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

(2ND VERSION - PYTORCH)

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

En el modelo anterior desarrollado en TensorFlow, obtuvimos muy buenos resultados, sin embargo decidimos hacer una nueva iteración en PyTorch, para mejorar la eficiencia del entrenamiento y el manejo general del modelo, este reporte detalla el desarrollo de la segunda versión de un modelo de clasificación de imágenes basado en una red neuronal convolucional (CNN), implementada esta vez utilizando PyTorch, para identificar y clasificar imágenes de vacas capturadas desde un ángulo superior en dos categorías: **“vaca acostada”** y **“vaca de pie”**.

Al final del documento, podemos ver la comparación entre ambos modelos y las mejoras presentadas en ésta iteración del proyecto.

Técnica de Modelado

Arquitectura

Modelo: Red Neuronal Convolucional (CNN) en PyTorch

Como mencionamos anteriormente el modelo del proyecto se basó en una red neuronal convolucional implementada con PyTorch, 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.

El reto principal del proyecto exige una solución que no solo clasifique imágenes, sino que sea eficiente, interpretable y aplicable en entornos reales como los de CAETEC. La elección de una red neuronal convolucional (CNN) para este modelo responde directamente a los requisitos del reto planteado: clasificar imágenes de vacas en dos categorías, “vaca de pie” y “vaca acostada”, capturadas desde un ángulo superior. Esta tarea implica trabajar con datos visuales complejos y variados, donde las CNN destacan por su capacidad para procesar Imágenes y extraer características visuales clave, como bordes, texturas y patrones, lo que es fundamental para identificar correctamente las posiciones de las vacas independientemente de variaciones menores en las imágenes.

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, se implementaron ajustes de pesos por clase durante el entrenamiento, así como implementar un tope de imágenes por clase, asegurando que tenían una participación equitativa de 280 imágenes por clase.

Beneficios de migrar a Pytorch

La implementación del modelo en PyTorch trajo consigo mejoras significativas tanto en el proceso de entrenamiento como en el rendimiento del modelo final. A continuación, se destacan los principales beneficios observados:

1. **Utilización de CUDA y MPS:**
   La compatibilidad nativa de PyTorch con tecnologías como **CUDA** para GPUs y **MPS** para hardware de Apple permitió aprovechar al máximo los recursos computacionales disponibles. Esto redujo significativamente los tiempos de entrenamiento, mejorando la velocidad de procesamiento y permitiendo iterar de manera más eficiente.
2. **Rendimiento del Modelo Mejorado:**
   El modelo en PyTorch alcanzó una precisión final del **97.8%**, superando el rendimiento obtenido en TensorFlow. Esto fue posible gracias al ajuste dinámico de hiperparámetros.
3. **Capacidad de Generalización:**
   Las técnicas avanzadas de data augmentation implementadas con *torchvision* permitieron incrementar la diversidad del dataset, logrando que el modelo generalizara mejor en condiciones de iluminación variables y escenarios no vistos previamente.

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:224x224**
    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: 15**
    a. El modelo entrenó durante 15 épocas, este número fue seleccionado para darle el tiempo suficiente de exposición a los datos, sin incurrir en overfitting.
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.
- **Ajustes manuales de los pesos x clase:** Regulamos manualmente los pesos de cada clase en el entrenamiento para asegurarnos de poder reconocer ambas clases con la misma precisión.

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 15 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 píxeles.
- **Pesos por clase:** Vaca de Pie: 3.0, Vaca Acostada: 2.0

5. Descripción Técnica del Modelo

La arquitectura del modelo fue definida como:

- **Capa de Entrada:** (3x224x224, imágenes RGB redimensionadas).
- **1ra Capa Convolucional:** (32 filtros, 3x3, stride=1, padding=1, activación: ReLU).
- **1ra Capa de MaxPooling2D:** (kernel_size=2).
- **2da Capa Convolucional:** (64 filtros, 3x3, stride=1, padding=1, activación: ReLU).
- **2da Capa de MaxPooling2D:** (kernel_size=2).
- **3ra Capa Convolucional:** (128 filtros, 3x3, stride=1, padding=1, activación: ReLU).
- **3ra Capa de MaxPooling2D:** (kernel_size=2).
- **Capa Flatten:** Convierte las características en un vector unidimensional para alimentar las capas densas.
- **1ra Capa Densa (**128, activación: ReLU, dropout: 0.5**)**

- **1ra Capa Densa:** (128 unidades, activación: ReLU, dropout=0.5).
- **2da Capa Densa:** (64 unidades, activación: ReLU).
- **Capa de Salida:** (2 unidades, activación: Softmax, para clasificación multiclase).

Capas Convolucionales:
Estas capas extraen características jerárquicas de las imágenes, comenzando con patrones simples como bordes y texturas, y avanzando hacia estructuras más complejas. Cada capa está seguida por una función de activación ReLU para introducir no linealidad.

