

Proyecto 02: Redes Neuronales convolucionales

Inteligencia Artificial

1st Jeffrey Leiva Cascante
Ingeniería en Computación
Instituto Tecnológico de Costa Rica
2021016720
Cartago, Costa Rica
jeffleivajr@estudiantec.cr

2nd Richard Osvaldo León Chinchilla
Ingeniería en Computación
Instituto Tecnológico de Costa Rica
2019003759
Cartago, Costa Rica
r29leonc@estudiantec.cr

Resumen—En este proyecto se comparan dos modelos de redes neuronales convolucionales en la tarea de clasificación de un conjunto de imágenes con tres categorías: Covid, Normal y Viral Pneumonia. Se aplican filtros en el preprocesamiento de las mismas para obtener tres conjuntos de datos distintos, se aplican los filtros *bilateral* y *canny edge detection*. El primer modelo; A, es una arquitectura resnet-50 preentrenada. Para el modelo B se plantean tres arquitecturas distintas, se realizan tres *proof of concept* de las mismas y se escoge la de mejor resultado. Ambos modelos; A y B, se entrena con los *datasets* crudo, *bilateral* y *canny* para determinar con que conjunto de datos se presenta el mejor rendimiento para ambos. El rendimiento de los modelos se evaluaron utilizando métricas de *Accuracy*, *Precision*, *Recall*, *F1-Score* y matriz de confusión. Se concluyó, que el Modelo B obtuvo los mejores resultados en general.

Index Terms—CNN, Deep Learning, Pytorch, Resnet50, Bilateral Filter, Canny Edge Detection, Covid19 dataset

I. INTRODUCCIÓN

El presente trabajo, se enfoca en aplicar las redes neuronales convolucionales (CNN) para la clasificación multiclas con imágenes utilizando aprendizaje supervisado.

Con base a la información de [1], las redes neuronales convolucionales es un algoritmo de *Deep Learning* que tiene como propósito trabajar con imágenes, tomando estas como *Input*, asignándole pesos a ciertas características en la imagen para así diferenciarla una de otra. Las tareas comunes de este tipo de redes son para detección o categorización de objetos, y clasificación de imágenes, este último siendo el objetivo de este proyecto.

El conjunto de datos base que se estará utilizando corresponde a *Covid-19 Image Dataset*, obtenido de la fuente [2], es un conjunto de datos que contiene radiografías de tórax a múltiples pacientes con el objetivo de detectar el *Covid-19*. Este contiene tres clases las cuales el modelo deberá ser capaz de clasificar cada imagen en: **Covid**, **Normal**, **Neumonía**.

Para la clasificación de imágenes, se hará el uso de dos modelos diferentes. El primero denominado **Modelo A**, construido mediante la red neuronal convolucional *Resnet-50*. El segundo modelo, denominado **Modelo B**, construido con la mejor de tres arquitecturas diseñadas manualmente.

Además, la construcción de ambos modelos se hará mediante el uso de *Pytorch*, que según [3], es una biblioteca

de aprendizaje automático de código abierto que facilita la creación y optimización de redes neuronales. Su capacidad para manejar el cómputo de tensores de manera eficiente, lo convierte en gran opción para proyectos de inteligencia artificial.

Primeramente, se realizará un preprocesamiento, que creará tres conjuntos de datos diferentes; uno siendo el original, el segundo con el filtro *Bilateral*, y el tercero con *Canny Edge Detection*. Esto con el fin de realizar tres entrenamientos distintos para cada modelo.

Segundo, se procederá con el *Data Augmentation*, el cuál es una técnica para aumentar la cantidad de ejemplos disponibles usando rotaciones, cambio de colores de canales, entre otros.

Tercero, se hará el entrenamiento de ambos modelos con los tres diferentes conjuntos de datos.

Y por último, la evaluación de ambos modelos para demostrar cual obtuvo los mejores resultados, realizando un análisis con la herramienta *Weights and Biases*.

II. PREPROCESAMIENTO

II-A. Lectura del conjunto de datos

El conjunto de datos *Covid-19 Image Dataset*, está subdividido en *Train* y *Test*, en las cuales cada uno, contiene las clases:

- **Covid**: Ver Figura 1
- **Normal**: Ver Figura 2
- **Neumonía**: Ver Figura 3

II-B. Aplicación de filtros

Antes de realizar los filtros, cada imagen se transforma a una escala de grises, es decir, un canal, y se le aplica un *resize* de 224x224 para el modelo A y 128x128 para el modelo B.

II-B1. Bilateral filter: Este filtro tiene como objetivo combinar pesos de distancia espacial e intensidad para preservar los bordes mientras reduce el ruido. [4]

Los parámetros utilizados fueron los siguientes:

- diámetro: 9
- sigmaColor: 25
- sigmaSpace: 25

Como ejemplo, se produjo el resultado observado en la Figura 4, en donde se redujo el ruido notoriamente, manteniendo los bordes.



Figura 1. Covid-Raw



Figura 4. Bilateral-Covid-224x224

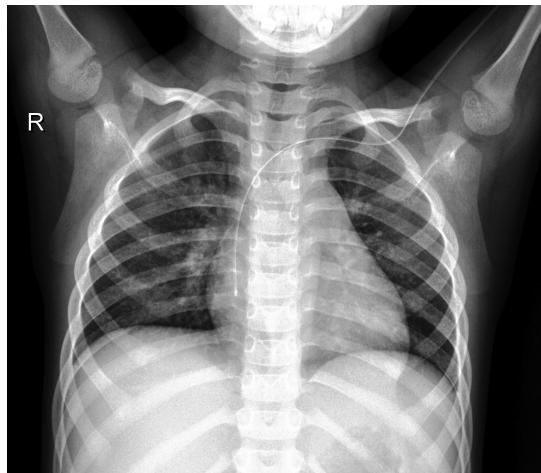


Figura 2. Normal-Raw



Figura 3. Neumonía-Raw

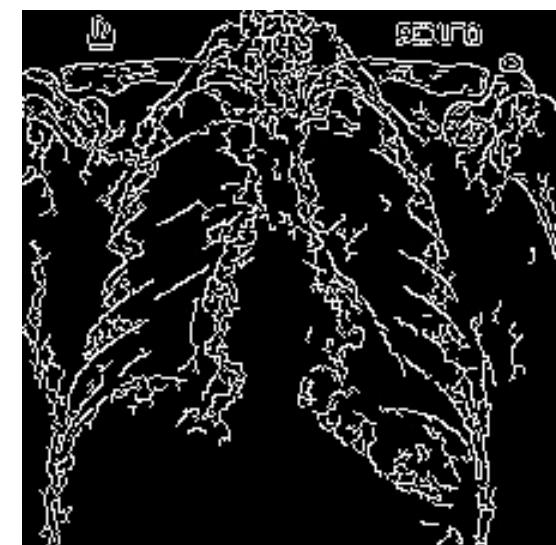


Figura 5. Canny-Covid-224x224

III. DATA AUGMENTATION

Las transformaciones que se aplicaron a las imágenes de los diferentes conjuntos de datos fueron las que se aprecian en el Cuadro I

Transformación	Descripción
Resize	A 224x224 o 128x128 píxeles
ToTensor	Convierte la imagen a tensor
Normalize	Con media 0.5 y desviación 0.5
RandomRotation	Rotación aleatoria de hasta 20°
RandomHorizontalFlip	Inversión horizontal aleatoria
ColorJitter	Variación de brillo y contraste

Cuadro I
TRANSFORMACIONES APLICADAS

IV. MODELO A

IV-A. Inicialización

Primero, se definen dos conjuntos de transformaciones: uno para el conjunto de entrenamiento y otro para el conjunto de prueba. Para las imágenes de entrenamiento, se aplica las que se vieron en la sección del *Data Augmentation*, ver Cuadro I, con una *resize* de 224x224, debido a que es el tamaño requerido por *Resnet*. Para las imágenes de prueba, se excluyen la rotación, la inversión y el ajuste de brillo-contraste.

