpetas para verificar que:
    - no quedaran imágenes completamente blancas,
    - no quedaran duplicados evidentes,
    - todas las imágenes mantuvieran contexto visual adecuado.

2. Luego se realizó la separación manual en las clases:
    - NO_NUDO
    - UNO_NUDO
    - DOS_NUDO

Organizadas en: dataset_4/dataset_etiquetado/ y finalmente consolidadas en: dataset_4_final/. Se renombraron los archivos para evitar colisiones entre sesiones (usando formato imagen_sesion).

Balance final por clase

El dataset final quedó estructurado así:

Clase	Imágenes
DOS_NUDO	839
NO_NUDO	1761
UN_NUDO	989

Tabla 4:Cantidad de imágenes separadas por clases.

Luego, nos quedan 3.589 imágenes limpias, consistentes y sin ruido.

Conclusiones del proceso de limpieza

El proceso de limpieza permitió transformar un dataset heterogéneo y con abundantes problemas en un conjunto de alta calidad listo para ser utilizado en entrenamiento.

Los beneficios directos fueron:

1. Reducción significativa de ruido visual, eliminando frames inválidos.
2. Disminución del riesgo de overfitting, eliminando duplicados exactos y casi idénticos.
3. Mayor representatividad de la variabilidad real, al remover redundancias temporales excesivas.
4. Coherencia total del flujo PyTorch–Unity, al eliminar imágenes RGBA y convertirlas a RGB.

Este pipeline de limpieza constituye ahora un procedimiento reproducible para futuras etapas del proyecto, incluyendo futuras sesiones RV o incremento del dataset.

5.4.2 Resultados de validación y prueba

El desarrollo del modelo definitivo para integrar en el simulador RV requirió la construcción de un pipeline de entrenamiento más robusto que en etapas anteriores, así como un dataset significativamente más diverso y controlado que los utilizados previamente. Este proceso comenzó con una fase de preentrenamiento estructurado, cuyo propósito fue estabilizar las redes y adaptarlas gradualmente a las características visuales propias de los nudos quirúrgicos capturados en el entorno virtual. Para ello se seleccionaron tres arquitecturas ampliamente utilizadas en visión por computador: ResNet18, ResNet50 y EfficientNet-B0, todas inicializadas con pesos preentrenados en ImageNet, lo cual proporcionó al modelo una base sólida de reconocimiento visual antes de especializarse en el dominio específico del proyecto.

El primer paso consistió en congelar todas las capas convolucionales e introducir una nueva capa clasificadora entrenada exclusivamente sobre las tres clases del problema. Esta etapa de "calentamieto" permitió que el modelo comenzara a identificar patrones distintivos del hilo quirúrgico sin alterar todavía las representaciones profundas heredadas del preentrenamiento. Posteriormente, se procedió a descongelar progresivamente bloques de capas, permitiendo que la red ajustara gradualmente sus filtros internos a la geometría, textura y configuraciones espaciales del hilo en el simulador RV. Este enfoque evitó la pérdida de información importante del preentrenamiento y facilitó una transición estable hacia el ajuste fino completo.

Una vez habilitadas todas las capas, se ejecutó el entrenamiento integral utilizando el optimizador AdamW, una tasa de aprendizaje inicial de 3×10^{-4} y un scheduler que reducía dinámicamente la tasa cuando la pérdida de validación dejaba de mejorar. Se aplicaron técnicas de data augmentation moderado, suficientes para aumentar la variabilidad sin alterar la estructura esencial del nudo. El criterio principal de evaluación en validación fue el F1 macro, métrica más apropiada en tareas triclase con distribuciones internas no uniformes.

Los resultados mostraron un rendimiento altamente competitivo entre las tres arquitecturas. En validación, ResNet18 y EfficientNet-B0 alcanzaron un F1 macro idéntico de aproximadamente 0.9891, mientras que ResNet50 obtuvo un F1 de 0.9843, ligeramente inferior debido a su mayor capacidad y a una tendencia más pronunciada al sobreajuste. Esta similitud entre EfficientNet-B0 y ResNet18 es coherente con sus arquitecturas más compactas y mejor regularizadas, las cuales favorecen un equilibrio óptimo entre complejidad y generalización. Bajo el criterio del pipeline, el modelo seleccionado para evaluación final fue EfficientNet-B0, por mostrar mayor consistencia entre épocas y una convergencia más estable.

La evaluación en el conjunto de prueba, completamente independiente del entrenamiento y la validación, confirmó la solidez del modelo. EfficientNet-B0 alcanzó un Test Loss de

0.0486, una exactitud del 98.52% y un F1 macro de 0.9822, valores que demuestran un excelente equilibrio entre sensibilidad y precisión. La matriz de confusión reveló que:

- DOS_NUDO: 131 correctamente clasificados, 5 confundidos con UN_NUDO.
- NO_NUDO: 257/257 clasificados correctamente (sin errores).
- UN_NUDO: 143 correctamente clasificados, 1 confundido con DOS_NUDO y 2 con NO_NUDO.

