UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3066 – Data Science

Sección 10

Proyecto 2

SEBASTIAN ARISTONDO PEREZ 20880

JOSE DANIEL GONZALEZ CARRILLO 20293

DIEGO JOSE FRANCO PACAY 20240

MANUEL ALEJANDRO ARCHILA MORAN 161250

JUAN DIEGO AVILA SAGASTUME 20090

Investigación de algoritmos

CNN

Una red neuronal convolucional (ConvNet o CNN) es una red neuronal artificial (RNA) que utiliza aprendizaje profundo algoritmos para analizar imágenes, clasificar elementos visuales y realizar tareas de visión por ordenador.La CNN aprovecha principios del álgebra lineal, como la multiplicación de matrices, para detectar patrones en una imagen. Como estos procesos implican cálculos complejos, requieren unidades de procesamiento gráfico (GPUs) para entrenar los modelos.(Pathak, 2022)

Capa Convolucional

La primera capa de una CNN es la capa convolucional. Es el bloque principal de la CNN, donde se realizan la mayoría de los cálculos. Necesita menos componentes, como datos de entrada, un mapa de características y un filtro.Una CNN también puede tener capas convolucionales adicionales. Esto jerarquiza la estructura de las CNN, ya que las capas posteriores pueden visualizar píxeles dentro de los campos receptivos de las capas anteriores. A continuación, las capas convolucionales transforman la imagen dada en valores numéricos y permiten a la red comprender y extraer patrones valiosos.(Pathak, 2022)

Capa totalmente conectada

Como su nombre indica, todos los nodos de una capa de salida están conectados directamente al nodo de la capa anterior en una capa totalmente conectada. Clasifica una imagen basándose en las características extraídas a través de las capas anteriores junto con sus filtros. Además, las capas FC suelen utilizar una función de activación softmax para clasificar correctamente las entradas en lugar de funciones ReLu (como en el caso de las capas pooling y convolucionales). Esto ayuda a producir una probabilidad de 0 o 1.(Pathak, 2022)

LSTM Bidireccional

Una LSTM bidireccional (memoria a largo plazo o a corto plazo) es un tipo de arquitectura de red neuronal utilizada en el aprendizaje profundo. Está diseñada específicamente para procesar datos secuenciales, como lenguaje natural o datos de series temporales. Las LSTM bidireccionales funcionan procesando los datos de entrada en dos direcciones: desde el principio hasta el final (dirección hacia adelante) y desde el final hasta el principio (dirección hacia atrás), lo que permite que el modelo capture el contexto pasado y futuro de los datos de entrada. (KeepCoding, 2023)

La salida del LSTM bidireccional se genera concatenando las salidas de las capas LSTM hacia delante y hacia atrás. Esto permite que el modelo aproveche la información de ambas direcciones, lo que puede mejorar la precisión de las predicciones. Las LSTM bidireccionales se usan comúnmente en tareas como el procesamiento del lenguaje natural, el reconocimiento de voz y el análisis de sentimientos, donde el contexto de los datos de entrada es importante para una predicción precisa.(KeepCoding, 2023)

Para obtener una predicción adecuada, esta red se debe entrenar. El entrenamiento es un proceso de aprendizaje iterativo en el que instancias de datos en las que se conoce la salida correcta se presentan a la red de una en una, ajustando cada vez los pesos asociados con los valores de entrada en cada capa. El proceso se repite varias veces con diferentes lotes de datos de entrada hasta que los pesos se ajustan lo suficientemente bien como para predecir la etiqueta correcta (o probabilidad) de cualquier muestra de entrada. Para crear las instancias de datos de entrenamiento, se registra una serie cronológica para cada ordenador durante un largo periodo de tiempo. A partir de este grupo de datos se extraen segmentos del tamaño apropiado. Estos segmentos se pueden etiquetar, ya que se puede comprobar si se ha producido un aumento en las entradas posteriores. Este conjunto se utiliza para entrenar los pesos en la red profunda.(Garzón, 2018)

