Tecnológico de Monterrey

Campus Querétaro

Modelado #1

Gamaliel Marines Olvera	A01708746
Uri Jared Gopar Morales	A01709413
José Antonio Miranda Baños	A01611795
María Fernanda Moreno Gómez	A01708653
Oskar Adolfo Villa López	A01275287
Luis Ángel Cruz García	A01736345

Inteligencia artificial avanzada para la ciencia de datos II
Grupo 501

Introducción

Este documento detalla el desarrollo de un modelo basado en Redes Neuronales Convolucionales (CNN, por sus siglas en inglés) diseñado específicamente para clasificar imágenes de camas de vacas en tres categorías: “cama vacía”, “vaca acostada” y “vaca de pie”. La clasificación precisa de estas imágenes tiene un propósito práctico en el monitoreo de ganado en instalaciones agrícolas, como lo es el CAETEC, facilitando la toma de decisiones en tiempo real y optimizando la gestión del espacio y el bienestar animal.

El proceso de desarrollo incluye el diseño de la arquitectura del modelo, la implementación de técnicas de aumento de datos para mejorar la generalización, y la evaluación de su rendimiento mediante métricas como el accuracy y la matriz de confusión. Cabe mencionar que para este modelo únicamente se utilizaron esas dos métricas, en la documentación de modelos futuros se pudieron haber implementado nuevas métricas de evaluación de acuerdo a los requerimientos del proyecto y de la misma precisión requerida, sin embargo, para este modelado solo se utilizaron las métricas anteriormente descritas. Además, se considera la infraestructura tecnológica requerida para implementar el modelo en un entorno real utilizando una Raspberry Pi, demostrando la viabilidad de la solución en un escenario práctico.

Selección de técnica de modelado

Existen múltiples aproximaciones posibles para la implementación del modelo de predicción. En el artículo *Model selection for 24/7 pig position and posture detection by 2D camera imaging and deep learning,* se utiliza un modelo FastCNN que arroja buenos resultados para detectar la posición y postura de cerdos en imágenes de día y noche. Por ello, se decidió utilizar un modelo de Red Neuronal Convolucional (CNN) simple como benchmark o modelo base. De acuerdo con los resultados de éste benchmark, se tomarán decisiones acerca de utilizar una arquitectura más compleja de CNN, o cambiar a modelos más complejos, como Transformers (Martin Riekert et al., 2021).

Técnica de modelado

Utilizamos una técnica de CNN debido a su eficacia para procesar datos en forma de cuadrícula, como las imágenes. La CNN está formada por varias capas: capas convolucionales, capas de agrupación y capas totalmente conectadas. Esta arquitectura se asemeja al procesamiento visual del cerebro humano y es adecuada para capturar patrones jerárquicos y dependencias espaciales dentro de imágenes.

Las capas que se utilizaron para esta arquitectura fueron las siguientes:

- **Capas convolucionales:** Aplican operaciones convolucionales a las imágenes. Sirven para detectar bordes, texturas y patrones complejos.

- **Capas de agrupación**: Reducen el tamaño de las dimensiones espaciales de la entrada con el fin de reducir la complejidad computacional.
- **Funciones de activación:** Utilizamos Rectified Linear Unit (ReLU) para introducir una propiedad no lineal al modelo con el fin de aprender patrones más complejos, evita el vanishing gradient[1] al no saturarse de entradas positivas, mitigando este problema y acelerando la convergencia durante el entrenamiento, además que es computacionalmente más eficiente que otras funciones al tener una forma matemática directa.
- **Capas completamente conectadas:** Estas son las responsables de hacer las predicciones basadas en los aprendizaes de las capas anteriores.

Las capas son entrenadas por medio de carpetas con imágenes clasificadas de acuerdo a las etiquetas de “cama vacía”, “vaca acostada” y “vaca de pie”, por lo que la red aprende a reconocer patrones y características asociadas a cada clase.

Para mejorar la generalización del modelo, se implementó la técnica de aumento de datos (data augmentation), que es el proceso de generar artificialmente nuevos datos a partir de datos existentes para entrenar un modelo, ya que los modelos de Machine Learning requieren de conjuntos de datos reales diversos y bastos para la generalización del modelo. En nuestro modelo, se realizaron las siguientes aumentaciones: redimensionamiento, giro horizontal y vertical de la imagen y rotación, esto con el fin de aumentar artificialmente los datos y tener un conjunto de datos de entrenamiento más robusto para que pueda realizar mejores predicciones el modelo.