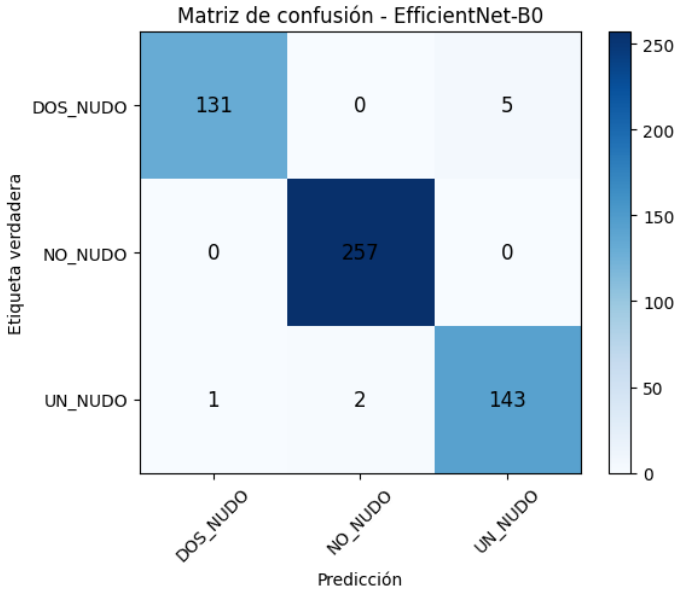

Figura 15: Matriz de confusión del Modelo EfficientNet-B0

El análisis clase por clase mostró un comportamiento notable:

- DOS_NUDO: F1 = 0.9776.
- NO_NUDO: F1 = 0.9961.
- UN_NUDO: F1 = 0.9728.

Estos resultados son especialmente destacables considerando la dificultad intrínseca del reconocimiento del nudo simple, que comparte patrones visuales intermedios tanto con el estado sin nudo como con el nudo doble. Las confusiones observadas fueron escasas y consistentes con la complejidad visual esperada del problema.

Como parte crítica del proceso de validación técnica previo a la integración en el simulador SECMA, el modelo final fue exportado tanto en PyTorch (.pt) como en ONNX, con el fin de garantizar que su comportamiento fuera replicable en tiempo de ejecución mediante el motor Barracuda de Unity. En esta etapa surgió un problema relevante: el modelo había sido exportado inicialmente utilizando opset 18, una versión reciente del

estándar ONNX que no es compatible con las versiones actuales de Unity Barracuda. Debido a esta incompatibilidad, Unity no lograba interpretar correctamente varios de los operadores presentes en el grafo del modelo, generando errores durante la importación y bloqueando su ejecución en el entorno RV.

Para resolver esta limitación, fue necesario reexportar el modelo utilizando opset 11, un estándar considerablemente más estable y ampliamente soportado por Barracuda. Este ajuste implicó la creación de un entorno alternativo de PyTorch y ONNX específicamente configurado para producir archivos compatibles. Una vez exportado el modelo con opset 11, Unity fue capaz de cargarlo sin errores, permitiendo avanzar en la verificación funcional y garantizando su futura integración en el simulador. Con base en esta experiencia, se estableció que opset 11 es el formato más amigable, confiable y recomendado para modelos destinados a ejecutarse en Unity Barracuda, evitando las incompatibilidades asociadas a opsets más recientes.

Una vez resuelto el proceso de exportación, se compararon los logits generados por PyTorch y ONNX utilizando imágenes reales del conjunto de prueba. La diferencia absoluta media fue de 1.86×10^{-4}**,** mientras que la máxima diferencia absoluta alcanzó solo 1.25×10^{-3}**,** valores que se encuentran dentro del rango esperado por diferencias numéricas entre implementaciones. Tras aplicar argmax a las salidas, las predicciones coincidieron en las 539 muestras evaluadas, resultando en un 100% de concordancia entre PyTorch y ONNX. Asimismo, las métricas de exactitud (0.9852) y F1 macro (0.9822) fueron idénticas en ambas plataformas.