Posteriormente, se cargan los conjuntos de datos de entrenamiento y prueba desde diferentes carpetas que contienen imágenes de rayos X de COVID-19, con tres versiones del conjunto: una normal, una con el filtro bilateral aplicado y otra con el filtro de detección de bordes Canny.

Por cada conjunto de datos, se reserva el 15 % de las imágenes del conjunto de entrenamiento para validación. Se aplican las transformaciones para finalmente crear los *Data Loaders* con un *Batchsize* de 32.

IV-B. Carga de Resnet-50 preentrenado

Se carga el modelo preentrenado *Resnet50* con los pesos preentrenados. Luego, se adapta la capa final del modelo para ajustarse al problema con las 3 clases, modificando la capa *fully connected* para que el número de salidas sea de 3.

Después, se congelan los pesos de las capas preentrenadas del modelo para que no se actualicen durante el entrenamiento, exceptuando la capa final utilizando el optimizador Adam con un *learning rate* de 0.001.

Finalmente, la función de pérdida se especifica mediante *CrossEntropyLoss*

IV-C. Entrenamiento

Los pasos realizados en el entrenamiento del Modelo A son los siguientes:

- **Ciclo de entrenamiento:** Por cada *epoch*, el modelo *Resnet-50* se pone en modo de entrenamiento y se empiezan a recorrer los datos de entrenamiento. El modelo genera predicciones para los *inputs* y *labels* y se calcula la función de pérdida. Se realiza el *backpropagation* y se actualizan los pesos.
- **Ciclo de validación:** Durante cada recorrido de todos los *epochs*, se evalúa el modelo en el conjunto de validación con el modelo en modo evaluación. Durante la validación, no se calculan los gradientes. Similar a la fase de entrenamiento se calcula la pérdida y el *accuracy*, pero esta vez sobre los datos de validación.

- **Mejor modelo:** Durante cada *epoch* en la validación, si el *accuracy* ha sido el mejor, se guarda el estado del modelo actual como el mejor modelo.

IV-D. Resultados de la fase de entrenamiento

Para el conjunto de datos crudo, con el filtro bilateral, y el canny, se obtuvieron los resultados mostrados en los Cuadros II, III, IV, respectivamente. Se aprecia que, se encontró el resultado óptimo en los primeros *epochs*, obteniendo en todos los entrenamientos un alto *accuracy* en la validación.

Como se aprecia en la Figura 6, el mejor rendimiento se obtuvo en el conjunto Bilateral, tanto para las métricas de pérdida como las de *accuracy*, lo que indica que el modelo está generalizando mejor en este conjunto de datos. A pesar de esto, los otros dos conjuntos de datos presentaron buenos resultados también. Puede visualizar todo el reporte en **Reporte completo**

Parámetro	Valor
N_Epochs	20
Epoch	5
Train Loss	0.3176
Train Acc	0.0900
Val Loss	0.3190
Val Acc	0.9459

Cuadro II
ENTRENAMIENTO DATASET CRUDO

Parámetro	Valor
N_Epochs	30
Epoch	3
Train Loss	0.2246
Train Acc	0.0758
Val Loss	0.1460
Val Acc	1.00

Cuadro III
ENTRENAMIENTO DATASET BILATERAL

Parámetro	Valor
N_Epochs	30
Epoch	6
Train Loss	0.3265
Train Acc	0.0711
Val Loss	0.3058
Val Acc	0.9189

Cuadro IV
ENTRENAMIENTO DATASET CANNY

IV-E. Resultados de la fase de pruebas

IV-E1. **Dataset Crudo:** En la Figura V, se observa que en comparación con los datos de entrenamiento, hubo un *Overfitting* considerable. El modelo predice correctamente aproximadamente el 77,27 % de las veces. En cuanto a la métrica de *precision*, de todas las predicciones que el modelo tomó como positivas, el 85,71 % realmente lo eran. En el *recall*, con un valor de 75 %, significa que el modelo detectó correctamente el 75 % de los casos positivos. Finalmente con

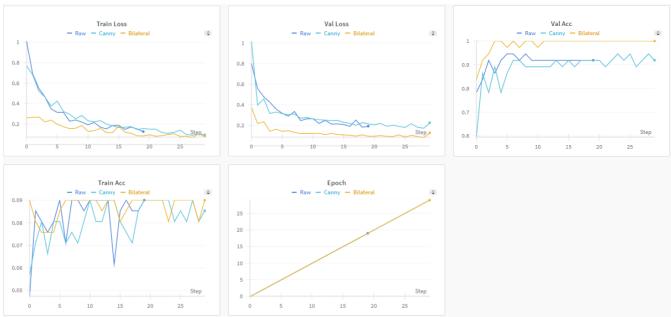


Figura 6. WB-Modelo A Training

el *F1-Score*, el balance indica un 70% que sugiere que el modelo aún tiene mucho margen de mejora para este conjunto de datos.

En la Figura 7, se observa la matriz de confusión, en donde el modelo clasifica correctamente todas las predicciones de la clase 0 (Covid). En la clase 1 (Normal), el modelo solo clasifica correctamente 5 de las 20 predicciones, mientras que 15 son clasificadas incorrectamente como clase 2 (Neumonía), lo que indica un alto índice de confusión en estas dos clases. Para la clase 2 (Neumonía), el modelo muestra un buen rendimiento clasificando correctamente 20 predicciones sin errores.