YOLO

Es una red neuronal que predice los cuadros delimitadores y las probabilidades de clase a partir de una imagen en una sola evaluación, de allí su nombre en inglés You only look once. Los modelos YOLO pueden procesar más de 60 fotogramas por segundo, por lo que es una arquitectura excelente para detectar objetos en vídeos. El funcionamiento se basa en una red neuronal convolucional, en la que usa características de imagen al completo para trazar los cuadros delimitadores (Bounding Box). Con ello se consigue que la red neuronal reaccione y detecte todos los objetos que hay en una imagen. La imagen se divide en una rejilla de tamaño S x S de forma que, si un objeto cae en una celda de la cuadrícula, esa celda es responsable de detectar ese objeto.(Telecomunicación et al., 2021)

Por otra parte, cada celda de la cuadrícula predice B cuadros delimitadores con sus puntuaciones confianza (score). Estas puntuaciones de confianza reflejan cómo de fiable es el modelo de si esa celda contiene un objeto y también cómo de preciso es al predecirlo. Formalmente definimos la confianza como Pr (Objeto)*IOU. Si no hay

un objeto en esa celda, las puntuaciones de confianza deben ser cero. De lo contrario, si hay un objeto en dicha celda, queremos que la puntuación de confianza sea igual a la intersección sobre unión (IOU). Un poco más adelante, lo explicaremos con mayor detalle. (Telecomunicación et al., 2021)

YOLO es construido en base de una red neuronal convolucional, ya que solo contiene capas convolucionales, a menudo se le suele denominar FCN o fully convolutional network. En las capas iniciales se extraen las características de la imagen y las capas completamente conectadas predicen las probabilidades de salida y las coordenadas. Esta red está inspirada en el modelo GoogleNet usado en la clasificación de imágenes. (Telecomunicación et al., 2021)

En cuanto a la función de pérdida, en YOLO se predicen muchos cuadros delimitadores (Bounding Box) por celda. Para calcular la pérdida de un verdadero positivo, solo se tiene en cuenta uno de los cuadros delimitadores, de todos los posibles y es aquel que al calcular la intersección sobre la unión IOU ofrece el resultado más alto con el cuadro de la anotación.(Telecomunicación et al., 2021)

Redes Neuronales Recurrentes (RNN)

Las Redes Neuronales Recurrentes, abreviadas como RNN, son un tipo de arquitectura de red neuronal ampliamente utilizada en el campo del aprendizaje profundo, especialmente en tareas que involucran secuencias de datos, como el procesamiento de lenguaje natural, el análisis de series temporales y la generación de texto, entre otros.

A diferencia de las redes neuronales convolucionales (CNN), que se especializan en el procesamiento de datos en forma de cuadrículas, como imágenes, las RNN están diseñadas para manejar datos secuenciales de longitud variable. Lo que hace que las RNN sean particularmente poderosas es su capacidad para modelar dependencias temporales en los datos. Esto significa que pueden recordar y utilizar información de pasos de tiempo anteriores en la secuencia para tomar decisiones en pasos de tiempo posteriores.

El componente clave que distingue a las RNN de otras redes neuronales es la introducción de "retroalimentación". Cada neurona en una capa RNN recibe una entrada, pero también toma como entrada la salida de su propia iteración anterior, lo que crea un bucle temporal en la red. Esta retroalimentación le permite a la RNN mantener una especie de "memoria" a corto plazo de eventos anteriores en la secuencia. Es esta capacidad de mantener estados internos que permite a las RNN capturar patrones y relaciones en datos secuenciales.

Sin embargo, las RNN tradicionales tienen limitaciones, como el problema del desvanecimiento del gradiente, que dificulta la captura de dependencias a largo plazo. Para abordar estas limitaciones, se han desarrollado variantes de RNN más avanzadas, como las Redes LSTM (Long Short-Term Memory) y las GRUs (Gated Recurrent Units), que son capaces de capturar relaciones a largo plazo de manera más efectiva.

