UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

TESIS

APLICACIÓN DE VISIÓN ARTIFICIAL EN LA ESTIMACIÓN DEL PESO CORPORAL DEL CUY

Para optar al título profesional de :
INGENIERO INFORMÁTICO Y DE SISTEMAS

Presentado por:
Br. ZAPATA TTITO, ABEL GABRIEL

Asesor:
M. Sc. ORMEÑO AYALA, YESHICA ISELA

CUSCO - PERÚ
2022

Agradecimientos

Agradezco a mi familia con especial mención a mis padres Celestino Rene y Aquilina, quienes siempre me apoyaron en todo momento para cumplir la meta de ser un buen profesional. También a mis hermanos Teofano Rene, Jackeline Katy, Denisse Elizabeth, Ebelyn Nancy y Yadira Melina por todo su apoyo y paciencia brindado en el cumplimiento de mis objetivos, les agradezco por acompañarme en todo momento.

Agradezco a la Universidad Nacional de San Antonio Abad del Cusco y la escuela profesional de Ingeniería Informática y de Sistemas donde me formé para llegar a ser un buen profesional, además de los docentes que me guiaron en este ambito profesional.

Agredezco de forma especial a mi asesora la profesora Yeshica Isela Ormeño Ayala por su ayuda, recomendaciones y sugerencias durante el proceso de investigación y desarrollo de mi proyecto de tesis.

A mis amigos de la universidad, que siempre me apoyaron y me incentivaron en el desarrollo de la tesis.

Agradezco también al Consejo Nacional de Ciencia, Tecnología e Innovación Tecnológica (CONCYTEC) y al Fondo Nacional de Desarrollo Científico, Tecnológico y de Innovación Tecnológica (FONDECYT), que han financiado el desarrollo de esta tesis.

Resumen

La estimación del peso es un tema de investigación que ha sido abordado usando técnicas de visión artificial, siendo la más reciente: la red neuronal convolucional (CNN), la cuál ha permitido obtener buenos resultados en la estimación del peso animal como: el cerdo, la vaca y el pescado. En la región del Cusco, el peso del cuy es un factor muy importante en la crianza de cuyes tanto en el sector comercial (productores) y el ámbito académico (investigadores), porque permite realizar un monitoreo al cuy durante su ciclo productivo. Este trabajo propone como objetivo general la aplicación de visión artificial mediante la adaptación de un modelo *Convolutional Neural Network* (CNN) en la estimación del peso del cuy. Para lograr este objetivo: primero, se realizó una revisión bibliográfica de investigaciones sobre la estimación del peso animal; segundo, se recolectó 1561 imágenes de cuyes y sus respectivos pesos para construir un dataset de imágenes, máscaras y pesos de cuyes; tercero, se adaptó el modelo Mask R-CNN añadiéndole una rama que permita estimar el peso del cuy; y cuarto, se analizó los resultados obtenidos después del entrenamiento del modelo para mostrar la precisión que se obtiene en la estimación del peso del cuy. También se procedió a la implementación de un prototipo que sirva como una herramienta en la estimación del peso del cuy. Como resultado se obtuvo un R^2=80 % de precisión en la estimación del peso del cuy y un MAPE de 12.41 %.

Palabras clave: visión artificial, máscara, Mask R-CNN, estimación de peso, prototipo, cuy

Abstract

Weight estimation is a research topic that has been approached using computer vision techniques, the most recent is: the convolutional neural network (CNN), which has allowed to get good results in animal weight estimation such as: pigs and fish. In the Cusco region, the weight of guinea pig is a very important factor in the raising of guinea pigs both in the commercial sector (producers) and the academic sphere (researchers), because it allows monitoring of the guinea pigs during their productive cycle. This work proposes as a main purpose the application of computer vision by adapting a CNN model for estimating guinea pig weight. To achieve this target: first, a bibligraphic review of research on the animal weight estimation was carried out; second, 1561 images of guinea pigs and their respective weights were collected to build a dataset of images, masks and weigts of guinea pigs; third, the Mask R-CNN model was modified by adding a new branch that allows estimating the weight of guinea pig; and fourth, ther results obtained after training the model were analyzed to show the precision obtained in guinea pig weight estimation. Also, a prototype was implemented to serve as a tool for estimating guinea pig weight. As a results, an R^2=80 % precision was obtained in guinea pig weight estimation and a MAPE of 12.41 %.

Keywords: computer vision, mask, Mask R-CNN, weight estimation, prototype, guinea pig

Índice general

Agradecimientos **II**

Resumen **III**

Abstract **IV**

Índice general **V**

Índice de figuras **VIII**

Lista de tablas **X**

Abreviaturas **XI**

Introducción **1**

1. Aspectos Generales **3**
1.1. Problema de investigación 3
1.1.1. Planteamiento del problema 3
1.1.2. Formulación del problema 4
1.2. Antecedentes 4
1.3. Justificación 9
1.4. Objetivos 10
1.4.1. Objetivo General 10
1.4.2. Objetivos Específicos 10
1.5. Alcances y Limitaciones 11
1.5.1. Alcances 11
1.5.2. Limitaciones 11
1.6. Metodología 11
1.7. Resultados esperados 13
1.8. Cronograma de actividades 14

2. Marco Teórico **15**
2.1. Cuy (*Cavia porcellus*) 15
2.1.1. Características morfológicas 16
2.1.2. Tipos de cuyes 16
2.1.3. Valor nutricional del cuy 19

2.1.4. Manejo productivo del cuy 19
2.2. Visión Artificial 23
2.2.1. Formación y representación de la imagen 24
2.2.2. Aplicaciones de visión artificial 24
2.2.3. Etapas en el proceso de la visión artificial 26
2.2.4. Aumento de datos 28
2.3. Machine Learning 29
2.3.1. Tipos de *machine learning* 29
2.3.2. Aprendizaje supervisado 30
2.3.3. Proceso del *Machine Learning* 31
2.4. Deep Learning 32
2.4.1. Red Neuronal Convolucional 33
2.4.2. Transfer learning 41
2.5. Detección de Objetos y Segmentación de Imágenes 43
2.5.1. Tipos de métodos de detección 44
2.5.2. Métodos basados en propuestas de regiones 44
2.5.3. Métricas para la detección de objetos 49
2.5.4. *Non-Maximun Supression* 53
2.5.5. Herramientas 54

3. Desarrollo del proyecto **57**
3.1. Recolección de datos 57
3.2. Procesamiento de imágenes 59
3.2.1. Selección de imágenes 60
3.2.2. Segmentación de imágenes 60
3.2.3. Aumento de datos 61
3.3. Construcción del dataset 63
3.3.1. Descripción del dataset 63
3.4. Adaptación de la arquitectura CNN 64
3.4.1. Recursos 65
3.4.2. Implementación 66
3.4.3. Determinar parámetros de entrenamiento 72
3.5. Prototipo implementado 73
3.5.1. Backend 73
3.5.2. Frontend 74

4. Resultados **76**
4.1. Métricas de evaluación 77
4.2. Evaluación del modelo A implementado 79
4.3. Evaluación del modelo B implementado 85
4.4. Refinamiento 90
4.5. Comparación con otras investigaciones 92
4.6. Detalles técnicos 92
4.6.1. Recursos hardware 92
4.6.2. Recursos software 93

Conclusiones **94**

Recomendaciones **96**

Bibliografía **97**

Índice de figuras

1.1. Esquema del funcionamiento de la estimación del peso en el cerdo . . . 7
1.2. Ejemplos de los datos del dataset de peces . . . 8
1.3. Flujo para la segmentación de ganado vacuno y extracción de contorno 9
1.4. Esquema de la metodología propuesta. . . . 12
1.5. Cronograma de actividades. . . . 14

2.1. Ejemplares de cuy criado en cautiverio . . . 15
2.2. Clasificación según la conformación del cuerpo . . . 17
2.3. Clasificación según el pelaje . . . 17
2.4. Clasificación según linea o raza . . . 18
2.5. Ciclo productivo del cuy . . . 20
2.6. Manejo integral de los cuyes . . . 20
2.7. Esquema general de visión por computadora . . . 24
2.8. Diagrama de bloques de un sistema de visión artificial . . . 27
2.9. Gráfica de clasificación y regresión . . . 31
2.10. Diagrama de Venn que muestra la relación de AI, ML y DL . . . 33
2.11. Arquitectura de una red neuronal convolucional . . . 34
2.12. Componentes de una típica capa de CNN . . . 35
2.13. Aplicación de una convolución . . . 36
2.14. Ejemplo del funcionamiento de una neurona artificial . . . 37
2.15. Gráfico de la activación sigmoide . . . 38
2.16. Gráfico de la activación *Rectified Linear Unit* (ReLU) . . . 38
2.17. Aplicación de *max-pooling* . . . 39
2.18. Capa totalmente conectada . . . 40
2.19. Transfer learning . . . 41
2.20. Formas cómo *trasnfer learning* puede mejorar el aprendizaje . . . 42
2.21. Ejemplo de detección de objetos . . . 43
2.22. Diagrama de flujo de R-CNN . . . 45
2.23. Arquitectura de Fast R-CNN . . . 46
2.24. Arquitectura de Faster R-CNN . . . 46
2.25. FPN . . . 47
2.26. Arquitectura de Mask R-CNN . . . 48
2.27. Verdad fundamental vs resultado de un modelo . . . 50
2.28. *Intersection over Union* (IoU) . . . 50
2.29. Ejemplo de curva de *precision/recall* . . . 51
2.30. Suavizado de la curva *precision/recall* . . . 52

2.31. Cálculo de *Average Precision* (AP) 53
2.32. Algoritmo de *Nom-Maximum Supression* 54
2.33. Espacio de trabajo de Labelbox 55

3.1. Diagrama de flujo para la recolección de datos 58
3.2. Captura de imágenes de un cuy 59
3.3. Diagrama de flujo del procesamiento de imágenes 59
3.4. Redimensión de una imagen 60
3.5. Segmentación de una imagen usando Labelbox 61
3.6. Distribución Normal o de Gauss 62
3.7. Aumento de datos 62
3.8. Comparación de histograma original vs aumentado 64
3.9. Arquitectura de Mask R-CNN 65
3.10. Resultados de Mask R-CNN para la segmentación de instancias 65
3.11. Arquitectura de ResNet50 66
3.12. Diagrama de la rama para la máscara y el peso 69
3.13. Diagrama de la rama para el peso 71
3.14. Arquitectura del prototipo 73
3.15. Proceso de inferencia del API 74
3.16. Interfaz de la Aplicación para la estimación del peso del cuy 75

4.1. Gráfica de dispersión de la inferencia ResNet50A Refinado (80 %) con el dataset de evaluación 76
4.2. Gráfica de la curva de pérdida del modelo ResNet50A y COCO 80
4.3. Gráfica de la curva de pérdida del modelo ResNet50A e ImageNet . . . 80
4.4. Gráfica de dispersión de la inferencia ResNet50A 82
4.5. Gráfica de la curva de pérdida del modelo ResNet101A y COCO 82
4.6. Gráfica de la curva de pérdida del modelo ResNet101A e ImageNet . . 83
4.7. Gráfica de dispersión de la inferencia ResNet101A 84
4.8. Gráfica de la curva de pérdida del modelo ResNet50B y COCO 85
4.9. Gráfica de la curva de pérdida del modelo ResNet50B e ImageNet . . . 86
4.10. Gráfica de dispersión de la inferencia ResNet50B 87
4.11. Gráfica de la curva de pérdida del modelo ResNet101B y COCO 87
4.12. Gráfica de la curva de pérdida del modelo ResNet101B e ImageNet . . 88
4.13. Gráfica de dispersión de la inferencia ResNet101B 89

Lista de tablas

2.1. Composición de la carne de cuy con relación a otras especies 19
2.2. Métricas para *Common Objects in Context* (COCO) 53

3.1. Fecha y muestra de recolección de datos 58
3.2. Comparación de las arquitecturas de redes troncales 66
3.3. Descripción de la rama A 70
3.4. Descripción de la rama B 72

4.1. Evaluación de las métricas sobre ResNet50A 81
4.2. Evaluación de las métricas sobre ResNet101A 83
4.3. Evaluación de las métricas sobre ResNet50B 86
4.4. Evaluación de las métricas sobre ResNet101B 88
4.5. Comparación de Resultados (R^2) 90
4.6. Evaluación de las métricas sobre el refinamiento 91
4.7. Comparación con resultados de antecedentes 92

Abreviaturas

AI *Artificial Intelligence*
ANN *Articial Neural Network*
AP *Average Precision*
API *Application Programming Interface*
AUC *Area Under Curve*
CNN *Convolutional Neural Network*
COCO *Common Objects in Context*
CV *Computer Vision*
DL *Deep Learning*
FC *Fully Connected*
FPN *Feature Pyramid Network*
GPU *Graphical Proccesor Unit*
ILSVRC *ImageNet Large Scale Visual Recognition Challenge*
IoU *Intersection over Union*
MAE *Mean Absolute Error*
mAP *Mean Average Precision*
MAPE *Mean Absolute Percentage Error*
MARE *Mean Absolute Relative Error*
ML *Machine Learning*
MLP *MultiLayer Perceptron*
MSE *Mean Squared Error*
NMS *Non-Maximun Supression*
R-CNN *Regions with CNN features*
ReLU *Rectified Linear Unit*
RMSE *Root Mean Squared Error*
RMSRE *Root Mean Squared Relative Error*
RoI *Region of Interest*
RPN *Region Proposal Network*
SVM *Support Vector Machines*

Introducción

El peso corporal del cuy es una medida importante en la crianza del cuy (*cavia porcellus*) porque con este dato se puede evaluar los factores que inducen a que un cuy obtenga un mayor peso durante su crianza, además de ser información útil en investigaciones donde el objetivo es mejorar la especie del cuy (mayor tamaño mayor peso) para que produzca más carne. El cuy es un animal muy importante en la cultura andina, por su adaptación a diversos ecosistemas, ciclo reproductivo corto y alimentación versátil; siendo parte de la dieta de la población de la región del Cusco.

Además, la visión artificial ha demostrado ser eficaz en varios tareas como en: la clasificación de imágenes, la detección de objetos y la segmentación de instancias al utilizar las *Convolutional Neural Network*s (CNN) para este propósito. Además que las técnicas de visión artificial han permitido realizar la automatización de diversas tareas, incluidas la estimación de peso, por lo que es importante su aplicación en animales de granja como los cuyes, a los cuáles no se les ha realizado alguna investigación usando métodos de visión artificial como las CNNs.

Por lo tanto, en la presente tesis se muestra la propuesta de la aplicación de visión artificial al utilizar una arquitectura de detección de objetos para ubicar la posición de un cuy (criado en pozas) y estimar mediante la imagen obtenida el peso del cuy, siendo una herramienta para investigaciones que contemplen el peso del cuy como una variable a optimizar, además de permitir a las personas acceder a un método no invasivo que estime el peso del cuy para su respectivo seguimiento durante su ciclo productivo.

La presente tesis tiene la siguiente estructura:

Capítulo 1 Aspectos generales. Capítulo que comprende: el problema de investigación, los antecedentes, la justificación, el objetivo general y los objetivos específicos, los alcances y limitaciones, la metodología usada, los resultados esperados y el cronograma de actividades

Capítulo 2 Marco teórico. Se presentan los conceptos importantes relacionados a la investigación, estos conceptos sientan la base para el desarrollo del proyecto.

Capítulo 3 Desarrollo del proyecto. Se desarrolla la aplicación de visión artificial desde la recolección de los datos hasta el desarrollo de la arquitectura para la estimación del peso del cuy, incluyendo el desarrollo de un prototipo.

Capítulo 4 Análisis de resultados. Capítulo donde se muestran los resultados obtenidos durante el entrenamiento del modelo implementado en la estimación del peso del cuy.

Capítulo 5 Conclusiones y trabajos futuros. Se desarrolla las conclusiones del trabajo de investigación y las recomendaciones para trabajos futuros.

Capítulo 1

Aspectos Generales

1.1. Problema de investigación

1.1.1. Planteamiento del problema

El cuy es un animal muy importante en la cultura andina que forma parte de la gastronomía peruana ya que su carne posee un alto valor nutritivo, caracterizándose por contener un alto valor proteico y bajo en grasa, además de colesterol de buena calidad, minerales y vitaminas. En el proceso de crianza de cuyes siempre se busca obtener un ejemplar con un buen peso, por lo que los productores monitorean su ciclo productivo que consta de las siguientes etapas: lactancia, recría, engorde y reproducción. Después de la etapa de lactancia y durante la etapa de engorde es importante pesar continuamente al cuy para realizar un seguimiento y determinar que ejemplares pueden ser utilizados como reproductores o pueden ser comercializados. Esta tarea resulta laboriosa tanto en tiempo como en personal cuando se tiene una granja con una gran cantidad de cuyes, además de provocarles estrés porque tienen que ser movilizados a una balanza, afectando en buena medida su salud y su correcto desarrollo. Por esta razón algunos productores realizan este seguimiento al final de su ciclo productivo, cuando el cuy ya está listo para la venta. Esto implica descuidar ciertos factores que afectan su desarrollo correcto, los cuales se podrían verificar realizando un seguimiento constante en su peso.

La estimación del peso animal ya ha sido realizada en algunos animales de granja como los cerdos (Buayai *et al.* (2019), Jun *et al.* (2018), Jensen *et al.* (2018), Wang *et al.* (2008)), las vacas y en peces (Konovalov *et al.* (2019)), usando diversas técnicas de visión artificial, para realizar un seguimiento al animal y evitar movilizar dicho animal durante la realización de la tarea. Esta tarea la realizaron con un conjunto de imágenes de los animales etiquetados con sus respectivos pesos, sobre este conjunto realizan procesos de procesamiento de imágenes, segmentación y extracción de características. En trabajos recientes se usó la técnica de red neuronal convolucional o también conocida

como CNN sobre la máscara ya procesada para estimar el peso del animal, obteniendo buenos resultados que fueron evaluados usando métricas de regresión, que brindan la precisión con la que una técnica de visión artificial estima el peso del animal. Esta precisión es importante porque permite verificar si el modelo desarrollado se ajusta a estimar de manera correcta el peso del animal.

A pesar del desarrollo de las investigaciones en la estimación del peso usando técnicas de visión artificial, todavía no existe alguna desarrollada en cuyes, además que estas investigaciones dependen de la segmentación de las imágenes como paso previo para poder estimar el peso del animal, realizando en tareas separadas. Por esta razón la solución planteada es aplicar visión artificial para estimar el peso del cuy, lo que conlleva a desarrollar un dataset de imágenes de cuyes etiquetados con sus pesos, seleccionar y adaptar un modelo de CNN para que estime el peso del cuy basado en imágenes, evaluar los resultados de los entrenamientos para verificar la precisión que brinda el modelo en la estimación del peso del cuy, esta precisión nos indicará el grado de confiabilidad que se podrá obtener en la estimación del peso del cuy. Para finalmente implementar un prototipo de aplicación móvil que sirva como herramienta en la estimación del peso en cuyes.

1.1.2. Formulación del problema

¿Cómo se podría desarrollar un sistema de visión artificial para la estimación del peso corporal del cuy?

1.2. Antecedentes

La estimación del peso corporal de un animal es un tema que se ha desarrollado usando diversas técnicas de inteligencia artificial (AI), los trabajos realizados con estas técnicas han mejorado el desempeño en la estimación de peso teniendo como entrada imágenes 2D y 3D. Además, que de indicar que no existen investigación sobre la estimación del peso aplicadas en el cuy. Las investigaciones mas relevantes son las siguientes:

- **Paper: *"Semi-automatic pig weight estimation using digital image analysis"*** P. Buayai, K. Piewthongngam, C.K. Leung, K.R. Siakaew (Buayai *et al.*, 2019)

  Este trabajo de investigación tiene como objetivo: desarrollar y diseñar un sistema para pesar cerdos usando visión artificial sin cambiar el ambiente y estructura de una granja. El proceso de pesar juega un rol importante en la crianza de los cerdos porque usualmente son vendidos por su peso, además que un enfoque tradicional maneja el pesado manual en el ambiente de la granja, el cuál es lento y estresante debido al movimiento de estos animales de granja en un espacio atestado.

Los materiales que usaron en esta investigación son cámaras de bajo costo ubicadas en la parte superior del corral para capturar los cerdos mientras se mueven dentro del corral, además de adoptar software de código abierto. La metodología que usaron se describe de la siguiente forma: 1) convertir el vídeo a imagen, 2) detectar y refinar los bordes del cerdo mediante un umbral adaptativo, 3) aplicar particionamiento adaptativo con los componente conectados para obtener un umbral de cada partición y la máxima entropía para identificar el valor del umbral de cada partición, 4) calibrar la cámara para la detección de esquinas, 5) extraer las características de la imagen del cerdo, 6) construir un modelo de predicción usando un *MultiLayer Perceptron* (MLP) entrenado con *back-propagation*, y 7) validar el resultado de la red usando la desviación porcentual media o *Mean Absolute Percentage Error* (MAPE).

Se recolectó 230 imágenes, separando 70 % para el entrenamiento y 30 % para la evaluación del modelo. En la experimentación se evaluaron diferentes características extraídas usando el software OpenCV además de evaluar las funciones de transferencia (identidad, sigmoide, umbral, lineal, sigmoide bipolar) que mejor resultados muestren con estos datos, evaluando el MAPE, tiempo y R^2.

Los resultados del trabajo son los siguientes: obtiene la imagen del cerdo dentro del corral en una rutina cotidiana, adaptando una red neuronal artificial (ANN) con un error de 2.84 % MAPE y R^2 0.84 con un tiempo de evaluación aproximado de 4.66 segundos. Comparando con enfoques tradicionales, este método permite obtener el peso del cerdo en el corral, siendo información útil para los productores.

- **Paper: *"Estimating pig weights from images without constraint on posture and illumination"*** Kyungkoo Jun, Si Jung Kim, Hyun Wook Ji (Jun *et al.*, 2018)

  El artículo propone un método de estimación del peso del cerdo basado en imagen diferente de otros trabajos porque: primero, no tiene restricciones en la postura del cerdo, segundo, se basa solo en imágenes 2D y tercero, usa técnicas de *Machine Learning* (ML).

  Este trabajo es importante para el monitoreo de los pesos de los cerdos, ya que los cambios en los pesos provee información sobre la salud y el estado de crecimiento de los cerdos. El trabajo busca diferenciarse de investigaciones previas que tienen limitaciones más marcadas sobre las condiciones del ambiente donde están los cerdos.

  El método de estimación de peso basado en imagen, parte con la instalación de una cámara a una altura fija sobre el corral. Luego procede con la segmentación de las imágenes obtenidas, esta acción se realiza al convertir la imagen a binario y luego aplicando operaciones morfológicas. La segmentación permite eliminar el ruido que tiene la imagen, además de eliminar la cabeza y la cola porque estas partes no se correlacionan con el peso del cerdo. Luego, mediante técnicas de ML se proceden a extraer características dentro de un vector. Este vector es la

entrada en una ANN, siendo desarrollados diferentes configuraciones durante el entrenamiento.

El entrenamiento lo realizan con 513 imágenes separados entre los datos de entrenamiento (80 %) y los de validación (20 %). Las métricas para la evaluación son: *Mean Absolute Error* (MAE), *Root Mean Squared Error* (RMSE) y el coeficiente de determinación R^2.

El proyecto consigue tener un error promedio de 3.15 kg con una configuración de 5 capas de la siguiente forma 9-64-32-16-1 que obtiene un MAE menor de 3.2 %, un RMSE cercano a 3.8 % y un $R^2 = 0{,}79$. A pesar de que los resultados no son notables, esta investigación no considera la postura del cerdo, evita la intervención humana para la selección de imágenes y estima en las condiciones ambientales propias del corral.

- **Paper: *"Automatic estimation of slaughter pig live weight using convolutional neural networks"*** Dan B. Jensen, Katarina N. Dominiak, Lene J. Pedersen (Jensen *et al.*, 2018)

  Este artículo inicia con la mención de que una estimación exacta de los pesos en cerdos acabados (cerdos para ser llevados a un matadero) es útil porque: primero, le permite al granjero conocer el tiempo óptimo para llevar al cerdo al matadero; segundo, es un monitor exacto de las curvas de crecimiento para detectar problemas en la formulación del alimento, enfermedades o problemas con el ambiente; finalmente, para la dosificación de medicina. Además que el manejo de cerdos es laborioso por la gran cantidad de cerdos que maneja un granjero, haciendo inviable el control de peso de los cerdos. Este estudio tiene como objetivos: demostrar que una CNN puede ser usada para estimar el peso individual del cerdo basado en imágenes de vídeo, comparar la precisión y la fiabilidad dado cierta resolución de imagen y comparar la precisión y fiabilidad dado la inclusión o exclusión de imágenes sin cerdos dentro de la imagen.

  Para la recolección de datos se procedió a colocar una cámara encima del corral, la cuál capturó imágenes de 17 cerdos de un corral que han sido enumerados en su espalda para su identificación. Cada semana son pesados manualmente para obtener su peso. Solo las imágenes con el número de identificación legible y el cuerpo entero en la imagen son seleccionadas.

  La preparación y modelado de datos se realiza usando el lenguaje R en una computadora con 64GB de RAM. Se hace uso del aumento de datos, volteando horizontalmente y verticalmente las imágenes, además de convertir a escala de grises todas las imágenes. Usando el paquete mxnet se procedió a entrenar la CNN en un máximo de 1000 iteraciones o hasta que *Root Mean Squared Relative Error* (RMSRE) sea menor que el 10 %. La arquitectura CNN consiste en 2 capas convoluciones y 2 capas totalmente conectadas. Cada capa convolucional usa ReLU como función de activación y *max-pooling* para la agrupación. La primera capa convolucional tiene 20 filtros y la segunda tiene 50 filtros, con kernels de 5x5

píxeles. La capa de salida usa la regresión lineal como función de activación. Para medir el desempeño se hace uso del coeficiente de determinación R^2, RMSRE y *Mean Absolute Relative Error* (MARE).

En los resultados se evidencia que la inclusión de imágenes que no contienen cerdos reduce el R^2, y no altera los valores de RMSRE y MARE. En las conclusiones se demuestra que una CNN puede ser entrenada para estimar el peso del cerdo. El mejor desempeño es dado por $R^2 = 0{,}95$ con un error promedio $\pm 7\,\%$. Además de mostrar que resoluciones grandes no garantizan el mejor desempeño de la CNN. En la Figura 1.1 se puede evidenciar cómo se realiza la estimación del peso del cerdo y como calcular el error al obtener la estimación en un cerdo.

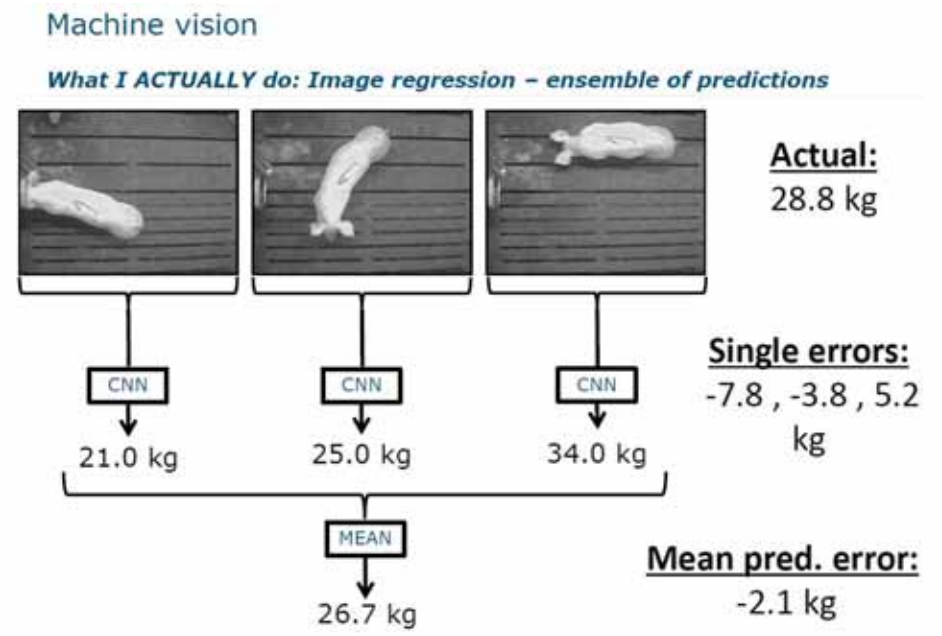


Figura 1.1: Esquema del funcionamiento de la estimación del peso en el cerdo
Fuente: Jensen *et al.* (2018)

- **Paper: *"Automatic weight estimation of harvest fish from images"*** Dimitry A. Konovalov, Alzayat Saleh, Dina B. Efrenova, Jose A. Domingos, Dean R. Jerry (Konovalov *et al.*, 2019)

  El aumento del volumen de animales y el incremento del costo de labor humana conducen al desarrollo de sistemas de visión artificial o *Computer Vision* (CV) dentro de la industria de la acuicultura, como en la medición o estimación automática de características morfológicas del pescado (longitud, ancho y masa). Donde las investigaciones previas hacen uso de modelos matemáticos y en datasets reducidos. El proceso de esta investigación fue diseñado para ser rápido y procesar varios *frames* de vídeo como imágenes individuales en tiempo real de una resolución de $480x480$ píxeles.

  El estudio usa tres dataset que contienen: 445, 600 y 1400 imágenes con sus

respectivos pesos, para eliminar la dependencia de los colores las imágenes son transformadas en escala de grises. Como se muestra en la Figura 1.2, (a-c) muestra las imágenes de los peces con sus pesos respectivos etiquetados, en (d) se observa el peso junto con un valor predecido por el conteo de píxeles, que no se ajusta al valor original, en (e) y (g) se muestra la máscara del pez eliminando las aletas y la cola, en cambio (f) y (h) muestra la aplicación de la máscara en el pez completo.

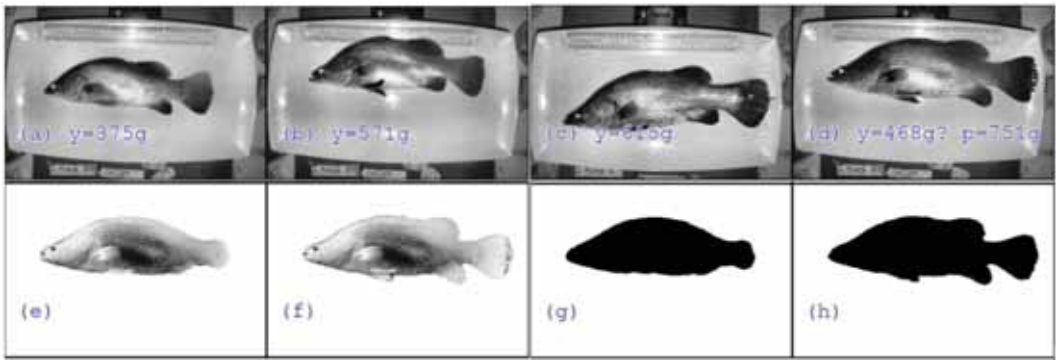


Figura 1.2: Ejemplos de los datos del dataset de peces
Fuente: Konovalov *et al.* (2019)

Para la segmentación semántica de las imágenes, se segmentaron manualmente alrededor de 300 imágenes con una escala de 1 mm-por-píxel, siendo escogidos porciones de cada dataset. Luego se escogió la CNN llamado LinkNet-34 que es un modelo de segmentación basado en ResNet-34.

El proceso de entrenamiento realizado es el siguiente: el dataset es dividido en un conjunto de entrenamiento (80 %) y uno de validación (20 %). Luego se carga en las capas de ResNet-34 los pesos entrenados de ImageNet para transferir ese conocimiento y se selecciona la función de activación sigmoide para la capa de salida. Para reducir el sobreajustamiento los pares de imagen-máscara son aleatoriamente rotados, escalados, recortados y volteados horizontalmente o verticalmente. El entrenamiento lo realizan sobre 100 épocas.