Parámetro	Valor
Accuracy	0.7727
Precision	0.8571
Recall	0.75
F1-Score	0.7091

Cuadro V
PRUEBAS DATASET CRUDO

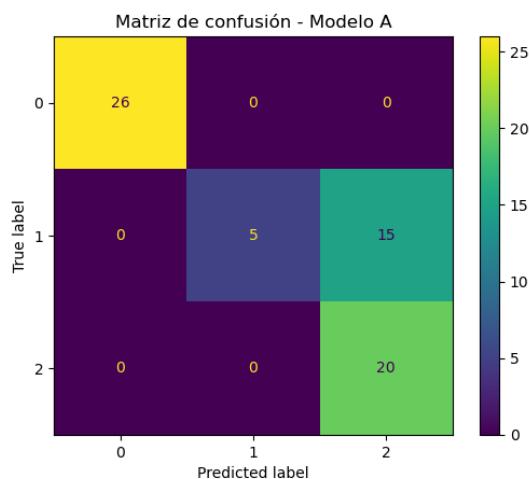


Figura 7. Matriz de Confusión - Raw - Modelo A

IV-E2. Dataset Bilateral: En el Cuadro VI, se aprecia lo siguiente:

- **Accuracy:** El modelo predice correctamente el 93,94 % de las veces. El cual indica que el modelo tiene buen rendimiento en este conjunto de datos.
- **Precision:** Todas las predicciones que el modelo hizo como positivas, el 93,68 % fueron correctas. Indica que el modelo tiene pocos falsos positivos y es bastante fiable al hacer predicciones positivas.
- **Recall:** El modelo es capaz de identificar correctamente el 93,33 % de los verdaderos positivos. Esto indica que el modelo está logrando detectar casi todos los casos positivos.
- **F1-Score:** El balance del modelo corresponde a 93,47 %, indicando un excelente balance entre el *precision* y *recall*.

En la matriz de confusión que se aprecia en la Figura 8, se observa un excelente rendimiento general. Para la clase 0 (Covid), el modelo clasificó correctamente las 26 predicciones.

Para la clase 1 (Normal), de las 20 predicciones, 18 fueron clasificadas correctamente, y solo 2 son incorrectas. Esto demuestra una mejora considerable respecto a la matriz de confusión del conjunto de datos crudo.

En la clase 2 (Neumonía), de las 20 predicciones, el modelo clasificó correctamente 18. Sin embargo, 1 predicción fue catalogada como clase 0 y la otra como clase 1.

Parámetro	Valor
Accuracy	0.9394
Precision	0.9368
Recall	0.9333
F1-Score	0.9347

Cuadro VI
PRUEBAS DATASET BILATERAL

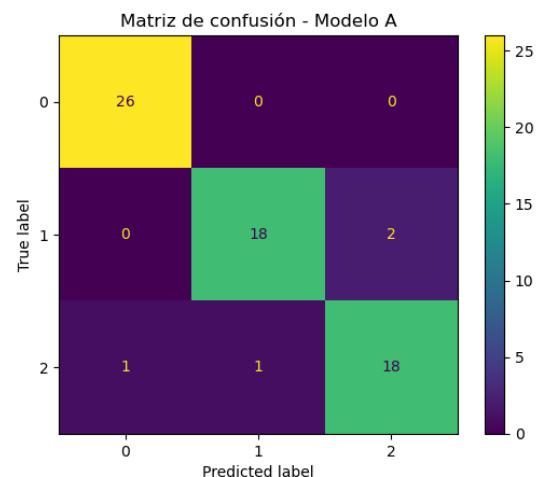


Figura 8. Matriz de Confusión - Bilateral - Modelo A

IV-E3. Dataset Canny: En el Cuadro VII, se observa lo siguiente:

- **Accuracy:** El modelo clasifica correctamente el 83,33 % de las predicciones. Es una precisión aceptable pero inferior al conjunto bilateral.

- **Precision:** Todas las predicciones que el modelo hizo como positivas, el 88,17 % fueron correctas. Indica que el modelo tiene pocos falsos positivos y es bastante fiable al hacer predicciones positivas.
- **Recall:** El modelo es capaz de identificar correctamente el 82,44 % de los verdaderos positivos. Esto indica que el modelo está perdiendo algunos positivos verdaderos, lo que genera falsos negativos.
- **F1-Score:** El balance del modelo corresponde a 81,80 %, indicando un balance aceptable entre el *precision* y *recall*.

En la matriz de confusión que se aprecia en la Figura 9, se observa un rendimiento general aceptable. Para clase 0 (Covid), el modelo clasifica correctamente 24 de las 26 predicciones de la clase Covid y comete solo 2 al predecirlas como la clase Neumonía.

Para la clase 1 (Normal), es donde el modelo presenta más dificultades para predecir. De las 20 predicciones, solo 11 fueron clasificadas correctamente, mientras que las otras 9 fueron incorrectamente clasificadas como clase Neumonía.

En la clase 2 (Neumonía), El modelo clasificó correctamente las 20 instancias de la clase Neumonía sin errores.

Parámetro	Valor
Accuracy	0.8333
Precision	0.8817
Recall	0.8244
F1-Score	0.8180

Cuadro VII
PRUEBAS DATASET CANNY

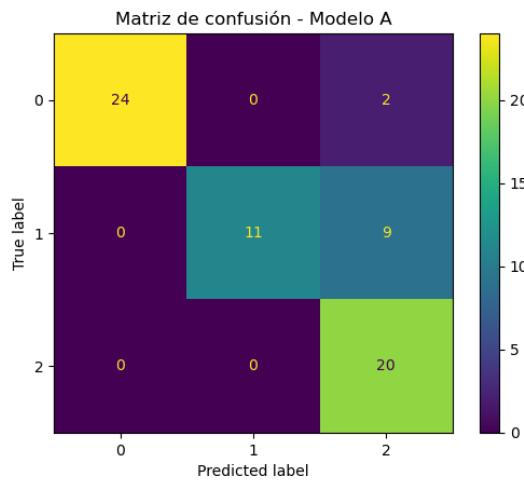


Figura 9. Matriz de Confusión - Canny - Modelo A

V. PROPUESTAS DE ARQUITECTURA DEL MODELO B

Para el modelo B se proponen tres arquitecturas, con las cuales se realizaron tres *proof of concept* con el fin de evaluar cual tiene el mejor rendimiento.

V-A. Arquitectura 1

Para la propuesta primera propuesta del modelo B tenemos la siguiente arquitectura:

- La primera capa es una capa de convolución de dos dimensiones que recibe la imagen de 1 canal (en escala de grises), aplica 64 filtros con tamaño de *kernel* 3 y *padding* de 1, lo que da como resultado la salida de 128x128x64.
- La segunda capa aplica *max pooling* con un *kernel* de 2 y *stride* de 2, lo que produce una salida de 64x64x64.
- Seguido de esto se tiene el módulo de *Inception*, que aplica varias operaciones de convolución y *pooling* en paralelo, lo que permite que el modelo aprenda mejor las características a diferentes niveles de complejidad. Cuenta con 4 ramas, en las que se aplican diversos tamaños de *kernels* y la salida de las ramas es de 64x64x128. Todas estas se concatenan y su salida es de 64x64x512.
- Después se vuelve a aplicar *max pooling* con *kernel* de 2 y *stride* de 2, lo que da como resultado una salida de 32x32x512, se hace el *flatten* de la misma y esta es la entrada de la primera capa *fully connected*.
- La capa primer capa *fully connected* produce 128 canales de salida. La última capa tiene 128 canales de entrada y 3 de salida correspondientes a las tres categorías de imágenes.

Se utiliza reLU como función de activación para las capas convolucionales y la primera capa *fully connected*. Todo esto se ve reflejado en la figura 10.

