Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

TC3007C

Inteligencia artificial avanzada para la ciencia de datos II

MODELO DE CLASIFICACIÓN DE POSICIONES PARA GANADO BOVINO (1ST VERSION-TENSORFLOW)

Autores:

A01368818 Joel Sánchez Olvera

A01661090 Juan Pablo Cabrera Quiroga

A01704076 Adrián Galván Díaz

A01708634 Carlos Eduardo Velasco Elenes

A01709522 Arturo Cristián Díaz López

Índice

Abstracto

Este reporte detalla el desarrollo de un modelo de clasificación de imágenes basado en una red neuronal convolucional (CNN), implementada utilizando TensorFlow, la cual fue desarrollada para identificar y clasificar de los datos entrantes que son imágenes de vacas, desde un ángulo superior, en dos categorías: **"vaca acostada"** y **"vaca de pie"**. A partir de un dataset de imágenes previamente clasificadas y organizadas en carpetas, aplicamos técnicas de preprocesamiento y ajuste de pesos por clase para manejar el desbalance en los datos y optimizar el rendimiento del modelo.

Técnica de Modelado

Arquitectura

Modelo: Red Neuronal Convolucional (CNN) en TensorFlow

Como mencionamos anteriormente el modelo del proyecto se basó en una red neuronal convolucional implementada con TensorFlow, la meta principal del modelo es clasificar correctamente una imagen dentro de las dos categorías mencionadas : **“vaca de pie”** y **“vaca acostada”**, para cumplir con dicha meta

Optimización y Entrenamiento

- Optimizador Adam
- Learning Rate 0.001.
- Función de pérdida: Binary Cross Entropy.

Las redes neuronales convolucionales (CNN) fueron seleccionadas para este proyecto debido a su destacada capacidad para analizar y clasificar imágenes con alta precisión y robustez. Las razones más importantes fueron:

1. **Eficiencia en el Procesamiento de Imágenes**: Las CNN son arquitecturas diseñadas específicamente para procesar datos espaciales, como imágenes, extrayendo automáticamente características visuales relevantes (bordes, texturas y formas).
2. **Capacidad de Abstracción**: A través de múltiples capas convolucionales y de agrupamiento (pooling), las CNN construyen una jerarquía de características, desde patrones simples hasta representaciones más complejas y abstractas, volviendolas ideales para clasificar imágenes con variaciones en iluminación, posición o perspectiva.

3. **Generalización Mejorada**: A través de técnicas de regularización como Dropout y Data Augmentation, las CNN son menos propensas al overfitting, asegurando que el modelo pueda generalizar mejor a datos no vistos.

Beneficios Comparativos sobre Otras Arquitecturas

1. **Redes Neuronales Densas (Fully Connected Networks)**: Las redes densas no están optimizadas para imágenes, ya que consideran cada píxel de manera independiente y no explotan la estructura espacial inherente de las imágenes.
2. **Redes Recurrentes (RNNs)**: Aunque las RNNs son efectivas para datos secuenciales como texto o series temporales, no son apropiadas para tareas de clasificación de imágenes debido a su incapacidad para captar relaciones espaciales en los datos.

Supuestos de Modelado

Para éste modelo de clasificación, los supuestos principales se refieren a la calidad de las imágenes y las circunstancias de iluminación en las que se encuentran, ya que dado a que las fotos se toman cada 5 minutos a cualquier hora del día, tenemos diferentes condiciones de Iluminación en el dataset:

- **Variaciones de Iluminación:** Dentro del dataset, tenemos imágenes tomadas en la mañana del día que aparecen tener una saturación de color muy alta pero en realidad es debido a la luz de la imagen, así como fotos de noche que son muy oscuras y poco distinguibles.
- **Brillo de las imágenes:** A pesar del desbalance natural entre clases y las condiciones de iluminación, las imágenes seleccionadas para el modelo representan principalmente condiciones de iluminación diurna.
- **Desbalance en las Clases**: La categoría "vaca acostada" contiene más ejemplos que "vaca de pie". Para mitigar este desbalance implementamos un tope de imágenes por clase, asegurando que tenían una participación equitativa de 280 imágenes por clase.

Diseño de las Pruebas

Para evaluar de manera efectiva el desempeño del modelo, se implementó un enfoque sistemático basado en la división del dataset, el preprocesamiento de imágenes y el uso de métricas de evaluación.

