UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Clasificación de la frescura del pescado utilizando una red neuronal convolucional

Tesis para obtener el Título Profesional de Ingeniero de Sistemas

Autores:

Edwin Wilson Mamani Flores

Roger Pedraza Huisa

Asesor:

Mg. Ferdinand Edgardo Pineda Ancco

Juliaca, diciembre de 2023

DECLARACIÓN JURADA DE ORIGINALIDAD DE TESIS

Yo Ferdinand Edgardo Pineda Ancco, docente de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que la presente investigación titulada: **"CLASIFICACIÓN DE LA FRESCURA DEL PESCADO UTILIZANDO UNA RED NEURONAL CONVOLUCIONAL"** de los autores Edwin Wilson Mamani Flores y Roger Pedraza Huisa tiene un índice de similitud de **10 %** verificable en el informe del programa Turnitin, y fue realizada en la Universidad Peruana Unión bajo mi dirección.

En tal sentido asumo la responsabilidad que corresponde ante cualquier falsedad u omisión de los documentos como de la información aportada, firmo la presente declaración en la ciudad de Juliaca, a los 04 días del mes de diciembre del año 2023

Ferdinand Edgardo Pineda Ancco

049

ACTA DE SUSTENTACIÓN DE TESIS

En Puno, Juliaca, Villa Chullunquiani, a 04 día(s) del mes de diciembre del año 2023, siendo las 15:00 horas, se reunieron los miembros del jurado en la Universidad Peruana Unión Campus Juliaca, bajo la dirección del (de la) presidente(a): Mgr. Benazir Francis Herrera Yucra el (la) secretario(a): Ing. Jenson Daniel Ghambi Aguilar y los demás miembros: Mg. Abel Angel Zullon Macahuya Mgr. Abel Ano y el (la) asesor(a) Mgr. Ferdinand Edgardo Pineda Ancco con el propósito de administrar el acto académico de sustentación de la tesis titulado: "Clasificación de la frescura del pescado utilizando una red neuronal convolucional" del(los) bachiller/es: a) Edwin Wilson Mamani Flores b) Royer Pedraza Huisa c) conducente a la obtención del título profesional de: Ingeniero de Sistemas

(Denominación del Título Profesional)

El Presidente inició el acto académico de sustentación invitando al (a la) / a (los) (las) candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por al (a la) / a (los) (las) candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Bachiller (a): Edwin Wilson Mamani Flores

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	18	A-	Muy Bueno	Sobresaliente

Bachiller (b): Royer Pedraza Huisa

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	18	A-	Muy Bueno	Sobresaliente

Bachiller (c):

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al (a la) / a (los) (las) candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente/a

Secretario/a

Asesor/a

Miembro

Miembro

Bachiller (a)

Bachiller (b)

Bachiller (c)

AGRADECIMIENTOS

Por razones significativas, agradezco primero a Dios, quien me brindó la oportunidad de la vida y por mi éxito actual. En segundo lugar, a mis padres, quienes me apoyaron en todo lo esencial. Agradecimientos especiales al Ingeniero Ferdinand Pineda por su orientación y ayuda durante esta investigación. También expreso mi gratitud a la Universidad Peruana Unión - Campus Juliaca, que a través de su cuerpo docente fomenta el desarrollo de profesionales exitosos. Roger Pedraza.

Quiero aprovechar este momento para expresar mi más profundo agradecimiento a todos aquellos que han sido parte de mi vida. A mis queridos hermanos y amigos de la Iglesia Balmaceda en Antofagasta, Chile, y a la Iglesia 'Las Américas' en Juliaca, Perú, por su apoyo incondicional hacia mi familia. A los respetados profesores de la Escuela de Ingeniería de Sistemas, por su dedicación y conocimiento compartido, que han contribuido de manera incomparable a mi educación. A mis compañeros de clase, por las alegrías, los desafíos que hemos superado juntos, por la sincera amistad que ha hecho cada paso memorable. A mis padres, cuyo amor y orientación han sido mi fortaleza en todo momento. Y finalmente, a Dios, por todas Sus bendiciones y constante guía en mi camino. Con humildad y profundo afecto, extiendo mi gratitud por haber sido parte de mi vida. Wilson Mamani

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

Clasificación de la frescura del pescado utilizando una red neuronal convolucional

RESUMEN

En el trabajo se analizó la frescura de la trucha (Oncorhynchus mykiss), siendo este aspecto muy importante para determinar su calidad. El objetivo, es proponer un modelo computacional basado en una CNN para clasificar la frescura de la trucha en función a los cambios de color de sus ojos y branquias, Para ello se creó un dataset de imágenes con las truchas adquiridas. Para obtener los resultados se realizaron 3 experimentos, el primero; con 2 clases (días 1 y 9), el segundo con 3 clases (días 1, 5 y 9) y el tercero con 5 clases (días 1, 3, 5, 7 y 9), todos los experimentos se ejecutaron en Google colab. Los resultados se validaron con una matriz de confusión curva ROC. Los mejores resultados dieron el modelo ResNeXt5032x4d, con 2 clases obtuvo un accuracy de 0.9833, con 3 clases un accuracy de 0.9222 y con 5 clases un accuracy de 0.8800.

Palabras clave: CNN, Transferencia de aprendizaje, precisión y Oncorhynchus mykiss.

Classification of fish freshness using a convolutional neural network

ABSTRACT

At work, the freshness of rainbow trout (Oncorhynchus mykiss) was analyzed, this aspect being very important to determine its quality. The objective is to propose a computational model based on a CNN to classify the freshness of the trout based on changes in the color of their eyes and gills. For this purpose, a dataset of images with acquired trout was created. To obtain the results, three experiments were conducted: the first with 2 classes (days 1 and 9), the second with 3 classes (days 1, 5, and 9), and the third with 5 classes (days 1, 3, 5, 7, and 9). All experiments were carried out in Google Colab. The results were validated using a confusion matrix ROC curve. The best results were obtained by the ResNeXt5032x4d model, with 2 classes achieving an accuracy of 0.9833, with 3 classes an accuracy of 0.9222, and with 5 classes an accuracy of 0.8800.

Keywords: CNN, Transfer learning, accuracy and Oncorhynchus mykiss.

1. INTRODUCCIÓN

La Organización Mundial de la Salud (OMS) reitera la recomendación de la ingesta regular de pescados y mariscos dos o tres veces por semana, dado el alto valor nutricional que proporcionan las especies marinas y acuícolas (*A healthy diet sustainably produced*, s. f.). La acuicultura en Perú ha experimentado una disminución en la producción en los últimos años, alcanzado el 2022 la cifra de 140 931 toneladas métricas (TM), -6,56% en comparación al 2021. A pesar de esta tendencia negativa, la trucha (Oncorhynchus mykiss) se destaca como una de las principales especies acuícolas, registrando un incremento en su producción a 61,572.8 TM, lo que equivale al 43.7% del total de la producción acuícola (*Anuario Estadístico Pesquero y Acuícola 2022*, s. f.). El pescado es un alimento de excelente valor nutricional, que proporciona proteínas de alta calidad, ricas en aminoácidos esenciales y una amplia variedad de minerales, incluidos el fósforo, magnesio, hierro, zinc y yodo (Ariño et al., 2013). Siendo la frescura del pescado uno de los principales atributos de calidad para su comercialización y consumo (Barat et al., 2008), permitiéndole conservar su valor nutricional y por ende un producto base de la canasta de alimentos del hogar (Mohammadi Lalabadi et al., 2020). Los procesos microbianos y químicos; la formación de aminas hipoxantinas y biogénicas así los cambios en la composición de proteínas y lípidos son las principales causas del deterioro del pescado fresco (Dowlati et al., 2013). Por lo cual su frescura puede tener un impacto negativo por los procesos de manipulación y almacenamiento, afectando la calidad del producto. Desde la extracción hasta los consumidores finales, es importante tener el control del tiempo de retención y la temperatura de almacenamiento del pescado siendo indicadores clave para mantener la frescura del pescado (Dutta et al., 2016).