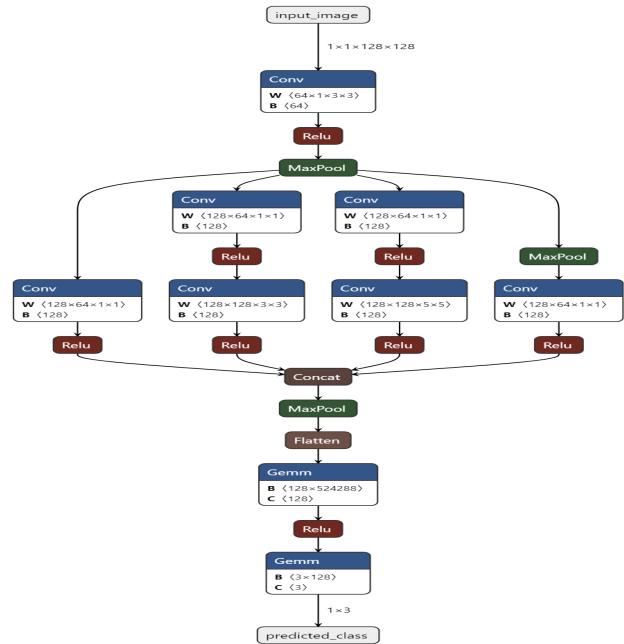


Figura 10. Propuesta de arquitectura #1

V-B. Arquitectura 2

Segunda propuesta de modelo B. Esta arquitectura de modelo es más compleja que la anterior, se aplican otras técnicas de normalización que ayuden al modelo a generalizar mejor los datos, en este se incluye un único módulo de *Inception*, además de varias capas de convolución, *max pooling* y *dropout*. Además se utiliza *Batch Normalization*, la cual es una técnica para normalizar las activaciones en capas intermedias de redes neuronales profundas, además ha demostrado mejorar el accuracy y acelerar el proceso de entrenamiento. [5]

- En primer lugar se tiene una capa de convolución en la que se aplican 32 filtros, con un *kernel* de 3x3 y *padding* de 1, su salida es de 128x128x32.
- La segunda capa de convolución aplica 64 filtros con un *kernel* de 3 y *padding* de 1, su salida es de 128x128x64.
- Seguido se aplica *max pooling* con *kernel* de 2 y *stride* de 2, la salida es de 64x64x64.
- Luego se aplica el 1er *dropout* con una probabilidad de 0.2.
- Seguido de esto se tiene el módulo de *Inception*, que aplica varias operaciones de convolución y *pooling* en paralelo. Cuenta con 4 ramas, la primera produce 32 *feature maps*, la segunda 64, la tercera 64 y la cuarta 32, se concatenan y dan como salida 64x64x192.
- La tercera convolución produce una salida de 64x64x128.
- Luego se aplica de nuevo *max pooling* lo que reduce las dimensiones a 32x32x128.
- Seguido, se aplica la cuarta convolución, que produce como salida 32x32x256.
- Se aplica después el segundo *dropout* con una probabilidad de 0.25.
- Se hace el *flatten*, la primera *fully connected* produce 512 salidas, la segunda produce 128 y la tercera produce 3 salidas correspondientes con las categorías de imágenes.

Se aplica la función de activación *ReLU* en las capas convolucionales y *fully connected*. Después de cada convolución se aplica la función *BatchNorm2d*. Se puede ver la arquitectura definida en la figura 11.

V-C. Arquitectura 3

Tercera propuesta de modelo B. Esta arquitectura de modelo es más compleja, se aplican técnicas de normalización que ayuden al modelo a generalizar mejor los datos como *BatchNorm* y *Dropout* para evitar el overfitting. Se aplica *Global Average Pooling*, que es otra estrategia que ayuda a evitar el overfitting en capas *fully connected*, además de que ayuda a reducir el almacenamiento necesario para matrices de gran tamaño en capas *fully connected*, sirviendo como un reemplazo de las mismas. [6] También incluimos dos módulos de *Inception* para extraer una mayor cantidad de *features* de las imágenes, y así tener un modelo con una mayor capacidad de aprendizaje.

- En primer lugar se tiene una capa de convolución en el que se aplican 32 filtros, con un *kernel* de 3x3 y *padding* de 1, su salida es de 128x128x32.

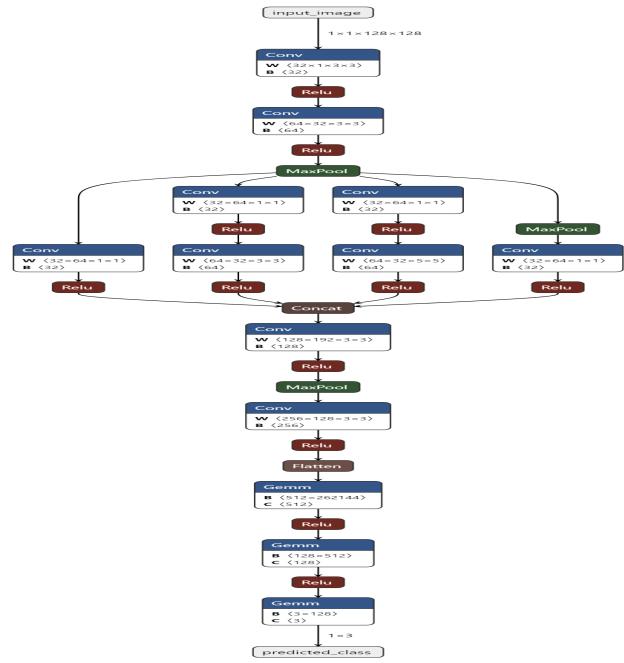


Figura 11. Propuesta de arquitectura #2

- La segunda capa de convolución aplica 64 filtros con un *kernel* de 3 y *padding* de 1, su salida es de 128x128x64.
- Seguido se aplica *max pooling* con *kernel* de 2 y *stride* de 2, la salida es de 64x64x64.
- Luego se aplica el primer módulo de *Inception*, que aplica varias operaciones de convolución y *pooling* en paralelo. Cuenta con 4 ramas, la primera produce 32 *feature maps*, la segunda 64, la tercera 32 y la cuarta 32, se concatenan y dan como salida 64x64x160.
- Seguido se aplica el segundo *max pooling* con *kernel* 2 y *stride* 2, la salida es de 32x32x160.
- Luego seguimos con el segundo módulo de *Inception*, que aplica varias operaciones de convolución y *pooling* en paralelo. Cuenta con 4 ramas, la primera produce 64 *feature maps*, la segunda 128, la tercera 64 y la cuarta 64, se concatenan y dan como salida 32x32x320.
- Despues se aplica otro *max pooling* con *kernel* 2 y *stride* 2, la salida es de 16x16x320.
- Posteriormente se aplica el primer *dropout* con probabilidad de 0.3.
- Despues se aplica la capa de *global average pooling*, la salida es de 320x1x1.
- Seguido de esto se hace el *flatten* y la salida se le da a la primera capa *fully connected*.
- La primera *fully connected* recibe 320 entradas y produce 128 salidas.
- Despues se aplica la segunda capa de *dropout* con probabilidad de 0.4.
- Seguido de esto, se pasa a la segunda capa *fully connected* que recibe 128 entradas y produce 64 salidas.
- Finalmente se pasa a la tercer y última capa *fully connected* que produce 3 salidas.

connected que recibe 64 entradas y produce 3 salidas correspondientes a las clases del set de datos.