El objetivo de este estudio fue el desarrollo de una mejor práctica para la estimación automática de los peces basados en imágenes. El objetivo es logrado aplicando peso-a-partir-del-área y peso-a-partir-de-la-imagen. Aquí se comparan modelos matemáticos que calculan el peso a partir del área teniendo el mejor resultado con un MAPE=4.33 %, en cambio cuando se uso una versión modificada llamada LinkNet-34R con una capa de regresión se obtiene un MAPE=4.20 %.

- **Paper:** ***"Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming"*** Yongliang Qiao, Matthew Truman, Salah Sukkarieh (Qiao *et al.*, 2019)

  En la ganadería, los enfoques basados en visión artificial se han utilizado ampliamente para obtener información individual sobre la salud y el bienestar del

ganado, como la puntuación sobre la condición corporal, el peso vivo y los comportamientos de actividad. Siendo necesaria segmentar con precisión cada imagen para obtener la información individual del ganado y así poder procesarla.

Este trabajo usa el enfoque de segmentación de instancias conocido como Mask R-CNN para la segmentación de instancias y la extracción de contornos en un entorno real. Como se muestra en la Figura 1.3, el enfoque consta de los siguientes pasos: extracción de fotogramas clave, mejoramiento de la imagen, segmentación del ganado y la extracción del contorno corporal.

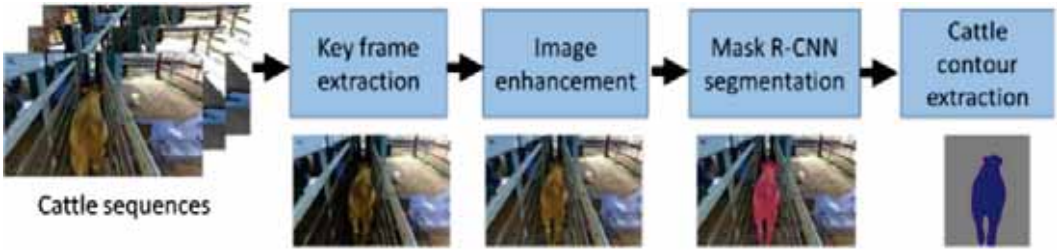


Figura 1.3: Flujo para la segmentación de ganado vacuno y extracción de contorno
Fuente: Qiao *et al.* (2019)

Para la extracción del contorno y segmentación de instancias se utilizó Mask R-CNN con la red troncal de ResNet101, usando en el entrenamiento 1188 imágenes de tamaño 960 x 540 píxeles. Para el entrenamiento se usó 988 imágenes, mientras que para la evaluación se usó 200 imágenes. Además de usar los pesos pre-entrenados con el dataset COCO. El entrenamiento duró casi dos días con un 0.0003 ratio de entrenamiento.

El trabajo muestra que los resultados de Mask R-CNN para la segmentación y extracción de contorno logró mejores resultados que otros modelos similares.

1.3. Justificación

La presente investigación es relevante porque permitirá: primero, explorar investigaciones respecto a la estimación del peso animal basado en imágenes, segundo, proporcionará el desarrollo de un dataset con imágenes de cuyes, máscaras y sus respectivos pesos, y tercero porque se medirá el desempeño de una arquitectura CNN de detección de objetos y segmentación de instancias en la estimación del peso del cuy.

El impacto social de esta investigación es proveer de una herramienta tanto a investigadores como a productores de cuyes. A los investigadores les permitirá tener una herramienta no intrusiva para realizar el seguimiento de la curva de crecimientos del cuy, y validar investigaciones sobre la alimentación o el ambiente de crianza para mejorar su producción cárnica. En caso de los productores, les permitirá una herramienta

para identificar los cuyes que están listos para su venta con su respectivo peso, y también permitirles realizar seguimiento a los pesos de los cuyes, permitiéndoles mejorar su sistema de crianza.

Las implicaciones prácticas de la presente investigación son que permitirán la estimación del peso del cuy basado en imagen, pudiendo ser extendido en otros animales de granja criados en condiciones similares como los cerdos o las vacas. Además de poder considerarse más detalles que sean relevantes a un investigador o productor en la crianza de estos animales.

Su valor teórico permite mostrar la aplicación de una arquitectura CNN diseñada para la detección de objetos y segmentación de instancias en la estimación del peso del cuy, mostrando los resultados de precisión en la estimación del peso. Con los resultados se espera conocer el nivel de precisión que puede tener esta arquitectura CNN en la estimación del peso del cuy.

Además, la investigación permitirá tener una herramienta para estimar el peso del cuy, que podrá servir para la recolección de datos en otras investigaciones. Y también permitirá sugerir la profundización de las investigaciones en la aplicación de técnicas de visión artificial, ML y DL para mejorar la crianza de los animales, como en este caso: el cuy.

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un sistema de visión artificial aplicado en la estimación del peso corporal del cuy usando una arquitectura de red neuronal convolucional.

1.4.2. Objetivos Específicos

- Desarrollar un dataset de imágenes y pesos de los cuyes.
- Seleccionar y adaptar una arquitectura CNN para la estimación del peso del cuy.
- Realizar entrenamientos con la arquitectura CNN adaptada y el dataset desarrollado.
- Evaluar la precisión obtenida después del entrenamiento de la arquitectura CNN propuesta.
- Implementar un prototipo para la aplicación de la estimación del peso del cuy.

1.5. Alcances y Limitaciones

1.5.1. Alcances

- La captura de imágenes se realizará desde una vista de arriba hacia abajo, en una poza al nivel del suelo, además que cada imagen solo contendrá un cuy.
- La segmentación del cuy será realizada manualmente usando el software en linea Labelbox.
- La construcción del dataset se realizará asociando una máscara y un peso en una imagen del cuy, separando el conjunto de entrenamiento del conjunto de validación en una relación de 80 % y 20 % respectivamente.
- La estimación del peso del cuy se realizará usando sólo una arquitectura de CNN para la detección de objetos y segmentación de instancias.
- La evaluación de la arquitectura se realizará sobre los datos del conjunto de entrenamiento y validación.

1.5.2. Limitaciones

- La recolección de datos se realizó en granjas familiares de cuyes en la región del Cusco, donde se tomaron muestras de la población de cuyes del galpón para la captura de información. La recolección de datos es supervisada por el encargado de la granja de cuyes, quién indicará que cuyes pueden ser utilizados para la recolección de datos (algunos productores evitan el traslado de cuyes en estado de gestación, cuyes reproductores o crías).

1.6. Metodología

La metodología de investigación es de tipo experimental, porque se mide la precisión de la arquitectura CNN como variable en la estimación del peso para el desarrollo del sistema de visión artificial (Hernández-Sampieri & Torres, 2018).

Según el tipo de pregunta planteado en el problema esta investigación es práctica-tecnológica porque plantea la aplicación una CNN en la estimación del peso del cuy (Mejía, 2005). En la Figura 1.4 se muestra la secuencia de pasos para el desarrollo de la estimación del peso usando redes neuronales convolucionales, este esquema se basa en las etapas de un sistema de visión artificial (González *et al.*, 2006).

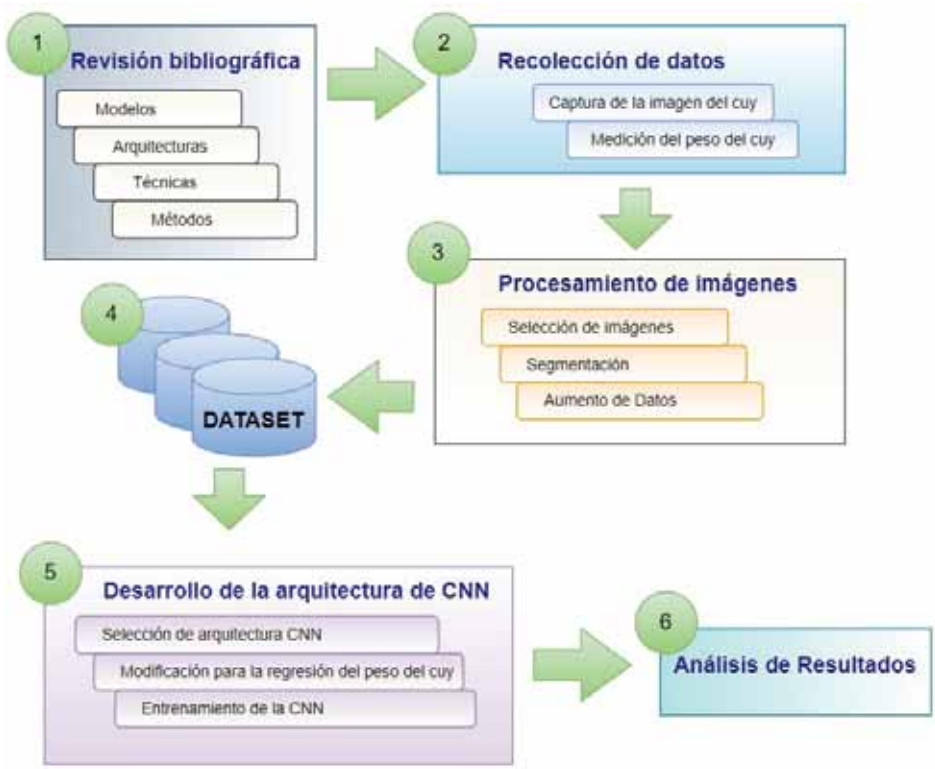


Figura 1.4: Esquema de la metodología propuesta.
Fuente: Elaboración propia

Por tanto, como se muestra en la Figura 1.4 el desarrollo metodológico es el siguiente:

1. **Revisión bibliográfica**

   La revisión bibliográfica se realiza en base al análisis de artículos científicos que tienen como objetivo la estimación del peso animal usando técnicas de visión artificial, ML y *Deep Learning* (DL). Su recopilación se realizó en bibliotecas digitales, de las cuales se obtuvieron los antecedentes para la investigación. Las investigaciones realizadas con imágenes bidimensionales tienen más énfasis en la recopilación de las investigaciones.

2. **Recolección de datos**

   El proceso de recolección de datos comprende desde la captura de la imagen usando una cámara fotográfica y la medición del peso del cuy usando una balanza. En la captura de imágenes se realizaron configuraciones en la cámara para enfocar el cuy respecto a su contexto.

3. **Procesamiento de imágenes**

   El procesamiento de imágenes parte desde la selección de las imágenes que contiene el cuerpo completo del cuy, luego por la segmentación de la máscara del cuy que resalta la ubicación del cuy dentro de la imagen. Finalmente, se aumenta los

datos mediante un volteado (*flipping*) horizontal y vertical de las imágenes.

4. **Construcción del dataset**

   La construcción del dataset se realiza agrupando los datos, para el entrenamiento (80 %) y la validación (20 %), así como su asociación entre la imagen, su máscara y el peso del cuy medido.

5. **Desarrollo de la arquitectura CNN**

   El desarrollo de la aplicación de la arquitectura parte desde la selección de la CNN a utilizarse, además de la modificación de una rama para la regresión del peso del cuy.

6. **Análisis de resultados**

   El análisis de resultados muestra los resultados obtenidos durante el entrenamiento, además de la aplicación de métricas de regresión para la evaluación de los pesos estimados frente a los pesos medidos del cuy.

1.7. Resultados esperados

1. Un dataset de imágenes de cuyes con sus respectivas máscaras y sus pesos.

2. Un modelo de arquitectura CNN aplicado en la estimación del peso corporal del cuy basado en imágenes, que servirá como herramienta de visión artificial.

3. Un prototipo de aplicación móvil y un API para aplicar la estimación del peso del cuy usando un teléfono celular.

1.8. Cronograma de actividades

El cronograma de actividades del presente trabajo se muestra en la Figura 1.5.

Figura 1.5: Cronograma de actividades.
Fuente: Elaboración propia

Capítulo 2

Marco Teórico

2.1. Cuy (*Cavia porcellus*)

El cuy (cobayo o curí) es una especie híbrida doméstica de roedor histricomorfo de la familia Caviidae, originario de la zona andina de Bolivia, Colombia, Ecuador y Perú. Por su capacidad de adaptación a diversas condiciones climáticas, los cuyes pueden encontrarse desde la costa hasta alturas de 4500 metros sobre el nivel del mar, y en zonas tanto frías como cálidas. (Chauca, 1997) Las ventajas de la crianza de cuyes incluyen su calidad de especie herbívora, su ciclo reproductivo corto, la facilidad de adaptación a diferentes ecosistemas y su alimentación versátil (Chauca, 1997). Mayormente la crianza de cuyes se realiza en pozas o en jaulas, la cantidad de cuyes que viven en una poza o jaula depende del tamaño de esta. En la Figura 2.1 se muestra ejemplares de cuy criados en pozas.

Figura 2.1: Ejemplares de cuy criado en cautiverio
Fuente: Fotografía propia

2.1.1. Características morfológicas

La forma de su cuerpo es alargado y cubierto de pelos desde el nacimiento. Las partes del cuerpo de los cuyes se describen a continuación.

- **Cabeza**. Relativamente grande en relación a su volumen corporal, de forma cónica y de longitud variable de acuerdo al tipo de animal. Las orejas por lo general son caídas. Los ojos son redondos vivaces de color negro o rojo, con tonalidades de claro a oscuro.
- **Cuello**. Grueso, musculoso y bien insertado al cuerpo, está conformado por siete vértebras.
- **Tronco**. De forma cilíndrica.
- **Abdomen**. Tiene como base anatómica a 7 vértebras lumbares, es de gran volumen y capacidad.
- **Extremidades**. En general cortas, siendo los miembros anteriores más cortes que los posteriores. Ambos terminan en dedos, provistos de uñas cortas en los anteriores y grandes y gruesas en las posteriores.

2.1.2. Tipos de cuyes

La clasificación de Chauca (1997) se realiza de acuerdo a su conformación, pelaje y tonalidad de pelaje.

- Clasificación según la conformación del cuerpo
  - Tipo A. Corresponde a cuyes mejorados que tienen una conformación enmarcada dentro de un paralelepípedo (Figura 2.2a), clásico en las razas productoras de carne. La tendencia es producir anímales que tengan una buena longitud, profundidad y ancho. Esto expresa el mayor grado de desarrollo muscular, fijado en una buena base ósea. Son de temperamento calmado, responden eficientemente a un buen manejo y tienen una buena conversión alimenticia.
  - Tipo B. Corresponde a cuyes de forma angulosa (Figura 2.2b), cuyo cuerpo tiene poca profundidad y desarrollo muscular escaso. Cuentan con una cabeza triangular y alargada y, tienen mayor variabilidad en el tamaño de la oreja. Es muy nervioso, lo que hace dificultoso su manejo.

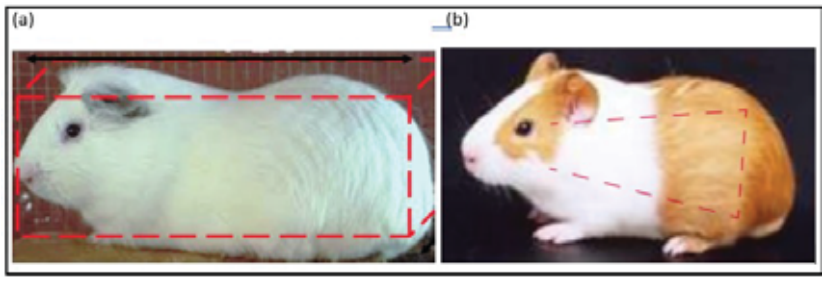


Figura 2.2: Clasificación según la conformación del cuerpo
Fuente: Ataucusi (2015)

- Clasificación según el pelaje

  - Tipo 1. Es de pelo corto, lacio y pegado al cuerpo (Figura 2.3a), es el tipo más difundido. Se encuentra de colores simples claros, oscuros o combinados. Es el que tiene el mejor comportamiento como productor de carne.
  - Tipo 2. Es de pelo corto y lacio, pero forma rosetas o remolinos a lo largo del cuerpo (Figura 2.3b), es menos precoz. Está presente en poblaciones de cuyes criollos, existen de diversos colores. Tiene buen comportamiento como productor de carne.
  - Tipo 3. Es de pelo largo y lacio (Figura 2.3c), presenta dos subtipos que corresponden al tipo 1 y 2 con pelo largo. No es buen productor de carne, más bien es utilizado como mascota.
  - Tipo 4. Es de pelo ensortijado (Figura 2.3d), tornándose erizado. Su forma de cabeza y cuerpo es redondeada, de tamaño medio. Tiene una buena implantación muscular y con grasa de infiltración, el sabor de su carne destaca a este tipo. La variabilidad de sus parámetros productivos y reproductivos le da un potencial como productor de carne.

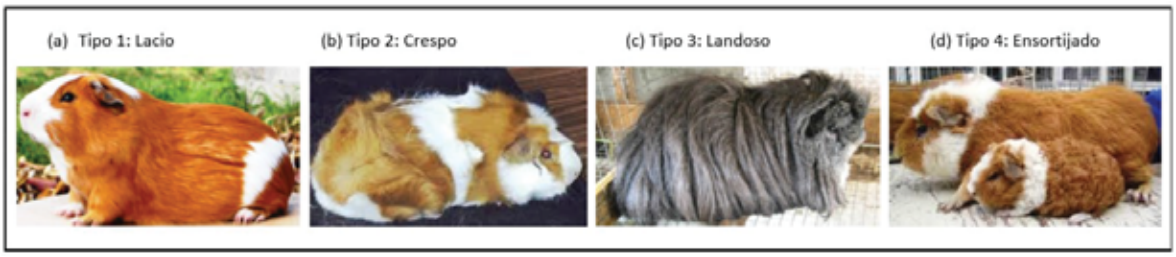


Figura 2.3: Clasificación según el pelaje
Fuente: Ataucusi (2015)

- Clasificación según la coloración del pelaje

  Existen dos tipos de pigmentos que dan coloración al pelaje de los cuyes, estos son: el granular (de variantes: rojo, marrón y negro) y el difuso (variando de color amarillo pálido a marrón rojizo).

La clasificación de acuerdo al color de pelaje se ha realizado en función a los colores simples, compuesto y a la forma como están distribuidos en el cuerpo.

- Pelaje simple. Lo constituyen pelajes de un solo color.
- Pelaje compuesto. Son tonalidades formadas por pelos que tienen dos o más colores.
- Overos. Son combinaciones de dos colores.
- Fajados. Tienen los colores divididos en secciones o franjas de diferentes colores.
- Combinados. Presentan secciones en forma irregular y de diferentes colores.

- Razas y lineas de cuyes

  - Raza Perú (Figura 2.4a). Es una raza pesada, con desarrollo muscular marcado, es precoz y eficiente convertidor de alimento. El color de su capa es alazán con blanco; puede se combinada o fajada, por su pelo liso corresponde al Tipo A. Se uso preferentemente es como macho reproductor.
  - Raza Andina (Figura 2.4b). Se caracteriza por su alta prolificidad y alta incidencia de gestación postparto. La raza andina se adapta a los ecosistemas de costa, sierra y selva alta. Se usa principalmente como reproductor hembra.
  - Raza Inti (Figura 2.4c). Se caracteriza por poseer un pelaje lacio y corto, además de presentar color bayo (amarillo) en todo el cuerpo o combinado con blanco. Posee una forma redondeada. Es la raza que mejor se adapta al nivel de los productores logrando los más altos índice de sobrevivencia. Es una raza intermedia entre la raza Perú y la Andina; es un animal prolífico y se adapta fácilmente a diferentes pisos altitudinales.
  - Raza Inca (Figura 2.4d). Se caracteriza por poseer un mayor número de crías por parto y cualidades productivas y reproductivas sobresalientes, como un menor índice de mortalidad de las crías y una óptima conversión alimenticia.

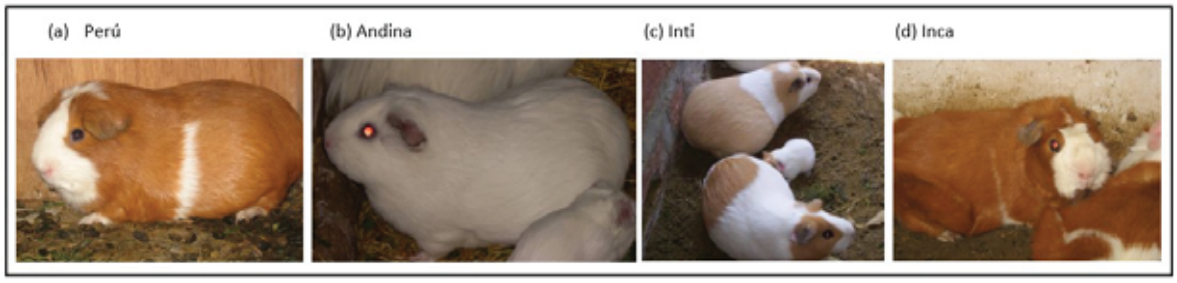


Figura 2.4: Clasificación según linea o raza
Fuente: Machaca (2020)

2.1.3. Valor nutricional del cuy

La carne del cuy tiene un alto valor nutricional, debido a que contiene los aminoácidos esenciales y ácidos grasos esenciales requeridos en la nutrición humana. Según las tablas peruanas de composición de alimentos 2017, elaborado por el Centro Nacional de Alimentación y Nutrición del Instituto Nacional de Salud (INS) del Ministerio de Salud (Minsa), la carne del cuy contiene 78,1 % de agua; 19 % de proteína; 1,6 % de grasa; 1,2 % de minerales y 0,1 % de carbohidratos totales y disponibles. (Andina, 2017)

La carne de cuy es magra, es decir con un porcentaje de grasa menor al 10 %, con alto contenido de proteínas (20,3 %), baja en contenidos de colesterol (65mg/100g) y sodio, por lo que es ideal para incluirla en una alimentación variada y equilibrada. Es una carne apta para todos los grupos poblacionales y en diversas situaciones fisiológicas, como por ejemplo el embarazo o la etapa de lactancia. (Santos, 2007)

Como se muestra en la Figura 2.1 el cuy tiene un alto porcentaje de proteína y de agua, además de un bajo porcentaje de grasa comparado a otras especies animales criadas para el consumo humano.

Especie	Humedad (%)	Proteína (%)	Grasa (%)	Carbohidratos (%)	Minerales (%)
Cuy	70.6	20.3	7.8	0.5	0.8
Aves	70.2	18.3	9.3	1.2	1.0
Cerdos	46.8	14.5	37.3	0.7	0.7
Ovinos	50.6	16.4	31.1	0.9	1.0
Vacuno	58.9	17.5	21.8	0.8	1.0

Tabla 2.1: Composición de la carne de cuy con relación a otras especies
Fuente: Adaptado de Agropecuaria (1981)

2.1.4. Manejo productivo del cuy

El manejo de los cuyes en granja o galpones se basa en el ciclo evolutivo de la especie que está constituido por tres etapas bien definidas: lactación, recría o engorde y reproducción. (Ataucusi, 2015)

Como se muestra en la Figura 2.5 el ciclo productivo puede ser definido en las siguientes etapas: reproducción, lactancia, recría y engorde (recría y engorde como dos etapas separadas), durando el ciclo aproximadamente 3 meses. Durante la etapa de recría y engorde es importante controlar los pesos de los cuyes, para que su optimizar su desarrollo cárnico.

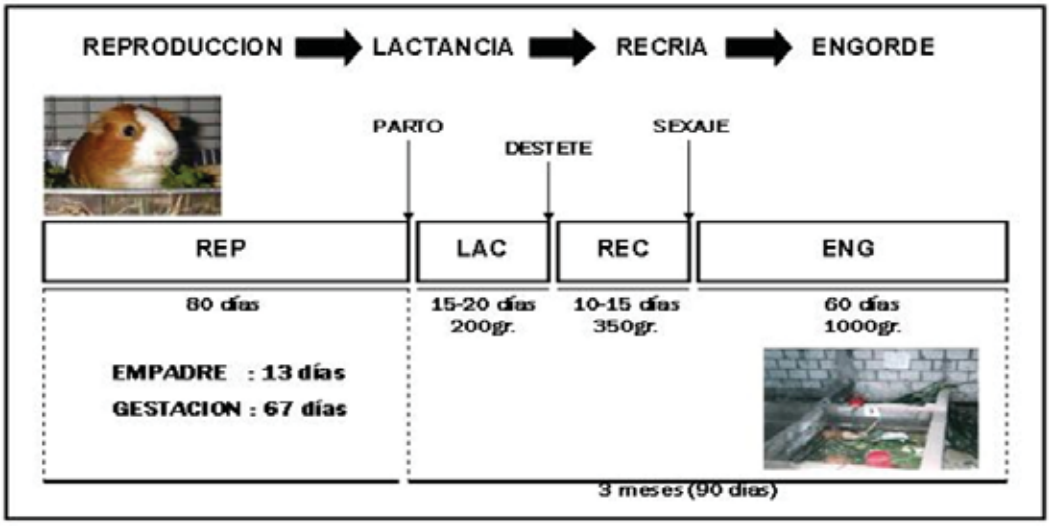


Figura 2.5: Ciclo productivo del cuy
Fuente: Machaca (2020)

En el ciclo productivo del cuy es importante el manejo eficiente de los cuyes reproductores para mejorar su fertilidad, prolificidad (porcentaje de crías nacidas en relación con el total de hembras paridas) y la sobrevivencia de las crías, siendo necesario conocer el comportamiento de los animales antes y durante su etapa reproductiva.

En la crianza de cuyes se sigue una serie de pasos pasos para obtener: animales más gordos, más crías por parto y menos enfermedades (Guerra, 2009). En la Figura 2.6 se muestra en un diagrama como es el manejo de los cuyes y algunas tiempos sugeridos para optimizar su producción.

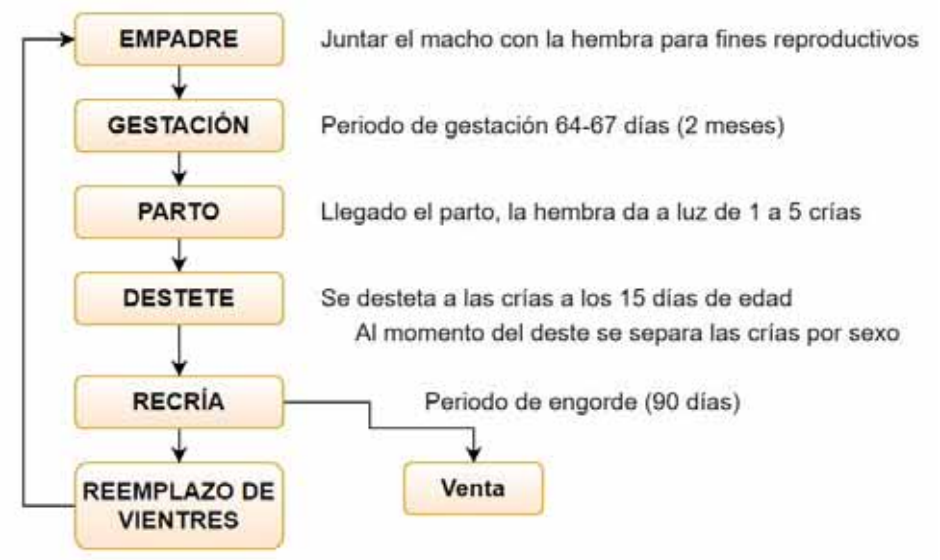


Figura 2.6: Manejo integral de los cuyes
Fuente: Adaptado de Guerra (2009)

Reproducción (Empadre)

La reproducción empieza con el empadre, que consiste en agrupar al macho con 6 a 10 hembras (depende del área de la poza) para su reproducción. La edad de empadre se relaciona con el peso y el grado de mejoramiento del cuy. En animales mejorados las hembras se empadran a partir de los 600 gr. de peso y a una edad promedio de 2 meses y medio, y en el caso de los machos a partir de los 900 gr. a partir de los 3 meses de edad.

Existen dos sistemas de empadre: controlado y continuo. El sistema controlado consiste en separar al macho de las hembras luego del empadre. En cambio, el sistema continuo aprovecha el celo postparto de la hembra para la reproducción, manteniendo el macho con las hembras durante toda su etapa reproductiva. El empadre continuo requiere de una alimentación eficiente para lograr un mayor número de crías.

Gestación

Esta se inicia después de la reproducción de los cuyes:

- Dura entre 63 a 70 días, en promedio 67 días.
- El tiempo de gestación depende del número de crías, a mayor número, menor tiempo de gestación.
- En esta etapa, se debe tener a las hembras en permanente estado de tranquilidad.
- En la última etapa de gestación (15 días), no se debe realizar limpiezas profundas ni traslados entre pozas.
- El cuy debe ser alimentado adecuadamente en cantidad, calidad y oportunidad.

Parto y Lactación

El parto se presenta al final del periodo de gestación.

- El proceso dura entre 10 a 30 minutos.
- Las crías nacen fisiológicamente maduras, con pelo, ojos abiertos y con capacidad de alimentarse solas.
- La lactación se inicia con el nacimiento de las crías.
- El tiempo de lactación puede ser de 7 a 21 días. Después de este tiempo, las crías estarán en la capacidad de alimentarse por sí solas y la madre se recuperará para el próximo empadre y gestación.

Recría (Destete)

La etapa de recría se realiza después del destete (separación de las crías de su madre). En ese momento se separa a las crías por sexo y se pesa a las crías. En la etapa de recría se consideran los cuyes desde el destete hasta la 4ta semana de edad. Los gazapos (cuyes comprendidos durante la recría) deben recibir una alimentación con altos porcentajes de proteína, durante esta etapa los gazapos alcanzan a triplicar su peso de nacimiento.

Engorde

La etapa del engorde se inicia a partir de la 4ta semana hasta la 9na o 10ma semana de edad. En esta etapa los cuyes están agrupados en lotes uniformes en edad, tamaño y sexo. En esta etapa los cuyes aumentan su peso para su comercialización al finalizar la etapa. En esta etapa también se seleccionan los cuyes que pueden servir como reemplazo de los reproductores.

Alimentación del cuy

La alimentación es uno de los factores de mayor importancia en el proceso productivo, ya que representa del 65 % al 70 % de los costos totales. La alimentación racional consiste en suministrar los alimentos conforme a sus necesidades fisiológicas y de reproducción con la finalidad de obtener el mejor aprovechamiento. El cuy requiere forraje y concentrado, consiste en 80 % de forraje (chala de maíz, avena, cebada, trébol y alfalfa) y 20 % de concentrado (afrecho de trigo, maíz molido, harina de alfalfa y sales minerales). (Ataucusi, 2015)

Importancia del peso

Durante toda la etapa del ciclo productivo del cuy, es importante el seguimiento del peso vivo del cuy, porque este valor nos indica si la alimentación y el ambiente de crianza son óptimos para su desarrollo. Siendo importante el peso de un cuy para su comercialización, ya que si tiene un mayor peso entonces la ganancia también es mayor.

2.2. Visión Artificial

Para definir lo que es la visión artificial o también conocida como *Computer Vision* (CV), se comienza por analizar los conceptos referidos a la visión, los más importantes son:

- "Visión es saber que hay y dónde mediante la vista" (Aristóteles).
- "Visión es recuperar de la información de los sentidos (vista) propiedades válidas del mundo exterior", (Gibson, 2014).
- "Visión es un proceso que produce a partir de las imágenes del mundo exterior una descripción que es útil para el observador y que no tiene información relevante", (Marr, 1982) .

Sucar & Gómez (2011) mencionan que la definición de Marr se acerca a la idea de visión artificial, porque existen tres aspectos importantes mencionados en esta definición que debemos tener presente: (i) visión es un proceso artificial, (ii) la descripción a obtener depende del observador y (iii) es necesario eliminar la información que no sea útil.