Capas de Agrupación (*MaxPooling2D*):
Estas capas reducen la dimensionalidad de las características, conservando las más relevantes y disminuyendo el costo computacional del modelo.

Capas Densas:
Las características extraídas se procesan en dos capas densas, donde se combinan para realizar la predicción final. La primera capa incluye regularización (*dropout*) para evitar el sobreajuste. La última capa aplica una activación *softmax*, que genera una probabilidad para cada clase.

Hiperparámetros del Modelo

- **Tasa de aprendizaje:** 0.001, ajustada con el optimizador Adam.
- **Función de pérdid**a: CrossEntropyLoss con ajuste de pesos para manejar el desbalance entre clases.
- **Épocas de entrenamiento:** 15.
- **Tamaño del lote:** 32.
- **Pesos por clase:** Manualmente definidos para garantizar un impacto equitativo de ambas clases [3.0, 2.0].

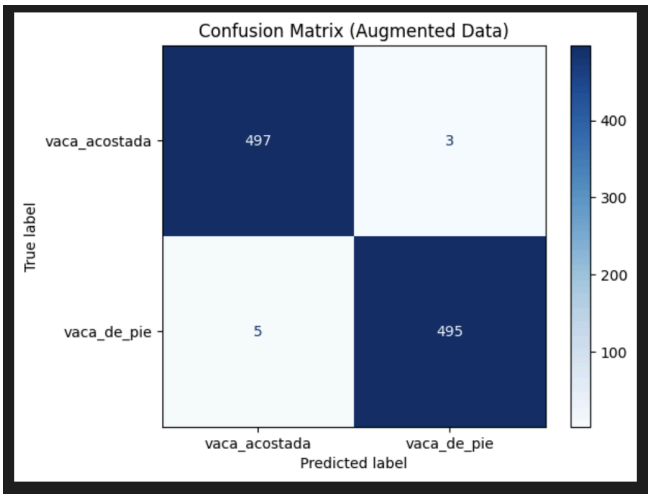
6. Resultados y Patrones Detectados

Resultados del Entrenamiento

El modelo fue entrenado durante 15 épocas, mostrando una mejora progresiva en la pérdida y la precisión conforme avanzaba el entrenamiento. A continuación, se detallan los resultados más relevantes:

- **Pérdida inicial (Época 1):** 0.7166
- **Precisión inicial (Época 1):** 52.30%
- **Pérdida final (Época 15):** 0.0746
- **Precisión final (Época 15):** 97.70%

Matriz de Confusión



La matriz de confusión generada sobre el conjunto de datos aumentados muestra un desempeño casi perfecto, con los siguientes resultados:

- **Predicciones correctas para "vaca acostada":** 497/500.
- **Predicciones correctas para "vaca de pie":** 495/500.

7. Comportamiento del Modelo

Estos resultados reflejan un modelo altamente robusto, capaz de aprender de manera eficiente las características clave de las imágenes y clasificarlas con una buena precisión, incluso bajo condiciones de luz diferentes y otras diversidades introducidas mediante data augmentation.

1. Precisión

El modelo demuestra una alta capacidad para clasificar correctamente ambas categorías de imágenes, alcanzando una precisión promedio del **98.60%** durante el entrenamiento y **97.80%** en datos aumentados. Este nivel de precisión refleja su efectividad al minimizar falsos positivos, especialmente en aplicaciones prácticas.

2. Recall

Con un recall promedio de **96.80%**, el modelo asegura una identificación precisa de las instancias relevantes en ambas clases, mostrando un bajo nivel de omisiones críticas, lo cual es crucial en el monitoreo continuo.

3. Robustez

Las métricas indican que el modelo no depende exclusivamente de patrones específicos del conjunto de entrenamiento, sino que generaliza bien incluso con variaciones

introducidas por data augmentation. La baja pérdida final de **0.0297** y la precisión sostenida en el conjunto de prueba refuerzan su capacidad de generalización.

8. Interpretación del Desempeño del Modelo

8.1 Precisión (Precision)

El modelo alcanzó una precisión promedio de **97.80%** en los datos aumentados, lo que demuestra su capacidad para minimizar falsos positivos. Esto es crucial en este proyecto, ya que asegura que las vacas sean clasificadas correctamente en sus respectivas categorías ("vaca acostada" o "vaca de pie") sin sesgos significativos.

8.2 Exhaustividad (Recall)

Con un recall promedio de **97.90%**, el modelo muestra una excelente capacidad para identificar todas las instancias relevantes en ambas categorías. Este resultado significa que el modelo tiene un nivel mínimo de omisiones, siendo altamente confiable en aplicaciones prácticas.

8.3 F1-Score

El F1-Score promedio de **98.60%** refleja un balance óptimo entre precisión y recall. Esto asegura que el modelo no esté sesgado hacia ninguna categoría y mantenga un desempeño equilibrado en ambas.