Se aplica la función de activación *ReLU* en las capas convolucionales y *fully connected*.

Después de cada convolución se aplica la función *Batch-Norm2d*.

Se puede observar esta estructura en la figura 12

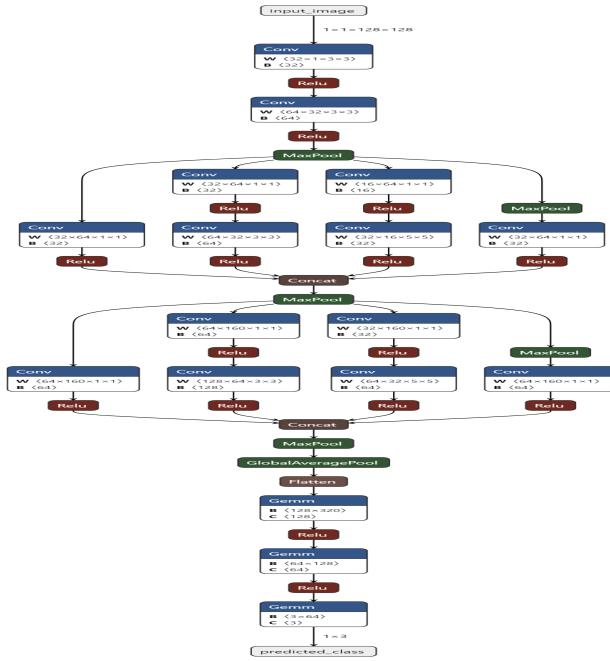


Figura 12. Propuesta de arquitectura #3

V-D. Selección de modelo

Para realizar la selección del modelo se realizaron tres *proof of concept* con un 20 % de los datos de entrenamiento y aplicando pocas transformaciones. Se realizó la prueba con 50 *epochs*, en *batches* de 8 imágenes, esto último debido a que se tienen muchos menos datos de entrenamiento para las pruebas, pasando de 251 a 52.

Además se guarda el el modelo que de mejor resultado de validación en un *epoch* específico, para comparar los mejores resultados de cada uno.

V-D1. Proof of Concept Arquitectura 1: Para esta prueba se utilizó el optimizador SGD, *Stochastic Gradient Descent* del módulo *optim* de *pytorch* con un *learning rate* de 0.001 y un *momentum* de 0.9. Este último es una técnica que se usa en Stochastic Gradient Descent para acelerar la convergencia del proceso de optimización. Se agrega una fracción de la actualización anterior a la actualización actual de los parámetros del modelo. [7]

Se entrenó al modelo con los datos y se midieron los resultados utilizando la métrica de *accuracy*.

En este caso, el mejor resultado para la validación fue de 84.62 %, lo que significa que de los ejemplos de *testing*, las clasificaciones fueron correctas en ese porcentaje.

V-D2. Proof of Concept Arquitectura 2: Se procede a realizar la prueba de concepto con el segundo modelo planteado.

Para el optimizador se utiliza el **Adam** que nos facilita el módulo *optim* de *pytorch* en este caso, este nos puede ayudar a converger más rápido , además cómo función de pérdida se utiliza *Cross Entropy Loss*. En este caso el *learning rate* se mantiene igual en un valor de 0.001.

Se entrenó al modelo con los datos y se midieron los resultados utilizando la métrica de *accuracy*.

En este caso, el mejor resultado para la validación fue de 92.31 %, lo que significa que de los ejemplos de *testing*, las clasificaciones fueron correctas en ese porcentaje.

Además esto representa una mejora con respecto a la prueba del primer modelo.

V-D3. Proof of Concept Arquitectura 3: Se procede a realizar la prueba de concepto con el segundo modelo planteado.

Para el optimizador se utiliza el mismo optimizador de la segunda prueba, **Adam** y el mismo *learning rate* de 0.001.

Se entrenó al modelo con los datos y se midieron los resultados utilizando la métrica de *accuracy*.

En este caso, el mejor resultado para la validación fue de 100 %, lo que significa que de los ejemplos de *testing*, las clasificaciones fueron correctas en ese porcentaje.

Además esto representa una mejora con respecto a la prueba del segundo modelo, siendo este el mejor de los tres.

V-D4. Comparación de resultados: La arquitectura del modelo que arroja los mejores resultados es la tercera, en la cual se tienen dos módulos de *Inception*, *Batch Normalization*, y *Dropout*.

Es por esta razón que se selecciona cómo modelo final y se procede a realizar el entrenamiento con los tres conjuntos de datos; crudos, bilateral y canny.

VI. MODELO B

VI-A. Entrenamiento

Se realizaron tres entrenamientos con el modelo B seleccionado.

Para las imágenes de entrenamiento, se aplica las que se vieron en la sección del *Data Augmentation*, ver Cuadro I. Se aplica el *resize* de 128x128 puesto que es la entrada que el modelo recibe. En este caso los datos se cargan en *batches* de 32 imágenes.

Los pasos realizados en el entrenamiento del Modelo B son los siguientes:

- **Ciclo de entrenamiento:** El entrenamiento finaliza hasta concluir el número de *epochs* estipulado. Por cada *epoch*, el modelo B seleccionado se pone en modo de entrenamiento y se empiezan a recorrer los datos de entrenamiento. El modelo genera predicciones para los *inputs* y *labels* de cada *batch* y se calcula la función de pérdida. Se realiza el *backpropagation* y se actualizan los pesos.
- **Ciclo de validación:** Después, en el mismo *epoch* se evalúa el modelo en el conjunto de validación con el modelo en modo evaluación. Durante la validación, no

se calculan los gradientes. Similar a la fase de entrenamiento se calcula la pérdida y el *accuracy*, pero esta vez sobre los datos de validación.

- **Mejor modelo:** Durante cada *epoch* en la validación, si el *accuracy* ha sido el mejor, se guarda el estado del modelo actual como el mejor modelo.

VI-B. Resultados de la fase de entrenamiento

VI-B1. Entrenamiento con dataset crudo: En la tabla VIII se muestran los parámetros utilizados así como los resultados del mejor *epoch* que en este caso fue el 47.

En este [enlace](#) se muestran las métricas que se registraron utilizando la herramienta de Weights and Biases.

Parámetro	Valor
N_Epochs	50
Optimizador	Adam
Loss Function	Cross Entropy Loss
Learning rate	0.001
Epoch	47
Train Loss	0.1872
Train Acc	0.9442
Val Loss	0.0696
Val Acc	0.9697

Cuadro VIII
ENTRENAMIENTO DATASET CRUDO

VI-B2. Entrenamiento con dataset bilateral: En la tabla X se muestran los parámetros utilizados así como los resultados del mejor *epoch* que en este caso fue el 12.