Suposiciones

Las suposiciones bajo las cuales funciona el modelo son las siguientes:

- Las imágenes de las camas tienen el mismo tamaño, 950 x 450 px.
- Las imágenes muestran una sola cama.
- Las imágenes no muestran personas u objetos diferentes.
- Las imágenes de las camas se toman desde arriba.
- Las imágenes fueron tomadas por un modelo específico de cámara y fueron pasadas por un proceso de aplanado, que coincide con el ojo de pescado[2] generado por dicho tipo de cámara en específico.

[1] **Vanising gradient:** Es un problema que ocurre en los entrenamientos de las CNN cuando los gradientes de la función de pérdida con respecto a los pesos de las primeras capas se vuelven extremadamente pequeños, resultando que las capas reciban poca o nula información de los pesos en la retropropagación (backpropagation).
[2] **Ojo de pescado:** Se refiere a las distancias focales que tienen ángulos muy anchos y que consiguen producir una distorsión visual de grandes capacidades.

Diseño de pruebas

Para las pruebas se utilizará el conjunto de datos que fue seleccionado para dicho propósito desde un inicio, lo que brinda la posibilidad de ver el rendimiento del modelo en imágenes que no fueron utilizadas para su entrenamiento, logrando de esta forma identificar si existe un sobreajuste[3].

Específicamente, se cuenta con la siguiente cantidad de imágenes por cada clase para el conjunto de prueba:

- Cama vacía: 1323
- Vaca acostada: 394
- Vaca de pie: 60

Una vez utilizado el modelo para evaluar los nuevos datos, se obtendrá como métrica principal el accuracy en forma de porcentaje, el cual se obtiene implementando la siguiente fórmula:

$$Accuracy = \frac{Correct\ predictions}{Total\ predictions} * 100$$

Buscando obtener un valor que no solo muestre que el ajuste del modelo es correcto, sino también que no existe una diferencia significativa entre dicho accuracy y el obtenido durante el entrenamiento.

Por último, se utilizará una matriz de confusión para poder identificar si existe tendencia a predecir una clase en específico, el cual es otro problema que se busca evitar, ya que al contar con pocas imágenes de “Vaca de pie”, podría no estar detectando correctamente dicha clase y seguir mostrando un accuracy alto.

Los valores que se muestran en la matriz de confusión son acorde a las clases, en donde se comparará la cantidad de valores predecidos correctamente para cada clase. A continuación se muestra un diagrama que ejemplifica los valores obtenidos en una matriz de confusión, en dicho diagrama, los valores que se deberían mostrar más altos son los verdaderos positivos (TP) y los verdaderos negativos (TN), ya que significa que el número de predicciones por cada clase fueron correctas.

[3] **Sobreajuste (overfitting):** Es un comportamiento en Machine Learning (ML) que se produce cuando el modelo da predicciones precisas para los datos de entrenamiento, pero no para nuevos datos, es decir, aprende de memoria los resultados, sin embargo, no aprende los patrones en realidad para datos no vistos.

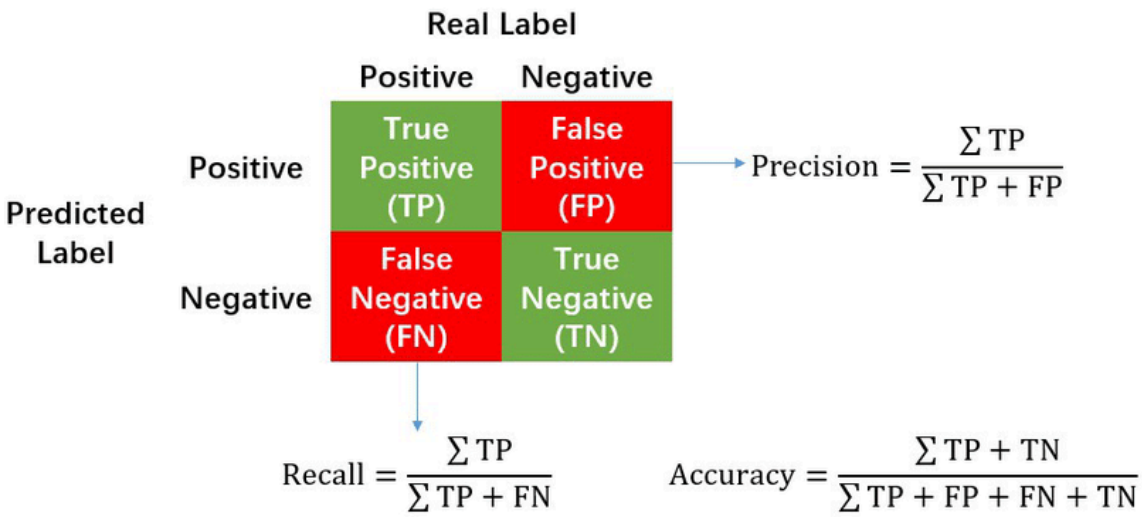


Imagen 1: Diagrama de construcción de una matriz de confusión.

