DETECCION DE VACAS POR INTELIGENCIA ARTIFICIAL

Uri Jared Gopar Morales, ITC A01709413, Tecnologico Campus Querétaro.

***Abstract*- En este documento se describen las técnicas utilizadas para la creación y entrenamiento de una red neuronal convolucional (CNN), diseñada para la detección de vacas y el análisis de su comportamiento a lo largo del día en dos de sus camas. El modelo alcanzó una precisión del 95% y opera de forma continua las 24 horas del día. Gracias a esta herramienta, fue posible identificar cuál de las camas es menos utilizada y proponer acciones que permitan un mejor aprovechamiento de los recursos disponibles.**

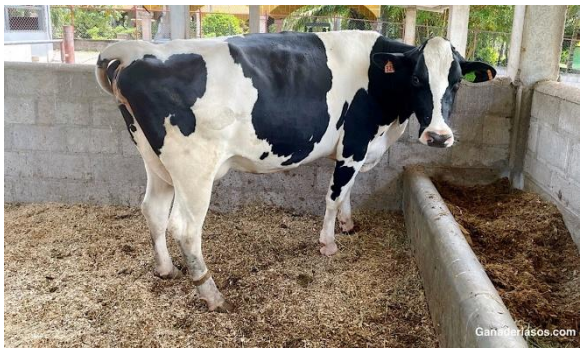
I. INTRODUCCION

En la industria lechera, optimizar la producción de leche es fundamental debido a la alta demanda de este producto. Para garantizar una leche de alta calidad, es necesario prestar atención a diversos factores. Las principales características de una buena leche incluyen:

- Alto contenido de grasa y proteína.
- Baja carga bacteriana.
- Ausencia de antibióticos u otros contaminantes.
- Buen sabor, color blanco uniforme y consistencia adecuada.

Para lograr estas características, es esencial que las vacas se encuentren en condiciones óptimas de bienestar, lo cual implica proporcionarles una alimentación balanceada, un ambiente limpio y cómodo, así como un estilo de vida saludable.

Una vaca saludable y bien cuidada puede producir entre 30 y 40 litros de leche al día, y en algunos casos incluso más. Considerando que una vaca lechera de raza Holstein tiene un valor aproximado de $60,000 MXN, sin incluir los costos de alimentación, cuidado veterinario y mantenimiento, es evidente que mantener una alta productividad es vital para asegurar la rentabilidad del negocio.



Uno de los factores clave para incrementar la producción de leche es que la vaca pase el mayor tiempo posible acostada, ya que esto favorece su descanso, rumiación y circulación sanguínea en la ubre. Por ello, en muchas granjas se utilizan camas de arena, diseñadas para ofrecer comodidad y minimizar el riesgo de lesiones.

En este contexto, nuestro modelo de inteligencia artificial basado en redes neuronales convolucionales (CNN) permite monitorear a las vacas durante las 24 horas del día, detectando su presencia en las camas y clasificando su comportamiento en tres categorías: **vaca parada**, **vaca acostada** o **cama vacía.**

El objetivo principal de este sistema es generar información útil sobre:

- Cuáles camas son más utilizadas.
- En qué periodos del día se usan.
- Qué posición adoptan las vacas en cada momento.

Con este análisis, es posible proponer estrategias para mejorar la distribución y aprovechamiento de las camas, lo que contribuye directamente a una mayor eficiencia y confiabilidad en la producción lechera.

II. MANEJO DE DATOS

Nuestro socio formador nos proporcionó inicialmente un dataset compuesto por 9,634 imágenes, capturadas a lo largo de varios meses. En cada imagen se observaban tres camas de arena utilizadas por las vacas durante las 24 horas del día. Para facilitar el análisis, se desarrolló un script en Python que dividía cada imagen en tres partes, correspondiendo cada una a una cama diferente. Después de eliminar las imágenes dañadas o corruptas, obtuvimos un total de 11,991 imágenes válidas, cada una mostrando una sola cama.

Dando una distribución de:

- 8,803 imágenes de camas vacías
- 2,625 imágenes de vaca acostada
- 397 vaca parada

Con este dataset se logró entrenar nuestra CNN.

Este conjunto de datos se utilizó para entrenar nuestra red neuronal convolucional (CNN).

Posteriormente, el socio formador nos proporcionó un segundo dataset, destinado exclusivamente para evaluar el desempeño del modelo en un contexto real. Este nuevo conjunto contenía 15,140 imágenes, cada una mostrando dos camas, lo que resulta en un total de 30,280 camas observadas. Las imágenes tienen una resolución de 1920x1080 píxeles y fueron destinadas únicamente para el proceso de predicción.