Teniendo en cuenta lo que se conoce como visión se puede definir la visión artificial de las siguientes formas:

- Sucar & Gómez (2011) define visión artificial como el estudio de los procesos de reconocer y localizar objetos en el ambiente mediante el procesamiento de imágenes al igual que la visión biológica, con el fin de entenderlos y construir máquinas con capacidades similares.
- González *et al.* (2006) define la visión artificial como una disciplina que engloba todos los procesos y elementos que proporcionan ojos a una máquina y se podría decir que: la visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo.
- Szeliski (2010) indica que la visión artificial intenta describir el mundo que vemos en una o más imágenes y reconstruir sus propiedades como forma, iluminación y distribuciones de color.

El objetivo de la visión artificial es extraer características de una imagen para su descripción e interpretación por la computadora. En visión artificial se busca obtener descripciones útiles para cada tarea a realizar. La Figura 2.7 muestra como una imagen de entrada es procesada para extraerle atributos, obteniendo como salida una descripción de la imagen analizada.

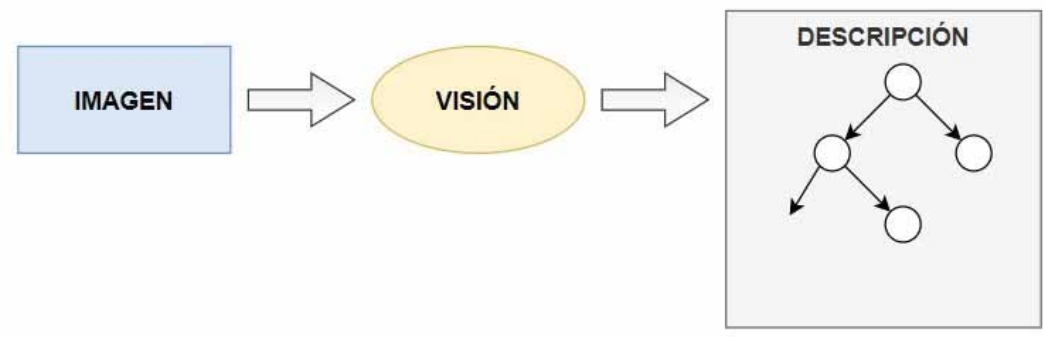


Figura 2.7: Esquema general de visión por computadora
Fuente: Adaptado de Sucar & Gómez (2011)

2.2.1. Formación y representación de la imagen

La formación de la imagen ocurre cuando un sensor (ojo, cámara) registra la radiación (luz) que ha interactuado con ciertos objetos físicos. La imagen obtenida por el sensor se puede ver como una función bidimensional, donde el valor de la función corresponde a la intensidad o brillantez en cada punto de la imagen.

Color

El color es un fenómeno perceptual relacionado con la respuesta humana a diferentes longitudes de onda del espectro visible (400 - 700 nm). Esto se debe a que existen tres tipos de sensores en el ojo que tienen una respuesta relativa diferente de acuerdo a la longitud de onda. Esta combinación de tres señales da la sensación de toda la gama de colores que percibimos. Existen diferentes formas de organizar o codificar los diferentes colores a partir de componentes básicas, lo que se conoce como espacios de color. Los modeles RGB y HSI son un ejemplo de tales modelos de color.

- **Modelo RGB**. Este modelo se basa en los tres sensores humanos, considerando que todos los colores son una combinación de tres colores básicos o primarios: R (rojo), G (verde), B (azul).
- **Modelo HSI**. Se considera que el modelo HSI es que mejor se aproxima a la percepción humana. Este modelo codifica el color en tres componentes: I (intensidad), H (croma) y S (saturación).

2.2.2. Aplicaciones de visión artificial

Sucar & Gómez (2011) menciona las siguientes aplicaciones prácticas de la visión artificial:

- Robótica móvil y vehículos autónomos. Utiliza cámaras y otros tipos de sensores para localizar obstáculos, identificar objetos y personas, etc.
- Manufactura. Se aplica visión para tareas como la localización e identificación de piezas, para el control de calidad entre otras.
- Interpretación de imágenes aéreas y de satélite. Se usa para mejorar las imágenes obtenidas, identificar distintos tipos de cultivo, ayudar en la predicción del clima, etc.
- Análisis e interpretación de imágenes médicas. Se aplica para interpretar diferentes clases de imágenes médicas como rayos-X, tomografía, ultrasonido, resonancia magnética y endoscopía.
- Interpretación de escritura, dibujos y planos. Se utiliza para el reconocimiento de textos, conocido como reconocimiento de caracteres. También se aplica para la interpretación de dibujos y mapas.
- Análisis de imágenes microscópicas. Se usa para interpretar imágenes microscópicas en química, física y biología.
- Análisis de imágenes de astronomía. Se usa para procesar imágenes obtenidas por telescopio, ayudando en localizar e identificar objetos en el espacio.
- Análisis de imágenes para compresión. Basado en la interpretación de imágenes recientemente se desarrollaron técnicas más sofisticadas de compresión.

Szeliski (2010) muestra algunas aplicaciones del mundo real, en las cuales se usa visión artificial:

- Reconocimiento de Caracteres Óptico. Se usa para leer códigos postales escritos a mano y en reconocimiento automático de números de placa vehicular.
- Inspección de máquina. En la inspección de calidad usando visión estéreo para medir la tolerancia de medidas en alas de avión y autopartes.
- Ventas. Reconocimiento de objetos para líneas de pago automatizadas.
- Construcción de modelos 3D (fotogrametría). Construcción de modelos totalmente automatizados basados en imágenes aéreas.
- Medicina. Registro de imágenes preoperatorias e intraoperatorias para realizar estudios de largo plazo de la morfología del cerebro de las personas a medida que envejecen.
- Seguridad automotriz. Detección de obstáculos inesperados como peatones en la calle, en condiciones en las que las técnicas de visión activa como el radar o Lidar no funcionan.
- Movimiento de coincidencia. Fusionar imágenes generadas por computadora (CGI) con imágenes de acción en vivo mediante el seguimiento de puntos característicos en el vídeo de origen para estimar el movimiento de la cámara 3D y la forma del entorno.
- Captura de movimiento. Uso de marcadores retrorreflectantes vistos desde múl-

tiples cámaras u otras técnicas basadas en la visión para capturar el movimiento de actores para la animación por computadora.

- Vigilancia. Monitoreo de intrusos, análisis del tráfico en las carreteras y monitoreo de piscinas para víctimas de ahogamiento.
- Reconocimiento de huellas dactilares y biometría. Para la autenticación de acceso automático y aplicaciones forenses.

2.2.3. Etapas en el proceso de la visión artificial

La visión artificial lleva una enorme cantidad de conceptos relacionados con hardware, software y desarrollos teóricos. González *et al.* (2006) indica los pasos para llevar a cabo una tarea de visión artificial, mostrando el flujo de esta tarea en la Figura 2.8.

- El primer paso es la adquisición de la imagen, donde se necesitan sensores y el hardware para digitalizar la señal producida por el sensor.
- El segundo paso es el preprocesamiento de la imagen, para que el objetivo tenga mejores resultados.
- El tercer paso es la segmentación, su objetivo es dividir la imagen en las partes que la constituyen o los objetos que la forman.
- El cuarto paso es la parametrización, o también conocida como selección de rasgos se dedica a extraer rasgos que produzcan alguna información cuantitativa de interés o rasgos para diferenciar un objeto de otro.
- El último paso es el reconocimiento y la interpretación. El reconocimiento es el proceso que asigna una etiqueta a un objeto basado en la información que proporcionan los descriptores. La interpretación lleva a asignar significado al conjunto de objetos reconocidos.

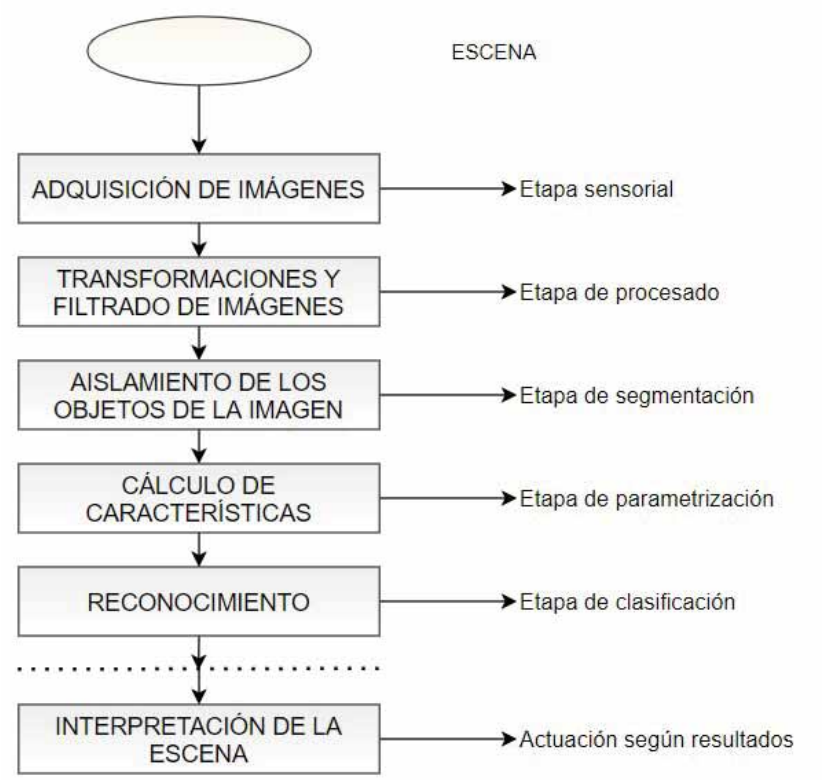


Figura 2.8: Diagrama de bloques de un sistema de visión artificial
Fuente: Adaptado de González *et al.* (2006)

2.2.3.1. Adquisición de imágenes

La adquisición de la imagen es la primera etapa del proceso de visión artificial. En este primer paso se trata de conseguir que la imagen sea la más adecuada posible para continuar con los siguientes pasos.

La correcta adquisición de la imagen supone un paso muy importante para que el proceso de reconocimiento tenga éxito. En esta etapa existen múltiples factores que están involucrados en la captura de la imagen, fundamentalmente por el hardware (cámara, óptica, tarjeta de memoria, ordenador y software), y el entorno y posicionamiento de los elementos (iluminación, fondo, posición correcta de la cámara, ruido eléctrico-óptico externo, etc) (González *et al.*, 2006).

2.2.3.2. Procesamiento de imágenes

La etapa del preprocesamiento de imágenes pretende reparar en la imagen los desperfectos producidos o no eliminados por el hardware: deformación, ruido, poco o mucho contraste o brillo, falta de ecualización apropiada.

2.2.3.3. Segmentación de imágenes

La segmentación consiste en diferenciar los diversos objetos del fondo, dividiendo en zonas disjuntas. Las técnicas que menciona González *et al.* (2006) son categorizadas en: transformaciones morfológicas y técnicas de segmentación (basada en un umbral, basada en bordes y orientada a regiones).

2.2.3.4. Extracción de características

Después del proceso de segmentación, donde se obtienen los bordes de los objetos, el siguiente paso es obtener parámetros que definan las características del objeto como: forma, textura, color, orientación, etc.

Para la selección de estos parámetros o descriptores, se debe tener en cuenta las siguientes características recomendadas por González *et al.* (2006):

- Ser discriminantes, es decir, que diferencien los objetos de una clase respecto a otras lo mejor posible.
- Ser independientes entre sí, para que sus valores no dependan de otros.
- Ser suficientes, tienen que delimitar de forma suficiente la pertenencia a un objeto.

2.2.4. Aumento de datos

Aumento de datos o *data augmentation* se refiere a métodos para construir algoritmos de optimización o muestreo iterativos mediante la introducción de datos no observados o variables latentes (Van Dyk & Meng, 2001).

Data augmentation es un método muy poderoso para lograr que el error de validación disminuya junto con el error de entrenamiento. Los datos aumentados representarán un conjunto de posibles punto de datos, minimizando así la distancia entre el conjunto de entrenamiento y el de validación, así como cualquier conjunto de prueba futuro (Shorten & Khoshgoftaar, 2019). Los métodos más utilizados se basan en manipulaciones básicas a la imagen, aunque existen otros más complejos, su uso depende del problema en el que será aplicado.

2.2.4.1. Aumento de datos basado en manipulaciones básicas de imágenes

- **Voltear (*flipping*)**. Consiste en la inversión en el eje horizontal o vertical de una imagen.
- **Espacio de color (*color space*)**. Como las imágenes están codificadas en un tensor de la dimensión (alto x ancho x canales de color). Esta técnica consiste

en alterar esos canales de color, como aislar un solo canal al agregar matrices de cero en los otros canales de color.

- **Recortar (*cropping*)**. Consiste en recortar partes de la imagen.
- **Rotación**. Consiste en girar la imagen hacia la derecha o la izquierda en un eje entre 1° y 359°.
- **Traslación**. Consiste en desplazar las imágenes hacia la izquierda, derecha, arriba o abajo para evitar el sesgo posicional en los datos. Se usa mas que todo en imágenes donde el conjunto de datos esté centrado.

2.3. Machine Learning

El *Machine Learning* (ML) también llamado aprendizaje automático o aprendizaje de máquinas, es un subcampo de las ciencias de la computación y una rama de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las computadoras aprendan.

Mitchell *et al.* lo define de la siguiente forma: ”se dice que un programa informático aprende de la experiencia E respecto a alguna clase de tareas T y una medida de desempeño P, si su desempeño en las tareas T, como su medida por P, mejora la experiencia E”. (Mitchell *et al.*, 1997)

El *Machine Learning* puede ser definido como el conjunto de técnicas computacionales enfocadas a proporcionar una cierta capacidad cognitiva a una máquina a través del entrenamiento de una red neuronal mediante ejemplos. El objetivo principal es educar a la maquinaria para fomentar su autonomía en la realización de una cierta tarea. Dado un conjunto de entradas y sus correspondientes salidas (valores de entrenamiento), el programa debe ser capaz de entrenar y ajustar la algoritmia interna para poder predecir el resultado de nuevas entradas que no pertenezcan al conjunto de entrenamiento. (Cantero Lorenzo, 2018)

El *Machine Learning* trata de hacer que las computadoras modifiquen o adapten sus acciones(ya sea que estas acciones estén haciendo predicciones) para que sus acciones se vuelvan más precisas, donde la precisión es medida por lo precisas que son las acciones elegidas frente a las correctas. (Marsland, 2015) Teniendo en cuenta que el aprendizaje se define como ”llegar a ser mejor en una tarea a través de la práctica”. (Marsland, 2015)

2.3.1. Tipos de *machine learning*

Marsland divide al *Machine Learning* en los siguientes tipos:

- **Aprendizaje supervisado**. Dado un conjunto de ejemplos con las salidas correctas

(objetivos) es provisto y, basado en el conjunto de entrenamiento, el algoritmo generaliza a responder correctamente todas las entradas posibles. También es llamado aprendizaje por ejemplos.

- **Aprendizaje no supervisado**. Las salidas no son provistas, por eso este algoritmo trata de identificar similitudes entre la entradas de tal forma que las entradas que tienen algo en común son categorizadas juntas. El enfoque estadístico del aprendizaje no supervisado es conocido como estimación densa.
- **Aprendizaje por refuerzo**. Este tipo está en algún lugar entre el aprendizaje supervisado y el no supervisado. El algoritmo es avisado cuando la respuesta es incorrecta, pero no es avisado si la respuesta es correcta. Este trata de explorar y probar diferentes posibilidades hasta que logre conocer como obtener la respuesta correcta. Este aprendizaje a veces es llamado aprendizaje con crítica porque este monitorea las puntuaciones de las respuestas, pero no sugiere mejoras.
- **Aprendizaje evolutivo**. Se basa en la evolución biológica, donde los organismos biológicos se adaptan para mejorar sus tasas de supervivencia y la posibilidad de tener descendencia en su entorno. Los modelos se basan en una idea de aptitud, que corresponde a una puntuación de cuán buena es la solución actual.

2.3.2. Aprendizaje supervisado

El aprendizaje supervisado consiste en realizar predicciones a futuro basado en comportamientos o características ya registradas, permitiendo la búsqueda de patrones de datos históricos relacionados con un campo especial.

Dado un dataset que consiste en un conjunto de datos de entrada junto con los datos objetivos (*target*), que son la respuesta que el algoritmo debe producir. Es usualmente escrito como un conjunto de datos (x_i, t_i), donde las entradas son x_i y las salidas son t_i, y el índice i sugiere que hay muchas piezas de información, indexadas por i desde 1 hasta N. Además que x_i y t_i son vectores, donde cada pieza de información tiene varias características.

Lo que hace que el ML sea mejor que la generalización, es que el algoritmo debe producir salidas sensibles para las entradas que no se encuentran durante el aprendizaje. Además que el resultado del algoritmo debe afrontar el ruido, que son pequeñas inexactitudes en la información que no son importantes en la medición del proceso real. (Marsland, 2015)

Dentro del aprendizaje supervisado destacan los algoritmos de regresión y los de clasificación, los cuáles se diferencian por el tipo de problema que solucionan. En la Figura 2.9 se muestra como son abordados este tipo de problemas por los algoritmos de ML.

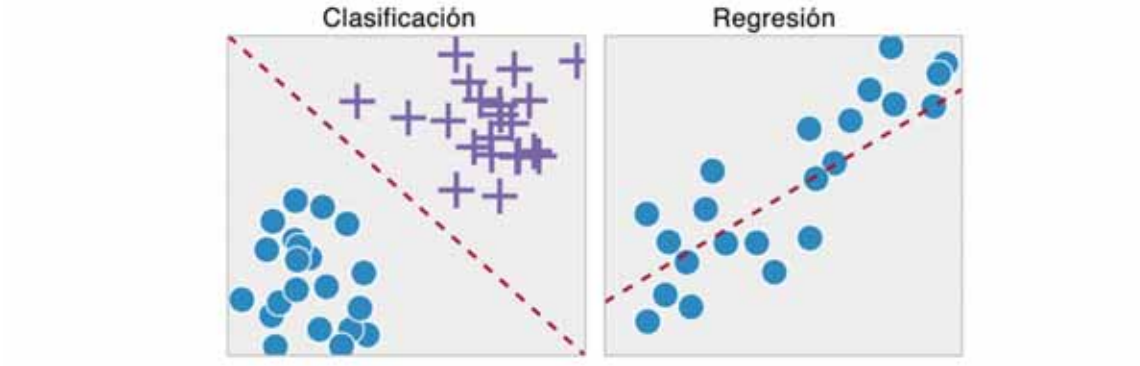


Figura 2.9: Gráfica de clasificación y regresión
Fuente: Buitrago (2020)

Regresión

En los problemas de regresión los valores que se predicen son valores continuos. Se debe identificar cuales son las variables independientes, para establecer una fórmula matemática, que se encargará de asociar el valor de la variable dependiente con las variables independientes. Esta variable dependiente es el valor a predecir. En términos generales, el modelo de regresión intenta explicar una variable dependiente en relación con otras variables independientes (Buitrago, 2020). En la Figura 2.9 para el problema de regresión la solución es una recta que pase lo más cercano a todos los puntos.

Clasificación

En los problemas de clasificación el resultado es una categoría o una clase. Se deben identificar las variables independientes con las cuales se entrenará el modelo. Es una forma de comparar los resultados predichos con los resultados esperados (Buitrago, 2020). En la Figura 2.9 para el problema de clasificación la solución es la agrupación de los puntos que estén mas cerca entre sí para dividirlos en dos grupos.

2.3.3. Proceso del *Machine Learning*

Marsland (2015) muestra el proceso mediante el cual los algoritmos de *Machine Learning* se pueden seleccionar, aplicar y evaluar para un determinado problema.

- **Recolección y preparación de los datos**. Este proceso consiste en recolectar una cantidad de datos con las características que son importantes. Para el aprendizaje supervisado, es necesario recolectar la información objetivo, lo que requiere inversión de tiempo o el involucramiento de expertos en ese campo. Además de considerar los datos que serán necesarios, los algoritmos de aprendizaje automático requieren cantidades de datos significantes, pero al incrementar el conjunto

de datos se incrementa el costo computacional.

- **Selección de características**. Este parte consiste en identificar posibles características que son más útiles para examinar el problema. Esto requiere de un conocimiento del problema y de los datos. Así como la identificación de las características, es necesario que no se requiera de mucho tiempo en su realización, además de evitar el ruido u otra corrupción que afecte al algoritmo.
- **Elección del algoritmo**. Dado un conjunto de datos, la elección del algoritmo es un paso importante para mostrar como se va a solucionar el problema.
- **Selección de parámetros y del modelo**. Para algunos algoritmos es necesario poner algunos parámetros manualmente, o que requieren que se identifiquen los valores apropiados.
- **Entrenamiento**. El entrenamiento es el uso de recursos computacionales para construir el modelo y predecir las salidas de los nuevos datos.
- **Evaluación**. Antes de que un sistema sea desplegado, este necesita ser probado y evaluado en la precisión con los datos que no fueron entrenados. Este puede incluir una comparación con un experto en el campo, y la selección de las métricas apropiadas para esta comparación.

2.4. Deep Learning

El aprendizaje profundo o *Deep Learning* (DL) es la jerarquía de conceptos que permite que la computadora aprenda conceptos complicados construyéndolos a partir de conceptos más simples. Esta jerarquía de conceptos permite a la computadora aprender conceptos complicados construyéndolos a partir de conceptos más simples. De modo más simple, si se dibuja un gráfico mostrando cómo estos conceptos están construidos uno encima del otro, el gráfico es profundo, con muchas capas (Goodfellow *et al.*, 2018).

Deep Learning es un campo dentro del *Machine Learning* donde la inteligencia es inducida en sistemas sin programación explícita usando algoritmos que han sido inspirados en el funcionamiento biológico del cerebro humano (Moolayil, 2019).

En la Figura 2.10 se muestra la relación entre los campos de *Artificial Intelligence* (AI), *Machine Learning* (ML) y *Deep Learning* (DL).

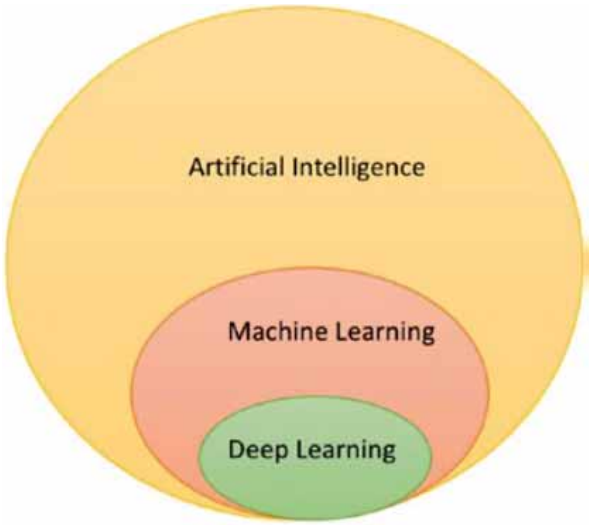


Figura 2.10: Diagrama de Venn que muestra la relación de AI, ML y DL
Fuente: Moolayil (2019)

2.4.1. Red Neuronal Convolucional

Una red neuronal convolucional (CNN) es denominada así por el concepto matemático de "convolución". La convolución es un tipo especializado de operación lineal. Las CNNs son simplemente redes neuronales que utilizan la convolución en lugar de la multiplicación de matrices general en al menos una de sus capas (Goodfellow *et al.*, 2018).

Las CNNs mantienen el concepto de capas, pero cada neurona no recibe conexiones entrantes de todas las neuronas, sino solo de algunas. Esto favorece a que una neurona se especialice en una región de la lista de números de la capa anterior y reduce drásticamente el peso y el número de multiplicaciones necesarias. Lo habitual es que dos neuronas consecutivas de una capa intermedia se especialicen en regiones solapadas de la capa anterior. (Pusiol, 2014)

Las CNNs son la clase de algoritmos de DL usados para casos de uso de visión artificial como clasificar una imagen o vídeo y detectar un objeto dentro de una imagen o incluso una región dentro de una imagen (Moolayil, 2019).

2.4.1.1. Arquitectura de una CNN

Las CNNs son una estructura compuesta de varias fases entrenables, aprendiendo de cada una de las características con diferentes grados de abstracción. La entrada y salida de cada una de estas etapas son conjunto de arreglos llamados mapas de características, la salida de cada mapa de características representa una característica particular extraída de la imagen de entrada.

Una típica arquitectura de CNN para clasificación supervisada está basada en varias etapas seguidas de un clasificador. Como muestra Figura 2.11, primero entra una imagen como una matriz de unos y ceros, luego se le aplican capas de convolución resultando en varias matrices del mismo tamaño; al aplicar las capas de convolución luego se reduce mediante funciones de muestreo que reducen las capas de convolución, este proceso se repite varias veces dependiendo de la arquitectura, finalmente se le aplica una capa totalmente conectado para clasificar la información.

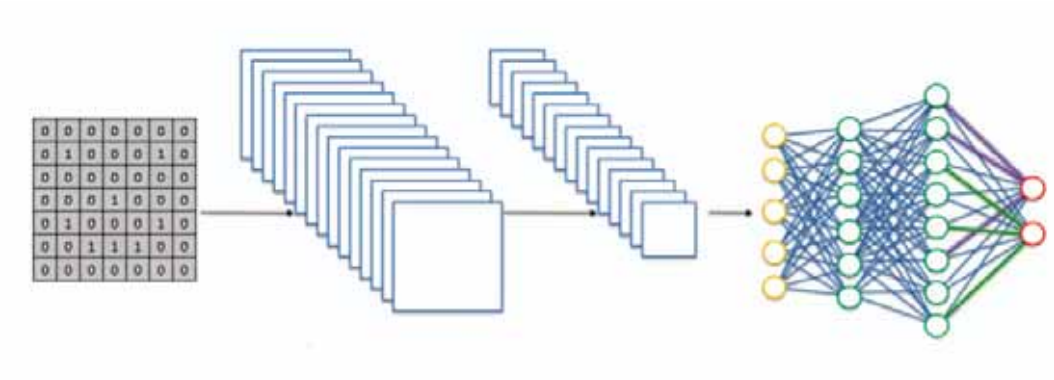


Figura 2.11: Arquitectura de una red neuronal convolucional
Fuente: Frogames (2020)

Goodfellow *et al.* indica que una capa típica de una red convolucional consiste de tres etapas (Figura 2.12 lado izquierdo). En la primera etapa, la capa realiza varias convoluciones en paralelo para producir un conjunto de activaciones lineales. En la segunda etapa, cada activación lineal se ejecuta a través de una función de activación no lineal, esta etapa a veces de denomina etapa de detección. En la tercera etapa, usamos una función de agrupación para modificar aún más la salida de la capa. En cambio la terminología simple (2.12 lado derecho), un red convolucional es vista como un número largo de simples capas, cada paso siguiente es considerado como una capa con su propia camino.

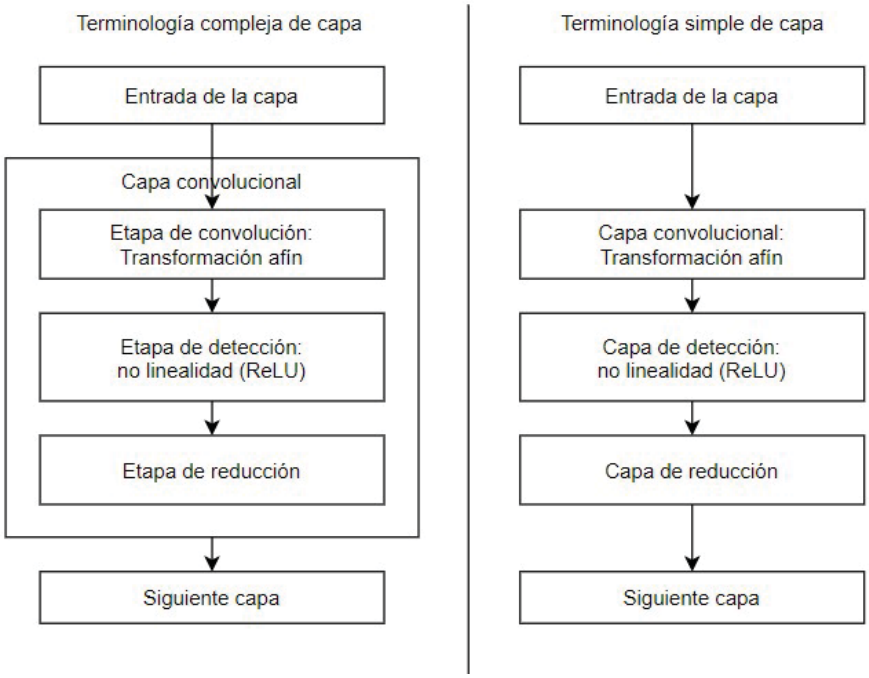


Figura 2.12: Componentes de una típica capa de CNN
Fuente: Adaptado de Goodfellow *et al.* (2018)

2.4.1.2. Capa de convolución

Esta capa esta constituida por neuronas convolucionales, cada una de estas neuronas están asociadas a un filtro (*kernel*). El objetivo de la convolución es extraer características de la imagen de entrada, para lograr este objetivo se hace uso de un filtro que es una matriz bidimensional de menor tamaño al de la imagen de entrada. En esta parte se toman grupos de píxeles cercanos y se opera matemáticamente (producto escalar) con el filtro. El conjunto de filtros aplicado en una convolución es conocido como mapa de características (*feature map*). A medida que se desplaza el filtro sobre la imagen, se va obteniendo nuevas imágenes que resaltan algunas características de la original. En la Figura 2.13 se muestra como la imagen de entrada es transformada después de aplicar una convolución con el filtro.

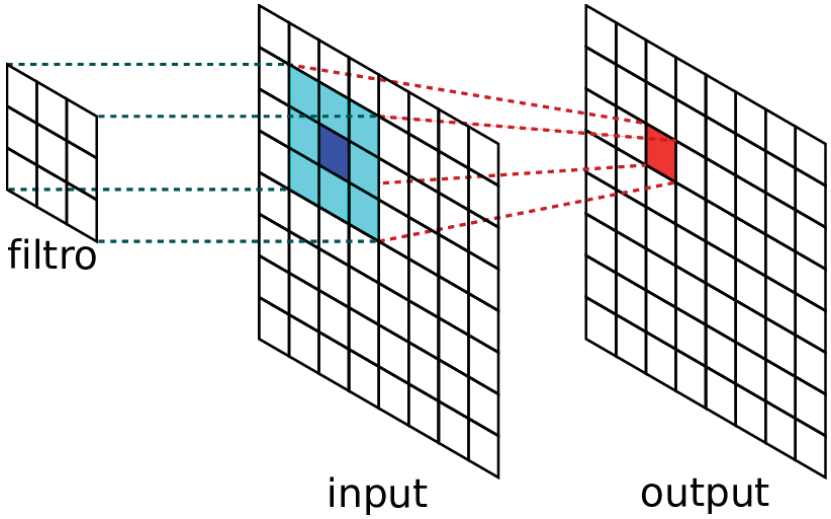


Figura 2.13: Aplicación de una convolución
Fuente: Redolfi (2018)

La convolución aprovecha tres ideas importantes que ayudan a mejorar un sistema de ML, las cuales son:

- Interacciones dispersas. Esto se logra haciendo que el filtro sea menor que la entrada. Por ejemplo, cuando se procesa una imagen, la entrada puede tener miles de millones de píxeles, pero se puede detectar pequeñas características significativas como bordes con filtros que ocupan solo cientos de píxeles. Esto reduce los requerimientos de memoria y mejora su eficiencia estadística.

- Uso compartido de parámetros. Se refiere al uso del mismo parámetro para más de una función en un modelo. En una CNN cada miembro de un filtro es usado en todas las posiciones de la entrada. Esto se muestra que en vez de aprender un conjunto de parámetros para cada ubicación, aprendemos solo un conjunto.

- Representaciones equivariantes. Se refiere a que si las entradas cambian, las salidas también cambian de la misma forma. Esto permite obtener características similares en varias imágenes.