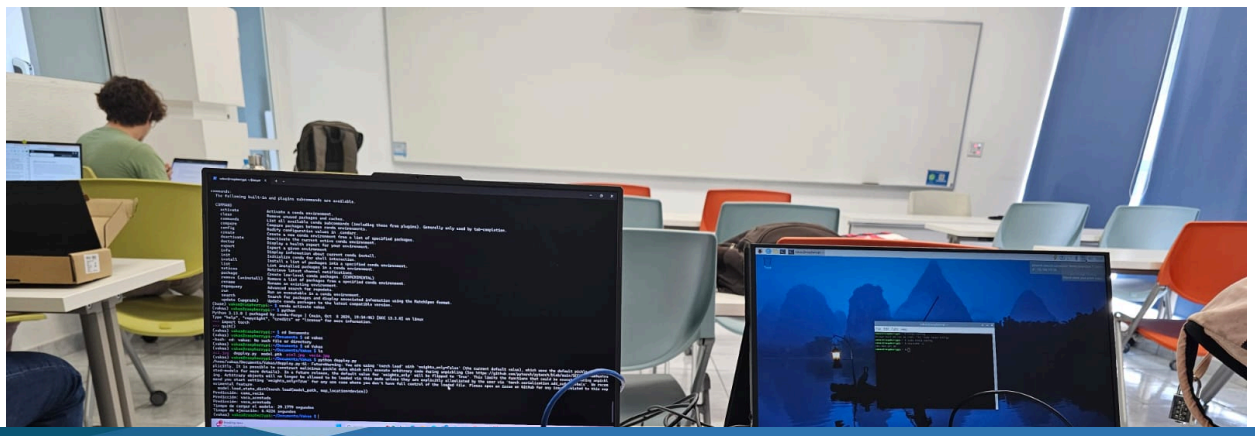
Prueba de arquitectura

Para garantizar que nuestra solución propuesta funcione correctamente en el entorno de producción y cumpla con los requisitos de despliegue establecidos por el CAETEC, se llevó a cabo una prueba de arquitectura. En esta prueba, se modeló la infraestructura de nuestro socio formador (el CAETEC) y se ejecutó nuestro modelo en dicha arquitectura.

Hardware requerido:

- Raspberry Pi Model 3 2015 64 bits.
- Una memoria SD de 64 GB
- Periféricos (Monitor, Teclado, mouse).
- Cable ethernet
- Laptop
- Cable HDMI
- Cargador tipo microSD.

Implementación:



Para lograr concluir con éxito esta prueba se realizaron una serie de pasos:

En la microSD, se cargó el sistema operativo Raspberry Pi OS de 64 bits, el cual nos permitirá aprovechar al máximo el hardware de la Raspberry Pi. Es necesario contar con un sistema de 64 bits para ejecutar nuestro modelo, ya que PyTorch solo es compatible con esta arquitectura.

Conectamos nuestra Raspberry Pi a los periféricos para simular una computadora y esperamos a que el sistema operativo se inicialice. Esto nos permitirá descargar lo necesario para ejecutar nuestro modelo de predicción.

Imagen 2: Fotografía de la implementación de la prueba de arquitectura.

La guia de ejecución para realizar la prueba de arquitectura se encuentra en el siguiente enlace, en este documento se presentará un resumen de lo realizado:

- GUIA DE ARQUITECTURA TC.

Activamos SSH con la herramienta FileZilla de para transferir los archivos necesarios desde nuestra laptop a la Raspberry Pi: el modelo entrenado, las imágenes a predecir y un script para inicializar el modelo.

Con estos archivos en Raspberry, instalamos PyTorch. Para ello, utilizamos Miniconda para crear un ambiente compatible en el que PyTorch pueda instalarse. Finalmente, ejecutamos el script en Raspberry Pi para que realice las predicciones.