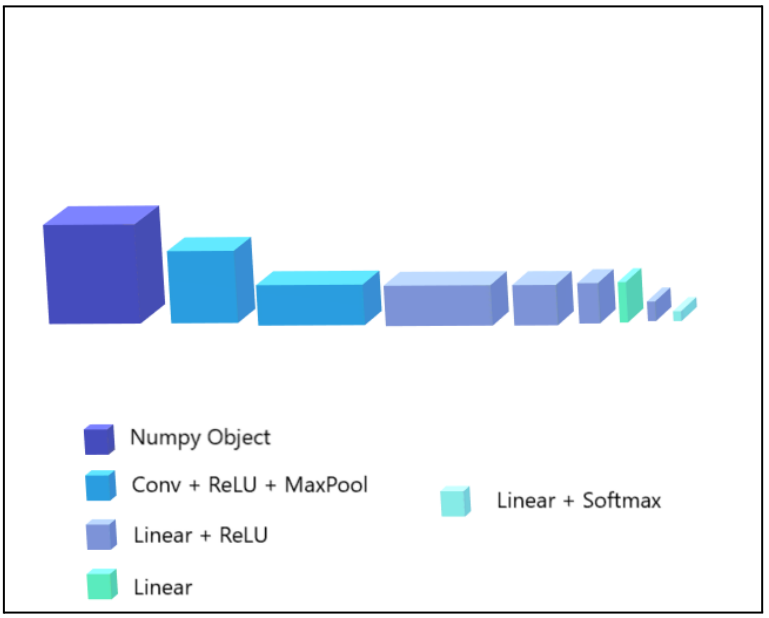
Selección de algoritmos a probar

Dada la naturaleza del conjunto de datos a trabajar estaremos probando 2 arquitecturas que trabajan muy bien con estos datos. El primer acercamiento a probar será la rama de arquitecturas basadas en CNN´s, estas arquitecturas son excelentes para la extracción de características de imágenes. Esto es de gran utilidad pues estamos trabajando con CTI´s y es necesario que el modelo sea capaz de identificar entre 200 a 600 imágenes donde y como se ve una fractura.

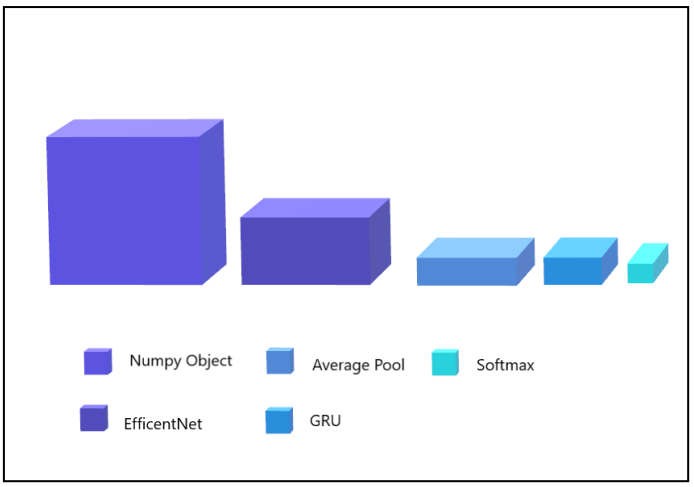
Por otro lado, el segundo acercamiento será la rama de arquitectura basadas en RNN. Cada paciente cuenta con un gran conjunto de CTI´s en las cuales se encuentran todas las cervicales, estas imágenes tienen relación y contexto entre sí. Por lo tanto, si el modelo puede mantener y comprender el contexto entre imágenes ayudará significativamente a procesar las imágenes y predecir de forma más acertada las cervicales que tienen una fractura.