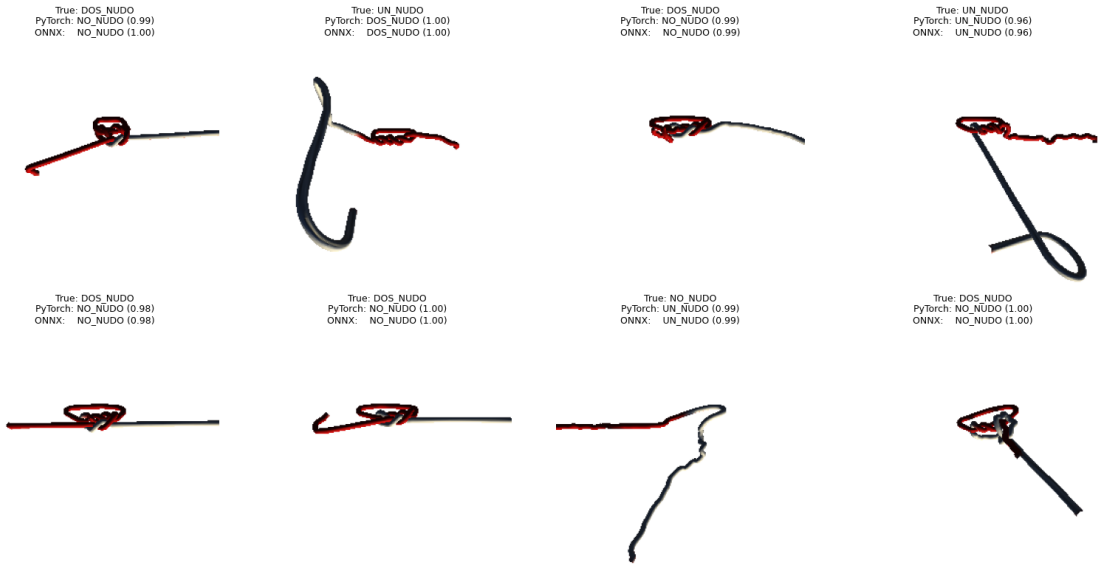

Figura 16:Predicciones del modelo en Pytorch y Onnx

Este resultado confirma que la exportación del modelo ya corregido al opset adecuado, no introduce distorsiones ni pérdida de precisión, y garantiza que su comportamiento dentro del simulador RV será consistente con el observado durante el entrenamiento y la evaluación académica. El pipeline consolidó así la compatibilidad del modelo final EfficientNet-B0 con el entorno Unity, cumpliendo uno de los requisitos funcionales más importantes del proyecto.

5.4.3 Funcionamiento en tiempo real del modelo elegido

El sistema virtual fue compilado y desplegado en el visor autónomo Meta Quest 3, generando una aplicación APK ejecutada directamente en el dispositivo. Durante la realización del ejercicio de sutura, el usuario puede observar en tiempo real los textos de clasificación correspondientes a los estados NO_NUDO, UN_NUDO y DOS_NUDOS, los cuales se actualizan a medida que el modelo analiza continuamente la escena.

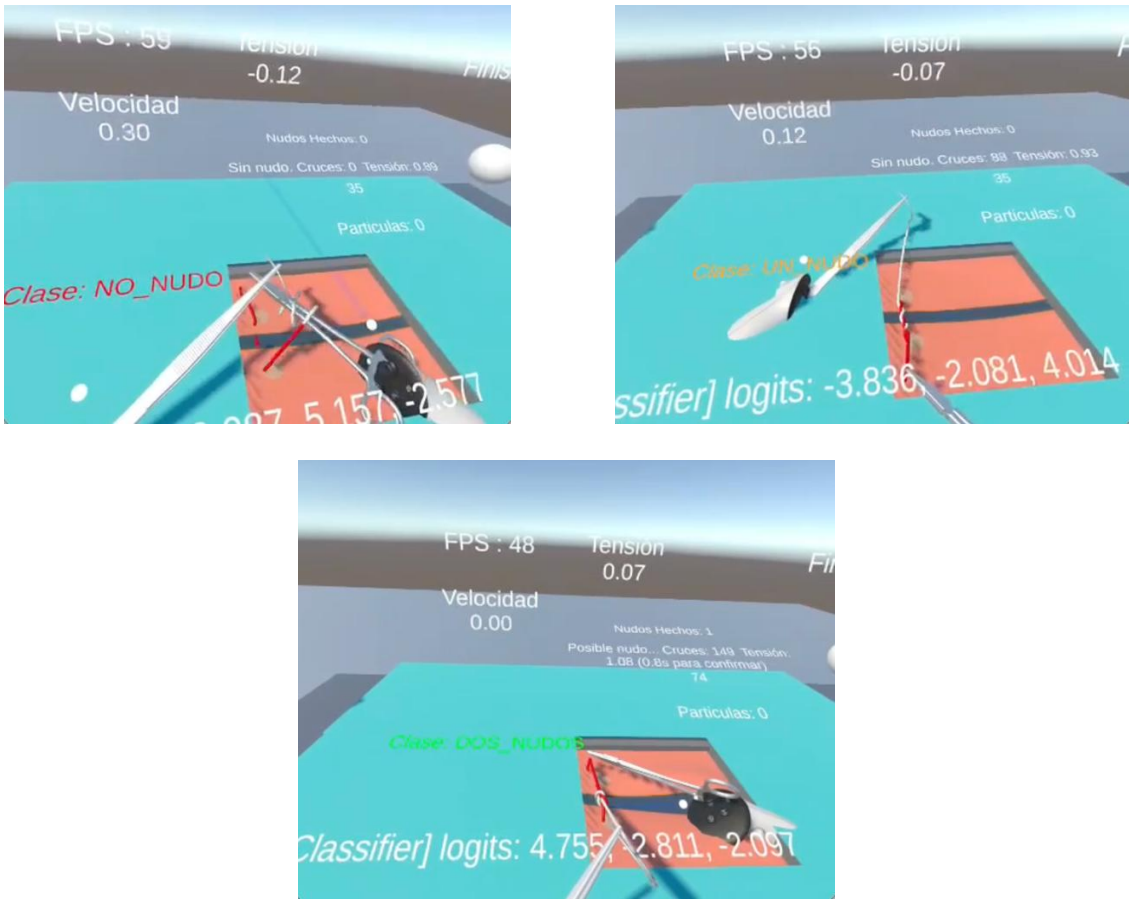

Figura 17: Secuencia en el simulador con las predicciones correctas.