2.4.1.3. Función de Activación

Basado en la funcionalidad de una neurona biológica humana, que consiste en tomar algunas entradas y disparar una salida. La neurona del ML es un marcador de posición para una función matemática, y su único trabajo es proporcionar una salida mediante la aplicación de la función a las entradas proporcionadas. La Figura 2.14 muestra la función de activación $h_{w,b}(x)$ como salida de la neurona dados los pesos w y los bias (sesgo) b para las entradas x. La función utilizada es denominada como función de activación, y existen varios tipos para diferentes problemas.

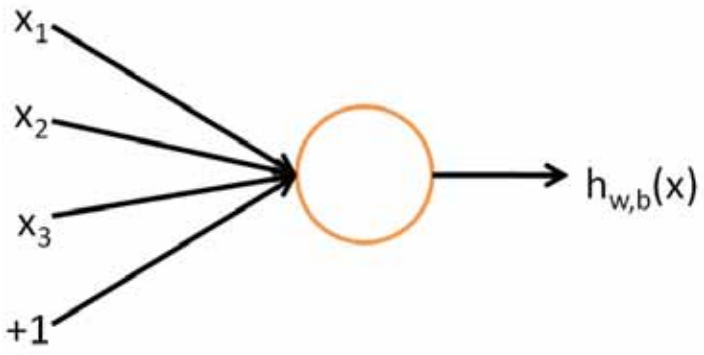


Figura 2.14: Ejemplo del funcionamiento de una neurona artificial
Fuente: Cornieles (2019)

Las funciones de activación se dividen en dos tipos: funciones de activación lineal y funciones de activación no lineales. Las funciones de activación más usadas en las CNNs son las siguientes:

Función de activación lineal

La función de activación lineal también conocida como función identidad donde la salida de la activación es proporcional a la entrada (Ecuación 2.1).

$$f(x) = x \tag{2.1}$$

Esta función tiene algunos problemas con el entrenamiento, ya que no puede usarse el algoritmo de *backpropagation* porque su derivada es una constante que no tiene relación con la entrada. Además de colapsar todas las capas de una red neuronal en una sola capa, al no alterar el resultado después de la salida de la primera capa. Es muy utilizada como capa de salida en problemas de regresión.

Función de activación sigmoide o logística

Esta función toma un valor real como entrada y retorna valores en el rango de 0 a 1. Mientras más grande sea el número (más positivo) la salida será 1, en cambio si el número se aleja de cero (más negativo), la salida sera 0 (Ecuación 2.2).

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Esta función es muy utilizada en modelos donde se tiene que predecir la probabilidad como una salida, siempre que sea en un rango de 0 a 1. Además que la función es diferenciable y provee una gradiente suave. La función sigmoide esta representada por una curva en forma de S (Figura 2.15).

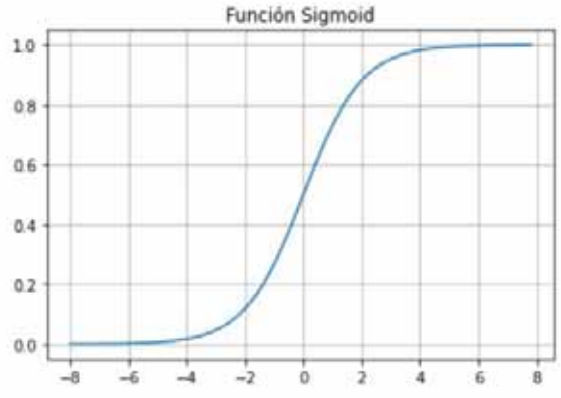


Figura 2.15: Gráfico de la activación sigmoide
Fuente: Elaboración propia

El problema con la función sigmoide son los valores de gradiente que produce ya que estos valores están entre el rango de -3 a 3. Esto implica que para valores mayores a 3 y menores que -3 la función tendrá pequeños gradientes. Cuando el gradiente llegue a cero, la red acabará de aprender y sufrirá el problema de *Vanishing gradient*.

Rectified Linear Unit

La función ReLU da la impresión de una función lineal (Figura 2.16), pero ReLU tiene una función derivada y permite la propagación hacia atrás haciéndolo computacionalmente eficiente. Con ReLU, las neuronas solo se desactivaran cuando la salida de la transformación es menor a 0 (Ecuación 2.3).

$$f(x) = max(0, x) \tag{2.3}$$

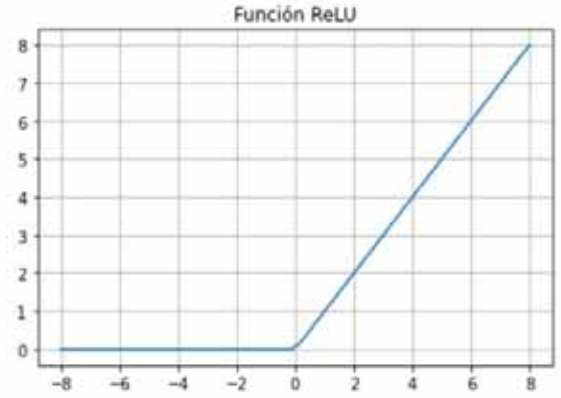


Figura 2.16: Gráfico de la activación ReLU
Fuente: Elaboración propia

Las ventajas de usar ReLU son: desde que solo activa cierto número de neuronas, la función es por lejos mas eficiente computacionalmente comparada con la función

sigmoide y tangencial; ReLU acelera la convergencia del descenso de gradiente hacia el mínimo global de pérdida debido a su propiedad lineal no saturante.

Las limitaciones que enfrenta esta función es llamado *Dying ReLU*, el cuál consiste en que el valor de las gradientes de las neuronas con valor cero no se actualicen durante la retropropagación, esto puede crear neuronas muertas que nunca se activan. Para la solución de este problema existen variantes de esta función como Leaky ReLU, Parametric ReLU y ELU.

Softmax

La función de activación Softmax se describe como una combinación de múltiples funciones sigmoides. Calcula las probabilidades relativas y devuelve una probabilidad para cada clase. Se usa más comúnmente como una función de activación para la última capa de la red neuronal en el caso de clasificación de clases múltiples. Su fórmula es la siguiente Ecuación 2.4.

$$softmax(Z)_j = \frac{e^{Z_j}}{\sum_{k=1}^{K} e^{Z_k}} \tag{2.4}$$

2.4.1.4. Capa de Reducción o Pooling

Una función de agrupación o pooling reemplaza la salida de la red en una ubicación determinada con una estadística resumida de las salidas cercanas (Goodfellow *et al.*, 2018). Es la encargada de reducir paulatinamente el consumo espacial de la representación de una imagen, así mismo el número de parámetros y el coste computacional de la red. Una de las técnicas que se suele aplicar es la de *max-pooling*, que como se muestra la Figura 2.17 consiste en seleccionar el valor mas alto de una región de valores de las neuronas.

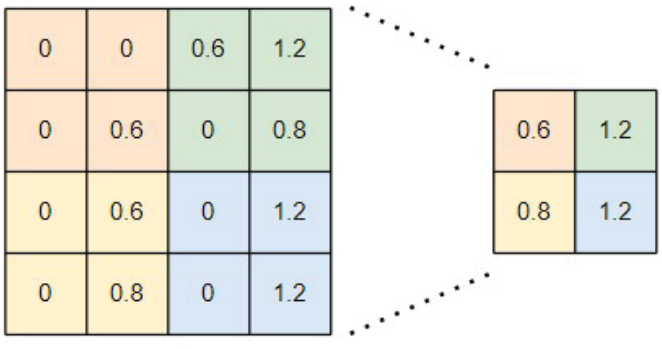


Figura 2.17: Aplicación de *max-pooling*
Fuente: Elaboración propia

2.4.1.5. Capa totalmente conectada

Después de aplanar el resultado de las capas convolucionales, estas pasan a la capa totalmente conectada. Las neuronas en una capa completamente conectada tienen conexiones completas con todas las activaciones en la capa anterior, como se muestra en la Figura 2.18, esta capa es similar a una *Articial Neural Network* (ANN) convencional. Por lo tanto, sus activaciones pueden calcularse con una multiplicación matricial seguida de un desplazamiento de polarización.

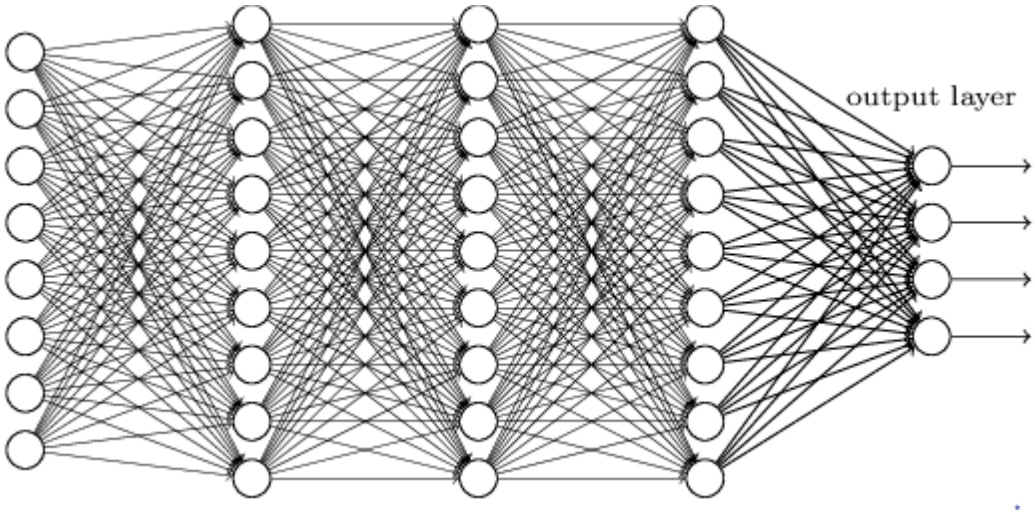


Figura 2.18: Capa totalmente conectada
Fuente: Nielsen (2015)

2.4.1.6. Entrenamiento de una CNN

El proceso de entrenamiento de una CNN se realiza utilizando el algoritmo de *back-propagation*, que calcula una función objetivo para minimizar el error mediante la retro propagación a las capas anteriores a la salida ajustando los pesos de las conexiones de las neuronas.

El algoritmo de entrenamiento trabaja de la siguiente forma:

- Se adquieren los datos de entrada.
- Propaga dichos datos hasta la capa de salida con pesos iniciales definidos o aleatorios.
- Calcula el error en la capa de salida.
- Propaga hacia atrás el error en las capas ocultas.
- Cambia los pesos de las conexiones.

2.4.2. Transfer learning

El aprendizaje por transferencia o *transfer learning* es un problema de experimentación en el *Machine Learning* que se centra en el almacenamiento del conocimiento adquirido en la resolución de un problema y su uso en un tema diferente pero relacionado (Barznji, 2020).

En *transfer learning* primero se entrena una red base en un conjunto de datos y una tarea base, y luego se reutilizan las características aprendidas, y se transfiere a una segunda red objetivo para ser entrenados en un conjunto de datos con una tarea objetivo. Este proceso tiende a funcionar si las características son generales, es decir, adecuadas para las tareas básicas y objetivo, en lugar de específicas para la tarea básica. Cuando el conjunto de datos de destino es significativamente más pequeño que el conjunto de datos base, el aprendizaje por transferencia puede ser una herramienta poderosa para permitir el entrenamiento de una red de destino grande sin sobreajuste (Yosinski *et al.*, 2014).

Transfer Learning es la mejora del aprendizaje en una nueva tarea mediante la transferencia de conocimientos de una tarea relacionada que ya se ha aprendido. Los métodos de transferencia tienden a depender en gran medida de los algoritmos de *Machine Learning* que se utilizan para aprender las tareas y, a menudo, pueden considerarse extensiones de esos algoritmos (Torrey & Shavlik, 2010).

En la Figura 2.19 se muestra como se puede visualizar el uso de una red pre-entrenada en un problema genérico de clasificación para aplicarlos en otra tarea de la segunda red, usando los parámetros transferidos de la primera red a la segunda.

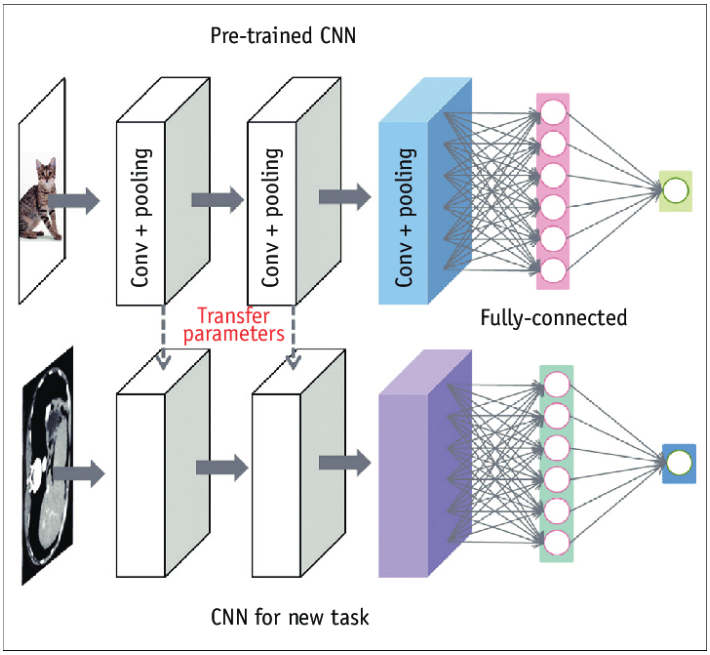


Figura 2.19: Transfer learning
Fuente: Do *et al.* (2020)

El objetivo del *transfer learning* es mejorar el aprendizaje en la tarea de destino aprovechando el conocimiento de la tarea de origen. Como se muestra en la Figura 2.20 existen tres medidas comunes mediante las cuales la transferencia puede mejorar el aprendizaje (Torrey & Shavlik, 2010).

- Primero, es el rendimiento inicial (*higher start*) alcanzable en la tarea de destino utilizando solo el conocimiento transferido, antes de que se realice cualquier aprendizaje adicional, en comparación con el desempeño inicial de un agente ignorante.
- Segundo, se encuentra la cantidad de tiempo que se necesita para aprender completamente la tarea objetivo (*higher slope*) dado el conocimiento transferido en comparación con la cantidad de tiempo para aprenderlo desde cero.
- Tercero, se encuentra el nivel de desempeño inalcanzable en la tarea objetivo (*higher asymptote*) en comparación con el nivel final sin transferencia.

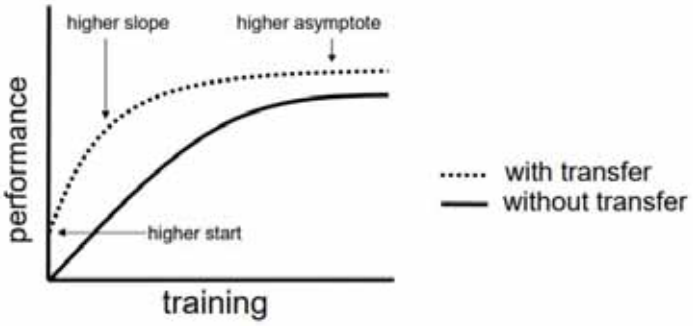


Figura 2.20: Formas cómo *trasnfer learning* puede mejorar el aprendizaje
Fuente: Torrey & Shavlik (2010)

2.5. Detección de Objetos y Segmentación de Imágenes

La detección de objetos es una técnica de visión artificial que permite identificar y localizar objetos en una imagen o vídeo. Con este tipo de identificación y localización, la detección de objetos puede ser usado para contar objetos en una escena y determinar y seguir su ubicación precisa, todo mientras se etiqueta con precisión. Específicamente, como se muestra en la Figura 2.21, la detección de objetos dibuja cuadros delimitadores alrededor de los objetos detectados, lo cuál permite localizar donde los objetos están en una escena dada.

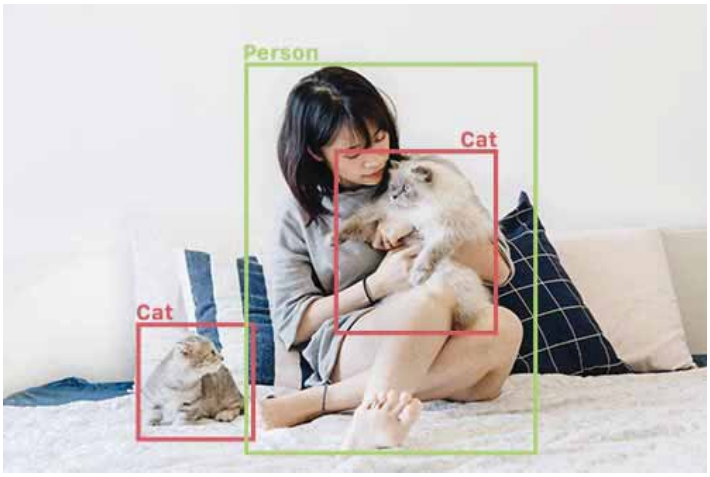


Figura 2.21: Ejemplo de detección de objetos
Fuente: AI (2021)

La definición del problema de la detección de objetos es determinar dónde se encuentran los objetos en una imagen determinada (localización de objetos) y a qué categoría pertenece cada objeto (clasificación de objetos) (Zhao *et al.*, 2019). El proceso de los modelos tradicionales de detección de objetos se divide principalmente en tres etapas: selección de región informativa, extracción de características y clasificación.

- **Selección de región informativa**. Como diferentes objetos pueden aparecer en cualquier posición de la imagen y tener diferentes proporciones o tamaños, es una elección natural escanear toda la imagen con una ventana deslizante de múltiples escalas. Es una operación computacionalmente costosa y produce demasiadas ventanas redundantes.
- **Extracción de características**. Para reconocer los objetos, se necesita características visuales que puedan proporcionar una representación semántica y robusta.
- **Clasificación**. Es necesario un clasificador para distinguir el objeto de destino de todas las demás categorías y hacer que las representaciones sean más jerárquicas, semánticas e informativas para el reconocimiento visual.

Gracias a la emergencia de los modelos de DL, se puedo conseguir desarrollar estas etapas con un mejor desempeño con la introducción de Regiones con características

CNN o *Regions with CNN features* (R-CNN). Estas arquitecturas mas profundas tenían la capacidad de aprender características más complejas.

Desde la propuesta de R-CNN, se han sugerido una gran cantidad de modelos generados, incluido Fast R-CNN que optimiza conjuntamente las tareas de clasificación y regresión del cuadro delimitador, Faster R-CNN que toma una subred adicional para generar propuestas de región y YOLO que logra la detección de objetos mediante una regresión de cuadrícula fija. Todos estos modelos aportan grados de mejoras en el rendimiento de la detección respecto al R-CNN primario y hacen que la detección de objetos sea precisa.

2.5.1. Tipos de métodos de detección

Zhao *et al.* clasifica la detección de objetos en dos tipos. Uno sigue el canal de detección tradicional, generando propuestas de región al principio y luego clasificando cada propuesta en diferentes categorías de objetos. El otro considera la detección de objetos como un problema de regresión o clasificación, adoptando un marco unificado para lograr resultados finales (categorías y ubicaciones) directamente. Los métodos basados en propuesta de región incluyen principalmente R-CNN, SPP-net, Fast R-CNN, Faster R-CNN, R-FCN, FPN y Mask R-CNN. Los métodos basados en regresión/clasificación incluyen principalmente MultiBox, AttentionNet, G-CNN, YOLO, SSD, YOLOv2, SSD La y DSOD.

2.5.2. Métodos basados en propuestas de regiones

Los métodos basados en propuestas de región, constan de un proceso de dos pasos, que coincide con el mecanismo de atención del cerebro humano, lo que proporciona una exploración aproximada de todos los escenarios y luego se centra en las regiones de interés.

2.5.2.1. R-CNN

En la detección de objetos es importante mejorar la calidad de los cuadros delimitadores y adoptar una arquitectura profunda para extraer características de alto nivel. Para resolver estos problemas Girshick *et al.* propuso R-CNN (Girshick *et al.*, 2014) y obtuvo un promedio de precisión media (mAP) de 53.3 % con una mejora de más de 30 % al mejor resultado en PASCAL VOC 2012.

Como muestra la Figura 2.22 este modelo se divide en tres etapas: el proceso inicia con la generación de propuestas de región, usando la búsqueda selectiva para generar 2000 propuestas de región para cada imagen, luego deforma y recorta cada propuesta de región para extraer un vector de características de 4096 dimensiones usando una CNN,

finalmente usa el algoritmo de *Support Vector Machines* (SVM) pre-entrenado para clasificar la presencia de un objeto dentro de la región. Además, el algoritmo también predice cuatro valores correspondientes al cuadro delimitador.

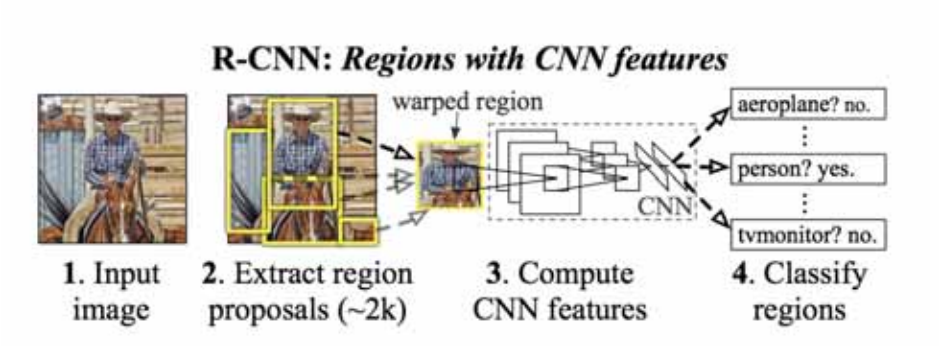


Figura 2.22: Diagrama de flujo de R-CNN
Fuente: Girshick *et al.* (2014)

A pesar de sus mejoras sobre métodos tradicionales y la importancia de llevar CNN a la detección de objetos, tiene las siguientes desventajas.

- Debido a la existencia de capas *Fully Connected* (FC), la CNN requiere de un tamaño fijo de la imagen de entrada (227x227), que conduce al cálculo de la CNN completa para cada región, lo que lleva mucho tiempo en el periodo de prueba.
- Es un proceso de varias etapas, al principio usa una CNN sobre propuestas de objetos bien ajustados, luego el clasificador *softmax* es reemplazado por SVM para que adapte las funciones de la CNN y finalmente se entrenan los regresores de los cuadros delimitadores.
- El entrenamiento es costoso en espacio y tiempo. Las características son extraídos de diferentes propuestas de región y almacenadas en el disco. Además de tomar mucho tiempo en entrenar un conjunto relativamente pequeño para redes profundas.

2.5.2.2. Fast R-CNN

El enfoque de Fast R-CNN (Girshick, 2015) es similar al algoritmo R-CNN. Pero, en lugar de enviar las propuestas de la región a la CNN, enviamos la imagen de entrada a la CNN para generar un mapa de características convolucional. A partir del mapa de características convolucional, identificamos la región de las propuestas y las deformamos en un tamaño cuadrado y, mediante el uso de una capa de agrupación de *Region of Interest* (RoI), las reformamos en un tamaño fijo para que se pueda alimentar a una capa totalmente convolucional. A partir del vector de características RoI, usamos una capa softmax para predecir la clase de la región propuesta y los valores del cuadro delimitador. En la Figura 2.23 se muestra la arquitectura de Fast R-CNN para la detección de objetos.

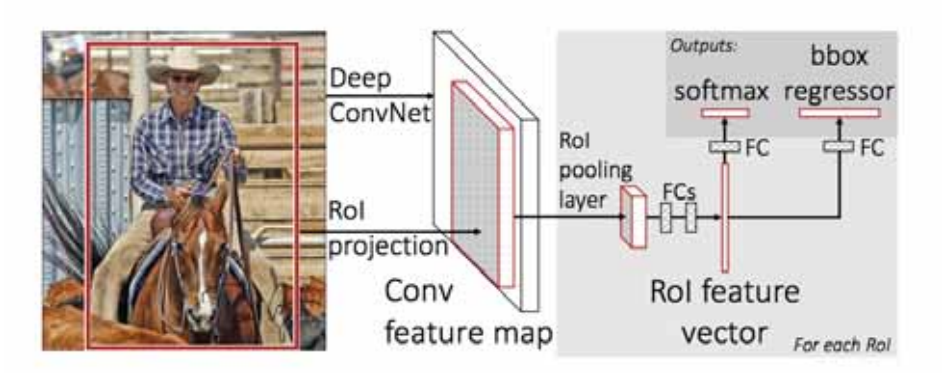


Figura 2.23: Arquitectura de Fast R-CNN
Fuente: Girshick (2015)

2.5.2.3. Faster R-CNN

Los algoritmos anteriores (R-CNN y Fast R-CNN) utilizan la búsqueda selectiva para encontrar las propuestas de región. La búsqueda selectiva es un proceso lento que afecta el rendimiento de la red. Para resolver este problema, Ren *et al.* introdujo la *Region Proposal Network* (RPN) (Ren *et al.*, 2015), la cual permite que la red aprenda las propuestas de región. La Figura 2.24 muestra la adición de la RPN dentro de la arquitectura de Faster R-CNN

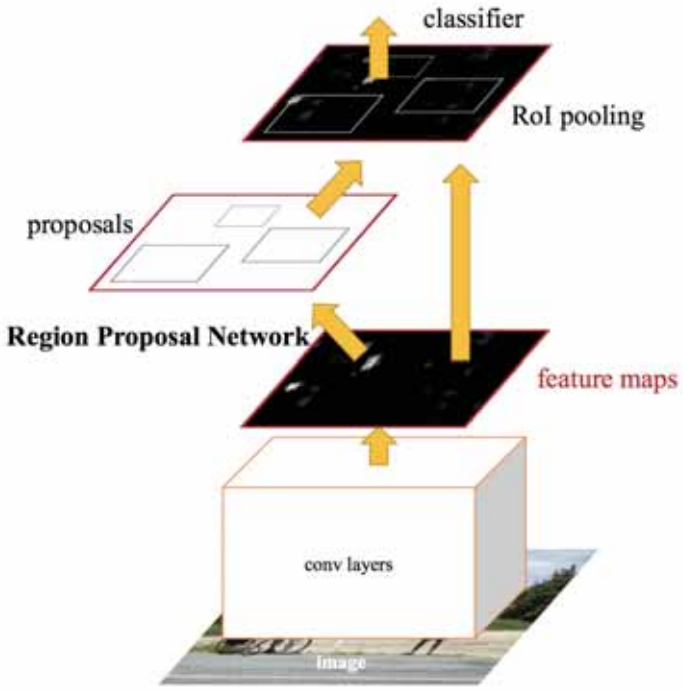


Figura 2.24: Arquitectura de Faster R-CNN
Fuente: Ren *et al.* (2015)

La RPN se logra con una red totalmente convolucional, que tiene la capacidad de predecir los límites y las puntuaciones de los objetos en cada posición simultáneamente.

La red se desliza sobre el mapa de características de convolución y se conecta completamente a una ventana espacial *nxn*. Se obtiene un vector de baja dimensión en cada ventana deslizante y se alimenta a dos capas FC para la clasificación del objeto y la regresión del cuadro delimitador. Para aumentar la no linealidad, ReLU se aplica a la salida d *nxn* de la capa de convolución.

Las regresiones hacia los cuadros delimitadores verdaderos se logra comparando propuestas en relación con cuadros de referencia (anclas).

Con la propuesta de Faster R-CNN, las arquitecturas CNN basadas en propuesta de regiones para la detección de objetos se pueden entrenar de manera integral. También logra una velocidad de 5 FPS (cuadros por segundo) en una *Graphical Proccesor Unit* (GPU) con precisión de detección de objetos. Sin embargo, el algoritmo de entrenamiento consume mucho tiempo y RPN produce regiones similares a objetos (incluidos fondos) en lugar de instancias de objetos y no es experto en el tratamiento de objetos con escalas o formas extremas.

2.5.2.4. *Feature Pyramid Network* (FPN)

Se trata de pirámides de características construidas sobre pirámides de imágenes que se han aplicado en muchos sistemas de detección de objetos para mejorar la invarianza de escala. FPN (Lin *et al.*, 2017) tiene una arquitectura con una vía de abajo hacia arriba, una vía de arriba hacia abajo y varias conexiones laterales para combinar características de baja resolución y semánticamente fuertes con características de alta resolución y semánticamente débiles, así como se muestra en la Figura 2.25.

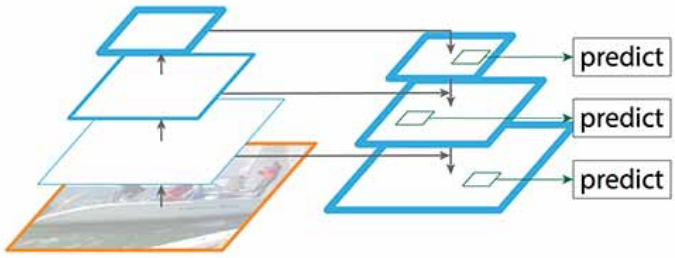


Figura 2.25: FPN
Fuente: Lin *et al.* (2017)

La ruta ascendente, que es la red troncal básica de ConvNet, produce una jerarquía de características correspondientes con un paso de 2. Las capas que poseen el mismo tamaño de mapas de salidas se agrupan en la misma etapa de red y la salida de la última capa de cada etapa se elige como el conjunto de referencia de mapas de características para construir la siguiente ruta de arriba hacia abajo.

Para construir la ruta de arriba hacia abajo, los mapas de características de las etapas superiores de la red se muestrean primero y luego se mejoran con los del mismo tamaño espacial de la ruta de abajo hacia arriba a través de conexiones laterales. Una capa de convolución de $1x1$ se agrega al mapa muestreado para reducir las dimensiones del canal y la fusión se logra mediante la adición de elementos. Finalmente, una convolución $3x3$ se agrega a cada mapa combinado para reducir el efecto de *aliasing* del *upsampling* y se genera el mapa de características final. El proceso se repite hasta que se genera el mapa de resolución más fina.

Como la pirámide de características puede extraer una rica semántica de todos los niveles y entrenarse de principio a fin con todas las escalas, se puede obtener una representación de vanguardia sin sacrificar la velocidad y memoria. Mientras tanto, FPN es independiente de las arquitecturas CNN troncales, se puede aplicar a diferentes etapas de detección de objetos.

2.5.2.5. Mask R-CNN

La segmentación de instancias es una tarea que requiere detectar todos los objetos de una imagen y segmentar cada instancia (segmentación semántica). Estas tareas generalmente se consideran dos procesos independientes, y un esquema multitarea crearía un borde falso y exhibiría errores sistemáticos en instancia superpuestas. Para resolver este problema, paralelamente a las ramas existentes en Faster R-CNN, Mask R-CNN (He *et al.*, 2017) agrega una rama para predecir máscaras de segmentación píxel a píxel, como se evidencia en la Figura 2.26.

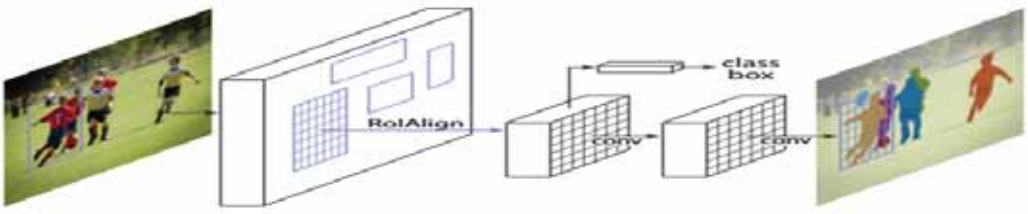


Figura 2.26: Arquitectura de Mask R-CNN
Fuente: He *et al.* (2017)

A diferencia de las otras ramas, que se colapsan en vectores de salida cortos por capas totalmente convolucionales, la rama de segmentación de máscara codifica una máscara de mxm para mantener el diseño espacial explícito del objeto. Además de las dos pérdidas para la clasificación y la regresión del cuadro delimitador, se define una pérdida adicional para la rama de la segmentación de la máscara para alcanzar una pérdida multitarea. Esta pérdida solo se asocia con la clase de verdad y se base en la rama de clasificación para predecir la categoría.