Modelos

Arquitectura 3D CNN



Arquitectura Efficient Net + GRU



Eficiencia de los modelos y visualización estáticas

Matrices de confusión 3DCNN

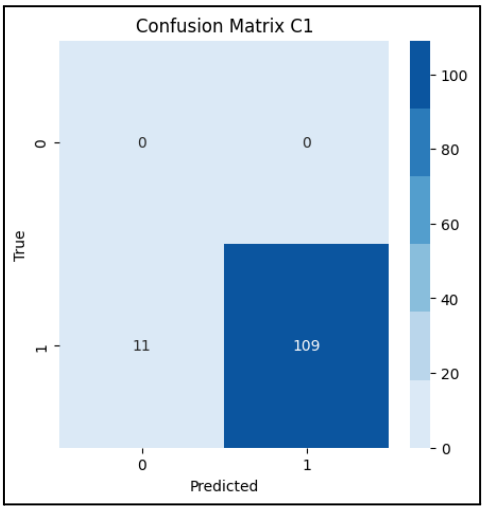


Figura No.1 - Matriz de confusión 3DCNN C1

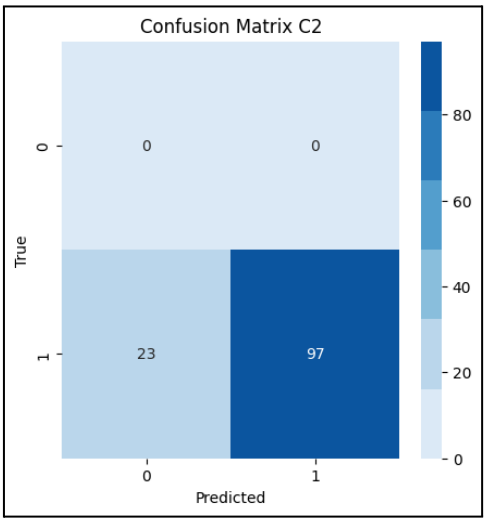


Figura No.2 - Matriz de confusión 3DCNN C2

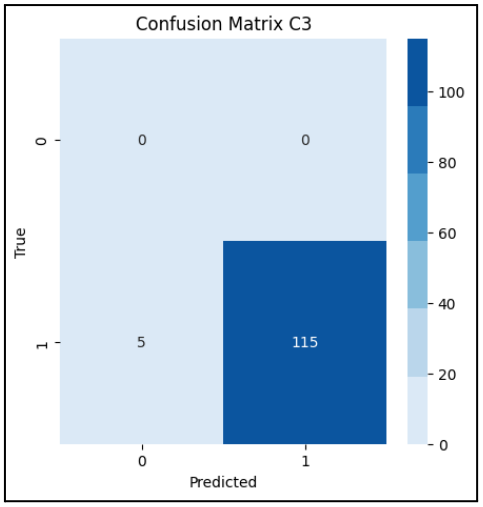


Figura No.3 - Matriz de confusión 3DCNN C3