División del Dataset

Las proporciones para la división del dataset, fueron las siguientes:

1. **Conjunto de Entrenamiento (80%)**:

    - Este conjunto se utilizó para ajustar los pesos del modelo mediante retropropagación, permitiendo al modelo aprender patrones clave de las imágenes.
2. **Conjunto de Prueba (20%)**:
    - Reservado para la evaluación final del modelo.
    - Este conjunto mide la capacidad del modelo para generalizar en datos no vistos anteriormente.

Preprocesamiento de las Imágenes

Cada imagen pasó por un proceso de preprocesamiento antes de ser utilizada en el modelo. Esto incluyó:

1. **Redimensionamiento**:
    - Las imágenes originales fueron escaladas a un tamaño estándar de **224x224 píxeles**.
2. **Normalización**:
    - Los valores de los píxeles se ajustaron al rango [0,1] para estabilizar el entrenamiento y acelerar la convergencia.
3. **Aumentos de Datos (Data Augmentation)**:
    - Para incrementar la diversidad del dataset y mejorar la capacidad del modelo para generalizar, se aplicaron transformaciones como:
        - **Rotaciones aleatorias**
        - **Volteos horizontales**
        - **Cambios en el brillo y contraste**
        - **Zoom aleatorio**

Método de Entrenamiento

El modelo fue entrenado utilizando un esquema iterativo por **8 épocas**, con un tamaño de lote (**batch size**) de **32 imágenes** por iteración. La función de pérdida utilizada fue **Binary Cross Entropy**, optimizada mediante el algoritmo **Adam**.

Durante el entrenamiento, se monitorearon métricas clave en el conjunto de validación, incluyendo:

- Precisión (Accuracy).
- Pérdida (Loss).

Ajuste de Parámetros

Parámetros:

1. **img:224**

   a. Se eligió un tamaño de imagen de 224 x 224 píxeles para el procesamiento del modelo. Este tamaño es estándar en arquitecturas de CNNs, permiten un equilibrio entre la preservación de detalles esenciales en las imágenes y el uso de memoria y tiempo de procesamiento.
2. **batchsize:32**
   a. Se utilizó un tamaño de lote (batch size) de 32 imágenes por iteración. Este valor nos ayuda a tener un equilibrio entre la demanda del aprendizaje y el uso eficiente de la memoria.
3. **epochs: 8**
   a. El modelo entrenó durante 8 épocas, este número fue seleccionado para proporcionarle el tiempo suficiente de exposición a los datos, sin incurrir en overfitting. La decisión de usar este valor fue tomada en base al monitoreo de la precisión y pérdida del modelo conforme fue avanzando en el entrenamiento.
4. **device: GPU (CUDA, MPS)**
   a. El entrenamiento del modelo se realizó en la GPU utilizando CUDA Y MPS, lo que aceleró significativamente el entrenamiento sobre los datos y la optimización de los pesos.
5. **Learning Rate:0.001**
   a. El Learning Rate fue configurado en 0.001. Este valor se seleccionó para garantizar que el modelo se actualiza y ajusta de manera estable, observamos que fue el Learning Rate con mejor resultado.

Descripción del Modelo

1. Modelo y Propósito

El modelo desarrollado es una **Red Neuronal Convolucional (CNN)** implementada en TensorFlow, diseñada específicamente para clasificar imágenes aéreas de vacas en dos categorías: "vaca acostada" y "vaca de pie". Este modelo tiene aplicaciones prácticas en la solución del reto pues nos da una clasificación precisa para usarse en el sistema de CAETEC.

2. Precisión y Robustez Esperada

Durante el entrenamiento, se utilizaron configuraciones específicas, como un tamaño de imagen de **224 x 224 píxeles**, un **batch size de 32** y un entrenamiento por **8 épocas**, para optimizar el modelo para este conjunto de datos. La arquitectura de la CNN fue seleccionada por su capacidad para manejar variaciones moderadas en las imágenes, como cambios en la iluminación y el contraste, con la arquitectura mencionada nos

aseguramos de que el modelo puede reconocer las imágenes en diferentes condiciones y la robustez del modelo y su resistencia a cambios es buena.