La integración del clasificador dentro del pipeline del simulador evidenció un funcionamiento correcto y estable. La inferencia se ejecuta de manera continua sin introducir retrasos perceptibles ni afectar negativamente la experiencia de usuario. En las

pruebas realizadas, la tasa de refresco del simulador se mantuvo consistentemente por sobre los **50 FPS**, lo que confirma que la ejecución asíncrona del modelo mediante corutinas permite compatibilizar la inferencia de visión por computador con las restricciones de rendimiento propias de un dispositivo autónomo.

Este comportamiento resulta particularmente relevante en hardware con recursos limitados como Meta Quest 3, donde mantener una ejecución fluida es un requisito crítico. En este contexto, el enfoque basado en aprendizaje profundo presenta ventajas claras frente a métodos tradicionales de detección de nudos basados en cálculos numéricos de tensiones, cruces geométricos o invariantes topológicos (por ejemplo, el polinomio de Alexander), los cuales requieren operaciones algebraicas o geométricas complejas que deben evaluarse de forma reiterada durante la simulación. En contraste, una vez entrenado, el modelo neuronal ejecuta inferencias de costo computacional constante, basadas en operaciones vectorizadas eficientes, independientes de la complejidad geométrica del nudo.

Un resultado adicional relevante es el potencial de transferencia del enfoque hacia el dominio físico. Dado que la entrada del modelo consiste únicamente en imágenes RGB, su funcionamiento no depende de la representación interna de la cuerda, del solver físico de Unity ni de las restricciones utilizadas en la simulación. El modelo aprende directamente patrones visuales asociados a la configuración del hilo y del nudo, lo que facilita su eventual adaptación a imágenes provenientes de entornos reales.

En particular, el mismo principio puede extenderse a imágenes capturadas desde maquetas de entrenamiento físico, videos adquiridos mediante cámaras convencionales o sistemas de visión artificial montados sobre bancos experimentales. El modelo no requiere información explícita sobre tensiones, curvaturas o geometría tridimensional del hilo, sino únicamente la observación visual de la configuración resultante. Esto permite plantear escenarios futuros de *fine-tuning* o reentrenamiento para reconocer nudos reales en distintos materiales, calibres de sutura y condiciones de iluminación.

Asimismo, los modelos de visión por computador tienden a aprender representaciones invariantes que se mantienen tanto en el entorno simulado como en escenarios físicos, tales como la forma del lazo, la superposición del hilo y la estructura global del nudo. Esta continuidad semántica favorece un proceso de transferencia Sim2Real considerablemente más directo que el requerido por métodos analíticos basados en geometría o topología del hilo.

Globalmente, estos resultados muestran que el sistema implementado no solo cumple con los requisitos de ejecución en tiempo real dentro del simulador SECMA, sino que además presenta un alto potencial de escalabilidad hacia aplicaciones de entrenamiento quirúrgico físico, abriendo la posibilidad de desarrollar sistemas automáticos de evaluación del desempeño directamente sobre maquetas reales sin modificar la mecánica del ejercicio de sutura.

6 Conclusiones, limitaciones y trabajo futuro

6.1 Conclusiones

El desarrollo de este proyecto permitió **obtener una solución funcional que habilita la incorporación automática de la detección y clasificación del estado del nudo quirúrgico dentro del simulador de realidad virtual SECMA**, cumpliendo de esta forma el **objetivo general** planteado. La integración exitosa de un modelo de visión por computador en el entorno operativo del simulador demuestra la factibilidad técnica de utilizar aprendizaje profundo como apoyo objetivo al entrenamiento de sutura en realidad virtual.

El trabajo no se limitó al entrenamiento de un modelo de clasificación, sino que se estructuró como un proceso integral orientado a abordar los desafíos reales asociados al uso de datos visuales generados artificialmente. Desde las primeras capturas realizadas en SECMA, se evidenció que el problema no se limitaba a la selección de una arquitectura de aprendizaje profundo adecuada, sino que dependía críticamente de la **caracterización y comprensión de las imágenes generadas**, dando cumplimiento al primer objetivo específico del proyecto. En este análisis se identificaron obstáculos relevantes, entre ellos la presencia de canales de transparencia no detectados, una redundancia extrema derivada de altas tasas de captura sin cambios visuales significativos, la fragmentación entre sesiones y un desbalance importante entre clases.

Estos factores motivaron una reformulación del enfoque inicial, situando la curación de datos como una etapa central del pipeline. Si bien técnicas perceptuales como perceptual hashing resultaron eficaces para eliminar duplicados evidentes, su alcance fue limitado frente a variaciones visuales sutiles, lo que condujo a la incorporación puntual de análisis basados en embeddings profundos para cuantificar similitud semántica y diagnosticar redundancias estructurales. Este proceso permitió consolidar una comprensión más precisa de la variabilidad real presente en las imágenes del simulador.

A través de un proceso iterativo de depuración del dataset, experimentación controlada y reconstrucción progresiva del pipeline, se consolidó un conjunto de datos final (Dataset 4) adecuado para entrenamiento y validación. Este dataset capturó distintas configuraciones del nudo quirúrgico bajo condiciones controladas, pero con suficiente diversidad como para permitir una generalización estable, sentando las bases para la **evaluación sistemática de modelos de clasificación**, en concordancia con el segundo objetivo específico.