Debido a que la agrupación de RoI, la operación principal en Faster R-CNN, realiza una cuantificación espacial aproximada para la extracción de características, se introduce

una desalineación entre la RoI y las funciones. Afecta poco a la clasificación debido a su robustez a las pequeñas traducciones. Sin embargo, tiene un gran efecto negativo en la predicción de la máscara píxel a píxel. Para resolver este problema, Mask R-CNN adopta una capa simple y sin cuantificación, llamada *RoI Align*, para preservar fielmente la correspondencia espacial explícita por píxel. RoI Align logra reemplazando la cuantificación severa de la agrupación RoI con interpolación bilineal, calculando los valores exactos de las características de entrada en cuatro ubicaciones muestreadas regularmente en cada contenedor de RoI. A pesar de su simplicidad, este cambio aparentemente menor mejora enormemente la precisión de la máscara, especialmente bajo estrictas métricas de localización.

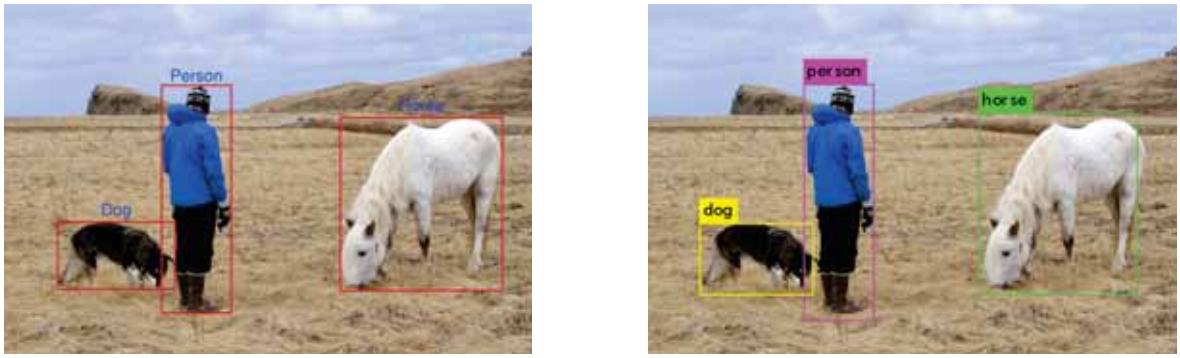
Dado la arquitectura de Faster R-CNN, la rama de máscara solo agrega una pequeña carga computacional y su cooperación con otras tareas proporciona información complementaria para la detección de objetos. Como resultado, Mask R-CNN es fácil de implementar con resultados prometedores de segmentación de instancias y detección de objetos.

2.5.3. Métricas para la detección de objetos

Cada imagen en un problema de detección de objetos podría tener diferentes objetos de diferentes clases, por eso es necesario evaluar tanto la clasificación como la localización de un modelo. Como la métrica utilizada en los problemas de clasificación no se puede aplicar directamente, entonces se usa el *Mean Average Precision (mAP)*. Para entenderlo completamente se necesita abarcar algunos conceptos.

Verdad fundamental

La verdad fundamental o *ground truth* incluye la imagen, los clases de los objetos en ella y los cuadros delimitadores verdaderos de cada uno de los objetos. En la Figura 2.27a se muestra la representación visual de la verdad fundamental mostrando las etiquetas de persona, perro y caballo junto con las cuadros delimitadores de cada objeto. Para el entrenamiento y la validación, todas las imágenes deben ser anotadas con esta información. En cambio en la Figura 2.27b se muestra el resultado de un modelo de predicción, para evaluar su precisión.



(a) *ground truth* (b) Resultado de un modelo

Figura 2.27: Verdad fundamental vs resultado de un modelo
Fuente: Shah (2018)

***Intersection over Union* (IoU)**

IoU es una relación entre la intersección y la unión de los cuadros predecidos con los cuadros de la verdad fundamental. Esta estadística también se conoce como Índice Jaccard y fue publicado por primera vez por Paul Jaccard a principios del siglo XX. Esta métrica evalúa la superposición entre dos cuadros delimitadores (Ecuación 2.5), para lo cual requiere un cuadro delimitador de la verdad fundamental (B_{gt}) y un cuadro delimitador predecido (B_p).

$$IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{2.5}$$

La Figura 2.28 muestra gráficamente el área de superposición de los cuadros delimitadores que son evaluados.

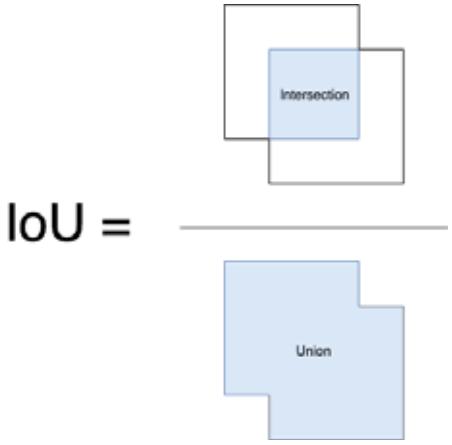


Figura 2.28: IoU
Fuente: Shah (2018)

Precisión y *Recall*

Para el cálculo de la precisión y el *recall* es necesario identificar los verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Para obtener los verdaderos positivos (TP) y falsos positivos (FP), se usa IoU con un umbral de confianza. El umbral más comúnmente usado es 0.5, es decir $IoU > 0{,}5$, se considera un verdadero positivo, de lo contrario es un falso positivo. Para la calcular el *recall* se necesita el recuento de los falsos negativos (FN), los cuales son los objetos que nuestro modelo ha perdido. Las Ecuación 2.6 muestra el cálculo de la precisión, el cuál es el porcentaje de predicciones correctas frente a todas las predicciones realizadas y la Ecuación 2.7 muestra el valor de *recall*, que es el ratio de el número de verdaderos positivos a el número total de objetos.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

2.5.3.1. *Average Precision* (AP) y *Mean Average Precision* (mAP)

La definición general de AP es encontrar el área que está por debajo de la curva de precisión/*recall* (Figura 2.29) y el mAP es el promedio de AP. En algunos casos AP es calculado para cada clase y el promedio es mAP, pero en contexto como la evaluación de la competición de COCO (Lin *et al.*, 2014) no hay diferencia entre estos dos conceptos.

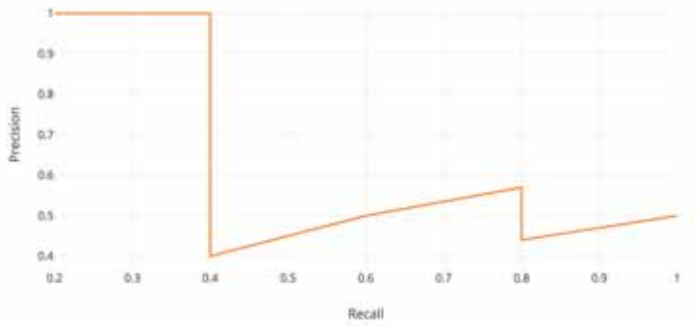


Figura 2.29: Ejemplo de curva de *precision/recall*
Fuente: Hui (2018)

La precisión y el *recall* están siempre entre 0 y 1. Por eso AP, siempre está dentro de este valor 0 y 1 (Ecuación 2.8).

$$AP = \int_0^b p(r)\, dr \tag{2.8}$$

Antes de calcular el AP se suaviza el patrón de zigzag en la curva, reemplazando con el valor de precisión máximo a la derecha de ese nivel de *recall*. En la Figura 2.30 se ve como la linea naranja se transforma en la líneas verdes, y la curva disminuye monótonamente en lugar de un patrón de zigzag.

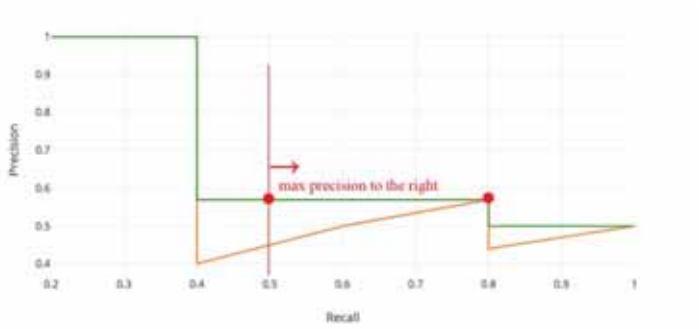
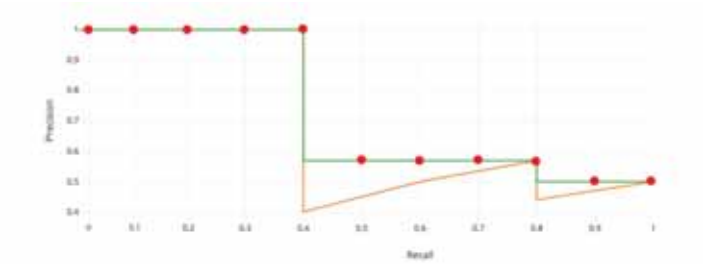
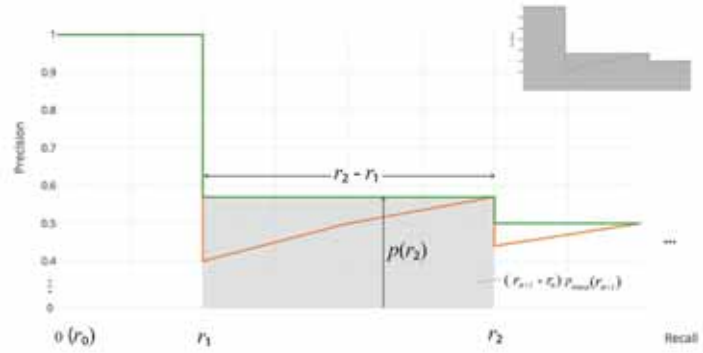


Figura 2.30: Suavizado de la curva *precision/recall*
Fuente: Hui (2018)

Para el cálculo de AP se realizada de dos formas: AP Interpolado y AP (*Area Under Curve* (AUC)). El AP interpolado calcula un promedio en diferentes puntos espaciado dentro de un rango, mientras que que AP AUC lo hace usando el área debajo de la curva. En la Figura 2.31a se muestra el cálculo del AP con 11 puntos, mientras que la Figura 2.31b se muestra el cálculo del AP usando el AUC.



(a) Ejemplo de AP interpolado en 11 puntos



(b) Ejemplo de AP AUC

Figura 2.31: Cálculo de *Average Precision* (AP)
Fuente: Hui (2018)

COCO mAP

Los trabajos de investigación más recientes tienden a dar resultados para el conjunto de dataset COCO. En COCO mAP, se utiliza una definición de AP interpolado con 101 puntos de cálculo. Para COCO, AP es el promedio sobre múltiples IoU, para la competencia se usa el AP de más de 10 niveles de IoU en 80 categorías (AP@[.50:.05:.95] comienza en 0.5 a 0.95 con pasos de 0.05). En la tabla 2.2 algunas métricas recopiladas para el conjunto de datos de COCO.

Average Precision	
AP	% AP con IoU=.50:.05:.95 (métrica principal de la competición)
$AP^{IoU=,50}$	% AP con IoU=.50 (métrica para PASCAL VOC)
$AP^{IoU=,75}$	% AP con IoU=.75 (métrica estricta)

Tabla 2.2: Métricas para COCO
Fuente: `https://cocodataset.org/#detection-eval`

2.5.4. *Non-Maximun Supression*

Denominada también Supresión No Máxima, este algoritmo busca reducir propuestas que son similares luego de la detección de propuestas (Figura 2.32). La entrada de este

algoritmo son una lista de cuadros de propuesta (B), sus puntuaciones de confianza (S) y el umbral de superposición (N); para dar como resultado una lista de propuestas filtradas (D). El algoritmo es como sigue:

1. Selecciona la propuesta con la puntuación de confianza más alta, la elimina de B y la agrega a la lista de propuestas final D (inicialmente D está vacía).
2. Ahora compara esa propuesta con todas las propuestas comparando el IoU. Si el IoU es mayor que el umbral N, elimine esa propuesta de B.
3. De nuevo, tome la propuesta con mayor confianza de las propuestas restantes en B y la elimina de B agregándola a D.
4. Calcule nuevamente el IoU de esta propuesta con todas las propuestas en B y elimine aquellas que tengan un IoU mayor que el umbral.
5. Este proceso se repite hasta que no queden más propuestas en B.

Figura 2.32: Algoritmo de *Nom-Maximum Supression*
Fuente: K (2019)

2.5.5. Herramientas

2.5.5.1. Labelbox

Labelbox es una plataforma de datos de entrenamiento construida para mejorar el ciclo de iteración de datos de entrenamiento. Está diseñado en torno a tres pilares básicos: la capacidad de anotar datos, diagnosticar el rendimiento del modelo y priorizar en función de sus resultados.

Labelbox es una infraestructura centrada en datos para equipos modernos de AI. Labelbox permite a los equipos de AI crear rápidamente datos de entrenamiento con una supervisión humana mínima y mejorar el rendimiento del modelo dentro de una plataforma unificada.

Labelbox permite crear clases de objetos (ontologías), las cuáles pueden ser anotadas para modelos de detección de objetos o modelos de segmentación de instancias. En la Figura 2.33 se muestra el espacio de trabajo de Labelbox para diferentes ontologías.

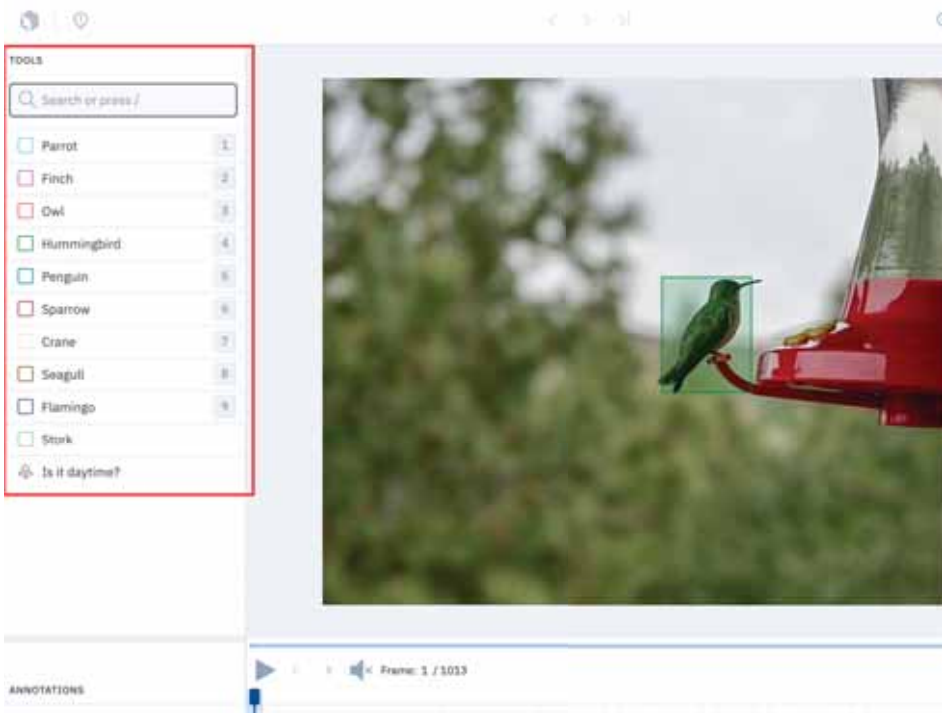


Figura 2.33: Espacio de trabajo de Labelbox
Fuente: Labelbox (2022)

Labelbox ofrece diferentes herramientas para anotar imágenes, entre las cuales existen: cuadro delimitador, máscara de segmentación, herramienta lápiz, herramienta borrador, herramienta de relleno, polígono, polilínea y punto; estas herramientas nos permiten anotar las instancias.

Además, Labelbox permite exportar los resultados de la anotación en un archivo de formato JSON, para luego descargar las máscaras un SDK que provee la plataforma.

2.5.5.2. Tensorflow

Tensorflow es una biblioteca de código abierto para ML, desarrollado por Google para satisfacer necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usado por los humanos.

Tensorflow puede correr en CPUs y GPUs (con extensiones opcionales de CUDA). Tensorflow está disponible para Windows, Linux, macOs, y plataformas móviles que incluyen Android y iOS. Tensorflow proporciona una API de Python, así como de C++,

Haskell, Java, Go y Rust; y existen bibliotecas de terceros para C#, Julia, R, Scala y Ocami.

El nombre de TensorFlow deriva de las operaciones que las redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como tensores.

2.5.5.3. Keras

Keras es una *Application Programming Interface* (API) de redes neuronales de código abierto escrita en Python. Está diseñada para posibilitar la experimentación en menos tiempo con redes de ML. Sus fortalezas se centran en ser amigables para el usuario, ser modular y extensible. Keras se conceptualiza como una API diseñada para seres humanos, siguiendo las mejores prácticas para reducir la carga cognitiva: ofrecer una API consistente y simple, minimizar la cantidad de acciones del usuario requeridas para caso de uso comunes y proporcionar mensajes de error claros y accionables. Además de contar con una extensa documentación y guía para desarrolladores.

2.5.5.4. CUDA Toolkit

CUDA Toolkit es un entorno de desarrollo proporcionado por NVIDIA para crear aplicaciones aceleradas por GPU de alto rendimiento. Con CUDA Toolkit se puede desarrollar, optimizar e implementar aplicaciones en sistemas integrados acelerados por GPU.

CUDA también se referencia como una plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo.

2.5.5.5. Flask

Flask es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código. Está basdo en la especificación WSGI de Werkzeug y el motor de templates Jinja2.

2.5.5.6. Expo

Expo es un framework y una plataforma para aplicaciones universales React. Es un conjunto de herramientas y servicios construidos alrededor de React Native y plataformas nativas que permiten desarrollar, construir, implementar e iterar rápidamente en iOS, Android y aplicaciones web desde la misma base de código JavaScript/TypeScript.

Capítulo 3

Desarrollo del proyecto

Para la aplicación de una técnica de visión artificial es necesario la construcción de un dataset (imágenes, máscaras y pesos) y la adaptación de una arquitectura CNN para la estimación del peso del cuy basado en imagen.

Este capítulo se basa en los pasos de la metodología propuesta (Figura 1.4), comprende la recolección de datos, el procesamiento de imágenes, la construcción del dataset y la adaptación de una CNN aplicada en la estimación del peso del cuy.

3.1. Recolección de datos

La recolección de datos comprende la captura de imágenes de cuyes y la obtención de sus pesos usando una balanza digital. Para lograr esta etapa se coordinó con algunos criadores de cuyes de la región Cusco para la realización de esta tarea en sus respectivas granjas.

La recolección de datos se realizó en seis granjas de cuyes, donde se realizó la captura de imágenes bajo la supervisión del dueño o encargado de la granja. En la tabla 3.1 se muestran las fechas en las que se realizó la recolección de datos, además de la cantidad de cuyes que fueron pesados. En cada visita se pesó una muestra de la población total de cuyes de la granja.

N°	Fecha	Ubicación	Muestra
1	07/02/2021	Tipon	36
2	07/06/2021	Saylla	5
3	12/06/2021	Tipon	30
4	17/06/2021	Huaro	25
5	18/06/2021	Huasao	20
6	24/06/2021	San Jerónimo	17
7	14/07/2021	Huasao	24
8	11/08/2021	Huaro	20
9	12/08/2021	San Jerónimo	22
10	13/08/2021	Huaso	26

Tabla 3.1: Fecha y muestra de recolección de datos
Fuente: Elaboración propia

Figura 3.1: Diagrama de flujo para la recolección de datos
Fuente: Elaboración propia

La etapa de la recolección de datos comprende la captura de la imagen del cuy mediante una cámara digital y el registro del peso del cuy usando una balanza. En la imagen 3.1 se muestra el diagrama de flujo para la recolección de datos para cada cuy. El proceso es como sigue:

- **Selección del cuy**. La selección del cuy comprende la separación del cuy de su poza o jaula para transportarlo a una poza vacía dentro de la misma granja.
- **Captura de imagen**. La captura de imagen del cuy se realiza dentro de la poza vacía, se captura imágenes donde se puede observar el cuerpo entero del cuy dentro de la poza, además que la distancia aproximada de la captura de la imagen es de 1 metro respecto al piso de la poza. Para cada cuy se tomaron aproximadamente entre 3 a 15 fotos en diferentes posiciones dentro de la poza (Figura 3.2).

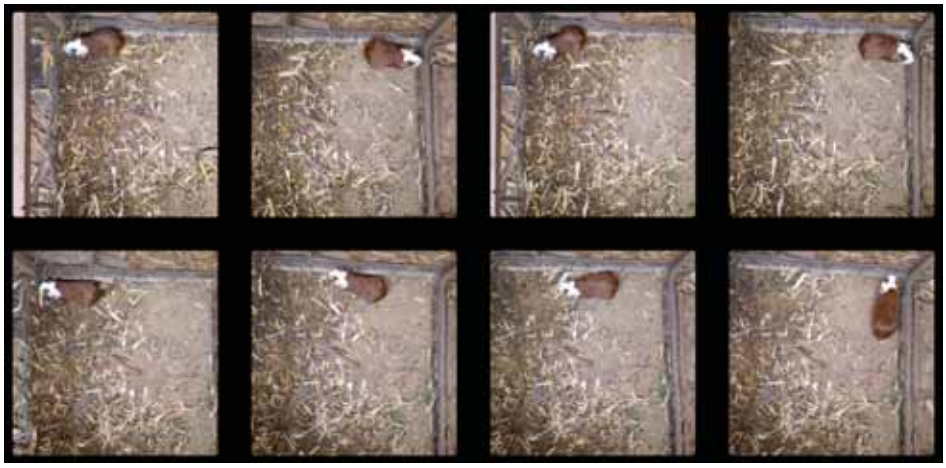


Figura 3.2: Captura de imágenes de un cuy
Fuente: Elaboración propia

- **Registro de peso**. Luego de la captura de las imágenes del cuy, se procede a mover el cuy de la poza vacía a una balanza digital para tomar registro de su peso en gramos, para finalmente devolver al cuy a su respectiva poza.

La recolección de datos estaba sujeto al tiempo disponible que tenía cada productor, tomando alrededor de 3 a 5 minutos la recolección de datos por cada cuy.

3.2. Procesamiento de imágenes

El procesamiento de imágenes comprende los pasos mostrados en la Figura 3.3, para así tener todos los datos debidamente etiquetados y listos para ser organizados en el dataset.

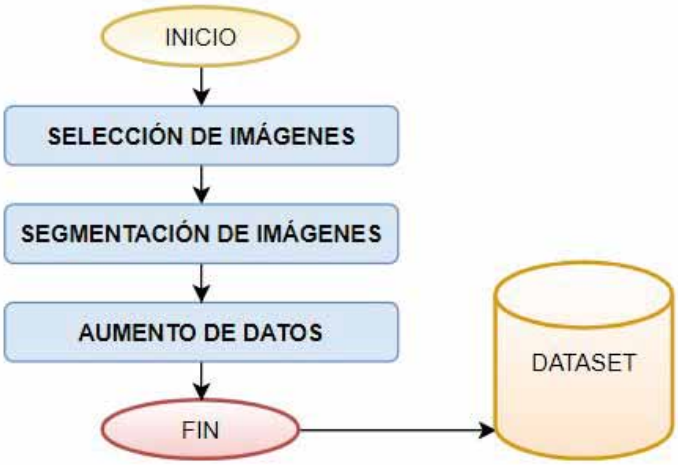


Figura 3.3: Diagrama de flujo del procesamiento de imágenes
Fuente: Elaboración propia

3.2.1. Selección de imágenes

La selección de imágenes consiste en determinar que imágenes recolectadas cumplen los requerimientos para formar parte del dataset, estos requerimientos son:

- El formato para la captura de imagen es JPEG.
- La imagen es capturada en un formato de 1x1 (de tamaño cuadrado).
- Sólo un cuy debe ser capturado en la imagen.
- Se debe visualizar el cuerpo entero del cuy dentro de la imagen.

En este paso se seleccionaron 1561 imágenes que cumplían con estos requisitos, luego de la selección se procedió a redimensionar las imágenes (Figura 3.4) de un tamaño de $3000x3000$ píxeles al tamaño de $640x640$ píxeles, para reducir el peso de cada archivo de imagen. Además se creo un documento de texto (.txt) donde se guardó la relación del nombre del archivo de imagen con el peso medido del cuy en dicha imagen.

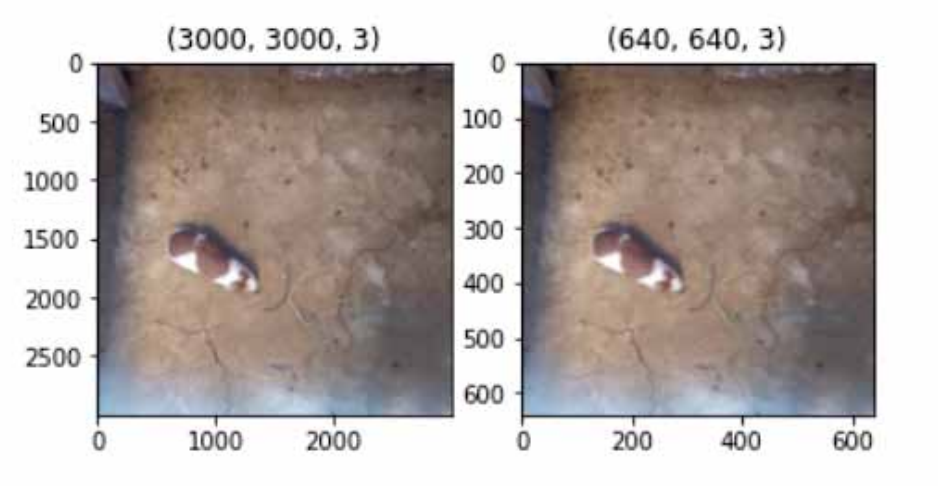


Figura 3.4: Redimensión de una imagen
Fuente: Elaboración propia

3.2.2. Segmentación de imágenes

Para la segmentación de datos se hizo uso de la plataforma de etiquetado Labelbox. La herramienta que se usó fue la herramienta lápiz, la cuál nos permite dibujar sobre el área que ocupa el cuy dentro de una imagen. En la Figura 3.5 se muestra el resultado de la segmentación de una imagen usando la plataforma en línea de Labelbox, en la cuál se puede ver con un color celeste la máscara para esa imagen. El tiempo para la realización del etiquetado fue de aproximadamente 45 segundos por cada imagen de cuy.

Figura 3.5: Segmentación de una imagen usando Labelbox
Fuente: Elaboración propia

La plataforma de Labelbox devuelve como resultado del proceso de etiquetado un archivo en formato JSON con la información de la imagen, su clasificación manual, la máscara y otros datos sobre el archivo.

Para descargar como un lote las 1561 máscaras, se hizo uso de una librería de utilidades para Mask R-CNN, desarrollado por Wilding-McBride & Pun en el lenguaje de programación Python, esta herramienta nos permitió descargar los archivos en formato PNG renombrándolos como la imagen original y una extensión 'mask'.

3.2.3. Aumento de datos

Para el aumentado de datos se hizo uso de la librería de MaskRCNN-utils (Wilding-McBride & Pun (2018)), que permite operar sobre la imagen y la máscara (en caso de rotación y volteado).

Antes de realizar el aumento de datos, se procede a dividir el dataset para el entrenamiento (80 %) y la validación (20 %). Luego se procede a convertir las imágenes de formato JPG a formato PNG para procesar el aumento de datos en este formato. El aumento de datos comprende las siguientes secuencias mostradas en el código 3.1.

El aumento de datos se hizo para agregar un total de 3 imágenes aumentadas por cada imagen original, además de alterar el registro de peso en las imágenes aumentadas con una distribución gaussina, con una desviación estándar de 5 gramos y una media de 0. En la Figura 3.6 se puede ver los valores que adquiere la distribución de Gauss al realizarse sobre 100000 iteraciones.

```
# Secuencia de rotación entre 90° y 270°
seq = iaa.Sequential([
    iaa.Rot90((1,3))
])
# Secuencia para la modificación del espacio de color de la imagen
# Adición, multiplicación y contraste Lineal que altera ligeramente
# la intensidad de un píxel
color_seq = iaa.Sequential({
    iaa.Add((-20,20)),
    iaa.Multiply((0.8,1.2)),
    iaa.contrast.LinearContrast((0.8, 1.2))
}, random_order=True)
```

Código Fuente 3.1: Configuración para el aumento de datos

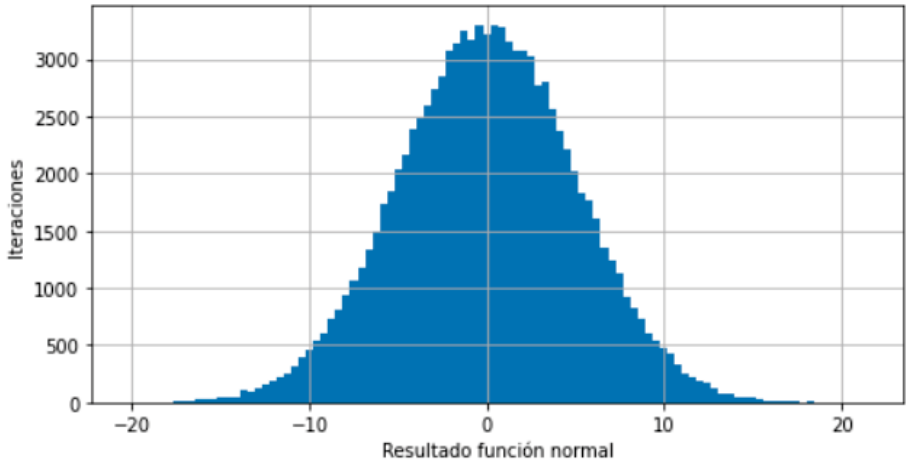


Figura 3.6: Distribución Normal o de Gauss
Fuente: Elaboración propia

El resultado del aumento de datos se puede ver en la Figura 3.7, donde por cada imagen original se obtienen 3 imágenes más, que han sido rotadas en 90°, 180° o 270°, además de modificar los canales de colores aleatoriamente.