3. Interpretación del Modelo y Dificultades

La CNN fue diseñada para extraer características relevantes, como bordes, texturas y formas, que permitan clasificar con precisión la posición de las vacas. Sin embargo, debido a la complejidad del modelo, tuvimos que realizar ajustes a los hiperparámetros como el Learning Rate y los pesos que se le daban a cada clase para reconocerla de manera más precisa.

Características Útiles del Modelo

Algunas características destacadas de la implementación incluyen:

- **Data Augmentation:** Técnicas como rotación, cambio de brillo y volteo horizontal ayudaron a incrementar la diversidad de los datos.
- **Regularización con Dropout:** Evitó el sobreajuste al introducir aleatoriedad en la activación de neuronas durante el entrenamiento.

4. Parámetros

El modelo fue entrenado utilizando los siguientes hiperparámetros:

- **Tasa de aprendizaje:** Se configuró una tasa inicial de 0.001 utilizando el optimizador Adam.
- **Épocas de entrenamiento:** Se establecieron 8 épocas, suficientes para alcanzar la convergencia sin incurrir en overfitting.
- **Tamaño del lote (*batch size*):** Se utilizó un tamaño de lote de 32 imágenes por iteración.
- **Tamaño de imagen:** 224 x 224 pixeles.

5. Descripción Técnica del Modelo

La arquitectura del modelo fue definida como:

- **Capa de Entrada** (224x224x3)
- **1ra Capa Convolucional** (32, 3x3, activación: ReLU)
- **1ra Capa de MaxPooling2d** (2x2)
- **2da Capa Convolucional** (64, 3x3, activación: ReLU)
- **2da Capa de MaxPooling2d** (2x2)
- **3ra Capa Convolucional** (128, 3x3, activación: ReLU)
- **3ra Capa de MaxPooling2d** (2x2)

- **Capa Flatten**
- **1ra Capa Densa** (128, activación: ReLU, dropout: 0.5).
- **Capa de Salida** (1, activación: Sigmoid).

Capa de Entrada:

- Dimensiones de las imágenes: 224x224x3.
- Preprocesamiento: Escalado de los valores de los píxeles al rango [0,1].

Capas Convolucionales:

- **Primera Capa**: 32 filtros con un tamaño de núcleo de 3x3, activación ReLU.
- **Segunda Capa**: 64 filtros con un tamaño de núcleo de 3x3, activación ReLU.
- **Tercera Capa**: 128 filtros con un tamaño de núcleo de 3x3, activación ReLU.

Capas de Agrupación (MaxPooling2d):

- Reducen la dimensionalidad mediante ventanas de 2x2, preservando las características más importantes y disminuyendo la carga computacional.

Capa Flatten:

- Convierte las matrices resultantes en vectores unidimensionales, preparando los datos para la etapa de clasificación.

Capas Densas:

- Una capa densa completamente conectada con 128 unidades, activación ReLU, y regularización mediante Dropout (tasa de 0.5).
- Capa de salida con una unidad y activación Sigmoid, adecuada para tareas de clasificación binaria.

6. Justificación de Arquitectura

Teniendo en cuenta las limitaciones y objetivos del proyecto, la arquitectura de la red neuronal convolucional (CNN) utilizada en el proyecto fue seleccionada específicamente para equilibrar precisión, eficiencia computacional y facilidad de implementación, es relativamente sencilla y sigue un diseño estándar para un problema de clasificación de imágenes binario. Así evitamos la sobre complicación del modelo, que podría llevar a un sobreajuste en lugar de generalización. El diseño y modelo de la arquitectura con sus utilidades:

1. **Capas Convolucionales**:

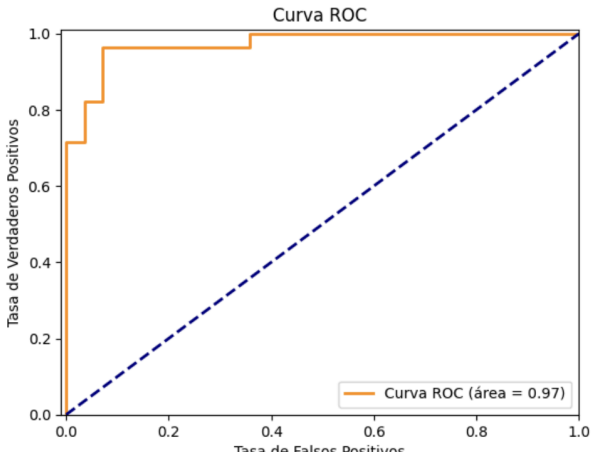
    i. Implementamos 3 bloques de capas convolucionales para capturar las características esenciales de las imágenes (como bordes, texturas y formas). Más capas no añadirían un beneficio significativo, dada la naturaleza y resolución de las imágenes (224x224 píxeles).
    ii. Los tamaños de filtro (3x3) y activaciones ReLU son estándar y probados para detectar patrones espaciales de manera efectiva.
2. **Capas de Agrupamiento (MaxPooling)**:
    - Estas capas reducen la dimensionalidad de los datos, manteniendo características clave. Esto no solo disminuye la carga computacional, sino que también ayuda a mitigar el riesgo de sobreajuste.
3. **Capas Densas y Regularización**:
    - La inclusión de una capa densa con 128 neuronas y activación ReLU permite combinar las características extraídas para realizar la clasificación final.
    - El dropout con una tasa de 0.5 ayuda a evitar el sobreajuste al introducir aleatoriedad durante el entrenamiento.
    - La capa de salida con activación sigmoid convierte los valores en probabilidades, lo que es apropiado para tareas de clasificación binaria.

7. Resultados y Patrones Detectados

Los resultados obtenidos durante el entrenamiento y prueba destacan el desempeño robusto del modelo:

- **Precisión (P):** 96%.
- **Exhaustividad (Recall):** 96%.
- **F1-Score promedio:** 96%.

Curva ROC



La curva ROC muestra un área bajo la curva (AUC) de 0.97, indicando que el modelo tiene una alta capacidad para distinguir entre las dos clases, minimizando falsos positivos y falsos negativos.

8. Comportamiento del Modelo

Durante el entrenamiento del modelo, identificamos que si bien presentan un buen rendimiento como se describe a continuación, el proceso de entrenamiento es un poco lento debido a la demanda computacional y el poder de cómputo disponibles.

1. **Precisión:** El modelo demuestra una alta capacidad para clasificar correctamente ambas categorías, con un 96% de precisión promedio.
2. **Recall:** Se asegura de identificar la mayoría de los casos relevantes, evitando omisiones críticas.
3. **Robustez :** Las métricas indican que el modelo no depende exclusivamente de patrones específicos del conjunto de entrenamiento, sino que generaliza bien a nuevas imágenes.

9. Interpretación del Comportamiento del Modelo

9.1 Precisión (Precision)

El modelo alcanzó una precisión promedio de **96%**, lo que indica su efectividad para minimizar los falsos positivos. Esto es crucial en el contexto del proyecto. El modelo es confiable para identificar correctamente a las vacas en sus respectivas categorías.

9.2 Exhaustividad (Recall)

Con un recall promedio de **96%**, el modelo demostró una alta capacidad para detectar todas las instancias relevantes en ambas categorías. El modelo tiene un bajo nivel de omisión de vacas en cualquiera de las categorías, lo que lo hace adecuado para aplicaciones de monitoreo continuo.

9.3 F1-Score

El F1-Score promedio de **96%** refleja un balance óptimo entre precisión y recall. Este equilibrio demuestra que el modelo no está sesgado hacia una categoría específica.

9.4 Curva ROC

La **Curva ROC** alcanzó un área bajo la curva (**AUC**) de **0.97**, lo que resalta la capacidad del modelo para distinguir correctamente entre las dos categorías.

Evaluación del Modelo

Precisión Global

- **Valor:** 96.43%.
- **Interpretación:**
    - El modelo logró clasificar correctamente el 96.43% de las imágenes del conjunto de prueba, confirmando un alto nivel de confiabilidad para esta tarea.

Pérdida

- **Valor Final en Prueba:** 0.1661.
- **Interpretación:**
    - Una pérdida baja indica que las predicciones del modelo se aproximan con precisión a las etiquetas reales, lo que sugiere que los parámetros ajustados y las configuraciones del modelo son apropiados.

Reporte de Clasificación

Clase	**Precisión**	**Recall**	**F1-Score**	**Soporte**
Acostada	96%	96%	0.96	28
Parada	96%	96%	0.96	28
Macro avg	**96%**	**96%**	**0.96**	**56**
Weighted avg	**96%**	**96%**	**0.96**	**56**

Ambas clases tienen un desempeño equivalente, lo que sugiere que el modelo no tiene sesgo hacia ninguna categoría.