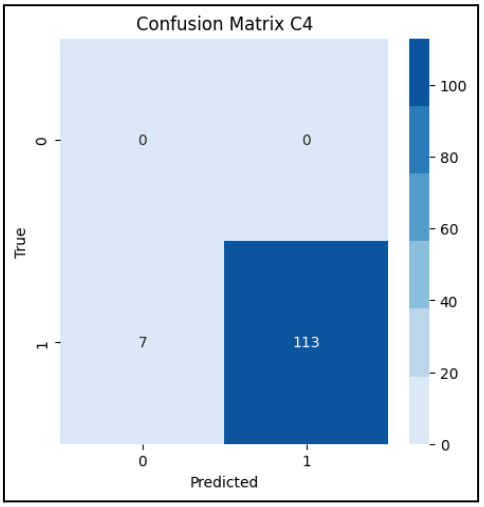


Figura No.4 - Matriz de confusión 3DCNN C4

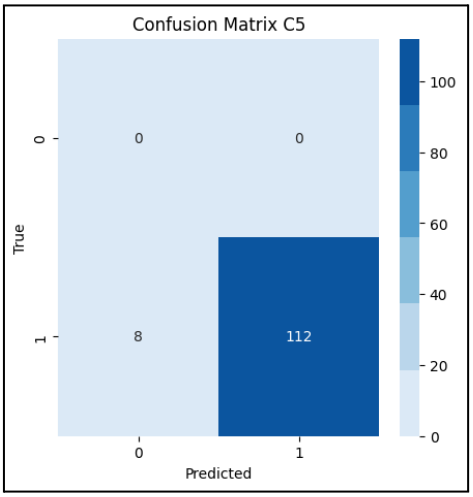


Figura No.5 - Matriz de confusión 3DCNN C5

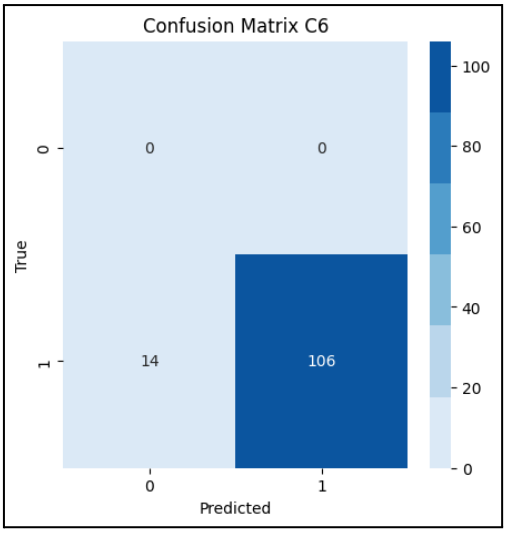


Figura No.6 - Matriz de confusión 3DCNN C6

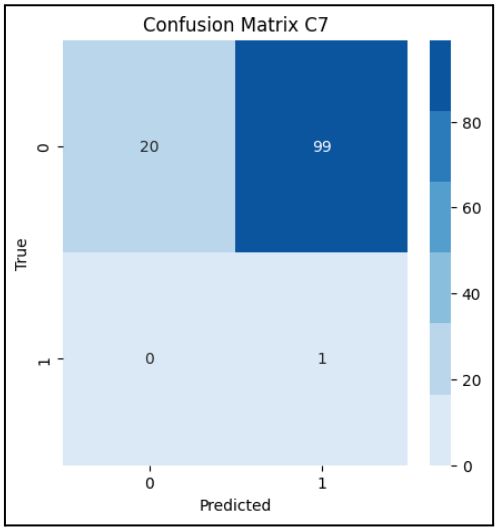


Figura No.7 - Matriz de confusión 3DCNN C7

Matrices de Confusión Efficient Net

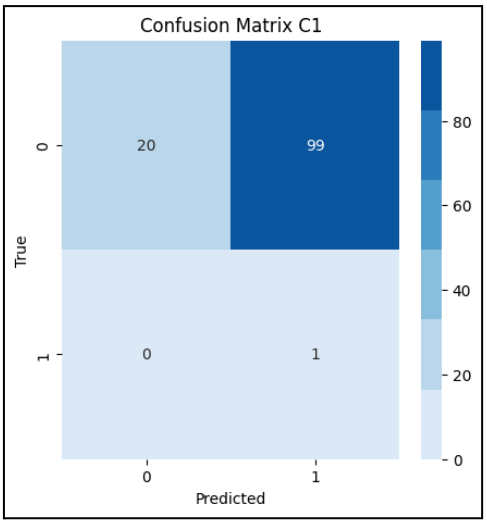