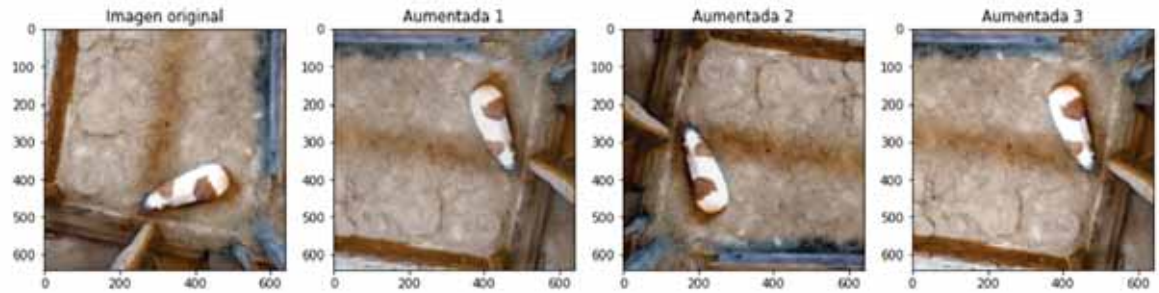


Figura 3.7: Aumento de datos
Fuente: Elaboración propia

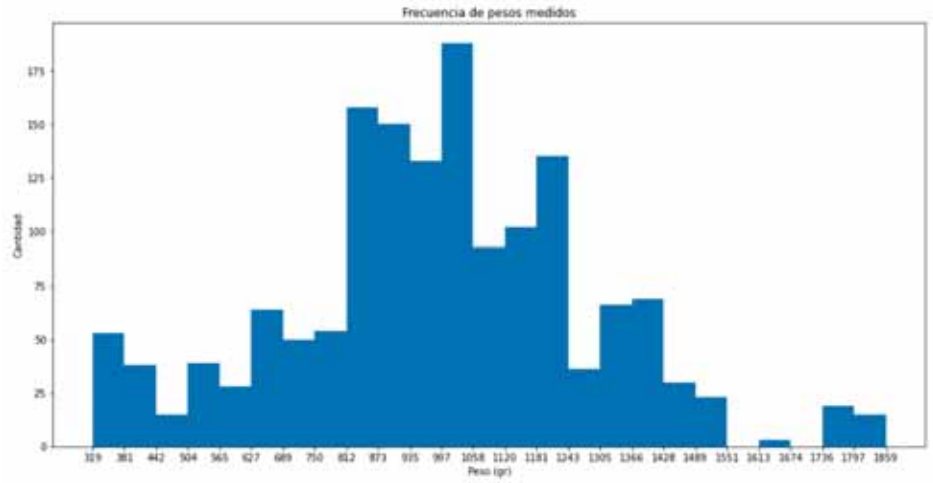
3.3. Construcción del dataset

Con los datos procesados, ya se tienen los datos del dataset creado con las imágenes aumentadas (6244) y los pesos medidos en gramos. La estructura del dataset es la siguiente:

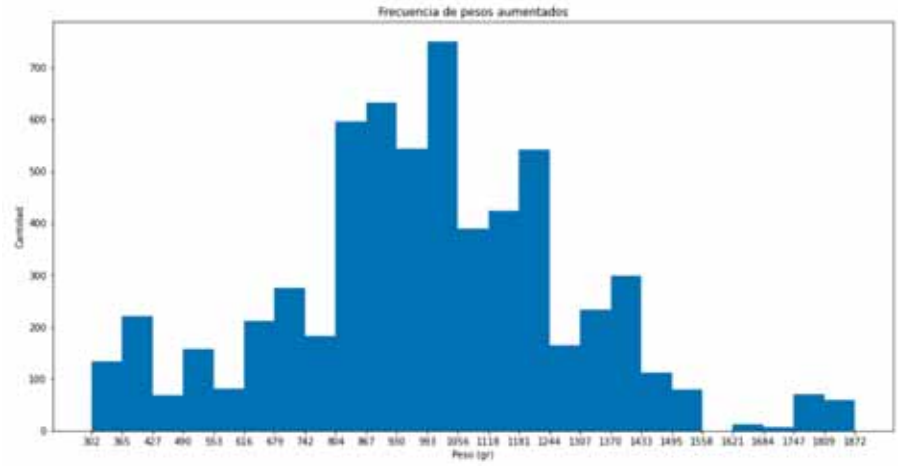
- masks (carpeta con todas las máscaras (6244) de todo el dataset en formato .PNG)
- train (carpeta con las imágenes (4992) de entrenamiento)
- val (carpeta con las imágenes (1252) de validación)
- cuy_bw.txt (documento de texto con los nombres de los archivos de imagen y su respectivo peso del cuy)

3.3.1. Descripción del dataset

El dataset original cuenta con 1561 imágenes de 225 cuyes, mientras que el dataset aumentado contiene 6244 imágenes. En la Figura 3.8a se evidencia cantidad de imágenes que se tiene por cada peso recolectado con los datos originales, en cambio en la Figura 3.8b se evidencia que al realizar el aumentado de datos la gráfica no varía significativamente, lo que muestra que el dataset aumentado tiene las mismas características que el dataset original.



(a) Histograma de los pesos (original)



(b) Histograma de los pesos (aumentado)

Figura 3.8: Comparación de histograma original vs aumentado
Fuente: Elaboración propia

3.4. Adaptación de la arquitectura CNN

Para la adaptación de la arquitectura que permitirá la estimación del peso del cuy se optó por el modelo Mask R-CNN, este modelo está diseñado para la segmentación de instancias, además que permite generar con precisión una máscara del objeto a identificar.

En la Figura 3.9 se puede observar la arquitectura de Mask R-CNN. El modelo toma una imagen como entrada y la procesar por una red troncal (*backbone network*), la salida de esta red troncal va hacia una *Region Proposal Network* (RPN) para generar las propuestas de región que serán aplicadas sobre el mapa de características resultante de la red troncal. Finalmente se dividen dos ramas: una para las clasificación y regresión del cuadro delimitador, y otra para la segmentación de la máscara de la instancia del

objeto.

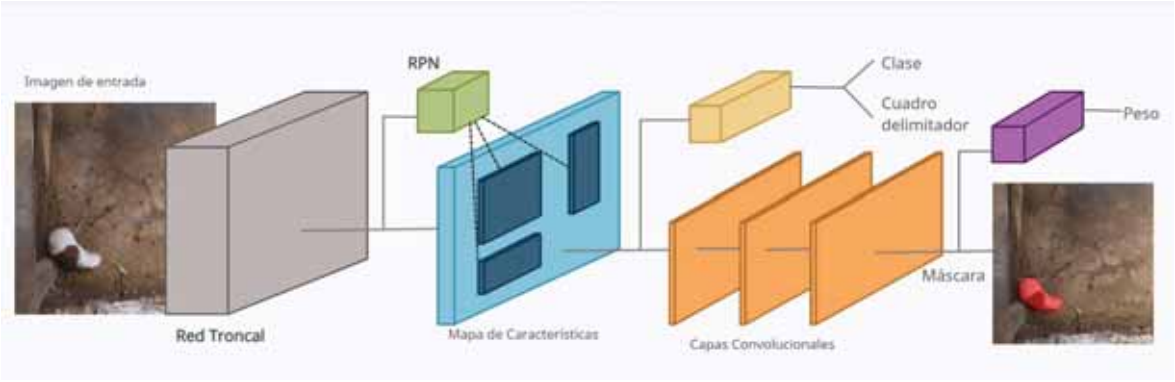


Figura 3.9: Arquitectura de Mask R-CNN
Fuente: Elaboración propia

En la Figura 3.10 se muestra los resultados de Mask R-CNN en la segmentación de instancias usando el dataset de COCO, obteniendo mejores resultados que los ganadores de COCO 2015 y 2016. La métrica para su evaluación es *mask AP*, similar a la de los cuadros delimitadores en la detección de objetos, pero ahora comparando la sobreposición de las máscaras. Para su mejor desempeño se usaron las arquitecturas de ResNet-101 con FPN y ResNeXt-101 con FPN.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
MNC [10]	ResNet-101-C4	24.6	44.3	24.8	4.7	25.9	43.6
FCIS [26] +OHEM	ResNet-101-C5-dilated	29.2	49.5	-	7.1	31.3	50.0
FCIS+++ [26] +OHEM	ResNet-101-C5-dilated	33.6	54.5	-	-	-	-
Mask R-CNN	ResNet-101-C4	33.1	54.9	34.8	12.1	35.6	51.1
Mask R-CNN	ResNet-101-FPN	35.7	58.0	37.8	15.5	38.1	52.4
Mask R-CNN	ResNeXt-101-FPN	**37.1**	**60.0**	**39.4**	**16.9**	**39.9**	**53.5**

Figura 3.10: Resultados de Mask R-CNN para la segmentación de instancias
Fuente: He *et al.* (2017)

3.4.1. Recursos

Para la implementación de la CNN en la estimación del peso del cuy, se comenzó en base a una implementación desarrollada con las librerías para ML Tensorflow y Keras. Esta implementación es una adaptación de la implementación original de Abdulla (2017) de Mask R-CNN, la adaptación de Paul (2018) fue modificada para la detección de puntos clave en la pose humana. En este caso se aprovechó de la rama que fue añadida para la detección de puntos de la pose humana, para usarla en la estimación del peso del cuy.

La implementación de Mask R-CNN (Abdulla, 2017) tiene la implementación de las redes troncales a las arquitectura de ResNet50 y ResNet101. En la Figura 3.11 se puede

observar la arquitectura de ResNet50 resumida en 5 etapas, cada una con sus capas convolucionales, de identidad y de normalización. Se tiene en cuenta que la arquitectura de ResNet101 es similar a la arquitectura Resnet50, solo que en la etapa 4 en vez de tener 5 capas convolucionales, tiene 22 capas convolucionales.

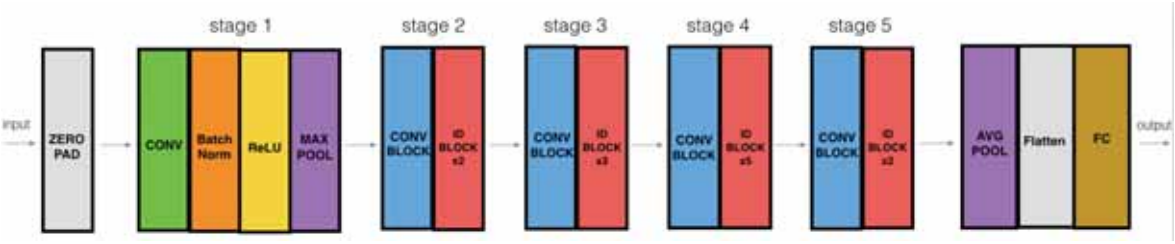


Figura 3.11: Arquitectura de ResNet50
Fuente: Dwivedi (2019)

Se elaboró la Tabla 3.2 para comparar algunas características de las redes troncales usadas en este proyecto, la tabla se basa en los resultados en la competición *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) 2012, que buscaba la clasificación de 1000 clases usando como dataset ImageNet.

Modelo	Año	Profundidad	# Parámetros	Top-5 (%) error
Resnet50	2015	50	25.6M	5.25 %
Resnet101	2015	101	44.5M	4.60 %

Tabla 3.2: Comparación de las arquitecturas de redes troncales
Fuente: Elaboración propia

3.4.2. Implementación

3.4.2.1. Archivos del modelo

La implementación desarrollado por Paul (2018) tiene los siguientes archivos:

- **config.py**. Archivo donde está la configuración predeterminada para el entrenamiento usando el dataset COCO.
- **model.py**. Archivo que contiene la implementación del modelo de Mask R-CNN y las subredes que utiliza, como la implementación de ResNet, FPN, Alineamiento RoI y algunos métodos para el entrenamiento y la inferencia de la red.
- **parallel_model.py**. Archivo que contiene la implementación para el entrenamiento en paralelo.
- **requeriment.txt**. Archivo que contiene los requerimientos de librerías que se deben instalar para la ejecución del entrenamiento y la validación.

- **utils.py**. Archivo con utilidades para la inspección del modelo y del dataset.
- **visualize.py**. Archivo que contiene herramientas para la visualización de los resultados en la inferencia del modelo.

3.4.2.2. Configuración para la carga del dataset de pesos de cuyes

Para la implementación del modelo de Mask R-CNN, se tiene que modificar el archivo de configuración del proyecto, el cuál tiene los parámetros para entrenar la red, así como un archivo para la carga del dataset. El dataset utilizado es diferente a la de la implementación para COCO, ya que solo tiene un objeto para detectar, en este caso el cuy, además de usar máscaras en formato PNG.

La clase que se tiene que sobreescribir tiene el nombre de "Dataset", como se muestra en el Código Fuente 3.2 se modificó tres métodos para la carga correcta del dataset; el primero respecto a la lectura completa del dataset (Id de la imagen, peso del cuy, ruta de la imagen), el segundo es para la carga de las instancias de máscaras y pesos, y el tercero es para la lectura de la ruta de una imagen.

```
class CuyDataset(Dataset):
    def load_dataset(self, dataset_dir, subset, weightsFile):
        """Carga un subconjunto del dataset en la clase Dataset
            dataset_dir: Directorio principal del dataset
            subset: Sub directorio del dataset (train, val)
            weightsFile: ruta del archivo de pesos
        """
    def load_mask(self, image_id):
        """ Carga las instancias de máscaras y pesos de una imagen
            image_id: Id de una imagen
            Return: mask: instancias de máscara en la imagen
                    weight: instancias de pesos en la imagen
                    class_ids: clase a la que pertenece cada instancia
        """
    def image_reference(self, image_id):
        """ Retorno la ruta de la imagen
            image_id: Id de una imagen
            Return: ruta de la imagen
        """
```

Código Fuente 3.2: Documentación de la clase Dataset

3.4.2.3. Modelo de la arquitectura Mask R-CNN

En el archivo "model.py", se encuentra la implementación de las siguientes partes de la arquitectura del modelo Mask R-CNN:

- *Resnet Graph*. Implementación de la arquitectura de Resnet50 y Resnet101 con sus respectivas capas.
- *Proposal Layer*. Capa para el filtrado de propuestas mediante el método de *Non-Maximun Supression* (NMS).
- *Pyramid ROIAlign*. Implementación de la pirámide de características en diferentes niveles junto con el alineamiento de RoIs.
- *Detection Target Layer*. Implementación de la capa que genera propuestas para los resultados objetivos, los cuales serán comparados con los verdaderos.
- *Detection Layer*. Implementación de la capa que genera las detecciones para la validación del modelo durante la inferencia.
- *Region Proposal Network*. Implementación de la capa que genera las propuestas de las regiones de interés.
- *Feature Pyramid Network Heads*. Implementación de las ramas, como la de clasificación, la regresión para el cuadro delimitador y la segmentación de la máscara.
- *Loss Functions*. Implementación de las funciones de pérdida, para cada salida (clase, cuadro delimitador y máscara).
- *Data Generator*. Métodos para la generación de datos en distintas partes de la red.
- *Mask R-CNN Class*. Implementación del modelo Mask R-CNN, agrupando las diferentes subredes que lo componen.
- *Data Formatting*. Métodos para dar forma a los datos que se utiliza en la red.

3.4.2.4. Modificaciones al modelo

Se realizaron algunas modificaciones al modelo principal, para que este funcione en la estimación del peso del cuy, las modificaciones que se realizaron son las siguientes:

- Se agregó una capa de entrada para los pesos, además se normalizó el valor de los pesos medidos con una escala de 2500, valor mayor que el máximo peso medido de los cuyes durante la recolección de datos.
- Se añadió como parámetro la detección de objetivos para el peso del cuy, en la capa de *Detection Target Layer*, la cuál genera propuestas a partir de los valores reales.
- Se diseño dos modelos de ramas finales aplicadas en la estimación del peso del cuy, la primera comparte las capas de convolución de la rama de la máscara, en

cambio la segunda contiene sus propias capas convolucionales como una rama separada.

- Se implementó la función de pérdida para la evaluación del peso estimado. Se usaron las métricas de: *Mean Squared Error* (MSE) que calcula la media de el cuadrado de la diferencia entre el valor real y el valor estimado; y *Mean Absolute Error* (MAE) que calcula la media entre la diferencia del valor real y el valor estimado.

3.4.2.5. Modelo A: Rama compartida para la máscara y el peso

El Modelo A, se muestra en la Figura 3.12, donde se observa que la rama para la estimación del peso se separa de la rama de la máscara después de las 4 capas convolucionales creadas para la segmentación. La rama para la estimación de peso tiene una capa de convolución de 256 *kernels* de tamaño (3x3) junto con una capa de *batch normalization* y una función de activación ReLU, luego de esta salida se realiza un aplanamiento para pasar a una capa densa de 1024 neuronas aplicandose como la función de activación lineal, que finalmente salen a la ultima capa de 2 neuronas (compuesta por el número de clases: cuy y *background*) con la función de activación lineal.

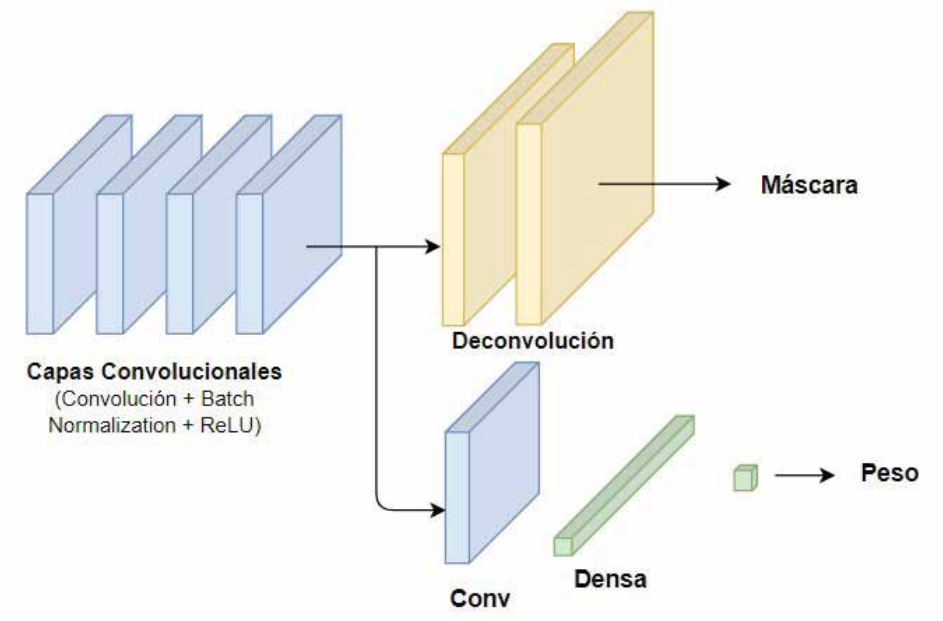


Figura 3.12: Diagrama de la rama para la máscara y el peso
Fuente: Elaboración Propia

En la Tabla 3.3 se muestra la descripción de las capas junto con el tamaño de las entradas y salidas de cada capa de la rama para la estimación del peso después de la capa del Alineamiento de RoIs, en la tabla se observa que el tamaño no varía en las primeras 4 capas convolucionales, más bien, después del aplanamiento recién se reduce

el tamaño de neuronas hasta tener solo las dos neuronas para la estimación del peso del cuy.

El primer valor (?) representa a la cantidad del *batch* con el que se va a entrenar, en este caso será de 2 imágenes. Ahora bien, el valor de 100 es la cantidad de RoI máxima para el entrenamiento, por defecto su valor está en 100, aunque si se disminuye no afecta en gran medida este entrenamiento.

Este modelo consta de 4 bloques de 3 capas, cada bloque esta compuesto por una capa convolucional, una capa de normalización y una función de activación ReLU hasta llegar a la rama que va a generar la máscara, en esta parte se utiliza un bloque de 3 capas para continuar con el procesamiento de los mapas de características, finalmente se realiza un aplanamiento y con una capa densa con función de activación lineal se procede a obtener la estimación del peso.

Capa	**Entrada**	**Salida**
mrcnn_mask_conv1	1x?x14x14x14x256	?x100x14x14x256
mrcnn_mask_bn1	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_conv2	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_bn2	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_conv3	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_bn3	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_conv4	?x100x14x14x256	?x100x14x14x256
mrcnn_mask_bn4	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
Salida compartida con la rama de la máscara		
mrcnn_cbw_conv1	?x100x14x14x256	?x100x14x14x256
mrcnn_cbw_nb1	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_cbw_flatten	?x100x14x14x256	?x100x50176
mrcnn_cbw_dense1	?x100x50176	?x100x1024
mrcnn_bodyweight	?x100x1024	?x100x2

Tabla 3.3: Descripción de la rama A
Fuente: Elaboración propia

3.4.2.6. Modelo B: Rama para el peso

El Modelo B, como se muestra en la Figura 3.13 esta conformada por una rama independiente compuesta por 3 capas convolucionales de 256 *kernels* y función de activación ReLU. Luego el proceso continua con dos capas convolucionales de 64 *kernels* seguidas de sus respectivas capas de reducción (*max-pooling*) con la función de activación ReLU. Para luego ser aplanadas y conectadas a una capa densa de 256 neuronas (con función de activación ReLU) y finalmente a la salida de dos neuronas (compuesta por el número de clases: cuy y *background*) y una función de activación Lineal.

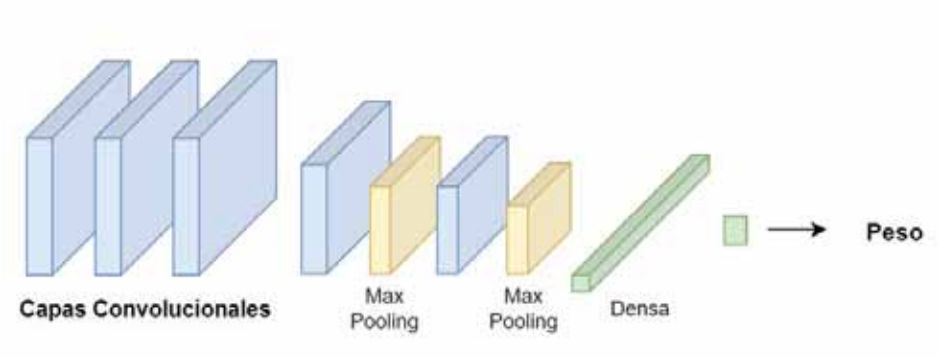


Figura 3.13: Diagrama de la rama para el peso
Fuente: Elaboración Propia

En la Tabla 3.4 se muestra el tamaño de las entradas y salidas de cada capa de la rama del modelo B para la estimación del peso luego de la capa de Alineamiento RoI, la diferencia más significativa con el Modelo A son las capas convolucionales de 64 *kernels* y la reducción de estas capas usando las capas de *max-pooling*, para así limitar las características en la capa densa.

En este modelo, para la estimación del peso se aplican dos bloques de 3 capas (convolución, normalización y función de activación ReLU) para seguir obteniendo mapas de características de interés, luego se aplican dos bloques de 2 capas (convolución y reducción) para reducir la complejidad de los mapas de características, después de la reducción se le aplica un aplanamiento a los resultados para con una capa densa y una función de activación lineal obtener el resultado de la estimación.

Capa	Entrada	Salida
mrcnn_bw_conv1	1x?x14x14x14x256	?x100x14x14x256
mrcnn_bw_bn1	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_bw_conv2	?x100x14x14x256	?x100x14x14x256
mrcnn_bw_bn2	?x100x14x14x256	?x100x14x14x256
ReLU activation	?x100x14x14x256	?x100x14x14x256
mrcnn_bw_conv3	?x100x14x14x256	?x100x14x14x64
mrcnn_bw_maxpooling1	?x100x14x14x64	?x100x7x7x64
mrccn_bw_conv4	?x100x7x7x64	?x100x7x7x64
mrcnn_bw_maxpooling2	?x100x7x7x64	?x100x3x3x64
mrcnn_cbw_flatten	?x100x3x3x64	?x100x576
mrcnn_bw_dense	?x100x576	?x100x256
mrcnn_bodyweight	?x100x256	?x100x2

Tabla 3.4: Descripción de la rama B
Fuente: Elaboración propia

3.4.3. Determinar parámetros de entrenamiento

Los parámetros se ajustaron en base a las características de los modelos seleccionados con la finalidad de obtener un buen rendimiento. La mayoría de los parámetros se mantienen como en la definición original de la implementación por Paul (2018). Los parámetros que se modificaron para realizar las pruebas son:

- **Número de épocas**. Es el número de veces que se le pasará como entrada el conjunto de entrenamiento al modelo implementado. Este parámetro esta definido en 50 épocas.
- **Tamaño de la imagen**. Las imágenes del dataset se determinaron a un tamaño de 640x640 píxeles.
- **Número de pasos por época**. El número de pasos por época indica la cantidad de veces en las que se procesará el conjunto de entrenamiento. Este parámetro lo definimos en 500 pasos por época.
- **Red troncal**. Se define a la red troncal con la que se realizará el entrenamiento. En este caso son ResNet50 y ResNet101.
- **Pesos pre-entrenados**. Son los pesos ya entrenados con las redes troncales, en este caso se usó los pesos pre-entranados en COCO y en ImageNet.
- **Rama**. Se define como el modelo de rama que estimará el peso del cuy, los modelos están definidos en A (peso y máscara en la misma rama) y B (rama solo para el

peso).

- **Imágenes por GPU**. Es la cantidad de imágenes que se entrenarán por GPU. Por el consumo de memoria del GPU este valor se define en 2 imágenes.
- **Tamaño de la Mini-máscara**. El modelo hace uso de una mini-máscara para acelerar el entrenamiento. El valor se define en 56x56 píxeles.

3.5. Prototipo implementado

En esta sección se muestra el desarrollo del prototipo de una aplicación móvil para la estimación del peso del cuy. En la Figura 3.14 se muestra el diseño de la arquitectura del prototipo compuesta por el *frontend* y el *backend*. El *frontend* esta compuesta por la interfaz de una aplicación móvil que captura una imagen del cuy y la consulta al API, para luego mostrar la respuesta del API con el peso estimado. El *backend* comprende la recepción de la imagen capturada por la aplicación móvil para procesarla de acuerdo a la configuración de entrada del modelo de CNN, este modelo estima el peso y luego se devuelve una imagen procesada con la máscara y el cuadro delimitador del cuy.

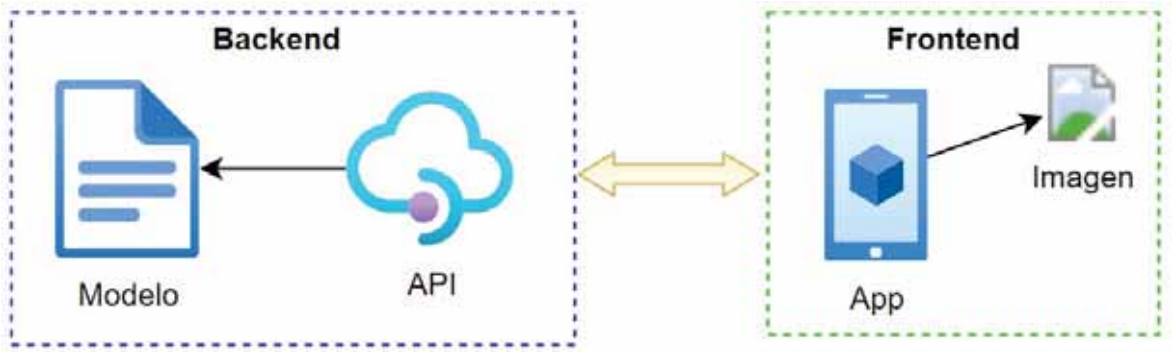


Figura 3.14: Arquitectura del prototipo
Fuente: Elaboración propia

3.5.1. Backend

El *backend* se encarga de procesar una imagen para realizar la estimación del peso del cuy usando un modelo entrenado en la estimación del peso del cuy.

Para la implementación de la API se usó el framework minimalista Flask, que permite el desarrollo rápido de aplicaciones web usando el lenguaje de programación Python. La ventaja de usar Flask, es que permite cargar todas las librerías usadas en la implementación y entrenamiento del modelo desarrollado. El proceso que sigue el API para la estimación del peso del cuy se muestra en la Figura 3.15.

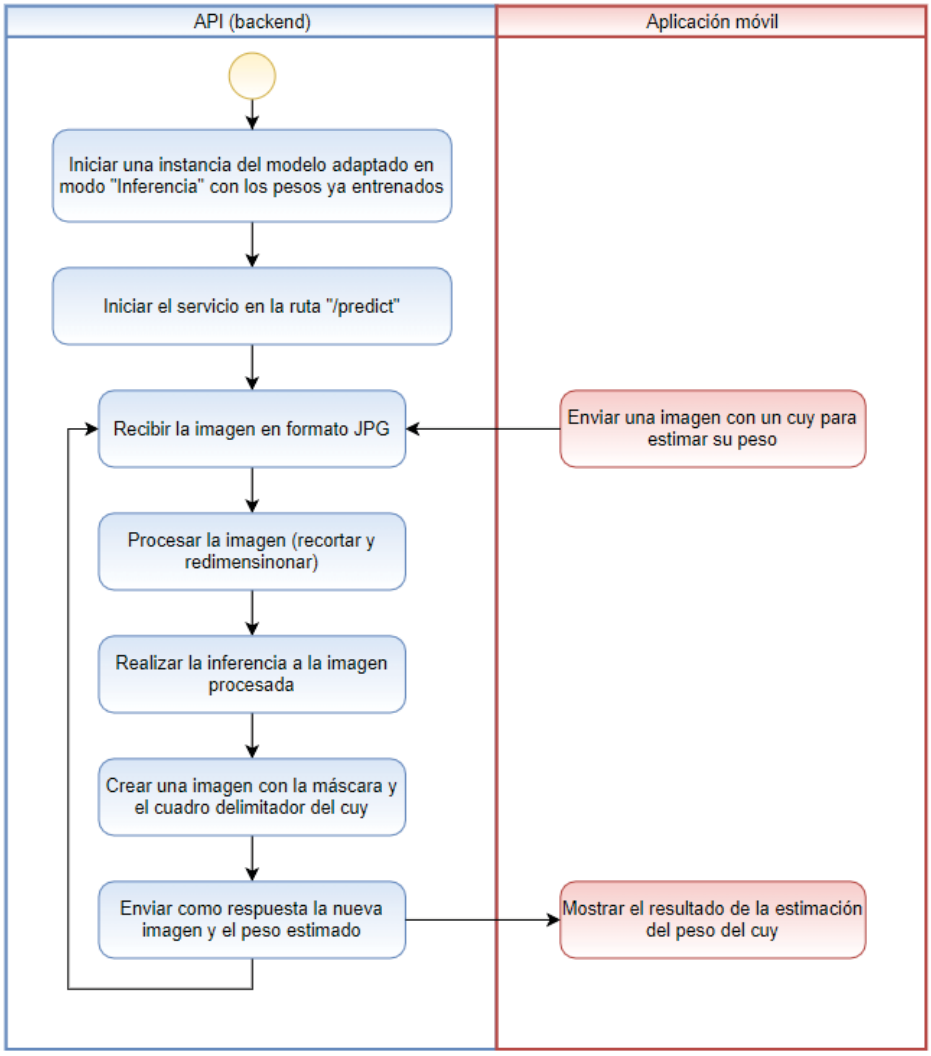


Figura 3.15: Proceso de inferencia del API
Fuente: Elaboración propia

3.5.2. Frontend

La implementación del *frontend* se hizo usando el framework Expo, que es una plataforma para el desarrollo de aplicaciones universales en Android, iOS y web, usando Javascript y React.

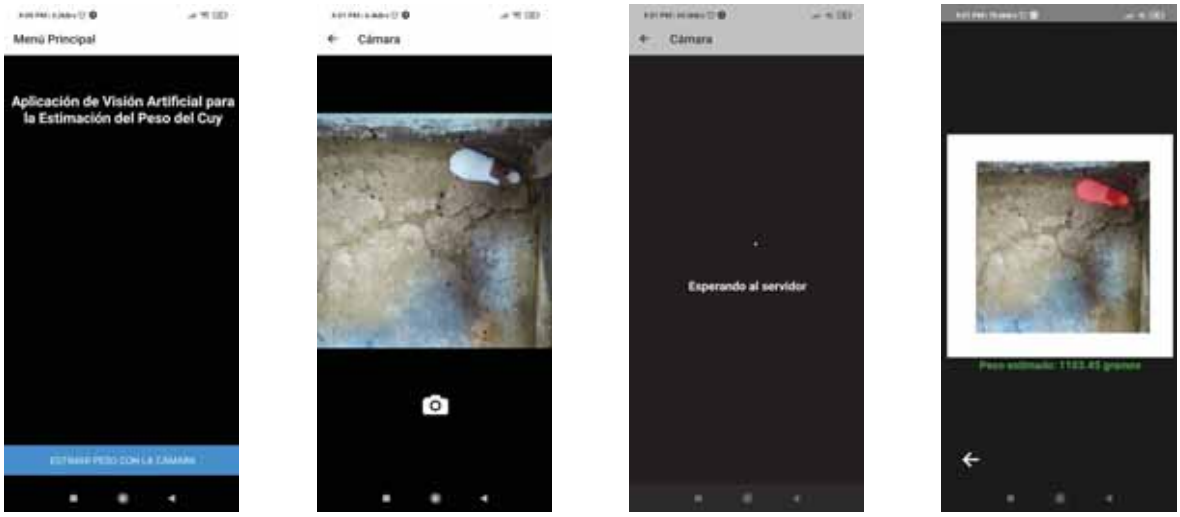
El framework Expo nos permite un desarrollo fácil y rápido de aplicaciones para el sistema operativo Android, además de tener una aplicación móvil llamada Expo Go para la depuración del desarrollo de las aplicaciones.