En este [enlace](#) se muestran las métricas que se registraron utilizando la herramienta de Weights and Biases.

Parámetro	Valor
N_Epochs	50
Optimizador	Adam
Loss Function	Cross Entropy Loss
Learning rate	0.001
Epoch	12
Train Loss	0.1566
Train Acc	0.9283
Val Loss	0.0525
Val Acc	1

Cuadro IX
ENTRENAMIENTO DATASET BILATERAL

VI-B3. Entrenamiento con dataset canny: En la tabla X se muestran los parámetros utilizados así como los resultados del mejor *epoch* que en este caso fue el 45.

En este [enlace](#) se muestran las métricas que se registraron utilizando la herramienta de Weights and Biases.

VI-C. Resultados de la fase de pruebas

Para el *testing* se toma en cuenta el modelo que se guarda en el *epoch* con el mejor valor de validación para cada entrenamiento.

Parámetro	Valor
N_Epochs	50
Optimizador	Adam
Loss Function	Cross Entropy Loss
Learning rate	0.001
Epoch	12
Train Loss	0.2776
Train Acc	0.8845
Val Loss	0.5720
Val Acc	0.8939

Cuadro X
ENTRENAMIENTO DATASET BILATERAL

VI-C1. Resultados dataset crudo: Para el *precision* se tiene un valor de 0.9667 lo que indica que de los ejemplos clasificados como Covid, Normal y Pneumonia, el 96.67 % realmente pertenecen a esas clases.

Para el *F1-Score* se tiene un valor de 0.9667 lo que indica un buen balance entre la precisión y el *recall*.

Para el *recall* se tiene un 0.9667 lo que indica que de todas las instancias de Covid, Normal y Pneumonia el 96.67 % se clasificaron correctamente como Covid, Normal y Pneumonia.

Para el *accuracy* se tiene un 0.9697 lo que indica que de todas las predicciones realizadas, un 96.97 % son correctas.

Se pueden observar estos datos en la figura XI.

Métrica	Valor
Accuracy	0.9697
Precision	0.9667
F1-Score	0.9667
Recall	0.9667

Cuadro XI
MÉTRICAS dataset CRUDO

También podemos ver la Matriz de Confusión en la figura 13, donde podemos ver que todos los ejemplos de Covid se clasificaron correctamente, de los ejemplos Normales, 19 se clasificaron correctamente y uno se clasificó como Pneumonia. Para los ejemplos de Pneumonia, 19 se clasificaron correctamente y 1 se clasificó como Normal.

VI-C2. Resultados dataset bilateral: Para el *precision* se tiene un valor de 1 lo que indica que de los ejemplos clasificados como Covid, Normal y Pneumonia, el 100 % realmente pertenecen a esas clases.

Para el *F1-Score* se tiene un valor de 1 lo que indica un excelente balance entre la precisión y el *recall*.

Para el *recall* se tiene un 1 lo que indica que de todas las instancias de Covid, Normal y Pneumonia el 100 % se clasificaron correctamente como Covid, Normal y Pneumonia.

Para el *accuracy* se tiene un 1 lo que indica que de todas las predicciones realizadas, un 100 % son correctas.

Se pueden observar estos datos en la figura XII.

También podemos ver la Matriz de Confusión de la figura 14, donde podemos ver que todos los ejemplos de Covid, Normal Y Pneumonia se clasificaron correctamente.

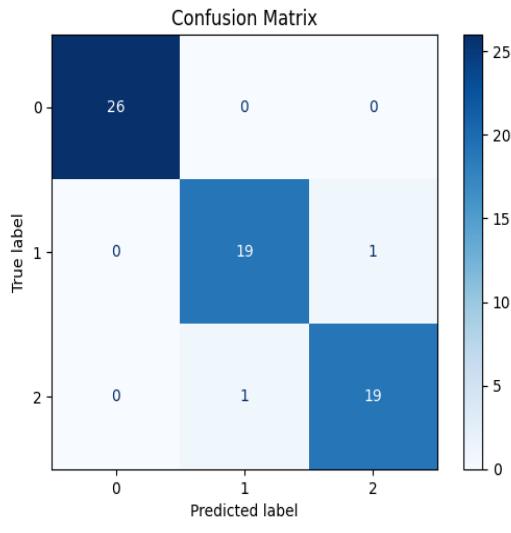


Figura 13. Matriz de Confusión Crudo

Métrica	Valor
Accuracy	1
Precision	1
F1-Score	1
Recall	1

Cuadro XII
MÉTRICAS dataset BILATERAL

VI-D. Resultados dataset canny

Para el *precision* se tiene un valor de 0.8960 lo que indica que de los ejemplos clasificados como Covid, Normal y Pneumonia, el 89.60 % realmente pertenecen a esas clases.

Para el *F1-Score* se tiene un valor de 0.8816 lo que indica un buen balance entre la precisión y el *recall*, pero menor que los otros dos conjuntos de datos.

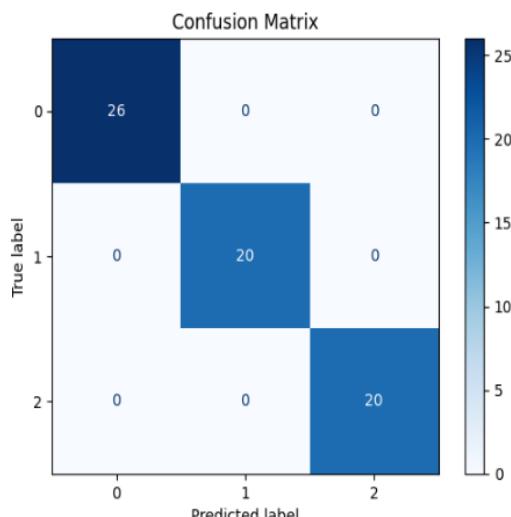


Figura 14. Matriz de Confusión Bilateral

Para el *recall* se tiene un 0.8833 lo que indica que de todas las instancias de Covid, Normal y Pneumonia el 88.33 % se clasificaron correctamente como Covid, Normal y Pneumonia, de igual manera menor que los otros dos.

Para el *accuracy* se tiene un 0.8939 lo que indica que de todas las predicciones realizadas, un 89.39 % son correctas.

Se pueden observar estos datos en la figura XIII.

Métrica	Valor
Accuracy	0.8939
Precision	0.8960
F1-Score	0.8816
Recall	0.8833

Cuadro XIII
MÉTRICAS dataset CANNY

También podemos ver la Matriz de Confusión en la figura 15, donde podemos ver que todos los ejemplos de Covid se clasificaron correctamente, de los ejemplos Normales, 14 se clasificaron correctamente, uno se clasificó como Covid y 5 se clasificaron como Pneumonia. Para los ejemplos de Pneumonia, 19 se clasificaron correctamente y 1 se clasificó como Normal.