Figura No.8 - Matriz de confusión efficient Net C1

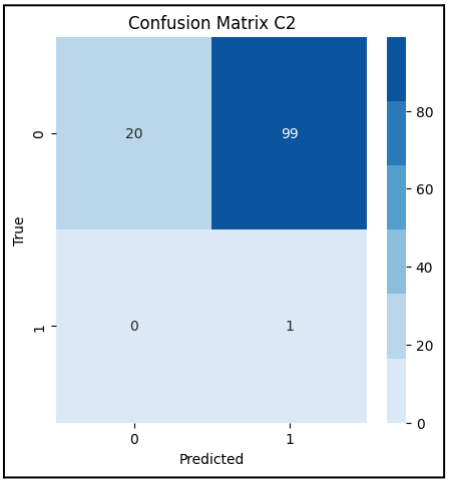


Figura No.9 - Matriz de confusión efficient Net C2

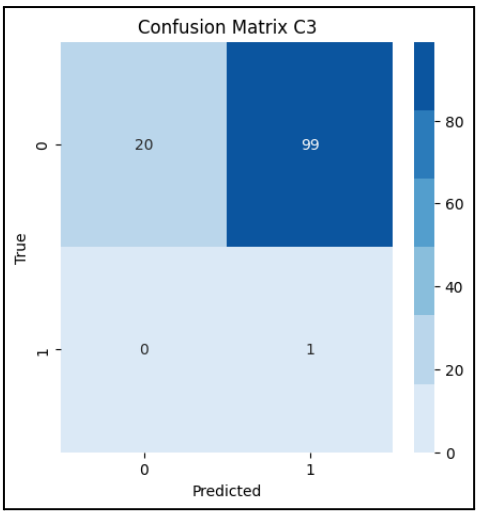


Figura No.10 - Matriz de confusión efficient Net C3

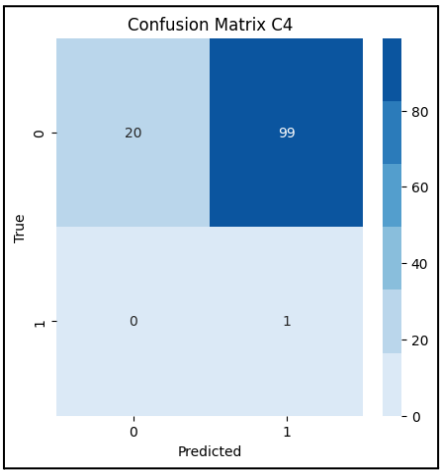


Figura No.11 - Matriz de confusión efficient Net C4

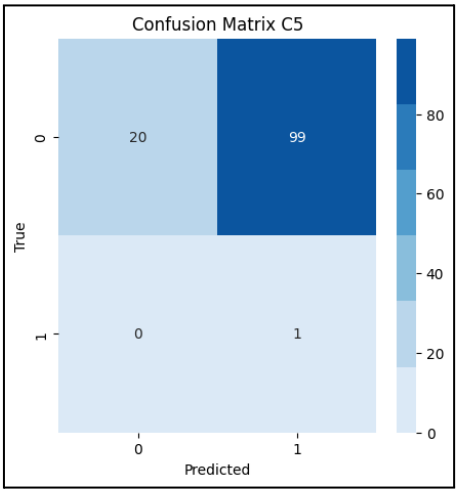


Figura No.12 - Matriz de confusión efficient Net C5