Para la implementación de la aplicación móvil se requiere el acceso a la cámara del móvil, en este caso solo se hace uso de la cámara posterior del móvil. La visualización previa se configura con el ratio de 4:3, además de mostrar solo una ventana de tamaño cuadrado para la captura de la imagen. En esta aplicación se optó por usar el tamaño

de imagen que este más próxima al de 640x640 píxeles, en la depuración de la aplicación fue de 1280x960 píxeles, aunque este valor varía dependiendo del dispositivo móvil.

La aplicación desarrollada tiene 4 interfaces (Figura 3.16) que muestran el proceso de la estimación del peso del cuy:

- La primera interfaz corresponde a la bienvenida de la aplicación, donde está el título de la aplicación y un botón para acceder a la cámara del móvil (Figura 3.16a).
- La segunda interfaz corresponde a la visualización previa de la cámara, aquí se puede capturar la imagen del cuy presionando en el botón de cámara. (Figura 3.16b).
- La tercera, es la interfaz de espera de la respuesta del API (Figura 3.16c), luego de enviar la imagen capturada por la cámara del móvil.
- La cuarta interfaz muestra el resultado que devuelve el API, en esta interfaz se muestra una imagen con el cuy detectado, además de un mensaje indicando el peso estimado en gramos (Figura 3.16d).



(a) (b) (c) (d)

Figura 3.16: Interfaz de la Aplicación para la estimación del peso del cuy
Fuente: Elaboración propia

Capítulo 4

Resultados

En este capítulo se procede a mostrar los resultados obtenidos del entrenamiento desarrollado al modelo en sus cuatro variantes (ResNet50A, ResNet101A, ResNet50B, ResNet101B). Para lo cuál primero se detallan las métricas que se utilizaron para evaluar la precisión de los modelos. Segundo, se muestran los resultados de cada modelo entrenado respecto a las métricas descritas. Tercero, se seleccionan los 3 mejores modelos entrenados y se procede a refinarlos para obtener una mejor precisión. Y cuarto, se compara los resultados frente a las investigaciones descritas en los antecedentes. También en este capítulo se muestran los detalles técnicos de las herramientas que se utilizaron para el desarrollo y evaluación de los modelos.

En la Figura 4.1 se muestra el resultado final con una precisión del 80 % en la estimación del peso del cuy usando el modelo Mask R-CNN adaptado.

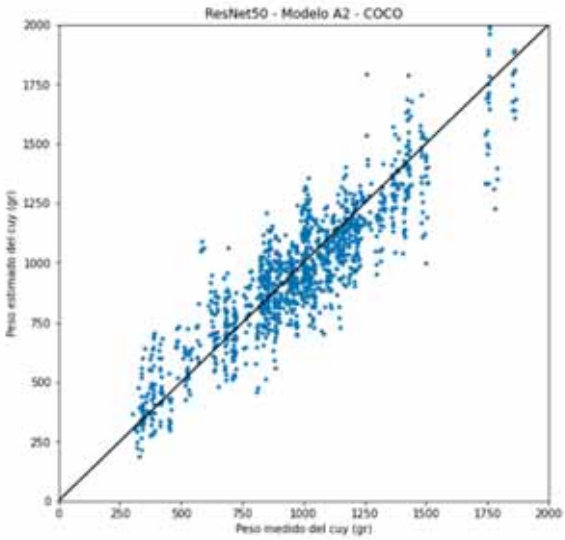


Figura 4.1: Gráfica de dispersión de la inferencia ResNet50A Refinado (80 %) con el dataset de evaluación

Fuente: Elaboración propia

4.1. Métricas de evaluación

Al finalizar el entrenamiento del modelo Mask R-CNN modificado, se realizó una evaluación para medir la precisión de la CNN aplicando métricas de evaluación tanto al subconjunto de entrenamiento y al de validación del dataset desarrollado en la estimación del peso del cuy.

Las métricas usadas se dividen en: las métricas que se utilizan para verificar la sobreposición de la máscara obtenida respecto a la máscara del valor fundamental, y las métricas que evalúan la correlación y error obtenido en la comparación del peso estimado frente al peso medido.

Métricas para la máscara

- **mAP**. Esta métrica mide la sobreposición entre la máscara obtenida por el modelo frente a la máscara del valor fundamental. Su valor se obtiene usando el cálculo de AP AUC (Figura 2.31b), donde se suman las regiones que están por debajo de la curva de *precision/recall* de cada imagen, usando la Ecuación 4.1, que muestra la sumatoria de cada región interpolada por el valor máximo de una región. El valor calculado es la media de los AP resultantes por cada imagen.

$$\begin{aligned} AP &= \sum (r_{n+1} - r_n) p_{interp}(r_{n+1}) \\ p_{interp}(r_{n+1}) &= max_{\tilde{r} \geq r_{n+1}}(\tilde{r}) \end{aligned} \tag{4.1}$$

- **mAP50**, mide la precisión con un umbral de IoU>0.50. Es la métrica más utilizada para la evaluación en la detección de objetos y segmentación de instancias.
- **mAP75**, mide la precisión de forma más estricta con el valor de IoU>0.75.
- **mAP[.50:.05:.95]**, mide el promedio de la precisión en 10 niveles de IoU, desde 0.50 hasta 0.95 con pasos de 0.05. Es la métrica principal de la competición de detección de objetos COCO.

Métricas para la estimación del peso

En la estimación del peso se busca que la diferencia entre la estimación y el valor real sean lo más cercanos, por eso se utilizan métricas que midan ese error en el conjunto evaluado.

En la estimación del peso del cuy, tenemos los valores medidos o la verdad fundamental denotándolo como y y los valores estimados por la CNN denotándolo como $\hat{y}$. Donde n es el tamaño del conjunto de evaluación. Las métricas usadas son las siguientes:

- ***Mean Absolute Error* (MAE)**. Calcula el error absoluto medio entre los valores medidos y los valores estimados (Ecuación 4.2). Los valores atípicos no tienen un gran efecto en esta métrica. Con esta métrica se busca que el valor se acerque a 0.

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i| \tag{4.2}$$

- ***Mean Squared Error* (MSE)**. Calcula el error cuadrático medio entre el valor medido y el valor estimado (4.3). Esta métrica obtiene la varianza del estimador. Los valores atípicos tienen un gran efecto en esta métrica, haciendo que el resultado sea muy grande, e indicándonos que la estimación esta fallando en ciertos valores dando resultados muy erróneos.

$$MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.3}$$

- ***Root Mean Squared Error* (RMSE)**. Calcula la raíz cuadrada del MSE (Ecuación 4.4). Nos muestra la desviación estándar entre los valores medidos y los valores estimados. Lo que indica el promedio en que los resultados se ajustan a la media de todos los valores estimados. Los valores óptimos tienden a acercarse a 0.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{4.4}$$

- ***Mean Absolute Percentage Error* (MAPE)**. Esta métrica expresa la exactitud como un porcentaje del error (Ecuación 4.5). Es similar a MAE pero obteniendo el resultado como porcentaje entre la diferencia del valor medido y el estimado en cada punto evaluado. El resultado se mide en una relación porcentual para comparar con otros resultados que utilizan otras unidades de masa. En este caso, se espera un resultado lo más cercano a 0 %.

$$MAPE(y, \hat{y}) = \frac{100}{n}\sum_{i=0}^{n-1}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{4.5}$$

- **Coeficiente de determinación (R^2)**. Representa la proporción de varianza (de y) que ha sido explicada por las variables independientes en el modelo. Proporciona una indicación de bondad de ajuste y, por lo tanto, una medida de lo probable que un modelo prediga las muestras invisibles, a través de la proporción de varianza explicada. El resultado del coeficiente de determinación oscila entre 0 y 1, indicando que cuando se acerque el resultado a 1, mayor es el ajuste del modelo a la variable que estamos intentando explicar. En la Ecuación 4.6 se puede ver su fórmula. Para una mejor explicación el resultado se representa en porcentaje,

siendo 100 % el resultado máximo.

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$
$$donde : \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.6}$$

4.2. Evaluación del modelo A implementado

En esta sección se muestran los resultados de la evaluación realizada al modelo A implementado, el cuál comparte las capas convolucionales con la rama de la máscara. Los modelos fueron entrenados con 50 épocas de 500 pasos por cada época para el entrenamiento y 50 pasos para la validación, además de usar como función de pérdida el MAE para la estimación de peso.

La evaluación comprende las gráficas de las funciones de pérdida durante el entrenamiento, los resultados de las métricas empleadas para la segmentación de instancias y las métricas de regresión para la estimación del peso, además de las gráficas de dispersión de los valores medidos frente a los estimados.

ResNet50A

En esta parte se muestra el resultado del entrenamiento realizado al modelo ResNet50A.

En la Figura 4.2 se muestra las gráficas de las curvas de pérdida del modelo ResNet50A durante el entrenamiento usando los pesos pre-entrenados con el dataset COCO.

La gráfica del lado izquierdo de la Figura 4.2 muestra la pérdida promedio de todas las funciones de pérdida del modelo implementado. En cambio en el lado derecho se muestra las curvas de cuatro funciones de pérdida (cuadro delimitador, estimación del peso, clasificación de clase y generación de la máscara). Las lineas continuas son los valores medidos con el conjunto de entrenamiento, mientras que las lineas punteadas son los valores medidos con el conjunto de validación.

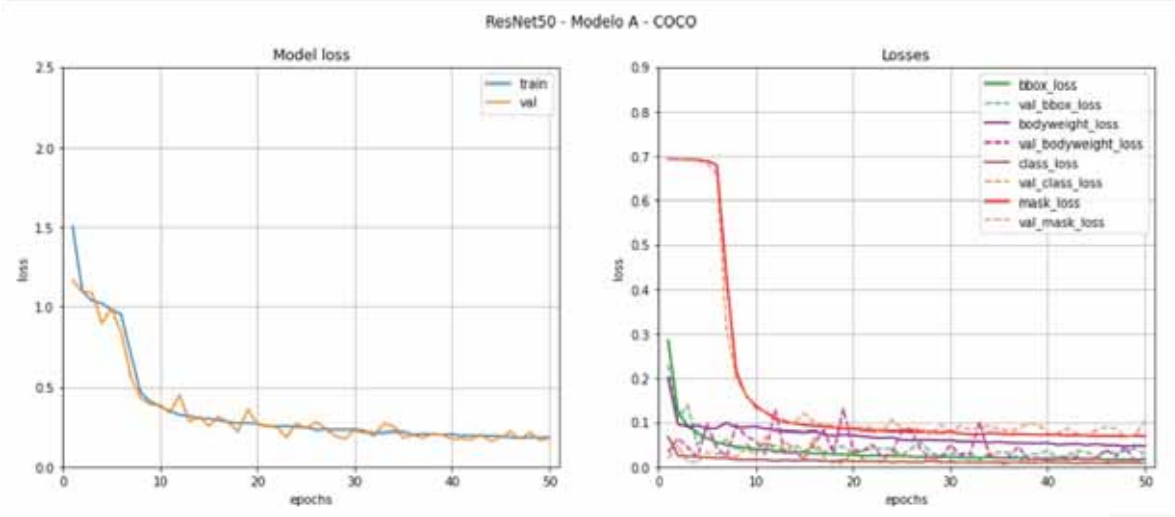


Figura 4.2: Gráfica de la curva de pérdida del modelo ResNet50A y COCO
Fuente: Elaboración propia

En la Figura 4.3 se muestra la curva de la pérdida del modelo ResNet50A durante el entrenamiento usando los pesos pre-entrenados con el dataset ImageNet. Igualmente a la gráfica de la Figura 4.2 se muestran la pérdida promedio en el lado izquierdo, y en el lado derecho las curvas de las cuatro funciones de pérdida principales.

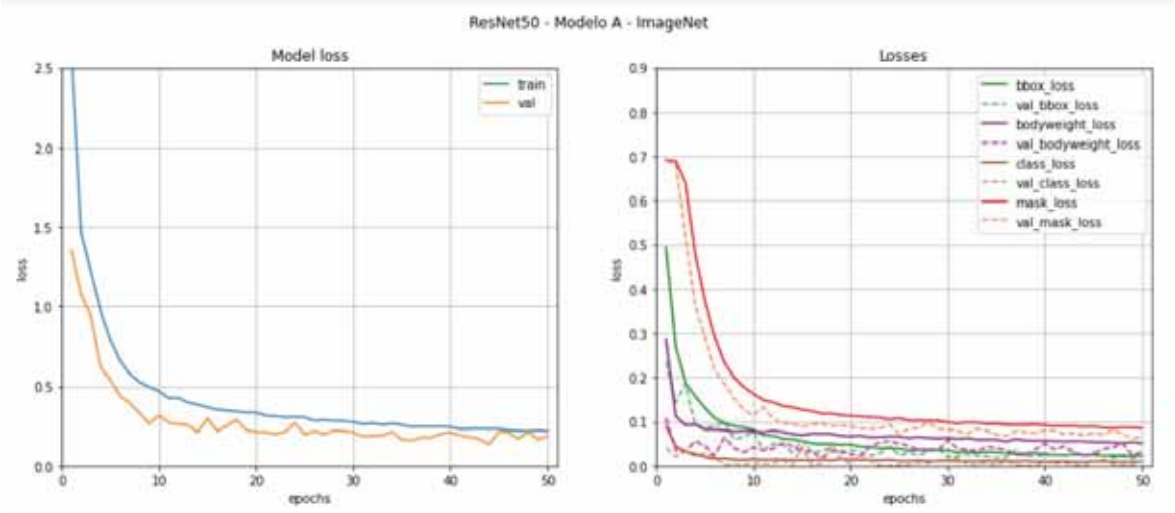


Figura 4.3: Gráfica de la curva de pérdida del modelo ResNet50A e ImageNet
Fuente: Elaboración propia

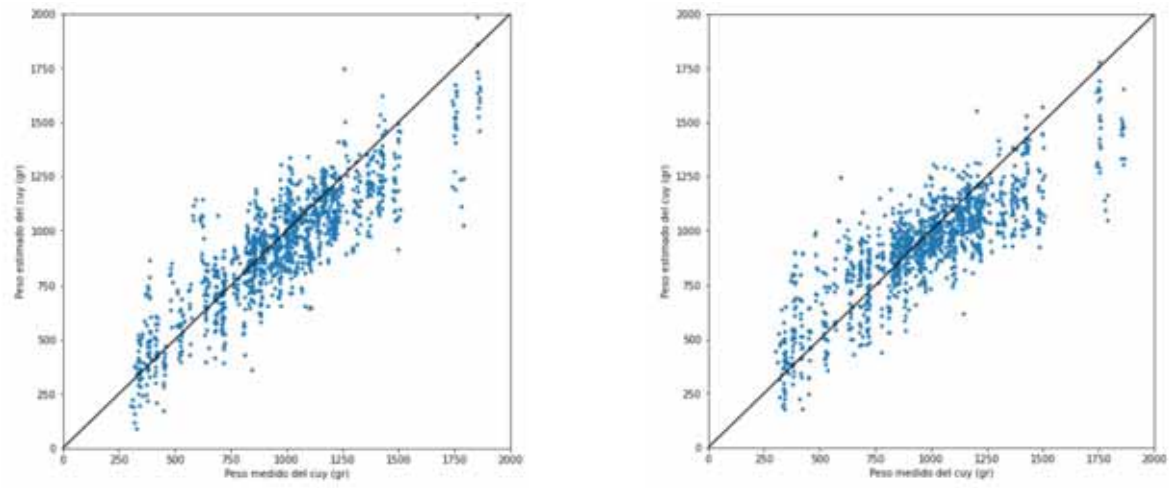
En la Tabla 4.1 se observa los resultados de la evaluación de la implementación del modelo ResNet50A, aplicados al conjunto de entrenamiento y al conjunto de validación. La evaluación también se dividió por el uso de los pesos pre-entrenados: (COCO e ImageNet). Las primeras 3 métricas se aplican en la precisión de la segmentación del cuy dentro de la imagen, mientras que las métricas de regresión son sobre la estimación del peso del cuy.

Se muestra que el resultado es mejor cuando se evalúa al conjunto de entrenamiento, pero similar al evaluado al conjunto de validación. En este caso, el mejor resultado para el conjunto de validación es de 72.38 %.

Resultados de la Inferencia - ResNet50A				
Métrica	**COCO**		**ImageNet**	
	train	val	train	val
mAP@50	1.00	1.00	1.00	1.00
mAP@75	1.00	1.00	1.00	0.99
mAP@[.50:.05:.95]	0.90	0.80	0.80	0.80
Conjunto Evaluado:	4941/4992	1250/1252	4988/4992	1251/1252
MAE	107.98	125.90	120.19	136.00
MSE	21312.11	26951.89	26410.68	32132.14
RMSE	145.99	164.17	162.51	179.25
MAPE	11.65 %	14.20 %	13.03 %	15.69 %
R^2	75.59 %	72.38 %	69.60 %	67.17 %
Tiempo de entrenamiento	12h 10m 14s		10h 4m 35s	

Tabla 4.1: Evaluación de las métricas sobre ResNet50A
Fuente: Elaboración propia

En la Figura 4.4 se muestra la gráfica de dispersión usando los datos de validación. Se compara el valor real frente al valor estimado por el modelo implementado. La Figura 4.4a muestra el resultado cuando se usó los pesos pre-entrenados de COCO, mientras que la Figura 4.4b muestra el resultado usando ImageNet. En ambos casos se observa que el modelo no estima correctamente cuando los pesos de los cuyes son mayores a 1750 gr., mostrando una mejor correlación el modelo entrenado usando los pesos de COCO.



(a) COCO (b) ImageNet

Figura 4.4: Gráfica de dispersión de la inferencia ResNet50A
Fuente: Elaboración propia

ResNet101A

En esta parte se muestran los gráficos y resultados del entrenamiento realizado al modelo ResNet101A.

En la Figura 4.5 se muestran las curvas de las funciones de pérdida del modelo implementado usando los pesos pre-entrenados de COCO para el modelo ResNet101A.

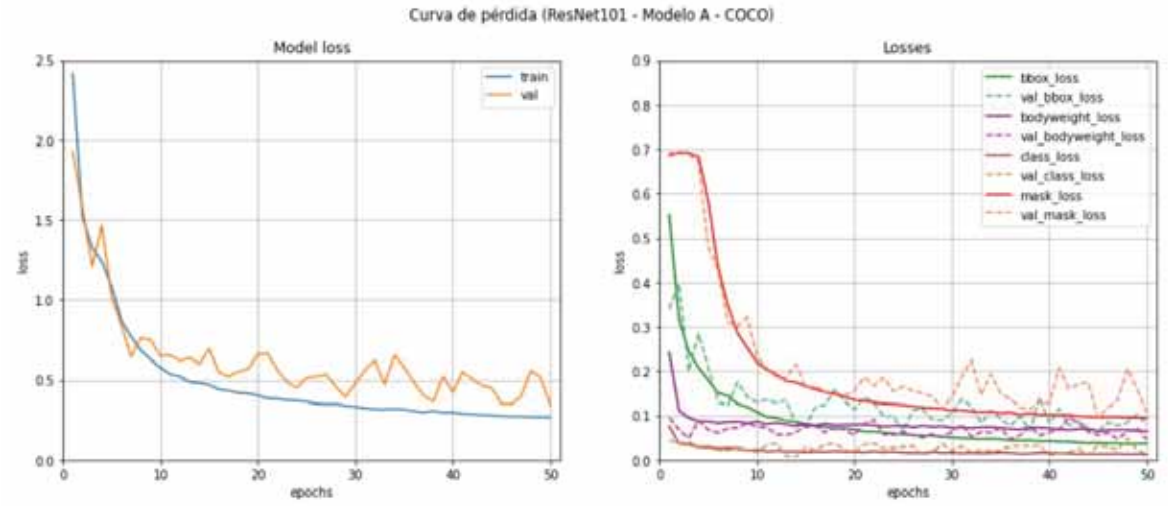


Figura 4.5: Gráfica de la curva de pérdida del modelo ResNet101A y COCO
Fuente: Elaboración propia

En la Figura 4.6 se muestra las curvas de las funciones de pérdida de ResNet101A usando los pesos pre-entrenados de ImageNet.

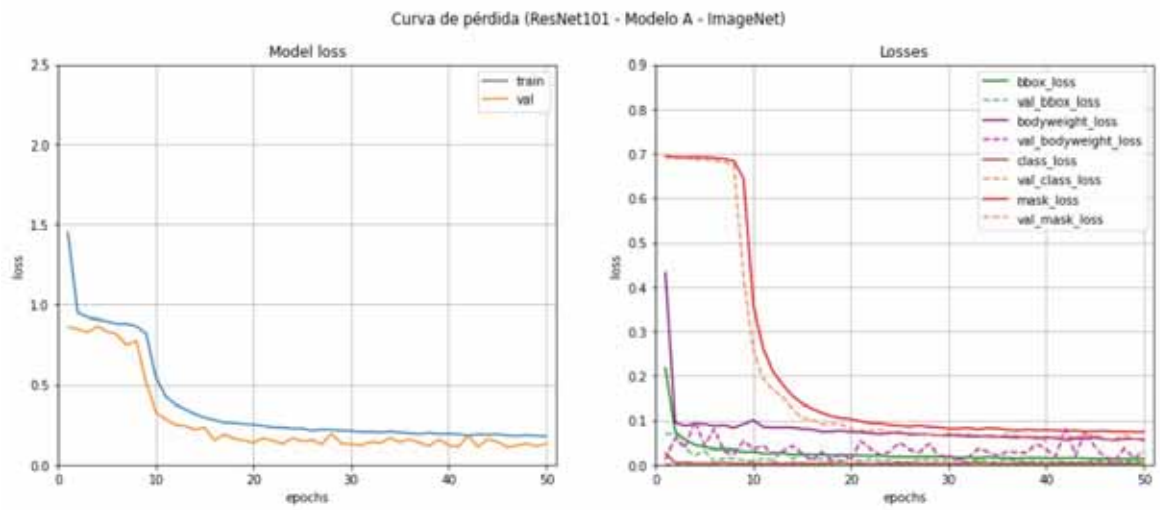


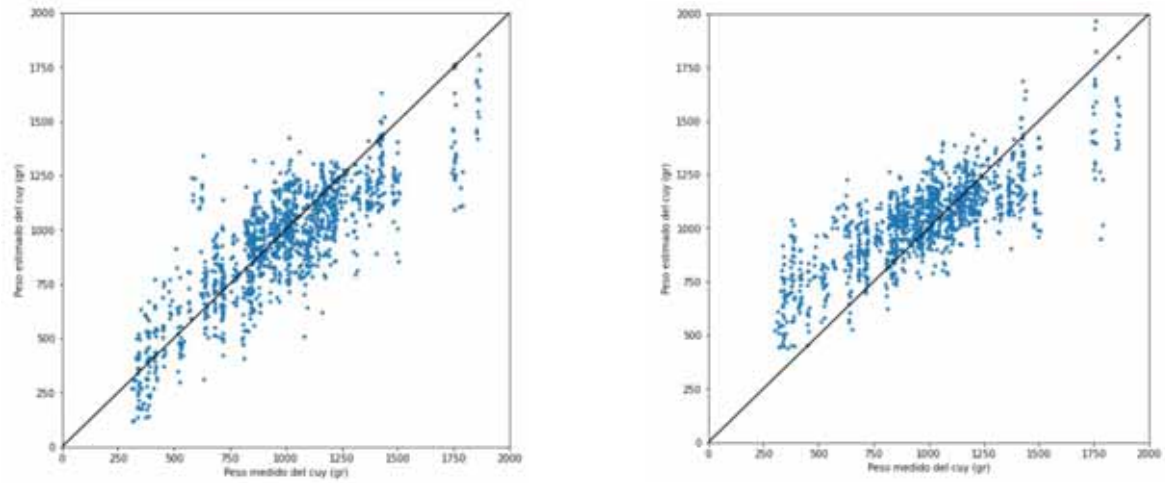
Figura 4.6: Gráfica de la curva de pérdida del modelo ResNet101A e ImageNet
Fuente: Elaboración propia

En la Tabla 4.2 se muestran los resultados de la evaluación del modelo ResNet101A, aplicados tanto al conjunto de entrenamiento como al conjunto de validación. El mejor resultado con el conjunto de validación fue de 66.36 % usando los pesos de COCO.

Resultados de la Inferencia - ResNet101A				
Métrica	**COCO**		**ImageNet**	
	train	val	train	val
mAP@50	1.00	1.00	1.00	1.00
mAP@75	1.00	1.00	1.00	1.00
mAP@[.50:.05:.95]	0.90	0.90	0.70	0.80
Conjunto Evaluado:	4964/4992	1249/1252	4931/4992	1240/1252
MAE	122.95	138.80	153.70	164.06
MSE	26983.71	32956.76	39410.61	43889.55
RMSE	164.27	181.54	198.52	209.50
MAPE	13.33 %	15.69 %	19.62 %	22.07 %
R^2	69.01 %	66.36 %	54.82 %	55.46 %
Tiempo de entrenamiento	10h 42m 18s		13h 40m 42s	

Tabla 4.2: Evaluación de las métricas sobre ResNet101A
Fuente: Elaboración propia

En la Figura 4.7 se muestran las gráficas de dispersión de la inferencia realizada al conjunto de evaluación con el modelo ResNet101A. La Figura 4.7a muestra la gráfica usando los pesos pre-entrenados con COCO, mientras que la Figura 4.7b muestra la gráfica usando los pesos de ImageNet.



(a) COCO (b) ImageNet

Figura 4.7: Gráfica de dispersión de la inferencia ResNet101A
Fuente: Elaboración propia

Respecto al modelo A, se observó que los resultados de la precisión respecto a la máscara corresponde con un valor de 1.0 cuando el umbral IoU es 0.5 y 0.75, demostrando que con estos umbrales el modelo siempre reconoce al cuy y lo segmenta con una buena precisión. Igualmente al evaluar el umbral con 10 niveles el resultado muestra una precisión entre 0.80 y 0.90, lo que ratifica el buen desempeño en la estimación del peso del cuy.

Para los resultados que se obtuvieron respecto a la estimación del peso, se observa que la desviación estándar del modelo es alta, siendo el resultado mayor a 140 gramos en las evaluaciones. Por las gráficas de dispersión se puede observar que cuando el modelo estima un cuy con un peso mayor a 1750 gramos el resultado se aleja por una gran diferencia.

También se observa que los resultados usando los modelos con la red troncal ResNet50 son mejores que los usados con ResNet101.

El mejor resultado obtenido fue del modelo ResNet50A con una correlación de 72.38 % y un MAPE de 14.20 %, evaluados con el conjunto de validación.

4.3. Evaluación del modelo B implementado

En esta sección se muestran los resultados de la evaluación realizada al modelo B implementado, el cuál se define como una rama aparte a la rama de la máscara. La evaluación comprende las gráficas de las curvas de pérdida durante el entrenamiento, y los resultados de las métricas usadas para la segmentación de instancias, además de las métricas de regresión sobre las estimaciones que realiza el modelo al conjunto de entrenamiento y validación.

Estos modelos fueron entrenados con 50 épocas de 500 pasos por cada época para el entrenamiento y 50 pasos para la validación, además de usar el MSE como función de pérdida para la estimación del peso del cuy.

ResNet50B

En esta parte se muestra el entrenamiento del modelo B (rama independiente de la máscara) usando la red troncal ResNet50.

En la Figura 4.8 se muestran las gráficas de la curva de pérdida del modelo Resnet50B durante el entrenamiento usando los pesos pre-entrenados del dataset COCO. Se evaluaron los conjunto de entrenamiento (*train*) y validación (*val*).

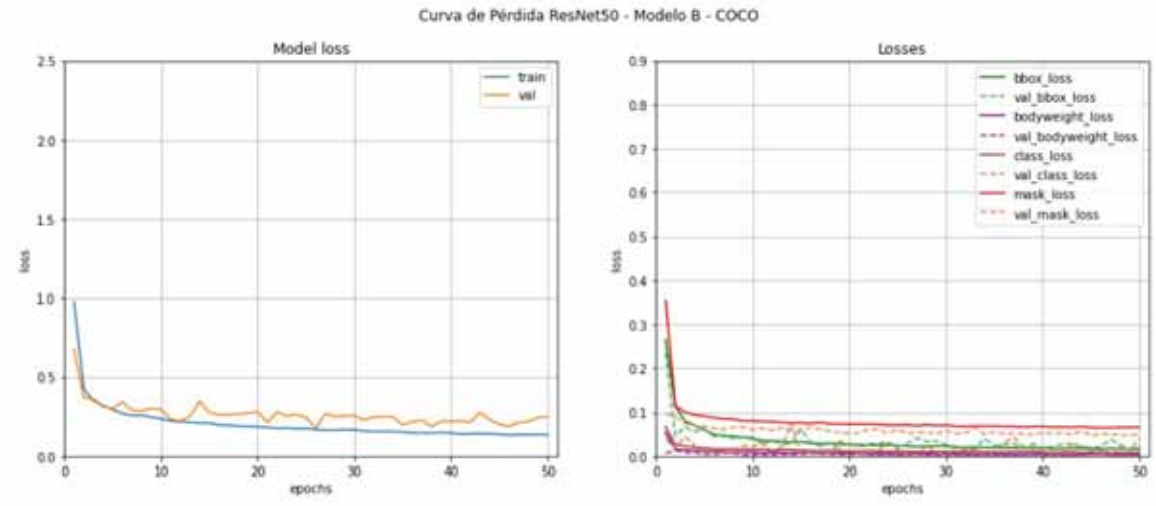


Figura 4.8: Gráfica de la curva de pérdida del modelo ResNet50B y COCO
Fuente: Elaboración propia

En la Figura 4.9 se muestran las gráficas de la curva de pérdida del modelo ResNet50B usando los pesos pre-entrenados del dataset ImageNet.