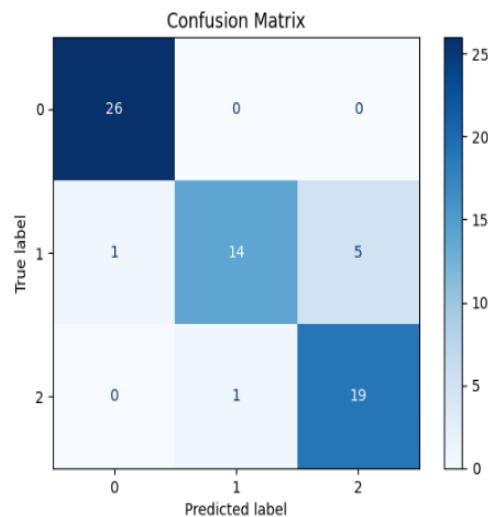


Figura 15. Matriz de Confusión Canny Edge

VI-D1. Comparación de resultados modelo B: Los mejores resultados para el modelo B se obtuvieron con el entrenamiento utilizando el *dataset* con el filtro *bilateral* aplicado a las imágenes, ya que en el mejor *poch* se logró clasificar correctamente el 100 % de las imágenes.

En segundo lugar se encuentra el *dataset* crudo, con un 96.67 % en sus métricas, lo que lo hace también una excelente alternativa para entrenar el modelo.

En tercer y último lugar se tuvo el *dataset* con el filtro *canny edge* aplicado a las imágenes, con un 89.60 % en *precision* y un poco menos en *f1-score* y *recall*, lo cual no lo hace malo, pero se tienen mejores resultados con los otros dos.

Se pueden ver los detalles del entrenamiento en el siguiente [enlace](#), como se ve en la figura 16

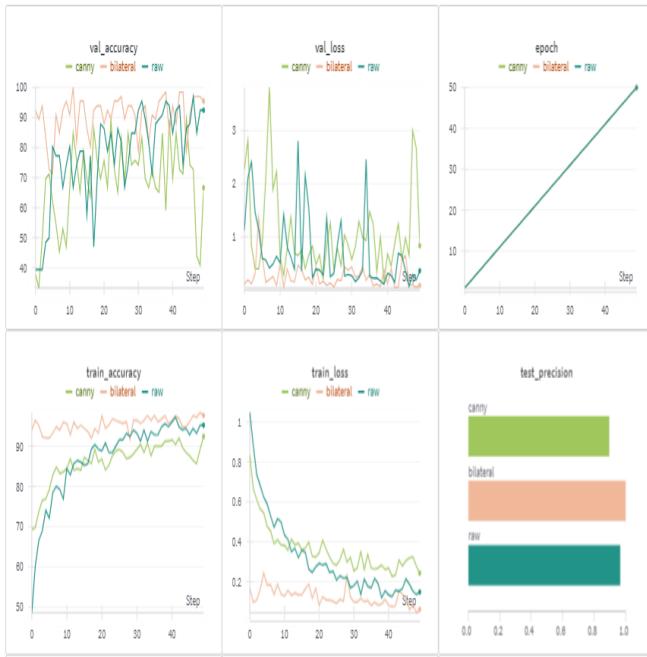


Figura 16. Comparación de entrenamientos

Allí se aprecia que el entrenamiento con *bilateral* tiene la menor pérdida y el mayor *accuracy*.

VII. DISCUSIÓN

VII-A. Modelo A

Para los resultados del modelo A, se tiene que las mejores métricas se obtuvieron con el entrenamiento y *testing* utilizando el *dataset* con el filtro *bilateral*. Las métricas de *accuracy*, *f1-score*, y *precision* son excelentes.

En el ámbito médico es de suma importancia el *recall*, dado que queremos clasificar cuando las clases son positivas; puesto que es más grave clasificar a una persona con Covid o Pneumonia como Normal, que clasificar a alguien Normal como que tuviera alguno de los padecimientos.

Para el *bilateral* la métrica de *recall* es de 0.9333 lo cual es la mejor de los tres entrenamientos realizados.

En segundo lugar se encuentra el entrenamiento con el *dataset* del filtro *canny edge detection*, las métricas obtenidas son buenas en general, tanto *accuracy*, *precision* y *f1-score*. Además, su *recall* es de 0.8244, siendo este también bueno, pero menor que el *bilateral*.

Además, los peores resultados para el modelo A se dieron con el *dataset* de datos crudos, siendo que para este los valores de *accuracy*, *f1-score* y *recall* son más bajos que los demás, y el *precision* es el único que se encuentra en el rango de 0.8 - 0.9 con un valor de 0.8571,

VII-B. Modelo B

Para el modelo B, las mejores métricas también se obtuvieron en el entrenamiento y *testing* utilizando el *dataset* con el filtro *bilateral*, siendo que se obtuvieron métricas del 100 %.

Como ya se dijo anteriormente, en este tipo de problemas de clasificación de problemas médicos, el *recall* es una métrica de suma importancia, puesto que es más grave clasificar a una persona con Covid o Pneumonia como Normal, que clasificar a alguien Normal como que tuviera alguno de los padecimientos.

En este caso el entrenamiento con el *dataset bilateral* nos da el mejor *recall* para el conjunto de datos, siendo este de un valor de 1.

En segundo lugar se encuentra el entrenamiento con el *dataset* crudo, las métricas obtenidas son buenas en general, tanto *accuracy*, *precision* y *f1-score*. Además, su *recall* es de 0.9667, muy cercano al *bilateral*.

En contraste con el modelo A, los peores resultados del modelo B se dieron cuando se entrenó con el *dataset* al que se le aplicó el filtro *canny edge detection*, ya que las métricas no llegan al 0.9, sin embargo siguen siendo buenos resultados de *accuracy*, *precision* y *f1-score*. Pero el *recall* es el más bajo de los tres con un valor de 0.8833.

VII-C. Mejor modelo general

En general los mejores resultados se presentaron al entrenar ambos modelos con el *dataset* aplicando el filtro de *bilateral*.

En la figura 17 se muestran las curvas de *validation* para el modelo A, siendo que la curva se aplana paulatinamente en el *loss*. Para el *accuracy*, este primero sube, se observan unos pequeños bajones y luego se mantiene constante.

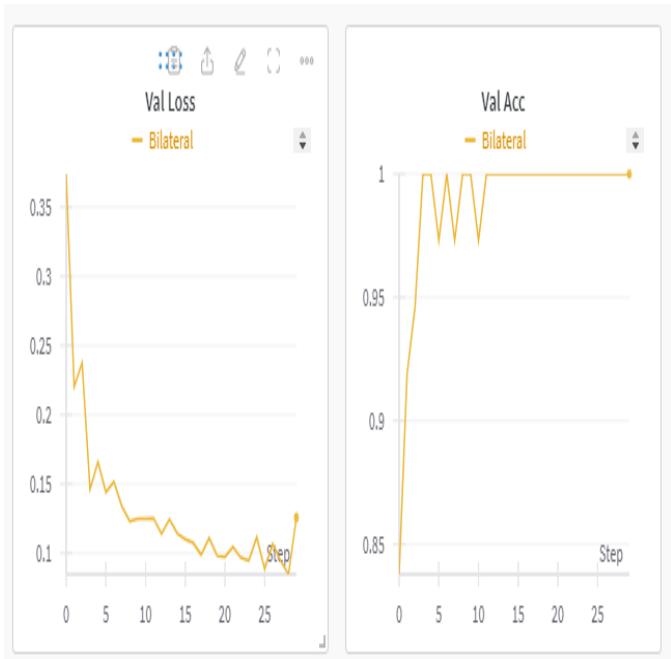


Figura 17. Curvas de *validation* en el modelo A

En la figura 18 se muestran las curvas de *validation* para el modelo B, siendo que la curva se aplana paulatinamente en el *loss*, pero en este caso se presentan picos en ciertos *epochs*, lo que puede deberse a la aleatoriedad y a la aplicación de capas como *dropout*. Para el *accuracy* este varía bastante, cuando

hay bajones estos se pueden producir porque el modelo no está generalizando bien en ciertos *epochs* pero luego se recupera en los siguientes, lo que le da el potencial de tener un mayor valor de accuracy que el modelo A, que se mantiene constante a partir de cierto punto.