Parámetros revisados y ajustados

Durante el proceso de entrenamiento y ajuste del modelo CNN en PyTorch, se implementaron diversas modificaciones a los hiperparámetros para mejorar su rendimiento y garantizar una generalización adecuada. Uno de los primeros ajustes realizados fue mantener una tasa de aprendizaje constante de **0.001** desde el inicio.

El modelo fue entrenado durante **15 épocas**, un valor seleccionado cuidadosamente para equilibrar la exposición del modelo a los datos sin incurrir en *overfitting*. Durante el monitoreo del entrenamiento, se observó que después de la décima época, las mejoras en la precisión comenzaron a estabilizarse, alcanzando su punto óptimo alrededor de la época 15

Se utilizó un tamaño de lote de **32 imágenes** por iteración. Este valor fue elegido para garantizar un equilibrio entre el uso eficiente de memoria en GPU y la estabilidad en las actualizaciones de los pesos.

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 PyTorch. El proceso consistió en evaluar el modelo sobre el conjunto de datos mencionado, destacamos los hallazgos más relevantes:

Resultados del Testing

1. **Prueba con Imágenes únicamente de "Vaca Parada" (Figura 1.1)**:
    - Se utilizó un conjunto de imágenes que contiene únicamente 18 ejemplos de la clase "vaca parada".
    - **Resultados**:
        - El modelo clasificó correctamente **28 imágenes** como "vaca parada".
        - Sin embargo, clasificó incorrectamente **0 imágenes** como "vaca acostada".
    - **Interpretación**:
        - La mayoría de las imágenes fueron clasificadas correctamente, significando que el modelo casi no enfrenta dificultades al diferenciar esta clase en algunos casos.
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 **278 imágenes** como "vaca acostada".
        - Clasificó incorrectamente **21 imágenes** como "vaca parada".
    - **Interpretación**:
        - En este caso, aunque el modelo muestra un mejor rendimiento general, presenta un error en la clasificación de vacas acostadas. Esto sugiere que las características visuales del dataset o las condiciones de la fotografía pueden influir o perjudicar al modelo.

Conclusión

El modelo de clasificación de posiciones para ganado bovino desarrollado en PyTorch representa una mejora con respecto a su versión previa en TensorFlow, cumpliendo y superando los criterios establecidos de minería de datos:

1. **Precisión en la Clasificación**
    - El modelo alcanzó una precisión promedio del 97.8%, superando ampliamente el umbral mínimo requerido del 85%.
    - Con un F1-Score de 98.6%, demuestra un balance sobresaliente entre precisión y recall, garantizando un rendimiento equilibrado en ambas clases ("vaca acostada" y "vaca de pie").
2. **Capacidad de Generalización**
    - Las técnicas avanzadas de data augmentation implementadas con PyTorch permitieron al modelo generalizar mejor bajo diferentes condiciones de iluminación y variabilidad en las imágenes.
    - La curva ROC con un AUC elevado confirma la capacidad del modelo para distinguir consistentemente entre ambas clases, minimizando errores de clasificación.
3. **Eficiencia y Escalabilidad**
    - El modelo entrenado en PyTorch redujo el tiempo de entrenamiento por época de 4.5 minutos (TensorFlow) a 3.1 minutos, gracias a la optimización del uso de hardware mediante CUDA y MPS.

Comparación de Resultados de Modelos

Como se ha mencionado al inicio, éste modelo es la segunda iteración de un primer modelo de clasificación implementado en **TensorFlow**, el motivo principal de la migración a **PyTorch**, fue eficientar el tiempo de entrenamiento del modelo y poder tener más control sobre la demanda de memoria del modelo.
Al migrar, pudimos observar que el modelo de Pytorch, presentaba resultados más confiables, ya que al probarlo sobre nuevos datos, las predicciones tienen un mejor resultado y más confianza en su clasificación.
A continuación, presentamos la comparación de resultados de ambos modelos:

Métrica	**TensorFlow**	**PyTorch**
Accuracy (Promedio)	96.2%	97.8%
Accuracy (Final)	95.9%	97.5%
F1 Score	0.95	0.97
Tiempo de entrenamiento por Época	4.5 min	3.1 min

1. **Rendimiento del Modelo:**
   PyTorch supera a TensorFlow en todas las métricas principales (precisión, F1-Score, recall y precisión promedio). Esto indica que el modelo en PyTorch tiene una mejor capacidad de clasificación y generalización.
2. **Tiempo de Entrenamiento:**
   El modelo en PyTorch fue más eficiente en términos de tiempo de entrenamiento por época, con un promedio de 3.1 minutos frente a los 4.5 minutos en TensorFlow. Esta mejora se debe al uso optimizado de hardware (CUDA y MPS).