Los experimentos realizados con distintas arquitecturas demostraron que, bajo un dataset correctamente depurado, modelos como EfficientNet-B0 y ResNet18 alcanzan desempeños comparables, con valores de F1 macro cercanos o superiores a 0.98 en validación y prueba. Este resultado sugiere que, en problemas con patrones visuales bien

definidos, la calidad y estructura del conjunto de datos pueden tener un impacto tan determinante como la elección de la arquitectura. El análisis de errores residuales indicó que las confusiones restantes se asocian principalmente a similitudes geométricas inherentes entre ciertos estados del nudo, más que a limitaciones del modelo.

Finalmente, la **integración del modelo seleccionado en el simulador SECMA**, cumpliendo el tercer objetivo específico, permitió evaluar su comportamiento en un entorno operacional completo, considerando factores propios de la realidad virtual como latencia, fluctuaciones de la tasa de refresco y cargas de procesamiento simultáneas. Las pruebas evidenciaron un funcionamiento estable y fluido, manteniendo tasas de refresco superiores a 50 FPS, lo que confirma que la inferencia asíncrona y el uso de corutinas permiten incorporar modelos de visión por computador sin comprometer la experiencia del usuario.

En términos generales, este trabajo demuestra no solo la viabilidad de entrenar modelos de clasificación visual para el reconocimiento de estados del nudo quirúrgico, sino también la **factibilidad de su despliegue efectivo dentro del entorno SECMA**. Más allá del modelo entrenado, el principal aporte del proyecto radica en la construcción de una infraestructura completa de captura, caracterización, curación, validación e integración, que sienta las bases para el desarrollo de simuladores quirúrgicos más inteligentes, autónomos y pedagógicamente efectivos.

6.2 Limitaciones

Aunque el modelo final logró un rendimiento sobresaliente en condiciones controladas, es importante reconocer las limitaciones estructurales del proyecto para evitar sobredimensionar su alcance.

En primer lugar, el sistema fue entrenado exclusivamente con imágenes procedentes de un entorno RV estático, con ángulos fijos, iluminación invariable y fondos completamente blancos. Esta homogeneidad, aunque beneficiosa para estabilizar el entrenamiento, restringe la capacidad del modelo para generalizar a escenarios más complejos o realistas. En un entorno de práctica quirúrgica real, o incluso en simuladores más avanzados con variabilidad dinámica, la iluminación, el punto de vista, la geometría del hilo y la presencia de elementos distractores podrían alterar significativamente la distribución de características visuales.

Asimismo, la captura de datos dentro del simulador generó redundancia temporal severa: múltiples frames consecutivos sin variación significativa. Aunque se aplicaron técnicas de deduplicación perceptual y profunda, el dataset conserva cierta linealidad interna, lo que puede sesgar el aprendizaje hacia patrones de una sesión específica. Este fenómeno se hizo evidente en el Dataset 3, donde la redundancia llegó casi al 90%, mostrando cuán dependiente es la calidad del dataset de las condiciones de captura.

6.3 Trabajo Futuro

Los resultados obtenidos abren múltiples líneas de desarrollo futuro orientadas a ampliar el alcance y la utilidad del sistema. Una primera dirección consiste en expandir el dataset incorporando sesiones con variaciones controladas en iluminación, ángulos de cámara, velocidad de ejecución y tensión del hilo. Esta diversificación permitiría entrenar modelos con representaciones más invariantes y reducir la dependencia de condiciones ideales. Asimismo, la inclusión de ejemplos con errores deliberados, nudos mal ejecutados o secuencias incompletas facilitaría avanzar hacia sistemas de evaluación más ricos y discriminativos.

Una segunda línea relevante es la incorporación explícita de la dimensión temporal. El proceso de anudado es inherentemente secuencial, por lo que arquitecturas que integren información temporal, como CNN combinadas con LSTM, ConvLSTM, Transformers temporales o Vision Transformers con atención en el tiempo. Estas podrían permitir no solo clasificar estados finales, sino también detectar errores en la ejecución, evaluar fluidez y analizar patrones dinámicos asociados a un desempeño experto.

Adicionalmente, resulta pertinente profundizar en la optimización de la integración en Unity, evaluando métricas como tiempo promedio de inferencia, impacto en el framerate bajo cargas simultáneas y sensibilidad a cambios de resolución o distancia de cámara. Estos análisis permitirían, de ser necesario, incorporar arquitecturas más ligeras o adaptativas (por ejemplo, MobileNetV3) o explorar aceleración mediante motores especializados como ONNX Runtime Mobile.

Finalmente, en una etapa más avanzada, podría explorarse la incorporación de retroalimentación pedagógica automática, donde el sistema no solo clasifica estados, sino que entrega sugerencias precisas sobre cómo mejorar la técnica, identifica errores recurrentes, compara el desempeño del usuario contra curvas de aprendizaje esperadas y genera métricas que puedan utilizarse en estudios clínicos o educativos.