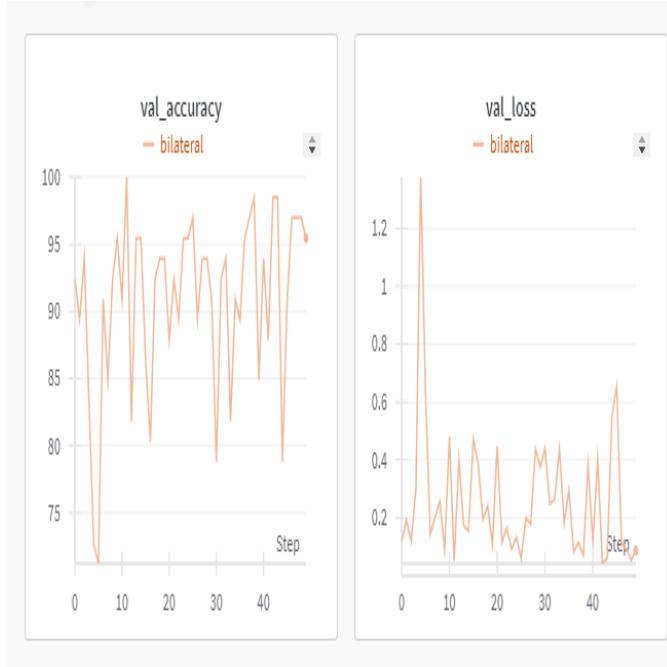


Figura 18. Curvas de *validation* en el modelo B

Comparando ambos modelos y su rendimiento utilizando este conjunto de datos para las pruebas, se tiene que el modelo B obtuvo los mejores resultados para la clasificación, con un *precision* de 1, *recall* de 1 y *F1-Score* de 1 para los datos de *testing*.

VIII. CONCLUSIONES

- Para ambos modelos, se concluye que el entrenamiento con el *dataset bilateral* arroja las mejores métricas en comparación con el *canny edge* y el crudo.
- La métrica de *recall* es de suma importancia a la hora de seleccionar un modelo para la clasificación de datos médicos, tanto el modelo A como el modelo B presentan excelentes índices de *recall* utilizando el filtro *bilateral*.
- Las técnicas de *Data Augmentation* le permiten al modelo mejorar su capacidad de generalización, creando pequeñas modificaciones a los datos existentes, aumentando la cantidad de ejemplos, y haciendo más robusto el modelo a posible ruido o distorsiones de datos que le puedan llegar como entrada.
- El modelo A, presentó dificultades especialmente para distinguir la clase Normal, ya que en los 3 entrenamientos, se equivocó bastante prediciendo muchas imágenes como clase Neumonía cuando realmente pertenecían a la clase Normal.
- En el preprocessamiento, el aplicar el *resize* antes de aplicar los respectivos filtros, hizo que los resultados de

aplicarlos hayan sido mejores que si se hubieran aplicado los filtros antes de realizar el *resize*

- El entrenamiento del Modelo B fue mucho más tardado que el modelo A debido a que este se hizo usando la plataforma *Colab* y no mediante GPU como fue en el caso del Modelo A.

REFERENCIAS

- [1] B. AI, “Intro a las redes neuronales convolucionales,” Disponible: <https://medium.com/@bootcampai/redes-neuronales-convolucionales-5e0ce960caf8>, [Accesado: Oct. 16, 2024].
- [2] P. Raikote, “Covid-19 image dataset,” Disponible: <https://www.kaggle.com/datasets/pranavraikokte/covid19-image-dataset>, [Accesado: Sept. 16, 2024].
- [3] A. Roldán, “Guía completa sobre pytorch para principiantes,” Disponible: <https://aplicaciones-ai.com/pytorch/>, [Accesado: Oct. 16, 2024].
- [4] Sumitkrsharma, “Image filtering in computer vision,” Disponible: <https://www.freedium.cfd/https://sumitkrsharma-ai.medium.com/image-filtering-in-computer-vision-ec60ec8a3e1>, [Accesado: Sept. 16, 2024].
- [5] J. Bjorck, C. Gomes, B. Selman, and K. Q. Weinberger, “Understanding batch normalization,” *arXiv (Cornell University)*, 1 2018. [Online]. Available: <https://arxiv.org/abs/1806.02375>
- [6] T.-Y. Hsiao, Y.-C. Chang, H.-H. Chou, and C.-T. Chiu, “Filter-based deep-compression with global average pooling for convolutional networks,” *Journal of Systems Architecture*, vol. 95, pp. 9–18, 2 2019. [Online]. Available: <https://doi.org/10.1016/j.sysarc.2019.02.008>
- [7] ListenToTheUniverse, “Stochastic Gradient Descent with momentum - ListenToTheUniverse - Medium,” 4 2023. [Online]. Available: https://medium.com/@ebinbabuthomas_21082/stochastic-gradient-descent-with-momentum-e3b87f30eca9#:~:text=A%20Momentum%20is%20a%20technique,update%20of%20the%20model%20parameters.

IX. RÚBRICA

Criterios	Puntuación máxima	Puntuación obtenida
Preprocesamiento aplicando filtros	5	
Data Augmentation	5	
Modelo A		
Criterios	Puntuación máxima	Puntuación obtenida
Entrenamiento con datasets	5	
Presentación de métricas y matriz de confusión	5	
Comparación de resultados	10	
Modelo B		
Criterios	Puntuación máxima	Puntuación obtenida
Diseño y justificación	15	
Entrenamiento con datasets	5	
Presentación de métricas y matriz de confusión	5	
Comparación de resultados	10	
Aspectos generales		
Criterios	Puntuación máxima	Puntuación obtenida
Comparación de los mejores modelos de A y B	5	
Registro de métricas de todos los experimentos en WaB	20	
Compleitud de entregables	5	
Estructura de artículo científico	5	
Aspectos Extra (Reporte WaB)		
Criterios	Puntuación máxima	Puntuación obtenida
Informe de WaB justificando el modelo con mejores resultados	5	
Presentación de saliency maps	5	
Total	110	