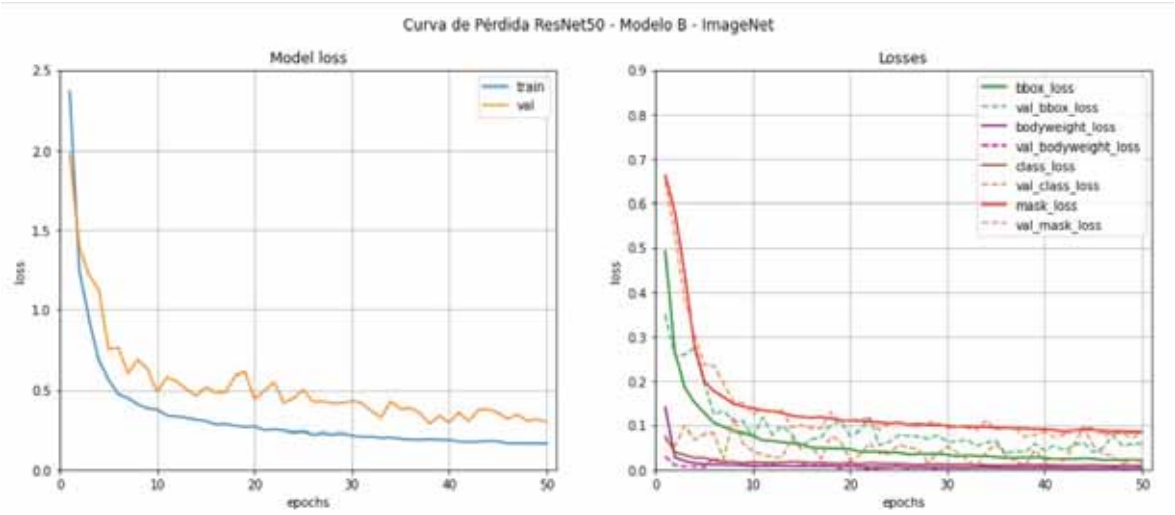


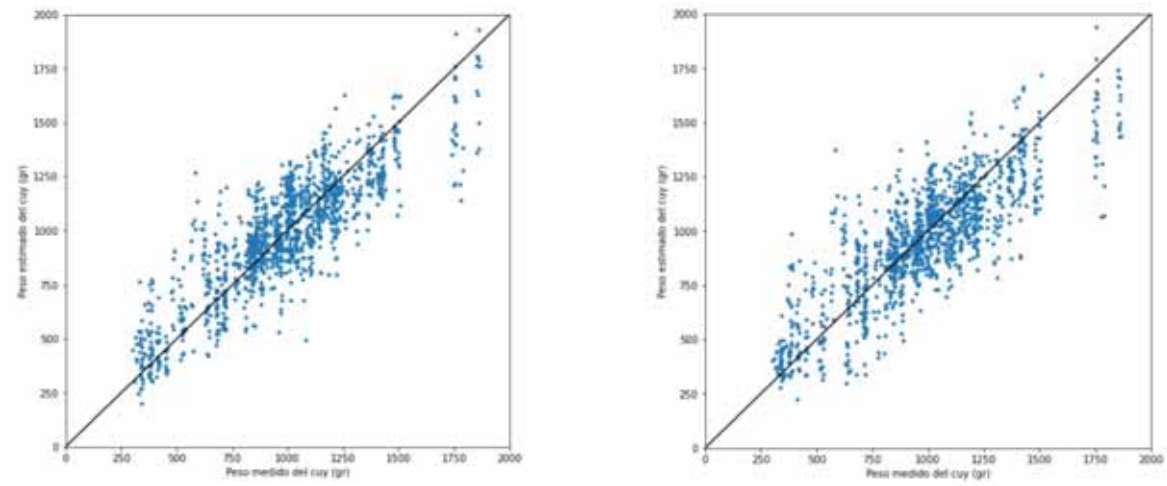
Figura 4.9: Gráfica de la curva de pérdida del modelo ResNet50B e ImageNet
Fuente: Elaboración propia

En la Tabla 4.3 se muestran los resultados obtenidos en la evaluación del Modelo ResNet50B, aplicados al conjunto de entrenamiento y de validación.

Resultados de la Inferencia - ResNet50B				
Métrica	**COCO**		**ImageNet**	
	train	val	train	val
mAP@50	1.00	1.00	1.00	1.00
mAP@75	1.00	1.00	1.00	1.00
mAP@[.50:.05:.95]	0.90	0.80	0.80	0.90
Conjunto Evaluado:	4975/4992	1249/1252	4985/4992	1252/1252
MAE	100.75	122.19	104.09	136.86
MSE	16671.16	25038.80	17866. 02	31086.08
RMSE	129.12	158.24	133.66	176.31
MAPE	11.22 %	14.17 %	11.46 %	15.45 %
R^2	80.83 %	74.44 %	79.45 %	68.22 %
Tiempo de entrenamiento	10h 24m 43s		9h 53m 48s	

Tabla 4.3: Evaluación de las métricas sobre ResNet50B
Fuente: Elaboración propia

En la Figura 4.10 se muestran las gráficas de dispersión de la inferencia realizada al conjunto de validación. Se compara el peso real frente el peso estimado por el modelo ResNet50B. La Figura 4.10a se muestra el resultado del entrenamiento usando los pesos pre-entrenados de COCO, mientras que 4.10b se muestra los resultados al usar los pesos de ImageNet.



(a) COCO

(b) ImageNet

Figura 4.10: Gráfica de dispersión de la inferencia ResNet50B
Fuente: Elaboración propia

ResNet101B

En esta parte se muestran los resultados del entrenamiento del modelo ResNet101B.

En la Figura 4.11 se muestran las gráficas de la curva de pérdida del modelo B durante el entrenamiento usando los pesos pre-entrenados con COCO. La gráfica nos muestra un buen ajuste entre la pérdida del conjunto de entrenamiento con la del conjunto de validación.

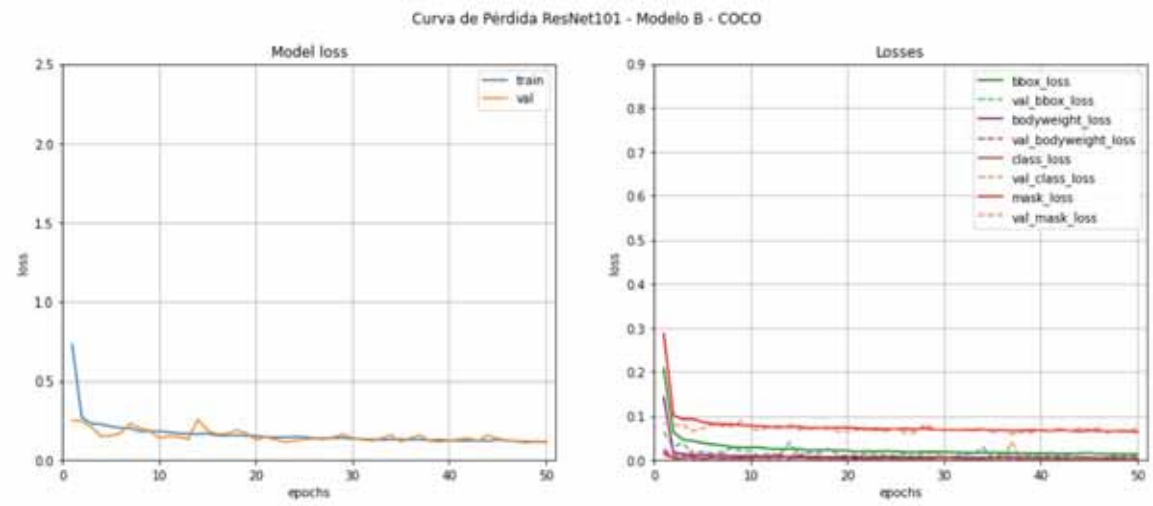


Figura 4.11: Gráfica de la curva de pérdida del modelo ResNet101B y COCO
Fuente: Elaboración propia

En la Figura 4.12 se muestran las gráficas de la curva de pérdida del modelo ResNet101B

con los pesos pre-entrenados de ImageNet. En este caso la gráfica muestra que el ajuste no es tan fino como el resultado obtenido con ResNet50.

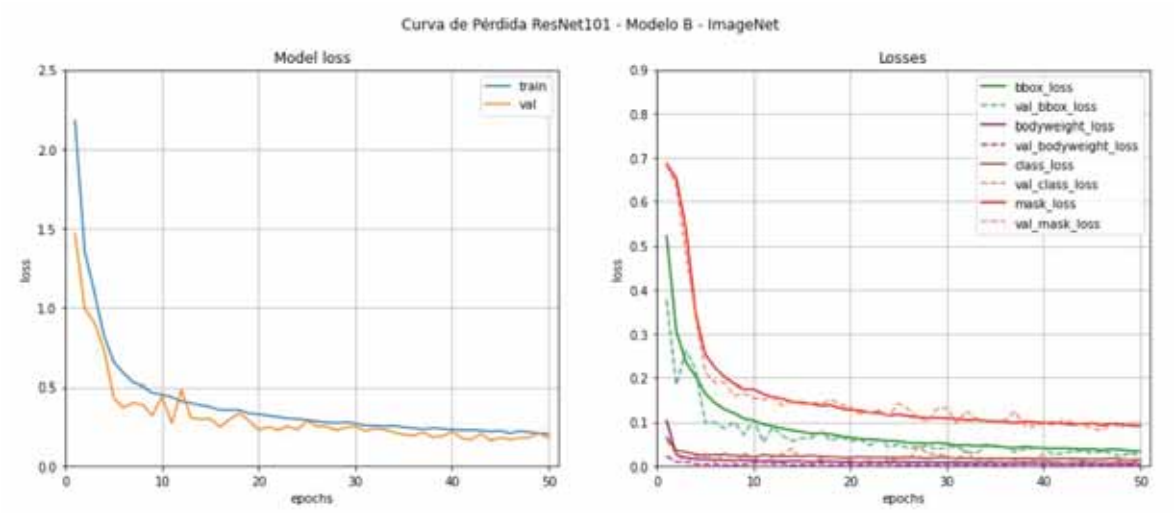


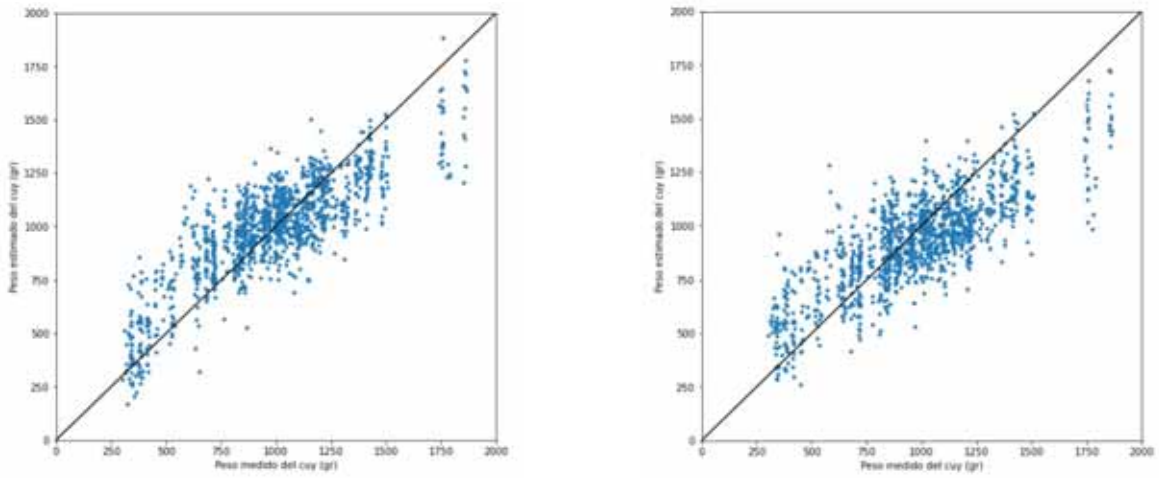
Figura 4.12: Gráfica de la curva de pérdida del modelo ResNet101B e ImageNet
Fuente: Elaboración propia

En la Tabla 4.4 se muestran los resultados obtenidos en la evaluación del modelo ResNet101B.

Resultados de la Inferencia - ResNet101B				
Métrica	**COCO**		**ImageNet**	
	train	val	train	val
mAP@50	1.00	1.00	1.00	1.00
mAP@75	1.00	1.00	1.00	1.00
mAP@[.50:.05:.95]	0.90	0.90	0.80	0.80
Conjunto Evaluado:	4983/4992	1252/1252	4954/4992	1251/1252
MAE	113.63	137.84	140.56	156.97
MSE	21010.86	30824.50	31696.99	38620.35
RMSE	144.95	175.57	178.04	196.52
MAPE	12.77 %	16.36 %	15.69 %	18.87 %
R^2	75.82 %	68.48 %	63.59 %	60.54 %
Tiempo de entrenamiento	11h 50m 29s		10h 42m 38s	

Tabla 4.4: Evaluación de las métricas sobre ResNet101B
Fuente: Elaboración propia

En la Figura 4.13 se muestran las gráficas de dispersión en la inferencia realizada al conjunto de validación del dataset. En la Figura 4.13a se muestra el resultado con los pesos pre-entrenados con COCO, mientras que 4.13b usa los pesos de ImageNet.



(a) COCO (b) ImageNet

Figura 4.13: Gráfica de dispersión de la inferencia ResNet101B
Fuente: Elaboración propia

Respecto a la evaluación de la precisión en la segmentación del cuy se observa resultados similares a la del modelo A, obteniendo valor de 1.0 cuando el umbral de IoU es 0.5 y 0.75. En cambio en la evaluación con 10 niveles los resultados son entre 0.8 y 0.9, lo que nos indica que la segmentación del cuy es realizada correctamente, identificándolos en todas las imágenes.

En la evaluación para la estimación del peso se observó un MAE mayor a 100 gramos, siendo este resultado menor que del modelo A. Pero, de forma similar, el modelo no estima correctamente cuando el peso del cuy es mayor a 1750 gramos.

En las gráficas de función de pérdida se muestra un mejor ajuste entre el conjunto de entrenamiento y el de validación, siendo menores a los resultados del modelo A. Pero, similarmente al modelo A, se obtiene mejores resultados usando ResNet50 que ResNet101.

El mejor resultado obtenido usando el modelo B fue de 74.44 % y un MAPE de 14.17 % usando el modelo ResNet50B y los pesos de COCO. Siendo, una leve mejora respecto al modelo ResNet50A, este modelo obtiene mejores resultados al evaluarse el conjunto de entrenamiento.

4.4. Refinamiento

El refinamiento busca mejorar los resultados de los modelos ya entrenados, ya que en un entrenamiento más largo se puede evidenciar si una red todavía puede aprender o no. En este caso la comparación se realiza de acuerdo con el coeficiente de correlación lineal (R^2), además de obtener un resultado promedio respecto al porcentaje de instancias que el dataset tiene, siendo 80 % para el entrenamiento y 20 % para la evaluación.

Basándonos en el resultado obtenido por el coeficiente de correlación (R^2), se comparó los resultados de todos las evaluaciones en la Tabla 4.5, donde se puede evidenciar que el mejor resultado promedio obtenido fue de 79.45 % y 77.20 % con el modelo ResNet50B. Mientras que los modelos ResNet50A y ResNet101B tienen resultados cercanos con un 74.95 % y 74.35 % respectivamente. Por lo que se eligió usar estos tres modelos para refinarlos y así obtener un mejor resultado en la correlación entre el peso estimado con el peso medido.

Modelo	Pesos	evaluación	resultado evaluación	resultado promedio
ResNet50A	COCO	train	75.59 %	74.95 %
		val	72.38 %	
	ImageNet	train	69.60 %	68.95 %
		val	67.17 %	
ResNet101A	COCO	train	69.01 %	68.48 %
		val	66.36 %	
	ImageNet	train	54.82 %	54.99 %
		val	55.68 %	
ResNet50B	COCO	train	80.83 %	79.55 %
		val	74.44 %	
	ImageNet	train	79.45 %	77.20 %
		val	68.22 %	
ResNet101B	COCO	train	75.82 %	74.35 %
		val	68.48 %	
	ImageNet	train	63.59 %	62.98 %
		val	60.54 %	

Tabla 4.5: Comparación de Resultados (R^2)
Fuente: Elaboración propia

El refinamiento se realizó entrenando los modelos con 50 épocas de 1000 pasos para el entrenamiento y 100 pasos para la validación, para que en cada iteración el modelo aprenda más del conjunto de entrenamiento y el de validación. También de configurar el *Mean Squared Error* (MSE) como función de pérdida para la estimación del peso del cuy. Los pesos utilizados son los resultados de los entrenamientos previos.

En la Tabla 4.6 se muestran los resultados del refinamiento de las redes: ResNet50A, ResNet50B y ResNet101B, donde se evidencia la mejora en la estimación del peso del cuy al evaluarlos en el conjunto de entrenamiento y validación.

Teniendo como mejor resultado un 80.09 % usando el modelo ResNet50A con un MAPE de 12.41 %. A pesar de que los modelos ResNet50B y ResNet101B se ajustaron mejor al conjunto de entrenamiento, su resultado con el conjunto de evaluación fue leve en comparación con el modelo ResNet50A.

Resultados de la Inferencia del Refinamiento						
Métrica	**ResNet50A**		**ResNet50B**		**ResNet101B**	
	train	val	train	val	train	val
mAP@50	1.00	1.00	1.00	1.00	1.00	1.00
mAP@75	1.00	1.00	1.00	1.00	1.00	1.00
mAP@ [.50:.05:.95]	0.90	0.90	0.90	0.80	0.90	0.90
Conjunto Evaluado:	4986/4992	1250/1252	4984/4992	1251/1252	4984/4992	1250/1252
MAE	75.05	108.64	65.65	117.16	70.59	125.92
MSE	9361.32	19500.64	7125.22	22943.97	8302.04	26832.32
RMSE	96.75	139.64	84.41	151.47	91.12	163.81
MAPE	8.12 %	12.41 %	7.37 %	13.83 %	7.70 %	14.23 %
R^2	89.23 %	80.09 %	91.80 %	76.55 %	90.45 %	72.56 %
Tiempo de entrena-miento	22h 55m 28s		21h 10m 39s		23h 33m 5s	

Tabla 4.6: Evaluación de las métricas sobre el refinamiento
Fuente: Elaboración propia

4.5. Comparación con otras investigaciones

Se realizó una comparación respecto a otras investigaciones que proponían estimar el peso del animal basado en imagen. En la Tabla 4.7 se muestra esta comparación respecto a los antecedentes de la investigación, donde se observa un mejor resultado del MAPE de 2.84 % con la estimación del cerdo realizada por Buayai *et al.* (2019), y un mejor resultado de correlación de 96 % respecto a la estimación del cerdo realizada por Jensen *et al.* (2018).

Pero, se debe resaltar que estas investigaciones parten de la imagen ya segmentada del cerdo. A comparación de nuestro modelo que parte de la imagen capturada y realiza la segmentación y estimación en un mismo modelo de arquitectura.

Autor	Animal de estudio	MAPE	R^2
Buayai *et al.* (2019)	Cerdo	2.84 %	87 %
Jun *et al.* (2018)	Cerdo	-	79 %
Jensen *et al.* (2018)	Cerdo	-	96 %
Konovalov *et al.* (2019)	Pescado	4.28 %	-
Modelo Propio	Cuy	12.41 %	80 %

Tabla 4.7: Comparación con resultados de antecedentes
Fuente: Elaboración propia

4.6. Detalles técnicos

Para el desarrollo del proyecto de investigación se hizo uso del siguiente hardware y software, los cuales se detallan a continuación.

4.6.1. Recursos hardware

- **Notebook**: Asus ROG Zephyrus S GX531GS. Procesador Intel Core i7-8750H (6 x 2.2 GHz, Coffee Lake-H), 16GB RAM DDR4, 512GB SSD, NVIDIA GeForce GTX 1070 Max-Q (Tamaño de memoria 8GB, 2018 núcleos)
- **Notebook**: Lenovo Legion Y7000. Procesador Intel Core i7-8750H (6 x 2.2 GHz, Coffee Lake-H), 16GB RAM DDR4, 256GB SSD + 1TB HDD, NVIDIA GeForce GTX 1060 (Tamaño de memoria 6GB, 1280 núcleos)
- **Teléfono Celular**: Xiaomi Redmi Note 7. Cámara principal de 48 megapíxeles y 5 megapíxeles, f/1.8
- **Balanza**: JBC Modelo DY130, Capacidad 40kg, Precisión 1-5gr.

4.6.2. Recursos software

- **Sistema Operativo**: Windows 10 x64.
- **Lenguaje de programación**: Python 3.7
- **Sistema de gestión de paquetes**: Anaconda 2.0.4
- **Librerías**: Numpy (1.16.4), scipy (1.2.2), Pillow, cython, matplotlib, sckit-image, tensorflow (1.14.0), tensorflow-gpu (1.14.0), keras (2.1.5), opencv-python, h5py (2.10.0), imgaug, pycocotools, cuda (7.0), cudnn
- **Sistema de control de versiones**: GitHub
- **Software de etiquetado**: Labelbox
- **Framework para el API**: Flask
- **Lenguaje de programación para la Aplicación Móvil**: Javascript
- **Framework para la Aplicación Móvil**: Expo (42.0.1)
- **Librerías para la Aplicación Móvil**: react (16.13.1), expo-camera (11.2.2)
- **Aplicación para la depuración de la Aplicación Móvil**: Expo Go (Android)

Conclusiones

- Conclusiones en base al objetivo general:

  Se concluye el desarrollo de un sistema de visión artificial usando el modelo Mask R-CNN para la estimación del peso corporal del cuy. El resultado del modelo fue con el conjunto de validación al obtener la correlación R^2=80.09 % y un MAPE de 12.41 % usando el modelo ResNet50A como red troncal, en cambio con el conjunto de entrenamiento la correlación se logró con la red troncal ResNet50B obteniendo la correlación de R^2=91.80 % y un MAPE de 7.37 %. Los resultados evidencian que se puede realizar la estimación del peso del cuy usando como fuente la imagen del cuy en un sistema de visión artificial con una arquitectura CNN como Mask-RCNN.

- Conclusiones en base al primer objetivo específico:

  Se logró construir un dataset de 1561 imágenes con los pesos de 225 cuyes, además de las máscaras etiquetadas de cada cuy. Con el aumento de datos se incrementó este dataset a 6244 imágenes de los cuyes manteniendo el mismo aspecto que el dataset original.

- Conclusiones en base al segundo objetivo específico:

  Se seleccionó el modelo de Mask R-CNN diseñado para la segmentación de instancias, modificándolo en dos versiones: la primera con una rama unida a la rama de la máscara, y la segunda con una rama independiente aplicada en la estimación del peso corporal del cuy, además de agregar la función de pérdida MSE para optimizar el entrenamiento del modelo en la estimación del peso del cuy.

- Conclusiones en base al tercer objetivo específico:

  Se realizaron los entrenamientos del modelo Mask R-CNN en las dos versiones adaptadas. El entrenamiento se empezó con el uso de los pesos pre-entrenados con los datasets COCO e ImageNet y seleccionando en cada entrenamiento las redes troncales ResNet50 y ResNet101. Los primeros entrenamientos duraron aproximadamente 10 a 12 horas cada uno con 50 épocas de 500 pasos cada una, y los entrenamiento finales duraron entre 23 a 24 horas cada uno con 50 épocas de 1000 pasos cada una.

- Conclusiones en base al cuarto objetivo específico:

Se realizó una evaluación de los modelos entrenados usando métricas de regresión en el modo de inferencia para así determinar su precisión en la estimación del peso corporal del cuy, la evaluación se desarrolló usando tanto el dataset de validación y el dataset de entrenamiento. Llegando a obtenerse la correlación de $R^2 = 80{,}09$ al evaluar la inferencia del modelo de ResNet50A con los datos de validación.

- Conclusiones en base al quinto objetivo específico:

  Se implementó un prototipo de aplicación móvil que se conecta a un servidor API, que sirve como herramienta de visión artificial que permite la estimación del peso corporal del cuy usando un dispositivo móvil.

Recomendaciones

- Se recomienda extender el trabajo de investigación adaptando otras redes troncales dentro del modelo desarrollado, para evaluar su desempeño en la estimación del peso del cuy.
- Se recomienda incrementar el tamaño de dataset, además de evaluar estadísticamente la población de cuyes necesaria para que los datos sean más uniformes. También se podría implementar un nuevo dataset para la aplicación de la estimación del peso en otros animales de granja similares como los cerdos, las vacas, las ovejas, etc.
- Se recomendaría incrementar el tiempo de entrenamiento a más épocas y más pasos por época, ya que esto puede mejorar la precisión del modelo en la estimación del peso animal.
- Se propone evaluar el modelo desarrollado en la captura de varios cuyes dentro de una imagen, además de evaluar si una captura en otra posición que no sea de arriba hacia abajo afecta en la estimación del cuy.
- Se propone realizar un estudio sobre la relevancia de aspectos como la iluminación, el ruido, y las características del hardware puedan afectar en la estimación del peso.
- Se recomienda desarrollar un sistema de gestión de la crianza del cuy para poder almacenar la información tanto del peso del cuy como de otras características relevantes que permitan a los investigadores como productores tomar mejores decisiones en la producción de cuyes.

Bibliografía

Abdulla, Waleed. 2017. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN.

Agropecuaria, Biblioteca. 1981. *Cuy Alimento Popular*.

AI, Fritz. 2021. *Object Detection Guide*. https://www.fritz.ai/object-detection/. Recuperado: 24 de agosto de 2021.

Andina. 2017. *Carne de cuy: estas son las bondades nutricionales de este alimento ancestral*. https://andina.pe/agencia/noticia-carne-cuy-estas-son-las-bondades-nutricionales-este-alimento-ancestral-756728.aspx. Recuperado el 20 de julio de 2021.

Ataucusi, Saturnino. 2015. Manejo técnico de la crianza de cuyes en la sierra del Perú. *Callao, Perú: Caritas del Perú*.

Barznji, Hemn Mela Karim. 2020. Transfer Learning as New Field in Machine Learning. 11.

Buayai, Prawit, Piewthongngam, Kullapapruk, Leung, Carson K, & Saikaew, Kanda Runapongsa. 2019. Semi-automatic pig weight estimation using digital image analysis. *Transactions of the ASABE*, 0.

Buitrago, Brayan. 2020. *Machine Learning - Modelos de Regresión I*. https://medium.com/iwannabedatadriven/machine-learning-modelos-de-regresi%C3%B3n-i-d293ae235e9a. Recuperado: 23 de agosto de 2021.

Cantero Lorenzo, Javier. 2018. Máquinas de aprendizaje y aplicaciones.

Chauca, Lilia. 1997. *Producción de cuyes (Cavia porcellus)*. Vol. 138. Food & Agriculture Org.

Cornieles, Pilar. 2019. *Entendiendo las redes neuronales: De la neurona a RNN, CNN y Deep Learning*. https://ia-latam.com/2019/02/06/entendiendo-las-redes-neuronales-de-la-neurona-a-rnn-cnn-y-deep-learning/. Recuperado: 25 de agosto de 2021.

Do, Synho, Song, Kyoung, & Chung, Joo. 2020. Basics of Deep Learning: A Radiologist's Guide to Understanding Published Radiology Articles on Deep Learning. *Korean journal of radiology*, **21**(01), 33–41.

Dwivedi, Priya. 2019. *Understanding and Coding a ResNet in Keras.* https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33.

Frogames. 2020. *La guía definitiva de las redes neuronales convolucionales.* https://frogames.es/la-guia-definitiva-de-las-redes-neuronales-convolucionales/. Recuperado: 18 de enero de 2021.

Gibson, James J. 2014. *The ecological approach to visual perception: classic edition.* Psychology Press.

Girshick, Ross. 2015. Fast r-cnn. *Pages 1440–1448 of: Proceedings of the IEEE international conference on computer vision.*

Girshick, Ross, Donahue, Jeff, Darrell, Trevor, & Malik, Jitendra. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. *Pages 580–587 of: Proceedings of the IEEE conference on computer vision and pattern recognition.*

González, Ana, Martínez, F, Pernía, A, Alba, F, Castejón, M, Ordieres, J, & Vergara, E. 2006. Técnicas y algoritmos básicos de visión artificial. *La Rioja: Universidad de la Rioja.*

Goodfellow, Ian, Bengio, Yoshua, & Courville, Aaron. 2018. *Deep learning.* Springer.

Guerra, César. 2009. Manual técnico de crianza de cuyes. Proyecto: Potenciando capacidades para el desarrollo sostenible de Chetilla y Magdalena–Cajamarca. 1–24.

He, Kaiming, Gkioxari, Georgia, Dollár, Piotr, & Girshick, Ross. 2017. Mask r-cnn. *Pages 2961–2969 of: Proceedings of the IEEE international conference on computer vision.*

Hernández-Sampieri, Roberto, & Torres, Christian Paulina Mendoza. 2018. *Metodología de la investigación.* Vol. 4. McGraw-Hill Interamericana Méxicˆ eD. F DF.

Hui, Jonatan. 2018. *mAP (mean Average Precision) for Object Detection.* https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173. Recuperado: 25 de agosto de 2021.

Jensen, D, Dominiak, K, & Pedersen, L. 2018. Automatic estimation of slaughter pig live weight using convolutional neural networks. *Pages 1–4 of: Proc. 2nd Int. Conf. Agro BigData Decis. Support Syst. Agricult.*

Jun, Kyungkoo, Kim, Si Jung, & Ji, Hyun Wook. 2018. Estimating pig weights from images without constraint on posture and illumination. *Computers and Electronics in Agriculture*, **153**, 169–176.

K, Sambasivarao. 2019. *Nom-maximum Supression (NMS).* `https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c`. Recuperado: 25 de agosto de 2021.

Konovalov, Dmitry A, Saleh, Alzayat, Efremova, Dina B, Domingos, Jose A, & Jerry, Dean R. 2019. Automatic weight estimation of harvested fish from images. *Pages 1–7 of: 2019 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE.

Labelbox. 2022. *Labelbox Ontologies*. `https://docs.labelbox.com/docs/labelbox-ontology`. Recuperado: 25 de enero de 2022.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, & Zitnick, C Lawrence. 2014. Microsoft coco: Common objects in context. *Pages 740–755 of: European conference on computer vision*. Springer.

Lin, Tsung-Yi, Dollár, Piotr, Girshick, Ross, He, Kaiming, Hariharan, Bharath, & Belongie, Serge. 2017. Feature pyramid networks for object detection. *Pages 2117–2125 of: Proceedings of the IEEE conference on computer vision and pattern recognition*.

Machaca, Abraham. 2020. *Producción de Cuyes*.

Marr, D. 1982. *Vision: A Computational Approach (San Fr*.

Marsland, Stephen. 2015. *Machine Learning An Algorithmic Perspective*. CRC Press.

Mejía, Elías. 2005. Metodología de la investigación científica. *Lima: Universidad Nacional Mayor de San Marcos*.

Mitchell, Tom M, *et al.* 1997. Machine learning.

Moolayil, Jojo. 2019. *Learn Keras for Deep Neural Networks*. Springer.

Paul, Kris. 2018. *Keypoints of humanpose with Mask R-CNN*. `https://github.com/chrispolo/Keypoints-of-humanpose-with-Mask-R-CNN`.

Pusiol, Pablo Daniel. 2014. *Redes convolucionales en comprensión de escenas*. B.S. thesis.

Qiao, Yongliang, Truman, Matthew, & Sukkarieh, Salah. 2019. Cattle segmentation and contour extraction based on Mask R-CNN for precision livestock farming. *Computers and electronics in agriculture*, **165**, 104958.

Redolfi, Javier. 2018 (05). *Aplicación en agricultura de precisión de esquemas actuales de reconocimiento visual*. Ph.D. thesis.

Ren, Shaoqing, He, Kaiming, Girshick, Ross, & Sun, Jian. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, **28**, 91–99.

Santos, V. 2007. Importancia del cuy y su competitividad en el mercado. *Arch. Latinoamérica de Producción Animal*, **15**(1), 216–217.

Shah, Tarang. 2018. *Measuring Object Detection models mAP What is Mean Average Precision?* https://tarangshah.com/blog/2018-01-27/what-is-map-understanding-the-statistic-of-choice-for-comparing-object-detection-models/. Recuperado: 25 de agosto de 2021.

Shorten, Connor, & Khoshgoftaar, Taghi M. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, **6**(1), 1–48.

Sucar, L Enrique, & Gómez, Giovani. 2011. Visión computacional. *Instituto Nacional de Astrofísica, Óptica y Electrónica. México*.

Szeliski, Richard. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media.

Torrey, Lisa, & Shavlik, Jude. 2010. Transfer learning. *Pages 242–264 of: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global.

Van Dyk, David A, & Meng, Xiao-Li. 2001. The art of data augmentation. *Journal of Computational and Graphical Statistics*, **10**(1), 1–50.

Wang, Y, Yang, W, Winter, P, & Walker, L. 2008. Walk-through weighing of pigs using machine vision and an artificial neural network. *Biosystems Engineering*, **100**(1), 117–125.

Wilding-McBride, Daryl, & Pun, Dil Kumar. 2018. *MaskRCNN utils*. `https://github.com/DiUS/MaskRCNN-utils`.

Yosinski, Jason, Clune, Jeff, Bengio, Yoshua, & Lipson, Hod. 2014. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*.

Zhao, Zhong-Qiu, Zheng, Peng, Xu, Shou-tao, & Wu, Xindong. 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, **30**(11), 3212–3232.