Bibliografía

[1] E. Akritidou *et al*., "Virtual reality simulation training in laparoscopic suturing and knot-tying: A narrative review," *Annals of Laparoscopic and Endoscopic Surgery*, 2023. [En línea]. Disponible en: https://ales.amegroups.org/article/view/10128/html

[2] A. Pérez *et al*., "Embodied hybrid bodystorming to design an XR suture training experience," Universidad Carlos III de Madrid, 2022. [En línea]. Disponible en: https://e-archivo.uc3m.es/rest/api/core/bitstreams/f0aacf6e-6430-472b-a553-80e281e61a57/content

[3] Osso VR, "Research continues to support the benefits of Osso VR for surgical training," *Osso VR Blog*, 2023. [En línea]. Disponible en: https://www.ossovr.com/post/research-continues-to-support-the-benefits-of-osso-vr-for-surgical-training

[4] UploadVR, "FundamentalVR expands its VR surgical simulation platform with new procedures," 2022. [En línea]. Disponible en: https://www.uploadvr.com/surgical-simulation-new-procedures/

[5] Unity Technologies, "Mixed reality surgical training with the VirtaMed LaparoS simulator," *Unity Blog*, 2021. [En línea]. Disponible en: https://unity.com/blog/industry/mixed-reality-surgical-training-with-the-virtamed-laparos-simulator

[6] QAwerk, "TechTalk with Oana Timis from VirtaMed," 2021. [En línea]. Disponible en: https://qawerk.com/blog/techtalk-with-oana-timis-from-virtamed/

[7] A. Rossi *et al*., "A low-cost Unity-based virtual training simulator for laparoscopic partial nephrectomy using HTC Vive," *Journal of Medical Systems*, 2023. [En línea]. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC10588702/

[8] Epic Games, "VR medical simulation from Precision OS trains surgeons five times faster," *Unreal Engine Spotlight*, 2022. [En línea]. Disponible en: https://www.unrealengine.com/en-US/spotlights/vr-medical-simulation-from-precision-os-trains-surgeons-five-times-faster

[9] J. Dequidt *et al*., "The Interactive Medical Simulation Toolkit (iMSTK): An open-source platform for surgical simulation," *Frontiers in Virtual Reality*, vol. 4, 2023. [En línea]. Disponible en: https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2023.1130156/full

[10] M. Zhang *et al*., “Innovating medical education using a cost-effective and scalable VR platform with AI-driven haptics,” *Scientific Reports*, vol. 15, 2025. [En línea]. Disponible en: https://www.nature.com/articles/s41598-025-10543-8

[11] J. Guzmán *et al*., “A new highly portable simulator (SECMA) based on virtual reality for teaching essential skills in minimally invasive surgeries,” Universidad del Desarrollo, 2023. [En línea]. Disponible en: https://repositorio.udd.cl/server/api/core/bitstreams/ff820476-ed3b-4f0c-8568-1a8d7656af39/content

[12] G. Sankaranarayanan et al., “A real-time knot detection algorithm for suturing simulation” *Studies in Health Technology and Informatics*, vol. 142, 2009. [En línea]. Disponible en: https://pubmed.ncbi.nlm.nih.gov/19377170/

[13] L. Qi et al., “Virtual interactive suturing for the Fundamentals of Laparoscopic Surgery (FLS),” *Journal of Biomedical Informatics*, vol. 75, , 2017. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S1532046417302071?via%3Dihub

Anexo A: Definiciones y conceptos fundamentales

A continuación, se presentan los principales términos técnicos utilizados en este trabajo, para garantizar una lectura precisa y accesible a distintos perfiles académicos.

Métricas de Evaluación:

Para evaluar el desempeño de los modelos entrenados en la tarea de clasificación de nudos quirúrgicos, se utilizaron métricas estándar de aprendizaje supervisado que permiten cuantificar la calidad de las predicciones y analizar los tipos de errores cometidos. Estas métricas se basan en la comparación entre las etiquetas reales y las predicciones del modelo, y utilizan como base los siguientes indicadores fundamentales:

- **TP (True Positives):** casos correctamente clasificados como pertenecientes a una clase.
- **TN (True Negatives):** casos correctamente clasificados como no pertenecientes a una clase.
- **FP (False Positives):** casos incorrectamente clasificados como positivos.
- **FN (False Negatives):** casos incorrectamente clasificados como negativos.

A partir de estas cantidades se definen las métricas utilizadas en este proyecto.

Accuracy (Exactitud): Mide la proporción total de predicciones correctas entre todas las evaluadas. Es útil como referencia general cuando las clases están balanceadas.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision (Precisión): Indica qué porcentaje de las predicciones positivas realizadas por el modelo son correctas. Es especialmente relevante cuando los falsos positivos tienen un impacto significativo.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (Sensibilidad): Mide la capacidad del modelo para detectar correctamente los casos positivos reales, penalizando los falsos negativos. Es útil en escenarios donde es importante no omitir instancias positivas.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-Score: Es la media armónica entre **Precision** y **Recall**, y proporciona un equilibrio entre ambas métricas. Es particularmente apropiada en situaciones de desbalance de clases, como ocurre en algunos de los datasets utilizados en este proyecto.