El modelo desarrollado alcanzó un buen desempeño en la clasificación de imágenes de vacas, cumpliendo con los objetivos planteados. Las métricas obtenidas reflejan su robustez y capacidad para generalizar.

Parámetros revisados y ajustados

Durante el proceso de entrenamiento y ajuste del modelo CNN, se realizaron diversas modificaciones a los hiperparámetros con el objetivo de mejorar su desempeño y garantizar su capacidad de generalización. La **tasa de aprendizaje** inicial de 0.01 fue reducida a 0.001 para estabilizar las actualizaciones de los pesos y mejorar la convergencia. Asimismo, se incrementó la **tasa de dropout** de 0.3 a 0.5 en las capas densas finales, lo que permitió reducir el riesgo de overfitting.

Otro aspecto clave fue el ajuste de los **pesos por clase** para tratar el desbalance entre "vaca acostada" y "vaca de pie", y limitamos las imágenes a un cierto número de imágenes por clase, lo que garantizó que ambas categorías tuvieran una representación equitativa en el cálculo de la pérdida. Por otro lado, el número de **épocas** fue reducido de 10 a 8, ya que se observó que las métricas principales se estabilizaron alrededor de la sexta época, evitando así el sobreajuste.

Para mejorar la robustez del modelo frente a variaciones en las imágenes, se implementaron técnicas avanzadas de **data augmentation**, como rotaciones aleatorias en un rango de [-15°, 15°], cambios de brillo y contraste, y volteos horizontales. Estas transformaciones incrementaron la diversidad del dataset, simulando condiciones reales de captura. Adicionalmente, el modelo fue ajustado para procesar imágenes de **224 x 224 píxeles**, en lugar de 128x128, lo que permitió preservar detalles visuales importantes para la clasificación.

Pruebas del Modelo

Una vez entrenado el modelo y teniendo un modelo eficiente para ser probado, tomamos una pequeña muestra de los datos con la cual no tiene ninguna interacción el modelo, con el fin de medir sus capacidades de reconocimiento y generalización para nuevas imágenes. Realizamos predicciones utilizando el modelo entrenado en TensorFlow. El proceso consistió en evaluar el modelo sobre el conjunto de datos mencionado, destacamos los hallazgos más relevantes:

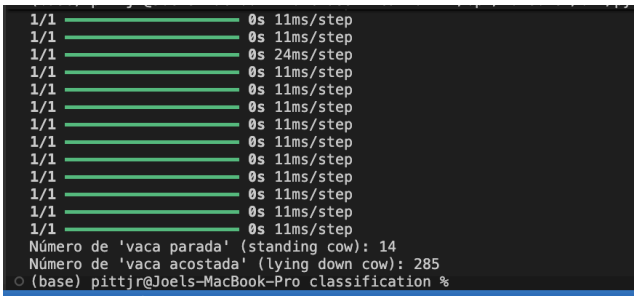


Figura 1.1.Resultados Prueba Vaca Parada Figura 1.2.Resultados Prueba Vaca Acostada

Resultados de las Pruebas

1. **Prueba con Imágenes únicamente de "Vaca Parada" (Figura 1.1)**:
    - Se utilizó un conjunto de imágenes que contiene únicamente ejemplos de la clase "vaca parada".
    - **Resultados**:
        - El modelo clasificó correctamente **23 imágenes** como "vaca parada".
        - Sin embargo, clasificó incorrectamente **5 imágenes** como "vaca acostada".
    - **Interpretación**:
        - Aunque la mayoría de las imágenes fueron clasificadas correctamente, la presencia de errores indica que el modelo enfrenta dificultades al diferenciar esta clase en algunos casos, posiblemente debido a similitudes visuales entre las clases o a un sesgo aprendido en los datos de entrenamiento.
2. **Prueba con Imágenes únicamente de "Vaca Acostada" (Figura 1.2)**:
    - Se utilizó un conjunto de imágenes que contiene únicamente ejemplos de la clase "vaca acostada".
    - **Resultados**:
        - El modelo clasificó correctamente **265 imágenes** como "vaca acostada".
        - Clasificó incorrectamente **34 imágenes** como "vaca parada".
    - **Interpretación**:
        - En este caso, aunque el modelo muestra un mejor rendimiento general, todavía presenta errores en una cantidad no trivial de imágenes. Esto sugiere que las características visuales del dataset pueden influir en la capacidad del modelo para distinguir correctamente entre ambas clases.