Figura No.13 - Matriz de confusión efficient Net C6

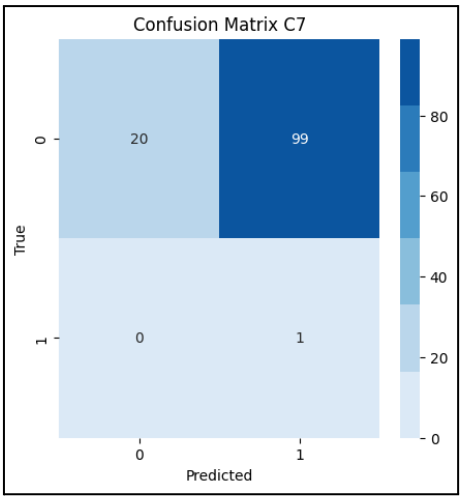


Figura No.14 - Matriz de confusión efficient Net C7

Eficiencia de 3DCNN

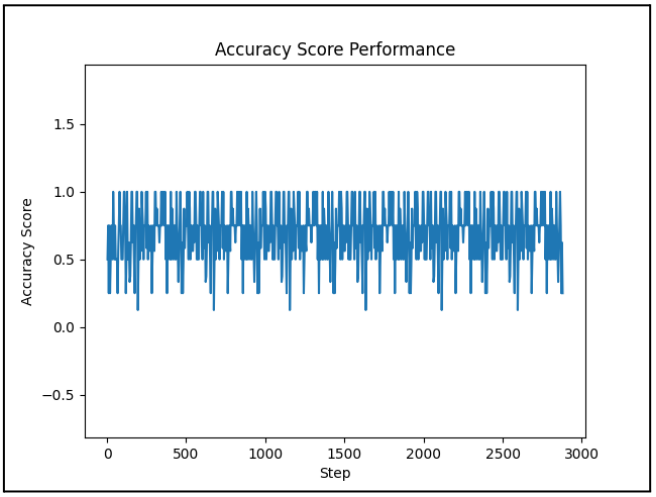


Figura No.15 - Accuracy 3DCNN

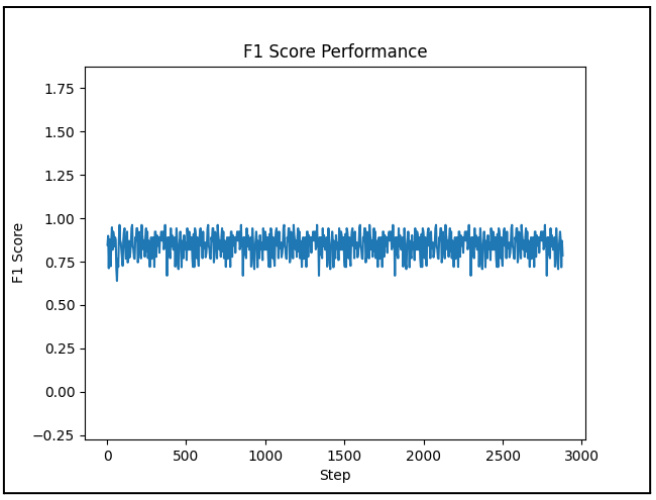


Figura No.16 - F1 Score 3DCNN

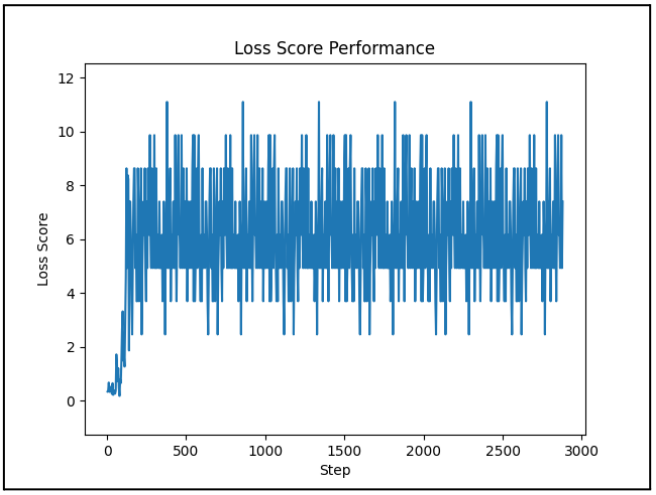


Figura No.17 - Loss score 3DCNN

Eficiencia de efficient Net + GRU