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Un F1 alto indica que el modelo mantiene simultáneamente buena precisión y buena sensibilidad. Variantes en problemas multiclase**:**

- **F1 micro:** Calcula métricas globales sumando todos los TP, FP y FN antes de aplicar la fórmula. Favorece a las clases mayoritarias.
- **F1 macro:** Calcula el F1 de cada clase por separado y luego promedia los resultados, otorgando el mismo peso a cada clase. Es útil cuando las clases están desbalanceadas.
- **F1 weighted:** Promedia el F1 de cada clase ponderándolo por el número de muestras de cada una. Refleja mejor el desempeño general cuando algunas clases dominan el dataset.

Matriz de confusión: Es una tabla de tamaño 3×3(para tres clases) en la que cada celda (i, j)indica cuántas muestras cuya clase real es ifueron clasificadas como clase j. Esta matriz permite identificar patrones de aciertos y errores, y es fundamental para analizar las confusiones entre estados del nudo, especialmente entre *UN_NUDO* y *DOS_NUDO*, que presentan similitudes geométricas.

Función de Pérdida (Cross Entropy Loss)

Es la función que se optimiza durante el entrenamiento. Cuantifica la diferencia entre la distribución predicha y la verdadera distribución one-hot.

$$\mathrm{L} = -\sum_{\mathrm{i}=1}^{\mathrm{C}} \mathrm{y_i} \log(\mathrm{p_i})$$

Donde C es el número de clases, $y_i \in \{0,1\}$ es la etiqueta one-hot, $p_i \in [0,1]$es la probabilidad predicha para la clase i.

Definición de pHash y conceptos asociados

El **perceptual hashing** es una técnica que genera un identificador numérico, denominado hash perceptual, que resume el contenido visual de una imagen de manera que imágenes similares producen hashes también similares, incluso si presentan variaciones menores como cambios de tamaño, compresión o leves variaciones de iluminación.

El **pHash** es una de las implementaciones más usadas de perceptual hashing. Basado en transformadas como la DCT (Discrete Cosine Transform)**,** extrae los componentes de baja frecuencia de la imagen, los cuales capturan la estructura visual global, y a partir de ellos genera un hash binario compacto.

Para comparar dos hashes perceptuales se utiliza la distancia Hamming, que mide cuántos bits difieren entre ambos:

dist_Hamming(h1, h2) = número de bits diferentes entre h1 y h2.

En pHash, un valor de distancia Hamming bajo, por ejemplo, menor o igual 5, indica que las imágenes son muy similares. Los valores altos sugieren imágenes diferentes.

Modelos CNN pre-entrenados

Las redes neuronales convolucionales (CNN) constituyen la base de la mayoría de los sistemas modernos de visión por computador. Su capacidad para aprender representaciones jerárquicas, desde patrones locales simples, como bordes y texturas, hasta estructuras visuales más complejas, las convierte en una herramienta fundamental para la clasificación de imágenes. En este proyecto se empleó la técnica de transfer learning, aprovechando modelos previamente entrenados en grandes conjuntos de datos como ImageNet, con el fin de reducir el tiempo de entrenamiento y mejorar la capacidad de generalización, especialmente en escenarios con conjuntos de datos limitados.

Desde el punto de vista computacional, una imagen no es interpretada por la red como una entidad visual, sino como un tensor numérico de dimensión $H \times W \times 3$, donde cada píxel está representado por tres valores de intensidad correspondientes a los canales rojo, verde y azul (RGB). Para que esta información pueda ser procesada por una CNN, las imágenes deben someterse a un conjunto de transformaciones previas que incluyen la conversión a tensores, la homogeneización espacial mediante redimensionamiento, y la normalización de los valores de intensidad. Estas transformaciones permiten situar los datos en un espacio numérico adecuado para la optimización, asegurando estabilidad durante el entrenamiento y coherencia dimensional en la entrada del modelo.

A lo largo del proyecto se evaluaron varias arquitecturas CNN reconocidas por su eficiencia, profundidad y capacidad de representación. Los modelos analizados fueron los siguientes:

ResNet18 pertenece a la familia de Redes Residuales (ResNet)**,** introducidas para mitigar el problema del desvanecimiento del gradiente mediante conexiones residuales. Estas permiten que el flujo de información atraviese la red sin degradarse, mejorando la estabilidad del entrenamiento en redes profundas. ResNet18 es una arquitectura

relativamente ligera, con 18 capas, lo que la hace rápida de entrenar y adecuada para tareas con recursos computacionales moderados.

ResNet50 es una versión más profunda de la arquitectura residual, con 50 capas y bloques convolucionales más complejos (bottleneck blocks). Su mayor capacidad expresiva permite capturar patrones más ricos, aunque también la vuelve más susceptible al sobreajuste cuando los datos son limitados.

EfficientNet-B0 forma parte de la familia EfficientNet, diseñada mediante un proceso de compound scaling que ajusta simultáneamente la profundidad, el ancho y la resolución de la red de manera equilibrada. Es reconocida por ofrecer una relación óptima entre rendimiento y eficiencia computacional.

MobileNetV3-Small es una arquitectura optimizada para dispositivos móviles y aplicaciones de bajo consumo, utilizando técnicas como depthwise separable convolutions y módulos squeeze-and-excitation. Aunque eficiente y liviano, su capacidad representacional es menor que la de los otros modelos evaluados.