Existen diferentes estudios en la evaluación de la frescura del pescado (Dubey & Jalal, 2014), (Taheri-Garavand et al., 2019), (Dowlati et al., 2012), (Tappi et al., 2017),

(Rocculi et al., 2019), (Macagnano et al., 2005) nos dan a conocer que se realizaron esfuerzos considerables para emplear la visión artificial y el procesamiento de imágenes en la evaluación de parámetros de pescado; tales como imágenes de peces enteros y segmentación de las branquias. Como resultados de estudios anteriores indicaron que se puede usar los dos atributos (ojos y branquias) para determinar la frescura de la trucha (Oncorhynchus mykiss). Por tanto, el presente estudio consideró los atributos (ojos y branquias), para analizar los cambios en los parámetros del color de los ojos y las branquias de la trucha, con un dataset con imágenes propias. Planteando como objetivo el desarrollar un modelo computacional basado en red neuronal convolucional (CNN) para clasificar la frescura de la trucha en función de los cambios de color de sus ojos y branquias. CNN (Convolucional Neural Network) es un método de clasificación de imágenes eficiente que utiliza capas complejas, agrupadas y completamente conectadas para el proceso de aprendizaje, esta red es un tipo de red neuronal multicapa que consta de neuronas con pesos y sesgos que se pueden entrenar (Liang et al., 2017). De esta forma beneficiar con este estudio a los consumidores (clientes mayoristas y clientes minoristas).

2. METODOLOGÍA

En el estudio se aplicó una metodología de cuatro pasos, a fin de brindar una solución para determinar la frescura del pescado, en este caso la trucha arcoíris, brindando una alternativa no invasiva al consumidor, para que este pueda conocer el estado de la frescura del pescado, determinando la cantidad de días que tiene de fresco, por tanto, se va a desarrollar un modelo computacional basado en red neuronal convolucional para clasificar la frescura de la trucha en función de los cambios de color de sus ojos y branquias. La metodología se puede apreciar en la figura 1:

Figura 1

Metodología del proyecto

Fuente: *Elaboración Propia*

2.1. Dataset

Se ha creado una dataset utilizando un total de 250 peces de trucha arcoíris (Oncorhynchus mykiss) que se compraron en Desaguadero Puno - Perú. Las muestras se redirigieron inmediatamente al laboratorio después de la compra. Las muestras de las truchas se colocaron en una placa de poliestireno expandido (EPS) para no maltratar las muestras y controlar su temperatura para un mantenimiento óptimo. Se utilizó bolsas de hielo y un refrigerador para enfriar las muestras, como se puede apreciar en la figura 2:

Figura 2

Conservación de la trucha arcoíris

Fuente: *Elaboración Propia*

Los peces estaban posicionados en 5 capas desde la parte inferior de la refrigeradora hasta la parte superior, y las bolsas de hielo se ubicaron entre los costados del refrigerador. Con estas modificaciones la temperatura llegó a todas las localizaciones. Se consideraron 5 clases de frescura sometiendo las muestras a las duraciones de almacenamiento en la refrigeradora con hielo en los días 1, 3, 5, 7 y 9.

En las primeras muestras del día 1, las imágenes se tomaron en un ángulo de 75° donde se podía visualizar las branquias de la trucha. Para obtener las imágenes de los ojos, se hizo del mismo ángulo que las branquias. Las imágenes de las branquias y ojos se obtuvieron de ambos lados de la trucha, así mismo se tomaron imágenes de la trucha entera para trabajos posteriores.

La herramienta utilizada fue un celular ZTE Blade V10 Vita con una cámara de 16 pixeles junto con una maqueta de poliestireno expandido (EPS) para que las muestras estén bien posicionadas al momento de tomar las imágenes, se clasificaron las muestras de las branquias, muestras de los ojos y muestras del pez entero.

Se utilizó como repositorio Google Drive para almacenar los archivos ordenados en carpetas, día 1 en 3 carpetas. Para los días 3, 5, 7, 9 se repitió el mismo procedimiento tomando muestras de las branquias de las truchas, los ojos y pez entero; como resultado final se obtuvo 2500 muestras en imágenes en formato JPG. Las imágenes tomadas en los días mencionados, tanto de los ojos y branquias, se puede apreciar en las siguientes figuras 3 y 4:

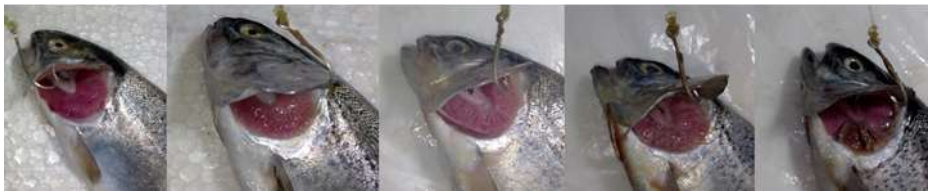
Figura 3

Trucha arcoíris-Ojos en los días 1,3,5,7 y 9

Fuente: *Elaboración Propia*

Figura 4

Trucha arcoíris-Branquias en los días 1,3,5,7 y 9



Fuente: *Elaboración Propia*

3. EXPERIMENTAL

Para los experimentos que se hizo en el estudio, se utilizó Google Colab Pro para ejecutar los notebooks y obtener los resultados. Así mismo se enlazó con Google Drive con Colab para obtener las imágenes.

Google Colab, también conocido como Colaboratory, es un servicio en la nube proporcionado de forma gratuita y de paga por Google, como el uso de sus recursos de las GPUs. Se basa en el entorno de Jupyter Notebook y está diseñado para la formación y la investigación en aprendizaje automático. Así mismo se utilizaron las siguientes librerías que están bajo Python:

- Scikit-learn, OpenCV
- PyTorch

3.1. Preprocesamiento

Las transformaciones que se utilizaron para el preprocesamiento como RandomResizedCrop, RandomRotation, RandomHorizontalFlip, ayudan a mejorar y aumentar la diversidad de los datos de entrenamiento como la capacidad de generalización de los modelos de aprendizaje automático en tareas de visión por computadora.

- **RandomResizedCrop:** realiza un recorte aleatorio de una imagen con su posterior redimensionamiento de (size = 500, scale = (0.8, 1.0)).

- **RandomRotation:** aplica rotaciones aleatorias a las imágenes, para este caso usamos una rotación de 15 grados.
- **RandomHorizontalFlip:** realiza volteos horizontales aleatorios en las imágenes.
- **CenterCrop:** recorta una porción central de la imagen original con un tamaño predefinido (224), es útil para enfocarse en la región de interés.
- **ToTensor:** convierte una imagen en un tensor numérico, lo que permite su procesamiento eficiente en modelos de aprendizaje automático.
- **Normalize:** Se aplica a un tensor o conjunto de datos con el objetivo de ajustar sus valores a una distribución específica, como la distribución normal ([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]).

El resultado del preprocesamiento de las imágenes originales se aprecia en la figura 5.

Figura 5

Resultado del preprocesamiento de las imágenes, un "antes y después"

Fuente: *Elaboración propia*

3.2. Modelos

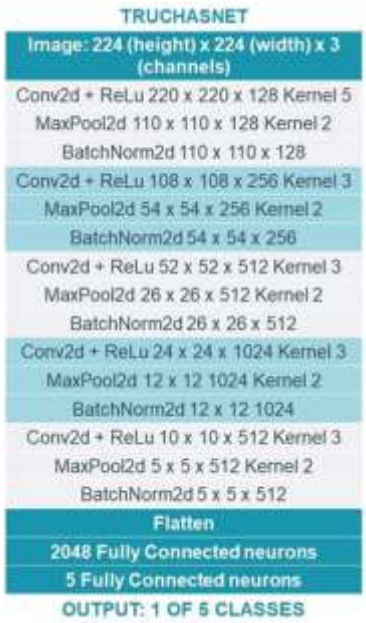
3.2.1. Modelo "TruchasNet"

La arquitectura de TruchasNet, está compuesta por 5 capas convolucionales, a cada capa, se aplicó una función de activación en forma de función ReLU que convierte las capas en capas no lineales, se aplicó MaxPool2d para implementar y reducir la capa de entrada a

un valor de salida pequeño y BatchNorm2d para acelerar el entrenamiento y utilizar tasas de aprendizaje más altas, facilitando el aprendizaje. Esta arquitectura también usa Flatten (aplana un tensor Ndimensional a un tensor1D) y 2 capas completamente conectadas. La arquitectura de TruchasNet se observa en la figura 6.