Los resultados obtenidos fueron los siguientes:

Imagen 3: Resultados obtenidos de la clasificación de las imágenes del modelo hospedado en la Raspberry. Se ejecutó a través del protocolo ssh.

Donde podemos observar que el modelo para ejecutarse la primera vez cuenta con un tiempo de 30 segundos, cabe destacar que esto únicamente sucede una vez al cargar el archivo

El tiempo de ejecución del cargado del modelo es de alrededor de 30 segundos. Este paso se necesita realizar una sola vez al iniciar el script.

Para realizar las predicciones, el modelo tarda 7 segundos, lo cual fue aprobado por el socio formador, ya que es un tiempo adecuado y entra dentro de la ventana de 5 minutos que tiene el modelo para hacer la predicción.

Modelo

Introducción

En este proyecto, el primer modelo implementado es una CNN diseñada para clasificar imágenes en tres categorías: *cama vacía*, *vaca acostada* y *vaca de pie*. Esta arquitectura está orientada a capturar características visuales esenciales, como bordes y patrones, permitiendo al modelo distinguir entre las diferentes clases de imágenes mediante un enfoque basado en capas convolucionales y totalmente conectadas.

Para garantizar un rendimiento óptimo, el modelo se entrena y evalúa utilizando un conjunto de datos etiquetados, dividido en tres subconjuntos: **entrenamiento (70%)**, **validación (15%)** y **prueba (15%)**. La diferenciación entre los subconjuntos de validación y prueba es esencial para un correcto desarrollo y evaluación del modelo:

- **Conjunto de Validación:** Se utiliza durante el entrenamiento del modelo para monitorear su rendimiento y ajustar hiperparámetros como es la tasa de aprendizaje con datos con los que nunca ha visto.
- **Conjunto de Prueba:** Es un subconjunto completamente separado y nunca expuesto al modelo durante el entrenamiento. Sirve como una métrica definitiva para evaluar el rendimiento del modelo en datos nuevos y no vistos, proporcionando una medida imparcial de su capacidad para generalizar.
- **Conjunto de Entrenamiento**: Ayuda para ajustar sus pesos y parámetros, minimizando la pérdida por época

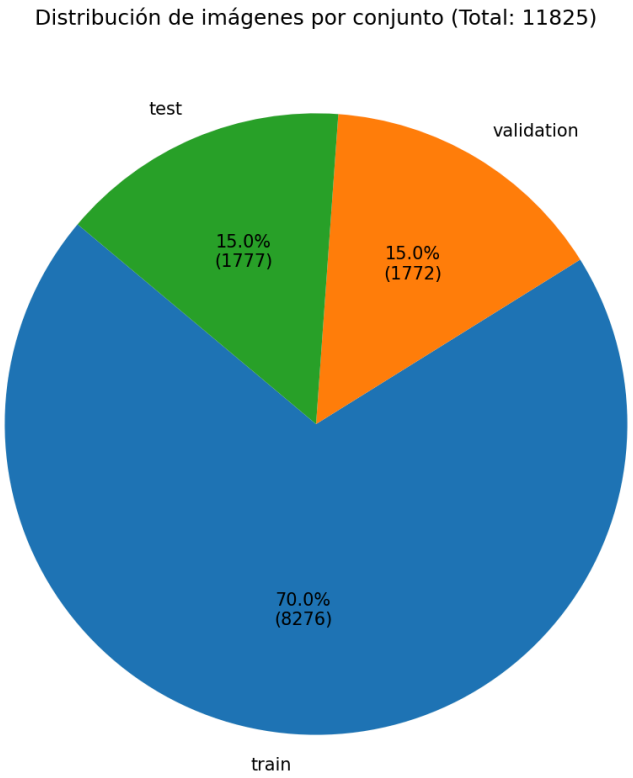


Imagen 4: Número de imágenes por sección de entrenamiento,prueba y validación.

Escogimos esta distribución de datos debido a que al tener un dataset grande, en el apartado de train de 8,276 imágenes se podrán apreciar los diferentes escenarios posibles, como son:

- Vaca acostada, en día y noche.
- Vaca parada, en día o noche.
- Cama vacía en día o noche.

Y como para nuestro apartado de prueba y validación aún cada uno cuenta con 1700 imágenes, lo adecuado para probar el rendimiento en un escenario real.