Hiperparámetros del entrenamiento

El proceso de entrenamiento de los modelos se estructuró en torno a un conjunto de hiperparámetros clave que determinan la dinámica de aprendizaje, la estabilidad de la optimización y la capacidad de generalización del sistema. La correcta selección y ajuste de estos parámetros resulta fundamental para obtener un desempeño robusto, especialmente en escenarios con conjuntos de datos limitados o con alta redundancia visual, como los utilizados en este proyecto.

En primer lugar, el número de **épocas** definió cuántas veces el modelo recorrió el conjunto de entrenamiento de manera completa. Cada época permite ajustar progresivamente los parámetros internos de la red, refinando su capacidad de representación. Complementariamente, el **batch size** determinó cuántas imágenes se procesaron simultáneamente antes de actualizar los pesos. Se adoptaron tamaños moderados (por ejemplo, 32 imágenes), lo que proporcionó un equilibrio adecuado entre estabilidad del gradiente y eficiencia computacional, tanto en GPU como en aceleradores MPS de Apple Silicon.

La **tasa de aprendizaje (learning rate)** controló la magnitud de las actualizaciones de los pesos en cada iteración. Valores excesivamente altos pueden causar inestabilidad y divergencia, mientras que valores demasiado bajos ralentizan la convergencia. Debido a esto, se utilizaron valores moderados acompañados de estrategias adaptativas. En particular, se emplearon los optimizadores **Adam** y **AdamW**: el primero por su estabilidad inicial y rápida convergencia, y el segundo como optimizador principal en los

modelos finales, dada su capacidad para desacoplar el término de regularización y manejar de mejor manera el ajuste fino de arquitecturas preentrenadas.

El hiperparámetro **weight decay** actuó como un término de regularización que penaliza pesos de gran magnitud, ayudando a evitar el sobreajuste, especialmente en fases donde el modelo comienza a memorizar patrones específicos del dataset. Para mejorar aún más la estabilidad del proceso, se utilizó un **scheduler** que reduce automáticamente la tasa de aprendizaje cuando la pérdida de validación deja de mejorar, permitiendo refinamientos más precisos en etapas avanzadas y mitigando oscilaciones indeseadas.

En el contexto de **transfer learning**, se aplicó una estrategia de entrenamiento en etapas, combinando **congelación parcial**, **descongelación progresiva** y **fine-tuning completo**. El **fine-tuning** consiste en ajustar los pesos de un modelo previamente entrenado en un gran conjunto de datos, como ImageNet, para especializarlo en una nueva tarea. Inicialmente, se congelaron todas las capas convolucionales para entrenar únicamente la última capa clasificadora, adaptando rápidamente la red al dominio de los nudos quirúrgicos. Posteriormente, se descongelaron gradualmente los bloques superiores para permitir que sus filtros se ajustaran a características más finas y específicas del problema. Finalmente, se ejecutó un fine-tuning total con una tasa de aprendizaje reducida, lo que permitió una adaptación profunda sin comprometer el conocimiento previamente adquirido.

Para mejorar la capacidad de generalización, se incorporaron técnicas de **data augmentation**, aplicando transformaciones visuales moderadas que no alteran la geometría esencial del nudo. Estas incluyeron ligeras rotaciones, variaciones sutiles de brillo y pequeños desplazamientos espaciales. Dichos aumentos permitieron incrementar la variabilidad efectiva del dataset, reduciendo el riesgo de sobreajuste y fortaleciendo la robustez del modelo frente a variaciones naturales del entorno de captura.

Finalmente, para la implementación:

ONNX es un formato abierto que permite intercambiar modelos de aprendizaje profundo entre distintos frameworks, como PyTorch o TensorFlow. Al exportar un modelo a ONNX, su estructura y pesos quedan en un formato estándar que puede ser ejecutado por diversas plataformas. En este proyecto se utilizó para trasladar el modelo entrenado en PyTorch hacia el entorno del simulador RV.

Unity es un motor de simulación y desarrollo interactivo ampliamente utilizado en videojuegos, realidad virtual y aplicaciones 3D. Dentro de este proyecto actúa como la plataforma base del simulador SECMA, proporcionando renderizado 3D en tiempo real, control de cámaras y escenas, interacción con dispositivos RV, integración con el motor de inferencia Barracuda. Unity permite presentar al usuario el entorno inmersivo donde se

ejecuta la tarea de anudado y donde se evalúan las imágenes procesadas por el modelo de IA.

Barracuda (Motor de inferencia de Unity) es la librería oficial de Unity para ejecutar modelos ONNX directamente dentro del motor, soportando dispositivos como CPU, GPU y aceleradores dedicados. Su función es interpretar el grafo del modelo, procesar tensores y producir predicciones en tiempo real.

Meta Quest es un visor de realidad virtual autónomo que proporciona seguimiento de manos, controladores hápticos (generan vibración) y renderizado estereoscópico en tiempo real. En el simulador SECMA, Meta Quest constituye el dispositivo a través del cual el usuario experimenta el entorno de entrenamiento quirúrgico, manipula las herramientas virtuales y ejecuta los movimientos del nudo. Su integración con Unity permite capturar imágenes desde cámaras virtuales internas del simulador, las cuales alimentan el modelo de clasificación para evaluar el estado del nudo durante el procedimiento