Figura 6

Arquitectura del modelo TruchasNet



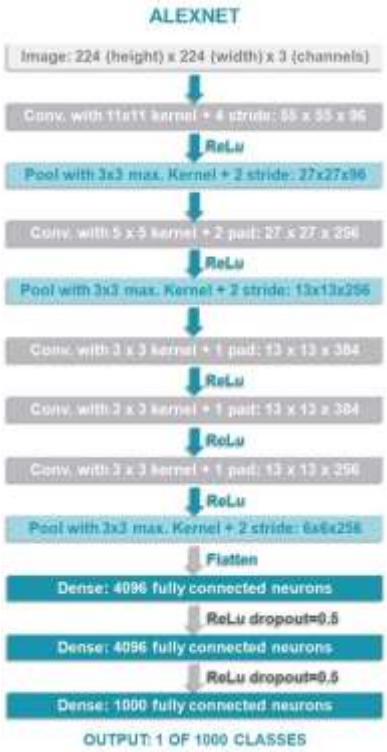
Fuente: *Elaboración Propia*

3.2.2. Modelo AlexNet

Es una arquitectura preentrenada que tiene ocho capas, de las cuales son básicamente 5 capas convolucionales con varios tamaños de kernel y varios filtros y 3 capas completamente conectadas. Una de las principales razones por las que AlexNet puede funcionar bien en el conjunto de datos de imágenes, son los diferentes tamaños de kernel que posee en su arquitectura, siendo pionero en el uso de GPUs para acelerar el entrenamiento de redes neuronales y popularizó técnicas como dropout (Krizhevsky et al., 2017). La arquitectura de AlexNet se observa en la figura 7.

Figura 7

Arquitectura del modelo AlexNet



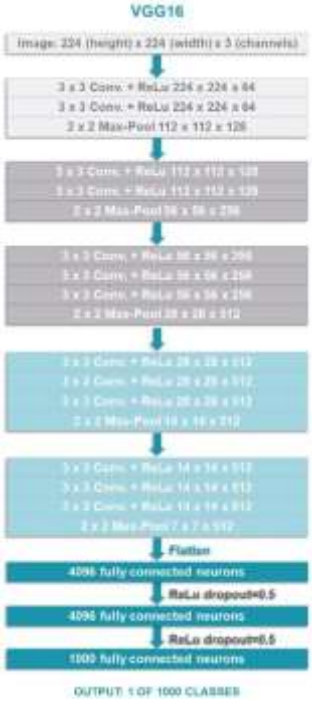
Fuente: *Elaboración Propia*

3.2.3. Modelo Vgg16

Es una arquitectura de red profunda preentrenada, fue desarrollada por Simonyan y Zisserman en el concurso ILSVRC 2014. Consta de 16 capas, incluyendo 13 capas convolucionales y 3 capas completamente conectadas, es conocido por su estructura simple y profunda que utiliza filtros de convolución de tamaño pequeño (3x3) y capas de agrupamiento para extraer características visuales de las imágenes, las capas completamente conectadas se utilizan para la clasificación final (Simonyan & Zisserman, 2015). La arquitectura de Vgg16 se observa en la figura 8.

Figura 8

Arquitectura del modelo Vgg16



Fuente: *Elaboración Propia*

3.2.4. Modelo ResNeXt-50 (32x4d)

Esta red se denomina ResNeXt-50 (32×4d) por su simplicidad, es una red neuronal profunda que consta de 50 capas y utiliza bloques con 32 subramas y transformaciones convolucionales 4D. Esta arquitectura permite aprender representaciones más complejas y puede ser utilizada para tareas de clasificación de imágenes y detección de objetos (Xie et al., 2017). La arquitectura de ResNeXt-50 (32x4d) se observa en la figura 9.

3.3. Métricas

3.3.1. Accuracy

Es una métrica de evaluación que indica la precisión general o la tasa de aciertos de un modelo de clasificación (*Classification*, s. f.). Se calcula dividiendo el número de

predicciones correctas entre el número total de realizadas, como también, se puede calcular en términos de positivos y negativos como se puede apreciar en la figura 10:

Fórmulas para calcular el Accuracy:

$$\text{Accuracy} = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

3.3.2. Matriz de confusión

Es una tabla que muestra la cantidad de predicciones correctas e incorrectas realizadas por un modelo de clasificación. Es una herramienta útil para evaluar el rendimiento y comprender los errores cometidos por el modelo en la clasificación de diferentes clases (Corso, 2009).

3.3.3. Curva ROC

Es una representación gráfica que muestra la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos a medida que se varía el umbral de clasificación en un modelo de clasificación binaria. Es una herramienta útil para evaluar y comparar el rendimiento de los modelos y proporciona información sobre la capacidad de discriminación del modelo (Fawcett, 2006).

3.4. *Primer experimento con 2 clases*

Para este primer experimento se han usado 4 modelos, el primer modelo es propio llamado TruchasNet y los otros tres son por Transfer Learning (AlexNet, Vgg16 y ResNeXt50 (32x4d)). Todos los modelos se han ejecutado con 20 y 30 épocas por cada modelo, se han modificado y agregado algunas capas y estas son:

- Al modelo de TruchasNet se agregó una capa de LogSoftmax para aumentar la estabilidad numérica del proceso de entrenamiento.
- A los modelos de Alexnet, Vgg16 y ResNeXt50 (32x4d) se modificó la salida de 1000 a 2 clases.

Todos estos modelos usan un optimizador SGD (Stochastic Gradient Descent) y una función de perdida CrossEntropyLoss para su entrenamiento. Los resultados de este primer experimento se observan en la tabla 1.

Tabla 1

Resultados de los notebooks de 2 clases

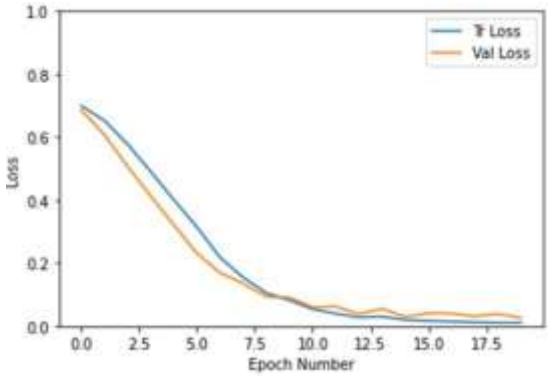
Entrenamiento	Train	Test
TruchasNet		
1. Arquitectura propia 20e (SGD, CrossEntropyLoss)	0.9596	0.9500
2. Arquitectura propia 30e (SGD, CrossEntropyLoss)	0.9730	0.9667
AlexNet		
1. Arquitectura AlexNet 20e (SGD, CrossEntropyLoss)	0.9946	0.9667
2. Arquitectura AlexNet 30e (SGD, CrossEntropyLoss)	1.0000	0.9833
Vgg16		
1. Arquitectura Vgg16 20e (SGD, CrossEntropyLoss)	1.0000	0.9333
2. Arquitectura Vgg16 30e (SGD, CrossEntropyLoss)	1.0000	0.9833
ResNeXt50		
1. Arquitectura resnext50-32x4d 20e (SGD, CrossEntropyLoss)	1.0000	***0.9833***
2. Arquitectura resnext50-32x4d 30e (SGD, CrossEntropyLoss)	1.0000	0.9833

El mejor resultado fue el modelo transfer learning resnext5032x4d con 20 épocas, obtuvo un accuracy de 0.9833, por tanto, se está mostrando los resultados del proceso de

entrenamiento y validación para la curva de pérdida y la curva de precisión (accuracy) en las figuras 11 y 12, también se está mostrando su matriz de confusión en la figura 13 y finalmente se muestra su Curva ROC en la figura 14.