Diseño y configuración del primer modelo

Este modelo se compone de una capa convolucional de 16 filtros de 3 canales con un kernel size de 3 y un padding de 1 para la extracción inicial de características,

seguidas de capas completamente conectadas de 128 neuronas a 3 para obtener un output de nuestra clase. La arquitectura es la siguiente:

- **Capa convolucional:** El modelo inicia con una capa convolucional que aplica 16 filtros de 3 canales, cada uno de tamaño de kernel de 3x3 y un padding de 1. Esta configuración permite que el modelo capture detalles en las imágenes como bordes y patrones esenciales para distinguir entre las diferentes clases.
- **Función de activación ReLU:** Después de la convolución, se utiliza la función de activación ReLU para introducir no linealidad[4].
- **Capas de agrupación (Max-Pooling):** Para reducir la dimensionalidad, y con ello, la carga computacional, se utiliza una capa de max-pooling con una ventana de 2x2, que selecciona los valores máximos en las regiones pequeñas de la imagen convolucionada. Esto permite la modelo conservar las características más relevantes mientras reduce el tamaño de la imagen.
- **Capas completamente conectadas (Fully Connected):** Después de aplanar las características extraídas, el modelo emplea una capa totalmente conectada de 128 neuronas, seguida de una capa de salida con 3 neuronas, correspondientes a cada clase de la imagen. La capa final utiliza una función de activación softmax[5] para generar las probabilidades de cada clase.

Transformaciones de imágenes

Para mejorar la capacidad de generalización del modelo y enfrentar la variabilidad de las imágenes, se aplicaron transformaciones en los datos (data augmentation). Las transformaciones incluyen:

- **Redimensionamiento:** Las imágenes se redimensionan a 450x950 píxeles para asegurar uniformidad.
- **Aumento de datos:** Se aplica un giro horizontal y vertical aleatorio con un 50% de probabilidad, y una rotación aleatoria de hasta 30°. Estas transformaciones ayudan a que el modelo aprenda variaciones en las posiciones de las vacas y las camas, aumentando así la robustez del modelo ante diferentes ángulos y posiciones de las imágenes.

Entrenamiento

Para entrenar el modelo, utilizamos un conjunto de datos etiquetado y dividido en tres subconjuntos: entrenamiento, validación y prueba, con una proporción de

[4] **No linealidad:** Se refiere a una situación en la que no existe una relación directa o lineal entre una variable independiente y una variable dependiente.
[5] **Softmax:** Es una función usada en la última capa de una CNN para tareas de clasificación que convierte los valores de entrada en un rango de 0 a 1 para que la suma de 1, de modo que se le dé un valor a cada una de las posibles respuestas.

70-15-15. El entrenamiento se lleva a cabo a lo largo de 5 épocas, utilizando el algoritmo de optimización Adam[6], con una tasa de aprendizaje de 0.001 y la función de pérdida de entropía cruzada, adecuada para problemas de clasificación multiclase.

Cada época de entrenamiento genera una métrica de pérdida acumulada, lo que permite evaluar la mejora del modelo a lo largo del tiempo. Posteriormente, evaluamos el rendimiento en el conjunto de validación para monitorear el accuracy y evitar el sobreajuste. Al final de cada época, se calcula el accuracy en el conjunto de validación y, en caso de ser superior al de épocas anteriores, se guarda el modelo como el mejor.

Métricas y pruebas en el conjunto de Test (Prueba)

En el primer modelo, las métricas utilizadas para evaluar el rendimiento en el conjunto de prueba fueron las siguientes, para dichas pruebas fue basado en el documento titulado "*[04]-General_Introduction_For_Modeling_TC*":

1. **Accuracy:** Se utilizó como métrica principal para medir el rendimiento global del modelo, mostrando el porcentaje de predicciones correctas sobre el total de predicciones realizadas.
2. **Curva ROC (Receiver Operating Characteristic):** Se implementó para evaluar el desempeño del modelo en términos de la relación entre falsos positivos y verdaderos positivos, proporcionando el área bajo la curva (AUC) como un indicador del desempeño general para cada clase.
3. **Gráficos de pérdida (loss) y precisión (accuracy) por época:** Se incluyeron para monitorear el comportamiento del modelo durante el entrenamiento y validar que la optimización y la generalización del modelo fueran adecuadas.
4. **Matriz de confusión:** Esta métrica permitió analizar de forma más detallada cómo el modelo clasifica las diferentes clases, identificando errores específicos, como confusiones frecuentes entre clases minoritarias.