Dado que el objetivo era realizar una estimación precisa sin necesidad de analizar la totalidad del conjunto, se optó por seleccionar una muestra representativa. Para ello, se utilizó la fórmula para el tamaño de muestra en poblaciones finitas:

$$n = \frac{N \cdot Z^2 \cdot p \cdot (1-p)}{(E^2 \cdot (N-1)) + (Z^2 \cdot p \cdot (1-p))}$$

Donde:

- n= tamaño de muestra requerido
- N= tamaño total de la población (30,280 camas)
- Z= valor para el nivel de confianza
- p= proporción esperada de clasificación correcta
- E= margen de error tolerable.

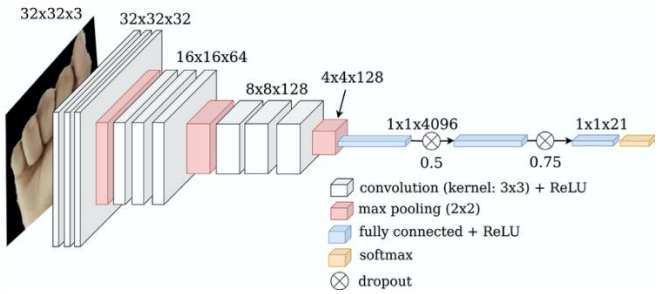
Aplicando esta fórmula, se determinó un tamaño de muestra mínimo de 379 camas clasificadas, lo que equivale aproximadamente a 190 imágenes. No obstante, debido a la variabilidad en el comportamiento de las vacas —un factor que no se puede controlar— y las posibles variaciones en las condiciones de iluminación de la cámara, se decidió duplicar el tamaño de la muestra para obtener un panorama más amplio y representativo de la situación real. De esta manera, se trabajó con una muestra final de 1,152 camas o lo que es equivalente a 576 imágenes, distribuidas de la siguiente forma:

- 48 vacas parada
- 283 vaca acostada
- 821 cama vacía.

Este nuevo conjunto permitió evaluar el modelo en condiciones reales, considerando distintos momentos del día y diferentes situaciones de iluminación.

III. MODELO

El modelo utilizado es un modelo de redes neuronales convolucionales (CNN), esto se debe a la arquitectura de este modelo es específicamente para procesar datos como imágenes. Este modelo tiene la capacidad para poder captar patrones espaciales y jerárquicos de las imágenes (bordes, texturas y diferentes formas) a través de sus capas convolucionales. Las cuales la detección de estos será de gran ayuda para el correcto funcionamiento de nuestro principal objetivo la detección de vacas y su posición en las camas a lo largo del día.



Capas Convolucionales: Detectan características locales de la imagen, como son los bordes, texturas o colores, estas aplican los filtros Kernels, los cuales son matrices pequeñas que realizan una operación de convolución.

Función de Activación: Introduce no linealidades, permitiendo que la red aprenda relaciones complejas.

Capa Pooling: Reduce las dimensiones espaciales alto y ancho esto para quedarse con las características más importantes.

Capa de Aplanamiento (Flattening): Convierte los mapas de características bidimensionales en un vector unidimensional para conectarlo con capas densas.

Capas Densas (Capas Conectadas): Realizan la clasificación usando las características aprendidas, lo que realiza es una conexión entre todas las neuronas de la capa anterior con la capa actual. Conectan todo lo aprendido anteriormente.

Softmax (Activación): Convierte las salidas en probabilidades para cada clase.

IV. IMPLEMENTACION

Se implementó una red neuronal convolucional (CNN) simple utilizando la biblioteca PyTorch, diseñada específicamente para la tarea de clasificación de imágenes en tres categorías: *vaca de pie*, *vaca acostada* y *cama vacía*.

La arquitectura del modelo comienza con una única capa convolucional compuesta por 16 filtros de tamaño 3×3 y un padding de 1, lo que preserva las dimensiones espaciales de la imagen de entrada. Esta capa convolucional toma como entrada imágenes RGB (3 canales) y aplica la función de activación ReLU para introducir no linealidad.

Posteriormente, se emplea una operación de *max pooling* con una ventana de 2×2 y un *stride* de 2, lo cual reduce a la mitad las dimensiones espaciales de la imagen procesada. A continuación, las salidas de la etapa convolucional se aplanan para formar un vector unidimensional que se conecta con una red totalmente conectada.