Figura 9

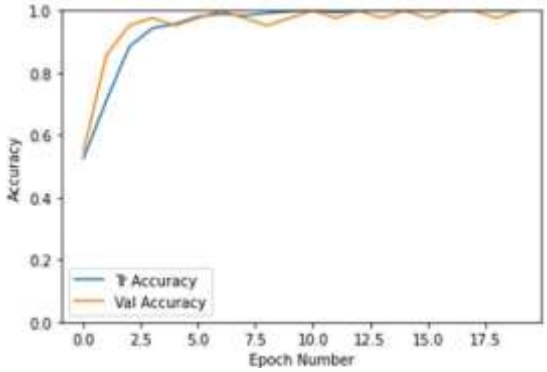
Curva de perdida de entrenamiento y validación



Fuente: *Elaboración Propia*

Figura 10

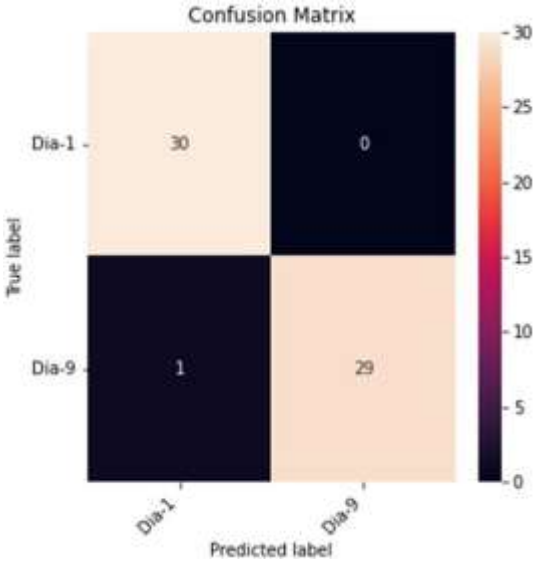
Curva de precisión de entrenamiento y validación



Fuente: *Elaboración Propia*

Figura 11

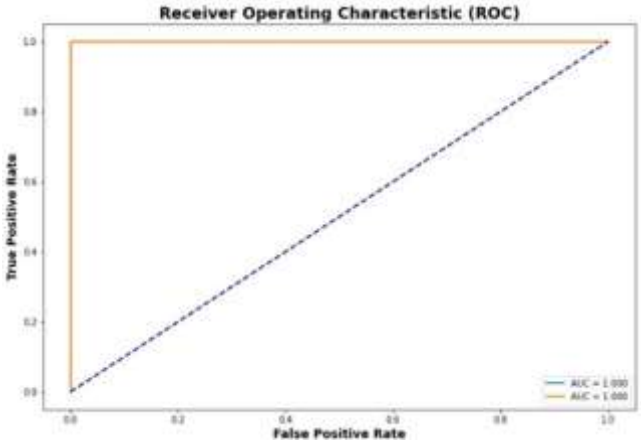
Matriz de confusión con 2 clases



Fuente: *Elaboración Propia*

Figura 12

Curva ROC con 2 clases



Fuente: *Elaboración Propia*

3.5. Segundo experimento con 3 clases

Para este segundo experimento se aplicaron los mismos modelos del primer experimento, el primer modelo es propio llamado TruchasNet y los otros tres son por Transfer Learning (AlexNet, Vgg16 y ResNeXt50 (32x4d)).

Todos los modelos se han ejecutado con 20 y 30 épocas por cada modelo, se han modificado y agregado algunas capas y estas son:

Entrenamiento	Train	Test
TruchasNet		
1. Arquitectura propia 20e (SGD, CrossEntropyLoss)	0.9593	0.9222
2. Arquitectura propia 30e (SGD, CrossEntropyLoss)	0.9440	0.9111
AlexNet		
1. Arquitectura AlexNet 20e (SGD, CrossEntropyLoss)	0.9186	0.9111
2. Arquitectura AlexNet 30e (SGD, CrossEntropyLoss)	0.9322	0.9111
Vgg16		
1. Arquitectura Vgg16 20e (SGD, CrossEntropyLoss)	0.9203	0.9000
2. Arquitectura Vgg16 30e (SGD, CrossEntropyLoss)	0.9288	0.9222
ResNeXt50		
1. Arquitectura Resnext50-32x4d 20e (SGD, CrossEntropyLoss)	0.9186	0.9111
2. Arquitectura Resnext50-32x4d 30e (SGD, CrossEntropyLoss)	0.9508	**0.9222**

- Al modelo de TruchasNet se agregó una capa de LogSoftmax para aumentar la estabilidad numérica del proceso de entrenamiento.
- A los modelos de Alexnet, Vgg16 y ResNeXt50 (32x4d) se modificó la salida de 1000 a 3 clases y se le agregó una capa de LogSoftmax a cada modelo respectivamente.

Todos estos modelos usan un optimizador SGD (Stochastic Gradient Descent) y una función de pérdida CrossEntropyLoss para su entrenamiento. Los resultados de este segundo experimento se observan en la tabla 2.

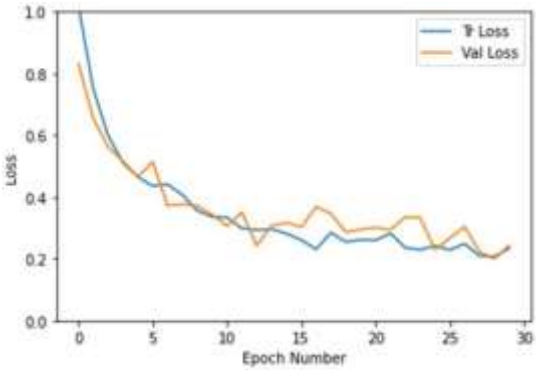
Tabla 2

Resultados de los notebooks de 3 clases

El mejor resultado fue el modelo transfer learning resnext5032x4d con 30 épocas, obtuvo un accuracy de 0,9222, por tanto, se está mostrando los resultados del proceso de entrenamiento y validación para la curva de pérdida y la curva de precisión (accuracy) en las figuras 15 y 16, también se está mostrando su matriz de confusión en la figura 17 y finalmente se muestra su Curva ROC en la figura 18.

Figura 13

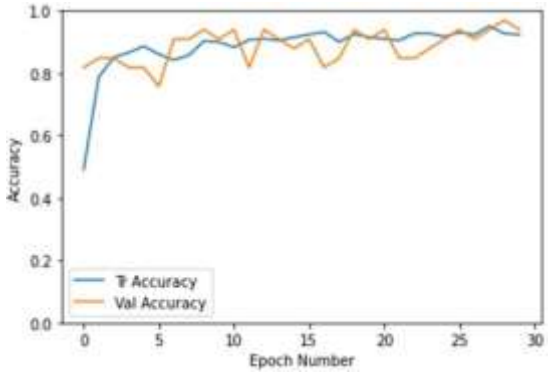
Curva de perdida de entrenamiento y validación



Fuente: *Elaboración Propia*

Figura 14

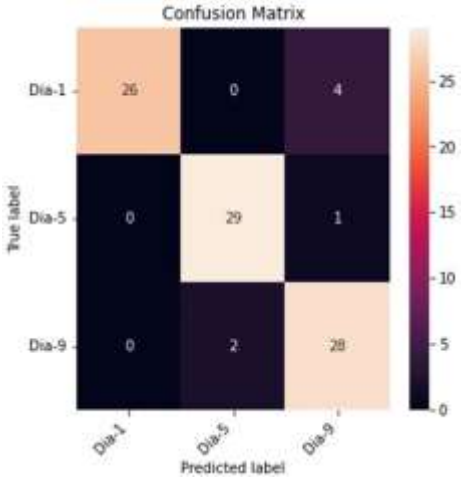
Curva de precisión de entrenamiento y validación