Estas métricas y visualizaciones permitieron obtener una visión general del desempeño del modelo, aunque evidenciaron limitaciones en la clasificación de clases minoritarias, particularmente la clase "vaca de pie."

[6] **Optimización Adam:** Es un algoritmo esencial para el aprendizaje eficiente y la convergencia en redes neuronales. Se usa para minimizar la pérdida durante el entrenamiento en CNN al utilizar los gradientes al cuadrado para escalar la tasa de aprendizaje.

Evaluación del modelo

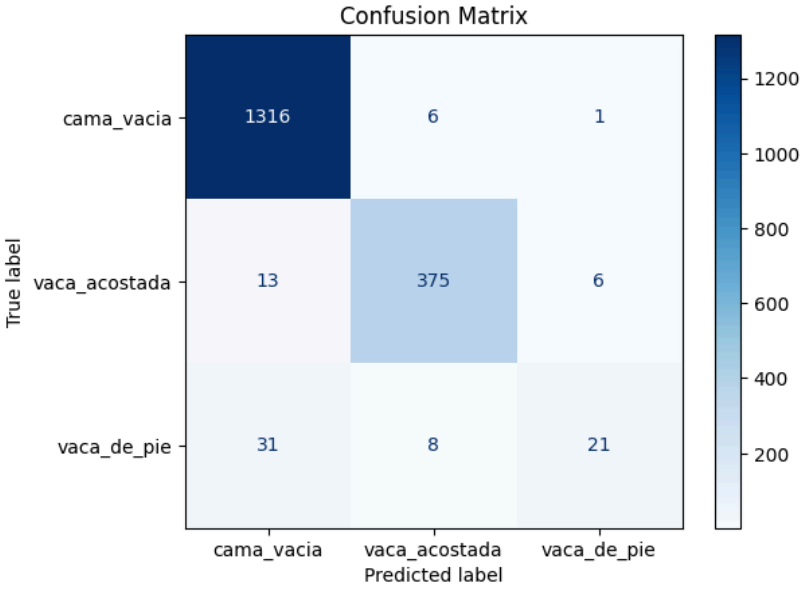


Imagen 5: Matriz de confusión del primer modelo.