Esta red densa está conformada por cuatro capas secuenciales:

La primera capa lineal que reduce la dimensión de entrada ($16 \times 225 \times 475$) a 32 neuronas, seguida de una capa de 64 neuronas, luego una de 128, y finalmente una capa de salida con 3 neuronas, cada una correspondiente a una de las clases objetivo.

Cada capa intermedia utiliza la función de activación ReLU para mantener la capacidad del modelo de aprender representaciones no lineales. La salida final del modelo consiste en tres valores sin aplicar la función Softmax, ya que esta suele incluirse junto con a nuestra función de Loss que la veremos en el apartado de entrenamiento.

Esta arquitectura fue diseñada para mantener la simplicidad estructural sin sacrificar la capacidad de aprendizaje, enfocándose en tareas de visión por computadora relacionadas con el monitoreo de comportamiento animal en entornos controlados.

V. ENTRENAMIENTO

El proceso de entrenamiento del modelo se llevó a cabo utilizando la biblioteca PyTorch, aprovechando la capacidad de cómputo de una GPU (4080 RTX), para acelerar la convergencia.

Para mejorar la generalización del modelo y simular variaciones del entorno real, se aplicaron varias transformaciones de aumento de datos durante la fase de preprocesamiento. Estas incluyeron redimensionamiento de las imágenes a 950×450 píxeles, volteos horizontales y verticales aleatorios, rotaciones de hasta 30 grados, transformaciones afines con escalada variable (entre 50% y 100%), y ajustes de saturación de color.

Todas las imágenes se convirtieron a tensores y se normalizaron con una media y desviación estándar de 0.5 por canal RGB. El conjunto de datos se dividió en tres subconjuntos: entrenamiento, validación y prueba, y se utilizaron DataLoaders para cargarlos en lotes de tamaño 32, con

aleatorización habilitada para los conjuntos de entrenamiento y validación.

El entrenamiento del modelo se realizó durante 50 épocas, utilizando la función de pérdida CrossEntropyLoss ponderada para corregir el desbalance de clases.

Se asignaron pesos específicos a cada clase (5.0, 1.0 y 9.0), priorizando la correcta clasificación de vacas acostadas y camas vacías, cuya detección es crítica para el análisis conductual.

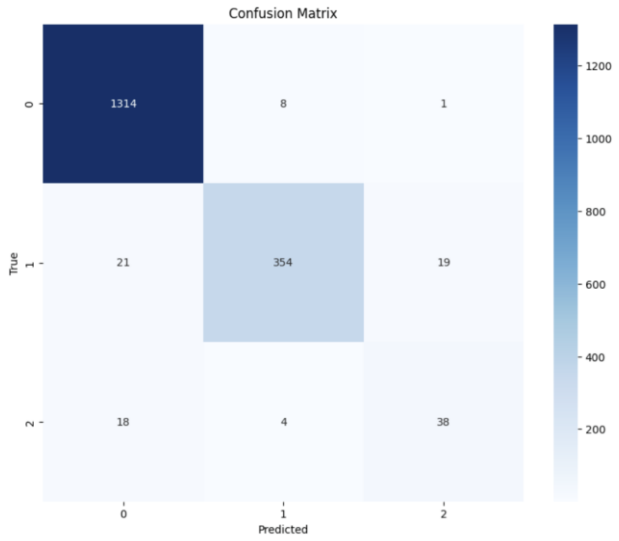
El optimizador empleado fue Adam, con una tasa de aprendizaje de 0.0001. Durante cada época, el modelo alternó entre los modos de entrenamiento y evaluación. En la fase de entrenamiento, se ejecutaron los pasos estándar de propagación hacia adelante (forward pass), retropropagación del error (backward pass) y actualización de pesos.

VI. RESULTADOS

Se evaluó el desempeño en el conjunto de validación, calculando la precisión como métrica principal. Se implementó un mecanismo de guardado automático del modelo cuando se superaba el umbral de precisión previamente registrado. Al finalizar el entrenamiento, se generaron gráficas que documentan la evolución de la pérdida en el conjunto de entrenamiento y la precisión en el conjunto de validación. Además, se construyó una matriz de confusión a partir del conjunto de prueba para analizar el rendimiento del modelo por clase. Para mejorar la experiencia del usuario, se incorporó una notificación sonora que indica la finalización del entrenamiento.

Esta metodología permitió evaluar su comportamiento de manera visual y cuantitativa a través de métricas relevantes y trazabilidad gráfica.

Al imprimir la matriz de confusión de mostraba los siguientes resultados:



0: Cama vacía, 1: Vaca acostada y 2: Vaca parada.