Fuente: *Elaboración Propia*

Figura 15

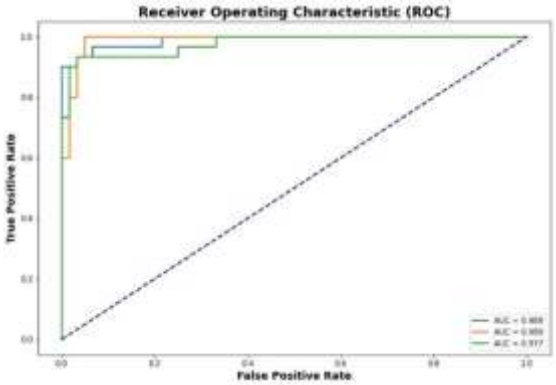
Matriz de confusión con 3 clases



Fuente: *Elaboración Propia*

Figura 16

Curva ROC con 3 clases



Fuente: *Elaboración Propia*

3.6. Tercer experimento con 5 clases

Para este tercer experimento se utilizaron los mismos modelos que los experimentos anteriores, el primer modelo es propio llamado TruchasNet y los otros tres son por Transfer Learning (AlexNet, Vgg16 y ResNeXt50 (32x4d)).

Todos los modelos se han ejecutado con 20 y 30 épocas por cada modelo, se han modificado y agregado algunas capas y siendo las siguientes:

- Al modelo de TruchasNet se agregó una capa de LogSoftmax para aumentar la estabilidad numérica del proceso de entrenamiento.
- A los modelos de Alexnet, Vgg16 y ResNeXt50 (32x4d) se modificó la salida de 1000 a 5 clases y se le agregó una capa de LogSoftmax igual que el segundo experimento.

Todos estos modelos usan un optimizador SGD (Stochastic Gradient Descent) y una función de pérdida CrossEntropy Loss para su entrenamiento. Los resultados de este tercer experimento se observan en la tabla 3.

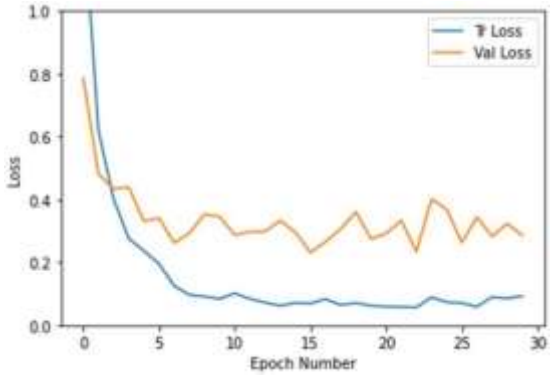
Tabla 3

Resultados de los notebooks de 5 clases

El mejor resultado fue el modelo transfer learning resnext5032x4d con 30 épocas, obtuvo un accuracy de 0.8800, por tanto, se está mostrando los resultados del proceso de entrenamiento y validación para la curva de pérdida y la curva de precisión (accuracy) en las figuras 19 y 20, también se muestra su matriz de confusión en la figura 21 y finalmente se muestra su curva ROC en la figura 22.

Figura 17

Curva de perdida de entrenamiento y validación



Fuente: *Elaboración Propia*

Entrenamiento	Train	Test
TruchasNet		
1. Arquitectura propia	0.8484	0.7100
2. Arquitectura propia	0.8244	0.7000
AlexNet		
1. Arquitectura AlexN	0.8964	0.8100
2. Arquitectura AlexN	0.9002	0.8200
Vgg16		
1. Arquitectura Vgg16	0.9328	0.8500
2. Arquitectura Vgg16	0.9367	0.8700
ResNeXt50		
1. Arquitectura resnex	0.9866	0.8600
2. Arquitectura resnext50-32x4d 30e (SGD, CrossEntropyLoss)	0.9904	**0.8800**

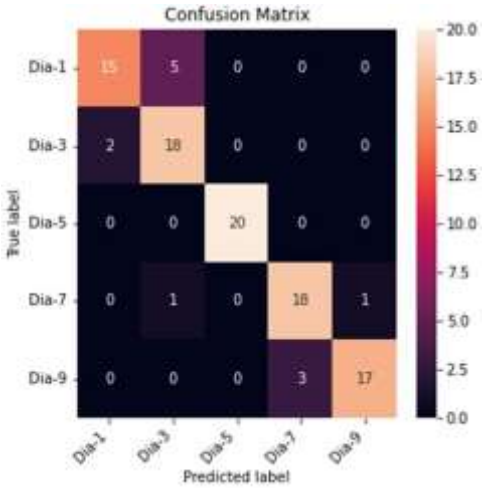
Figura 18

Curva de precisión de entrenamiento y validación

Fuente: *Elaboración Propia*

Figura 19

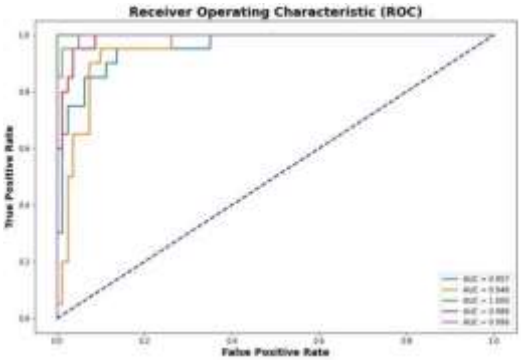
Matriz de confusión con 5 clases



Fuente: *Elaboración Propia*

Figura 20

Curva ROC con 5 clases



Fuente: *Elaboración Propia*

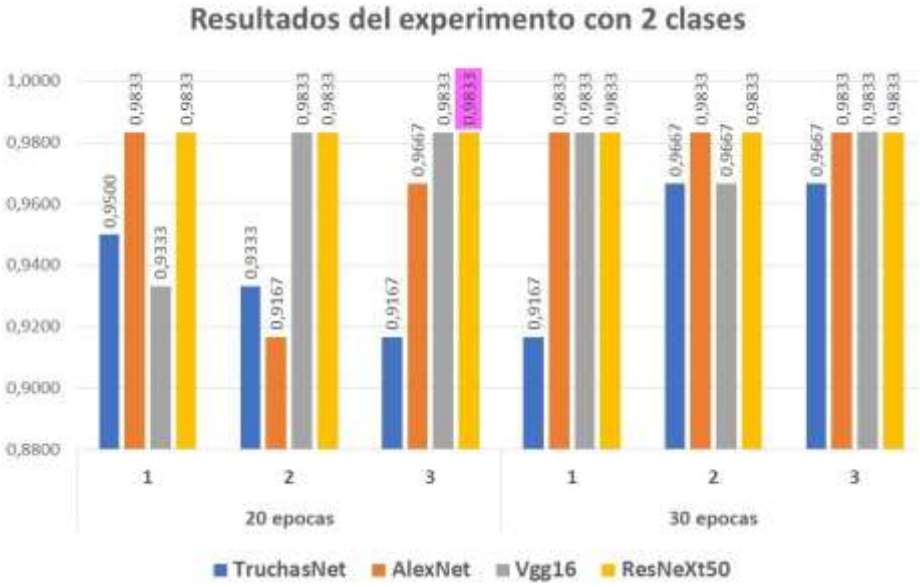
4. RESULTADOS

En el primer experimento para la clasificación de la frescura de la trucha, se realizó con 2 clases (día 1 y 9), obteniendo los resultados de los 4 modelos mencionados en la tabla 1, estos modelos se ejecutaron con 20 y 30 épocas; las cuales dieron resultados excelentes, siendo que el modelo transfer learning ResNeXt50-32x4d con 20 épocas dio uno de los mejores resultados con un accuracy de 0.9833.

Podemos apreciar todos los resultados del primer experimento con 2 clases, cada modelo se ejecutó 3 veces con 20 épocas y lo mismo con 30 épocas, como se muestra en la figura 23.