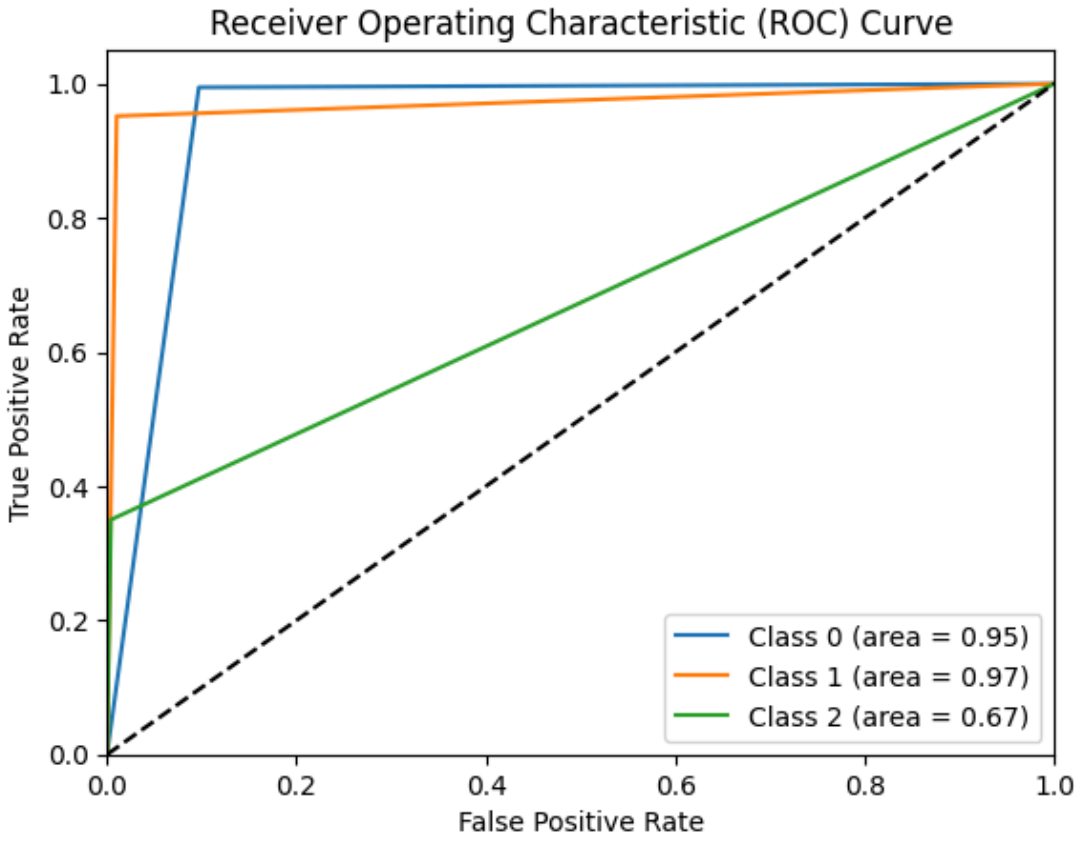


Imagen 6: Gráfico ROC, que representa la proporción de falsos positivos y falsos negativos entre las 3 clases: Clase 0 “Cama vacía”, Clase 1 “Vaca acostada” y Clase 2 “Vaca de pie”.

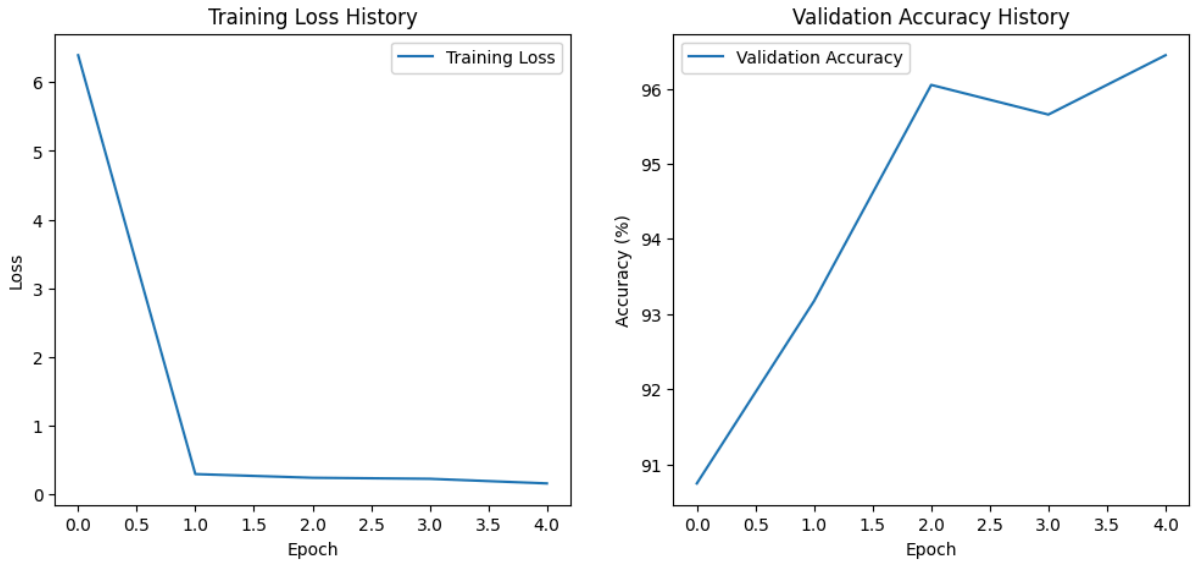


Imagen 7: Gráficos de loss (pérdida) y accuracy (precisión) del primer modelo por época.

```
Epoch [1/5], Loss: 6.394981656482436
Validation Accuracy: 90.74492099322799%

Epoch [2/5], Loss: 0.292740216151483 93
Validation Accuracy: 93.17155756207674%

Epoch [3/5], Loss: 0.23919793592791097
Validation Accuracy: 96.04966139954853%

Epoch [4/5], Loss: 0.224110534829968 54
Validation Accuracy: 95.65462753950338%

Epoch [5/5], Loss: 0.156940960198547 34
Validation Accuracy: 96.44469525959369%
```

Imagen 8: Gráficos en la consola de los resultados de las primeras 5 épocas del primer modelo.

Como podemos observar, en el primer modelo CNN diseñado para la clasificación de imágenes de tres clases (“cama vacía”, “vaca acostada” y “vaca de pie”) mostró un rendimiento aceptable, con ciertas áreas de mejora identificadas a partir de los resultados de las métricas de evaluación.