Eficiencia General

1. **Desempeño por Clase**:
    - El modelo muestra un desempeño aceptable en ambas pruebas, con un número mayor de predicciones correctas que incorrectas.
    - No obstante, los errores observados en ambas pruebas indican que el modelo aún tiene limitaciones en la generalización y robustez, especialmente frente a variaciones visuales no presentes en el conjunto de entrenamiento.
2. **Posibles Causas de los Errores**:
    - **Variabilidad de los Datos**: Las imágenes de prueba pueden tener características que el modelo no encontró en los datos de entrenamiento, como diferencias en iluminación, ángulo o calidad.
    - **Desbalance de Clases**: Aunque se implementaron ajustes para equilibrar las clases durante el entrenamiento, el modelo aún puede tener un leve sesgo hacia patrones aprendidos de una clase en particular.

Conclusión

El modelo de clasificación de posiciones para ganado bovino basado en una red neuronal convolucional (CNN) implementada en TensorFlow cumple los criterios establecidos de minería de datos:

1. **Precisión en la Clasificación**
    - El modelo logró una precisión promedio del 96%, superando ampliamente el umbral requerido del 85%.
    - El balance entre precisión y exhaustividad se evidencia en un F1-Score promedio de 96%, demostrando que el modelo es confiable tanto para identificar "vacas acostadas" como "vacas de pie".
2. **Capacidad de Generalización**
    - La curva ROC con un AUC de 0.97 indica una alta capacidad para distinguir entre las dos categorías, minimizando falsos positivos y negativos.
3. **Optimización y Eficiencia**
    - Aunque el modelo muestra un desempeño destacado, el tiempo de entrenamiento fue elevado debido a las demandas computacionales. Esto justifica la propuesta de migrar a PyTorch para aprovechar mejor los recursos de hardware y optimizar el proceso.

Por lo tanto, el modelo no solo satisface los criterios técnicos establecidos, sino que se posiciona como una herramienta eficiente y precisa para la clasificación de posturas en las vacas.

Mejoras a Implementar (Migración a Pytorch)

Los resultados obtenidos durante las pruebas, combinados con las observaciones anteriores, justifican la implementación de mejoras tanto en el modelo como en los datos utilizados.

Durante el desarrollo del modelo en TensorFlow, se identificaron ciertas limitaciones que justifican una migración a PyTorch para las siguientes iteraciones del proyecto. Siendo la mayor de éstas el tiempo que se tardaba el modelo en procesar las imágenes, si bien el modelo tenía un buen rendimiento, el proceso de entrenamiento fue más lento debido a su alta demanda computacional.

Así mismo, se nos presentó el problema de que el modelo,a pesar de ajustar los hiper parámetros, estaba teniendo un buen resultado en su precisión y otras métricas, pero al usarlo en datos nuevos, el modelo carece un poco en la precisión para clasificar y generalizar los datos.

Además, aunque TensorFlow permite aprovechar GPUs, la configuración y el manejo del hardware resultaron menos eficientes en comparación con PyTorch. La migración a PyTorch nos permitió superar estas limitaciones al aprovechar mejor tecnologías como **CUDA** y **MPS**, lo que aceleró significativamente el entrenamiento al reducir el tiempo de cómputo por época.

A continuación, se explican las principales razones detrás de esta decisión:.

1. **Mejor Optimización en Hardware:**
   PyTorch optimiza de manera más eficiente el uso de GPU, permitiéndonos usar **CUDA** y **MPS**. Esto aceleraría el proceso de entrenamiento y permitiría manejar configuraciones más complejas sin aumentar el tiempo de procesamiento.
2. **Pipeline de Datos Más Robusto:**
   La integración con *torchvision* puede facilitar el manejo avanzado de datos, como la aplicación de transformaciones en tiempo real y la creación de datasets personalizados con mayor flexibilidad. Ayudando así a mejorar el desempeño en el entrenamiento del modelo.
3. **Experiencia del Equipo:**
   PyTorch es el framework con el que el equipo tiene mayor experiencia y dominio. Esto permitió un desarrollo más eficiente, optimizando el tiempo dedicado a la implementación