Figura 21

Resultados generales del experimento 1



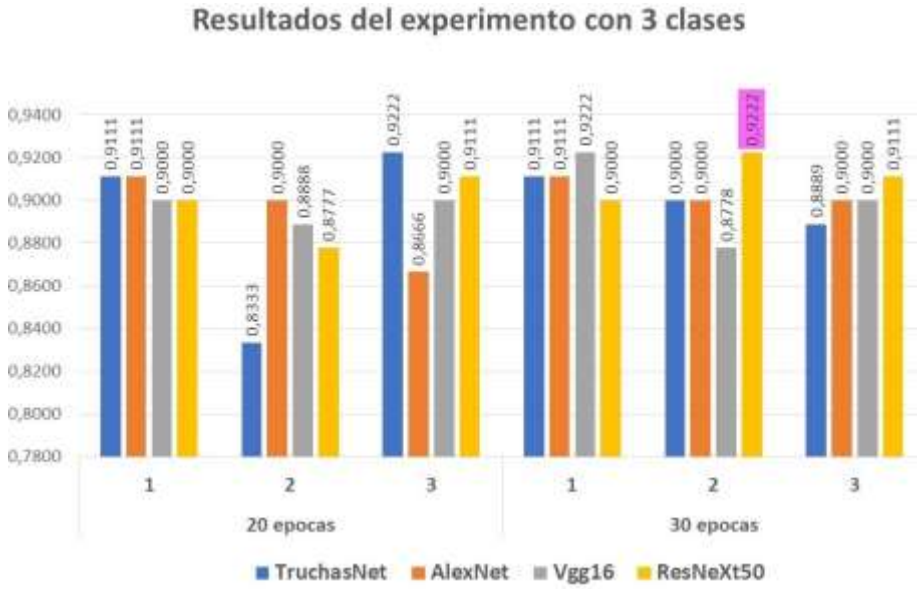
Fuente: *Elaboración Propia*

En el segundo experimento para la clasificación de la frescura de la trucha, se realizó con 3 clases (día 1, 5 y 9), se obtuvieron los resultados de los 4 modelos mencionados en la tabla 2, estos modelos se ejecutaron con 20 y 30 épocas; las cuales dieron resultados muy buenos, siendo que el modelo transfer learning ResNeXt50-32x4d con 30 épocas dio uno de los mejores resultados con un accuracy de 0.9222.

De la misma forma podemos apreciar todos los resultados del segundo experimento con 3 clases, cada modelo se ejecutó 3 veces con 20 épocas y lo mismo con 30 épocas, como se puede apreciar en la figura 24.

Figura 22

Resultados en general del experimento 2



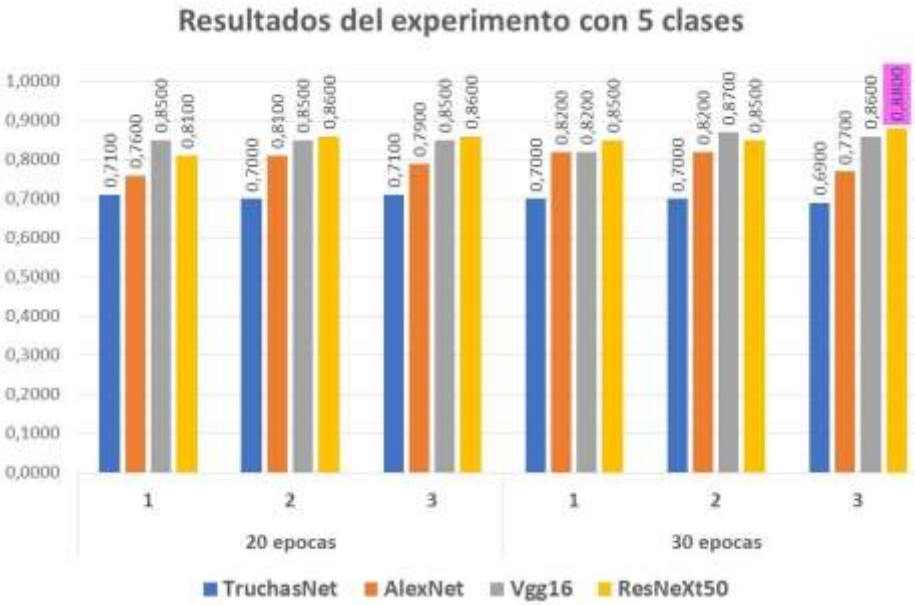
Fuente: *Elaboración Propia*

En el tercer experimento para la clasificación de la frescura de la trucha, se realizó con 5 clases (día 1, 3, 5, 7 y 9), los cuales se obtuvieron los resultados de los 4 modelos mencionados en la tabla 3, estos modelos se ejecutaron con 20 y 30 épocas; las cuales dieron resultados buenos, siendo que el modelo transfer learning ResNeXt50-32x4d con 30 épocas dio el mejor resultado con un accuracy de 0.8800.

En este último gráfico podemos apreciar todos los resultados del tercer experimento con 5 clases, cada modelo se ejecutó 3 veces con 20 épocas y lo mismo con 30 épocas, como se puede apreciar en la figura 25.

Figura 23

Resultados en general del experimento 3



Fuente: *Elaboración Propia*

5. CONCLUSIONES

Los resultados obtenidos en todos los experimentos realizados con TruchasNet, un modelo basado en redes neuronales convolucionales (CNN), ha demostrado un rendimiento muy bueno en la clasificación de la frescura de las truchas. Estos resultados validan la capacidad de nuestro modelo para realizar dicha clasificación de manera efectiva.

Además, el trabajo realizado con TruchasNet proporciona una base sólida para futuras investigaciones en segmentación y detección de truchas. Estas áreas de estudio tienen el potencial de ampliar las capacidades del modelo y explorar nuevos enfoques para la identificación y análisis de características específicas de las truchas.

Asimismo, los resultados de este trabajo pueden adaptarse para el desarrollo de un software de clasificación automático y fácil de usar para determinar la frescura del pescado.

Por último, la creación y publicación del dataset para el medio peruano puede impulsar una amplia gama de trabajos en inteligencia artificial. Esta disponibilidad de la

dataset permite abordar desafíos únicos y fomentar la colaboración y el intercambio de conocimientos en el campo de la inteligencia artificial.

6. REFERENCIAS

A healthy diet sustainably produced: Information sheet. (s. f.). https://www.who.int/publications-detail-redirect/WHO-NMH-NHD-18.12

Anuario Estadístico Pesquero y Acuícola 2022. (s. f.). Recuperado 18 de septiembre de 2023, de https://ogeiee.produce.gob.pe/index.php/en/oee-documentos-publicaciones/publicaciones-anuales/item/1116-anuario-estadistico-pesquero-y-acuicola-2022

Ariño, A., Beltrán, J. A., Herrera, A., & Roncalés, P. (2013). Fish and seafood: Nutritional Value. En B. Caballero (Ed.), *Encyclopedia of Human Nutrition (Third Edition)* (pp. 254-261). Academic Press. https://doi.org/10.1016/B978-0-12-375083-9.00110-0

Barat, J. M., Gil, L., García-Breijo, E., Aristoy, M.-C., Toldrá, F., Martínez-Máñez, R., & Soto, J. (2008). Freshness monitoring of sea bream (Sparus aurata) with a potentiometric sensor. *Food Chemistry*, *108*(2), 681-688. https://doi.org/10.1016/j.foodchem.2007.10.034

Classification: Accuracy | Machine Learning. (s. f.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy

Corso, C. L. (2009). Aplicación de algoritmos de clasificación supervisada usando Weka. *Córdoba: Universidad Tecnológica Nacional, Facultad Regional Córdoba*.