La **imagen 5** muestra la matriz de confusión que revela que el modelo tiene un alto desempeño en la clasificación de las clases de “cama vacía”, con 1316 predicciones correctas de un total de 1323. Sin embargo, las clases minoritarias, en particular “vaca de pie”, presentan un desempeño inferior. La clase “vaca de pie” fue confundida con frecuencia con las otras clases, lo que **sugiere una necesidad de ajustar el modelo para mejorar su precisión en la detección de clases menos representadas**.

La **imagen 6** muestra la curva ROC, que enseña el buen desempeño para las clases “cama vacía” y “vaca acostada”, con áreas bajo la curva (AUC) de 0.95 y 0.97 respectivamente. La clase “vaca de pie” tuvo un AUC de 0.67, lo que indica una menor capacidad de discriminación del modelo para esta clase. Esta diferencia en el AUC destaca la **necesidad de optimizar el modelo para mejorar la sensibilidad hacia esta clase en particular**.

En la **imagen 7** se muestran los gráficos de pérdida y precisión, durante el entrenamiento muestra una disminución constante en la pérdida y un aumento en el accuracy en el conjunto de validación, alcanzando una precisión de validación de 96.44% al final de la quinta época. **Esto indica que el modelo fue capaz de aprender los patrones presentes en el conjunto de entrenamiento sin evidenciar un sobreajuste significativo.**

El log de entrenamiento de la **imagen 8** detalla la disminución progresiva de la pérdida y el aumento en la precisión de validación con cada época. Desde una pérdida inicial de 6.39 y una precisión de validación de 90.74% en la primera época, el modelo logró reducir la pérdida a 0.15 y aumentar la precisión a 96.44% en la última época, **demostrando una mejora sostenida en su rendimiento**.

En conclusión, el primer modelo de CNN alcanzó buenos resultados en términos de precisión y aprendizaje general. Sin embargo, las métricas de clasificación y el análisis de la matriz de confusión sugieren que el modelo tiene dificultades para reconocer correctamente la clase “vaca de pie”, probablemente debido a la menor cantidad de ejemplos en esta clase.

Se sugiere que para el siguiente modelo, se ajusten los pesos en la función de pérdida, aumentando el peso de las clases minoritarias para reducir el sesgo a

las clases más representadas. Además, se pueden aplicar más transformaciones para incrementar la variabilidad en el conjunto de datos de entrenamiento y también poner más épocas para asegurar que el modelo capture los patrones.

Referencias

Amazon Web Services (AWS). (n.d.). What is data augmentation? Recuperado de https://aws.amazon.com/what-is/data-augmentation/#:~=Data%20augmentation%20is%20the%20process,machine%20learning%20(ML)%20models

Amazon Web Services (AWS). (n.d.). What Is Overfitting? Recuperado de https://aws.amazon.com/es/what-is/overfitting/

Amanatulla, M. (2023). Vanishing Gradient Problem in Deep Learning: Understanding, Intuition, and Solutions. Medium. Recuperado de https://medium.com/@amanatulla1606/vanishing-gradient-problem-in-deep-learning-understanding-intuition-and-solutions-da90ef4ecb54

Bargainfotos. (n.d.). Todo sobre los objetivos de ojo de pez. Recuperado de https://bargainfotos.com/blog/todo-sobre-los-objetivos-de-ojo-de-pez/

GeeksforGeeks. (n.d.). Convolutional neural network (CNN) in machine learning. Recuperado de https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/

Investopedia. (n.d.). What Is Nonlinearity? Recuperado de https://www.investopedia.com/terms/n/nonlinearity.asp

Martin Riekert, Svenja Opderbeck, Andrea Wild, Eva Gallmann. (2021). Model selection for 24/7 pig position and posture detection by 2D camera imaging and deep learning. Computers and Electronics in Agriculture, 187, 106213. Recuperado de https://doi.org/10.1016/j.compag.2021.106213

Srivastava, S. (2023). Understanding the difference between ReLU and Sigmoid activation functions in deep learning. Medium. Recuperado de https://medium.com/@srivastavashivansh8922/understanding-the-difference-between-relu-and-sigmoid-activation-functions-in-deep-learning-33b280fc2071

Sue, N. (2023). What Is the Softmax Function Used in Deep Learning? Illustrated in an Easy-to-Understand Way. Medium. Recuperado de https://medium.com/@sue_nlp/what-is-the-softmax-function-used-in-deep-learning-illustrated-in-an-easy-to-understand-way-8b937fe13d49