Lo que nos dice que nuestro modelo se confunde mas entre vaca parada y vaca acostada, pero al terminar todo el entrenamiento nos da un accuracy del 95%, a lo largo de las 24 horas, sin embargo, hay que tomar en cuenta que esto se basa en la visualización y entrenamiento del primer dataset proporcionado por el socio formador, esto quiere decir en donde en la imagen se apreciaban 3 camas con una iluminación buena.

VII. PREDICCIONES

Una vez finalizado el entrenamiento del modelo, se procedió a evaluarlo utilizando un nuevo conjunto de imágenes proporcionado por el socio formador. Este nuevo conjunto presenta un escenario distinto al del entrenamiento original: únicamente se visualizan dos camas por imagen, y la posición de la cámara genera condiciones desfavorables de iluminación, especialmente durante la noche, lo que dificulta la distinción clara de las siluetas de las vacas.

Para esta evaluación, se utilizó una muestra etiquetada manualmente compuesta por 1,152 imágenes. Este nuevo dataset permite estimar el desempeño del modelo en condiciones reales y diferentes a las del entorno de entrenamiento, proporcionando una medida más representativa del error de generalización.

Al aplicar por primera vez el modelo entrenado sobre este nuevo conjunto de datos, se observaron errores en las predicciones, lo cual era esperado dado el cambio en las condiciones visuales y de disposición espacial. En términos de etiquetas reales, la distribución de clases fue la siguiente:

- 48 imágenes con vacas de pie
- 283 imágenes con vacas acostadas
- 821 imágenes con camas vacías

Mientras que el modelo predijo lo siguiente:

```
Conteo total de clases:
cama_vacia       686
vaca_acostada    412
vaca_parada       54
```

Los resultados iniciales mostraron que el modelo tendía a confundir camas vacías con vacas acostadas. Este comportamiento puede explicarse por la baja iluminación, que provoca que las sombras y formas presentes en una cama vacía durante la noche sean similares a las de una vaca recostada, afectando la capacidad del modelo para distinguir correctamente entre ambas clases. Esta observación sugiere la necesidad de ajustar el modelo, ya sea mediante técnicas de *fine-tuning* con datos del nuevo entorno o implementando mejoras en el preprocesamiento y la adquisición de imágenes, como el uso de iluminación infrarroja o filtros adaptativos.

VIII. MEJORAS

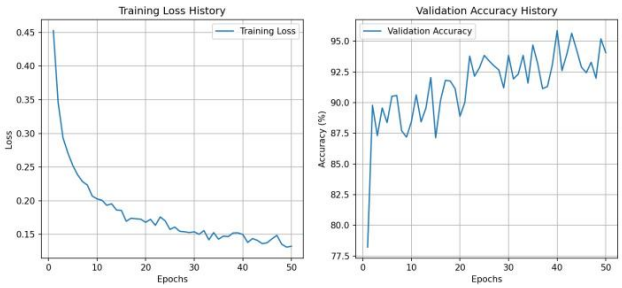
Posteriormente, se realizó un reentrenamiento del modelo incorporando las nuevas imágenes clasificadas al dataset original, con el objetivo de mejorar su desempeño ante las condiciones cambiantes del entorno, especialmente en lo referente a la iluminación y la nueva disposición de las camas. Esta actualización del conjunto de datos permitió al modelo exponerse a mayor variabilidad visual, fortaleciendo su capacidad de generalización.

Durante esta fase de reentrenamiento, se ajustaron los pesos asignados a las clases en la función de pérdida, incrementando la penalización para las clases de *vaca de pie* y *cama vacía*. Esta decisión se basó en los resultados previos, donde se observó una mayor tasa de error en estas categorías, particularmente en la distinción entre vacas

acostadas y camas vacías bajo condiciones de baja iluminación.

Adicionalmente, se modificaron las transformaciones aplicadas a las imágenes en el preprocesamiento, con el fin de resaltar características visuales importantes. Se incorporó un ajuste de saturación, brillo y contraste mediante la transformación transforms.ColorJitter(brightness=0.3, contrast=1.2, saturation=0.5). Esta técnica permitió simular diferentes condiciones de iluminación y mejorar el contraste de los objetos presentes en las imágenes, facilitando al modelo la identificación de bordes, siluetas y texturas relevantes durante el entrenamiento.

Estos cambios en conjunto permitieron optimizar la capacidad del modelo para adaptarse a entornos con iluminación deficiente y estructuras visuales más complejas, mejorando así su precisión y robustez en escenarios reales.



MEJORAS OPTENIDAS