Dowlati, M., de la Guardia, M., Dowlati, M., & Mohtasebi, S. S. (2012). Application of machine-vision techniques to fish-quality assessment. *TrAC Trends in Analytical Chemistry*, *40*, 168-179. https://doi.org/10.1016/j.trac.2012.07.011

Dowlati, M., Mohtasebi, S. S., Omid, M., Razavi, S. H., Jamzad, M., & de la Guardia, M. (2013). Freshness assessment of gilthead sea bream (Sparus aurata) by machine vision based on gill and eye color changes. *Journal of Food Engineering*, *119*(2), 277-287. https://doi.org/10.1016/j.jfoodeng.2013.05.023

Dubey, S. R., & Jalal, A. S. (2014). Fruit disease recognition using improved sum and difference histogram from images. *International Journal of Applied Pattern Recognition*, *1*(2), 199. https://doi.org/10.1504/IJAPR.2014.063759

Dutta, M. K., Issac, A., Minhas, N., & Sarkar, B. (2016). Image processing based method to assess fish quality and freshness. *Journal of Food Engineering*, *177*, 50-58. https://doi.org/10.1016/j.jfoodeng.2015.12.018

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861-874. https://doi.org/10.1016/j.patrec.2005.10.010

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90. https://doi.org/10.1145/3065386

Liang, T., Xu, X., & Xiao, P. (2017). A new image classification method based on modified condensed nearest neighbor and convolutional neural networks. *Pattern Recognition Letters*, *94*, 105-111. https://doi.org/10.1016/j.patrec.2017.05.019

Macagnano, A., Careche, M., Herrero, A., Paolesse, R., Martinelli, E., Pennazza, G., Carmona, P., D'Amico, A., & Natale, C. D. (2005). A model to predict fish quality from instrumental features. *Sensors and Actuators B: Chemical*, *111-112*, 293-298. https://doi.org/10.1016/j.snb.2005.06.028

Mohammadi Lalabadi, H., Sadeghi, M., & Mireei, S. A. (2020). Fish freshness categorization from eyes and gills color features using multi-class artificial neural network and support vector machines. *Aquacultural Engineering*, *90*, 102076. https://doi.org/10.1016/j.aquaeng.2020.102076

Rocculi, P., Cevoli, C., Tappi, S., Genovese, J., Urbinati, E., Picone, G., Fabbri, A., Capozzi, F., & Rosa, M. D. (2019). Freshness assessment of European hake (*Merluccius merluccius*) through the evaluation of eye chromatic and morphological characteristics. *Food Research International*, *115*, 234-240. https://doi.org/10.1016/j.foodres.2018.08.091

Simonyan, K., & Zisserman, A. (2015). *Very Deep Convolutional Networks for Large-Scale Image Recognition* (arXiv:1409.1556). arXiv. https://doi.org/10.48550/arXiv.1409.1556

Taheri-Garavand, A., Fatahi, S., Banan, A., & Makino, Y. (2019). Real-time nondestructive monitoring of Common Carp Fish freshness using robust vision-

based intelligent modeling approaches. *Computers and Electronics in Agriculture, 159*, 16-27. https://doi.org/10.1016/j.compag.2019.02.023

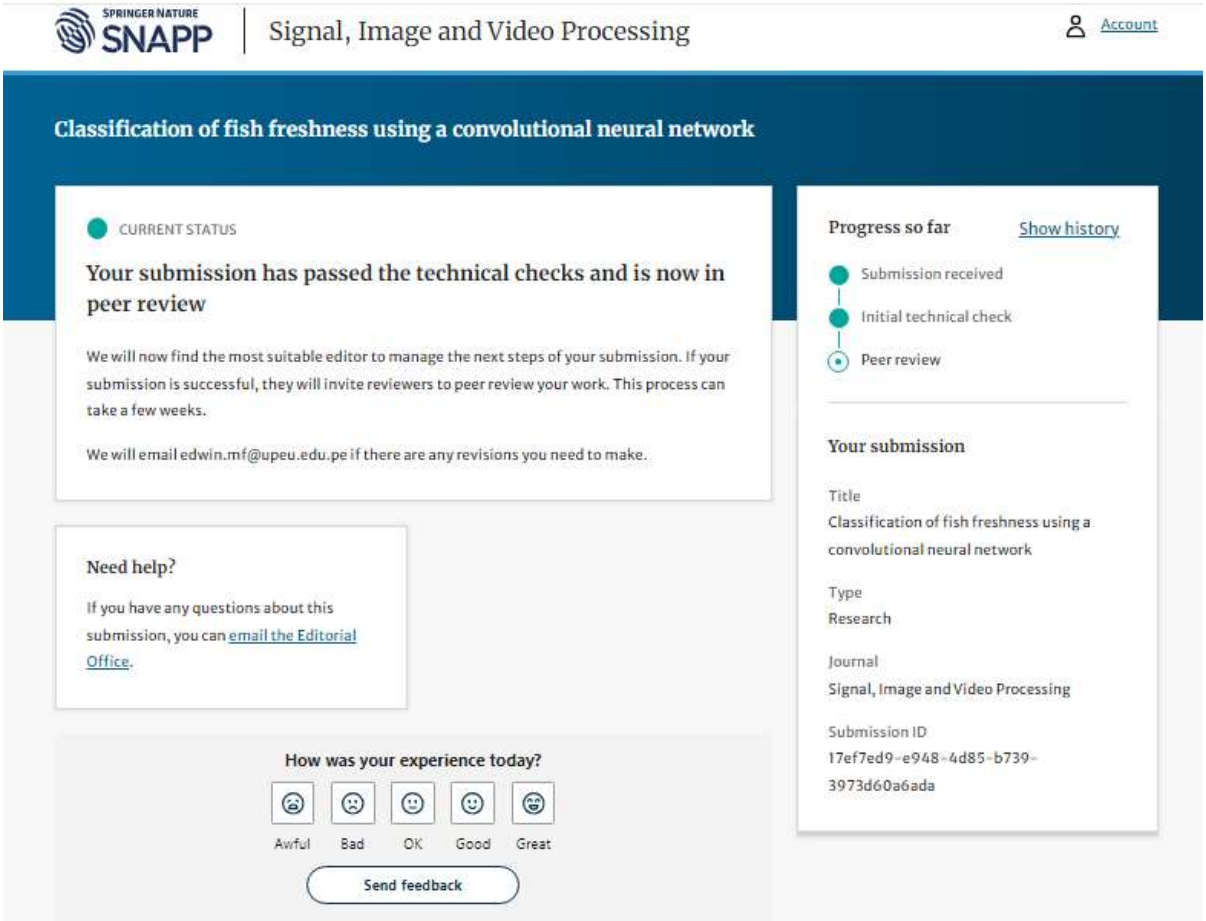
Tappi, S., Rocculi, P., Ciampa, A., Romani, S., Balestra, F., Capozzi, F., & Dalla Rosa, M. (2017). Computer vision system (CVS): A powerful non-destructive technique for the assessment of red mullet (Mullus barbatus) freshness. *European Food Research and Technology, 243*(12), 2225-2233. https://doi.org/10.1007/s00217-017-2924-0

Xie, S., Girshick, R., Dollar, P., Tu, Z., & He, K. (2017). *Aggregated Residual Transformations for Deep Neural Networks*. 1492-1500. https://openaccess.thecvf.com/content_cvpr_2017/html/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.html

ANEXOS

ANEXO A

Evidencia de submission del artículo a la revista Signal, Image and Video Processing



ANEXO B

Resolución de inscripción del perfil de proyecto de tesis aprobado por el Consejo de facultad

"AÑO DEL FORTALECIMIENTO DE LA SOBERANÍA NACIONAL"

RESOLUCIÓN N° 0403-2022/UPeU-FIA-CF-T

Lima, Ñaña 24 de mayo de 2022

VISTO:

El expediente de **Edwin Wilson Mamani Flores**, identificado(a) con Código Universitario Nº 201712142 y **Royer Pedraza Huisa**, identificado(a) con Código Universitario Nº 201520625, de la Escuela Profesional de Ingeniería de Sistemas de la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión;

CONSIDERANDO

Que la Universidad Peruana Unión tiene autonomía académica, administrativa y normativa, dentro del ámbito establecido por la Ley Universitaria Nº 30220 y el Estatuto de la Universidad;

Que la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión, mediante sus reglamentos académicos y administrativos, ha establecido las formas y procedimientos para la aprobación e inscripción del perfil de proyecto de tesis en formato artículo y la designación o nombramiento del asesor para la obtención del título profesional;

Que **Edwin Wilson Mamani Flores** y **Royer Pedraza Huisa**, han solicitado: la inscripción del perfil de proyecto de tesis titulado "Clasificación de la frescura del pescado a partir de las características de color de los ojos y las branquias utilizando una red neuronal convolucional" y la designación del Asesor, encargado de orientar y asesorar la ejecución del perfil de proyecto de tesis en formato artículo;

Estando a lo acordado en la sesión del Consejo de la Facultad de Ingeniería y Arquitectura de la Universidad Peruana Unión, celebrada el 24 de mayo de 2022, y en aplicación del Estatuto y el Reglamento General de Investigación de la Universidad;

SE RESUELVE:

Aprobar el perfil de proyecto de tesis en formato artículo titulado **"Clasificación de la frescura del pescado a partir de las características de color de los ojos y las branquias utilizando una red neuronal convolucional"** y disponer su inscripción en el registro correspondiente, designar como asesor a **Mg. Ferdinand Edgardo Pineda Ancco** para que oriente y asesore la ejecución del perfil de proyecto de tesis en formato artículo el cual fue dictaminado por: **MSc. Fredy Abel Huanca Torres** y **Mg. Abel Angel Sullon Macalupu**, otorgándoles un plazo máximo de doce (12) meses para la ejecución.

Regístrese, comuníquese y archívese.

Dra. Erika Inés Acuña Salinas
DECANA

Dr. Santiago Ramírez López
SECRETARIO ACADÉMICO

cc:
-Interesado
Asesor
Dirección General de Investigación
Archivo

ANEXO C

Base de Datos creado de imágenes de la trucha Arco Iris

La base de datos creado por los autores se encuentra en una carpeta de OneDrive, el cual esta publicada abiertamente para uso de los interesados que tengan el enlace siguiente:

URL: https://upeuedupe-my.sharepoint.com/:f:/g/personal/edwin_mf_upeu_edu_pe/Eio3Bprck91As0VsaO0SrtQBbVGllsYJGm-QmBjnoWQYqg?e=YQPyUC

ANEXO D

Código fuente del modelo ALEXNET

```
#m_alexnet.classifier[4] = nn.Linear(4096, 2048)
m_alexnet.classifier[6] = nn.Linear(4096, 5)
m_alexnet.classifier.add_module("7", nn.LogSoftmax(dim = 1))
m_alexnet.cuda()
```

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=5, bias=True)
    (7): LogSoftmax(dim=1)
  )
```

```
def test_model(model):
    model.eval()
    test_loss = 0.0
    test_corrects = 0
    criteria = nn.NLLLoss()
    #phase ='test2'

    for inputs, labels in test_data_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)

        with torch.set_grad_enabled(True):
            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)
            loss = criteria(outputs, labels)

        test_loss += loss.item() * inputs.size(0)
        test_corrects += torch.sum(preds == labels.data)

    epoc_loss = test_loss / test_data_size
    epoc_acc = test_corrects / test_data_size

    print('Test: Loss: {:.4f} Acc: {:.4f}'.format(epoc_loss, epoc_acc))
    return test_model
```

```
criteria = nn.CrossEntropyLoss()
optimizer = optim.SGD(m_alexnet.parameters(), lr=0.001, momentum=0.9)
scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
# Number of epochs
```

ANEXO E

Código fuente del modelo VGG16

```
# Change the final layer of AlexNet Model for Transfer Learning
#m_vgg16.classifier[3] = nn.Linear(4096, 512)
m_vgg16.classifier[6] = nn.Linear(4096, 5)
m_vgg16.classifier.add_module("7", nn.LogSoftmax(dim = 1))
m_vgg16.cuda()
```

```
VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (18): ReLU(inplace=True)
    (19): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
```

```
    (22): ReLU(inplace=True)
    (23): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (24): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (25): ReLU(inplace=True)
    (26): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (27): ReLU(inplace=True)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=5, bias=True)
    (7): LogSoftmax(dim=1)
  )
```

```
criteria = nn.CrossEntropyLoss()
optimizer = optim.SGD(m_vgg16.parameters(), lr=0.001, momentum=0.9)
scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
# Number of epochs
eps=30
```

```
model_vgg16, history = train_model(m_vgg16, criteria, optimizer, scheduler, eps, 'cuda')

torch.save(history, dataset+'_history.pt')
```

ANEXO F

Código fuente del modelo RESNEXT50_32X4D

```
# Change the final layer of ResNet50 Model for Transfer Learning
fc_inputs = m_resnext50.fc.in_features
m_resnext50.fc = nn.Sequential(
    nn.Linear(2048, 5),
    nn.LogSoftmax(dim=1) # For using NLLLoss()
            )

m_resnext50.cuda()
```

```
    (4): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (5): Bottleneck(
      (conv1): Conv2d(1024, 512, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(512, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (layer4): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(1024, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (1): Bottleneck(
      (conv1): Conv2d(2048, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (2): Bottleneck(
      (conv1): Conv2d(2048, 1024, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32, bias=False)
      (bn2): BatchNorm2d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(1024, 2048, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(2048, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Sequential(
    (0): Linear(in_features=2048, out_features=5, bias=True)
    (1): LogSoftmax(dim=1)
  )
)
```

ANEXO G

Código fuente del modelo propio TRUCHASNET

```
class Network(nn.Module):
    def __init__(self):
        super(Network, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(
                in_channels=3,
                out_channels=128,
                kernel_size=5,
                stride=1,

            ),
            nn.ReLU(),    # activation
            nn.MaxPool2d(kernel_size=2),
            nn.BatchNorm2d(128)
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(128, 256, 3, 1),
            nn.ReLU(),  # activation
            nn.MaxPool2d(2),
            nn.BatchNorm2d(256)
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(256, 512, 3, 1),
            nn.ReLU(),
            nn.MaxPool2d(2),
            nn.BatchNorm2d(512)
        )
    )
    self.conv4 = nn.Sequential(
        nn.Conv2d(512, 1024, 3, 1),
        nn.ReLU(),
        nn.MaxPool2d(2),
        nn.BatchNorm2d(1024)
    )
    self.conv5 = nn.Sequential(
        nn.Conv2d(1024, 512, 3, 1),
        nn.ReLU(),
        nn.MaxPool2d(2),
        nn.BatchNorm2d(512)
    )
    self.avgpool = nn.AdaptiveAvgPool2d(output_size=(1,1))
    self.fc = nn.Sequential(
        nn.Linear(512*5*5,2048),
        nn.Linear(2048,5),   # fully connected layer, output 5 classes
        nn.LogSoftmax(dim=1)
    )
 def forward(self, x):
    x = self.conv1(x)
    x = self.conv2(x)
    x = self.conv3(x)
    x = self.conv4(x)
    x = self.conv5(x)
    x = x.view(-1, self.num_flat_features(x))
    #x = x.view(x.size(0), -1)
```

```
        output = self.fc(x)
        return output

    def num_flat_features(self, x):
        size = x.size()[1:]
        num_features = 1
        for s in size:
            num_features *= s
        return num_features


network = Network()
```

```
network2 = network.to(device)
network2.cuda()
```

```
criteria = nn.CrossEntropyLoss()
optimizer = optim.SGD(network2.parameters(), lr=0.001, momentum=0.9)
scheduler = lr_scheduler.StepLR(optimizer, step_size=7, gamma=0.1)
# Number of epochs
eps=30
```

```
model_network2, history = train_model(network2, criteria, optimizer, scheduler, eps, 'cuda')

torch.save(history, dataset+'_history.pt')
```