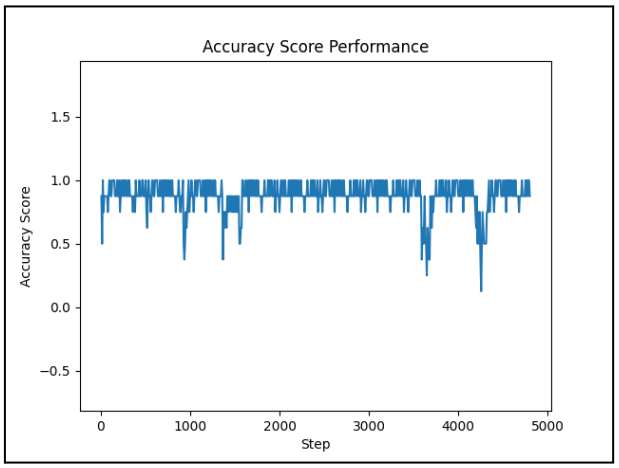


Figura No. 18 - Accuracy EfficientNet + GRU C1

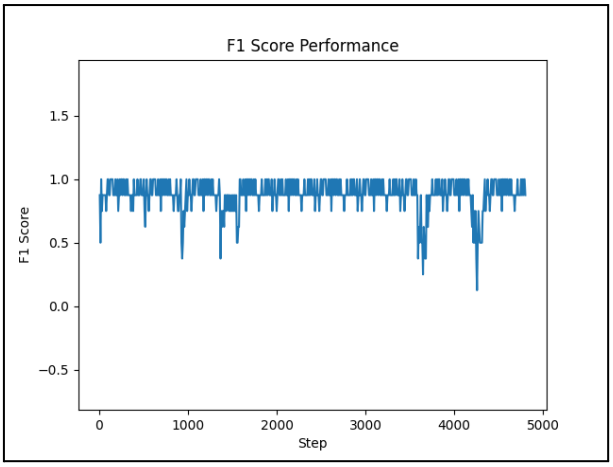


Figura No. 19 - F1 score EfficientNet + GRU C1

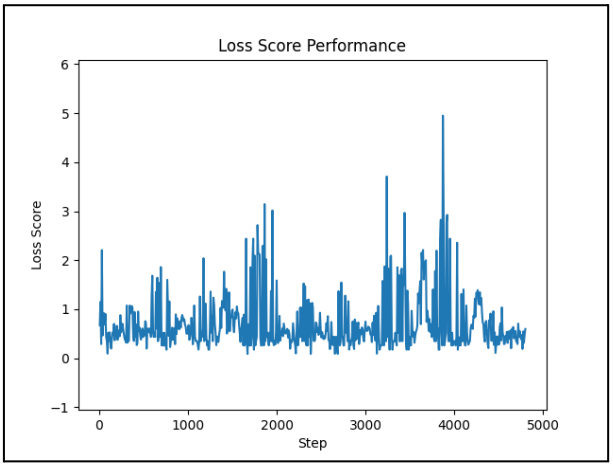


Figura No. 20 - Loss EfficientNet + GRU C1

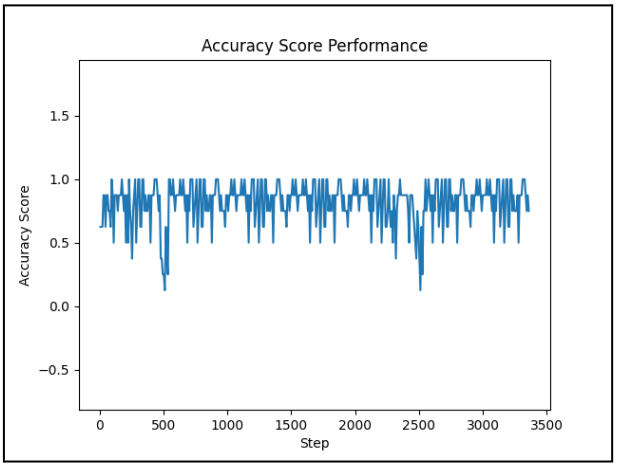


Figura No. 21 - Accuracy EfficientNet + GRU C2

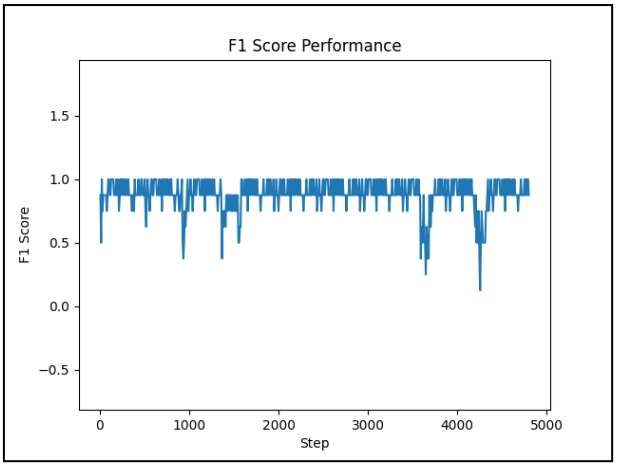


Figura No. 22 - F1 score EfficientNet + GRU C2

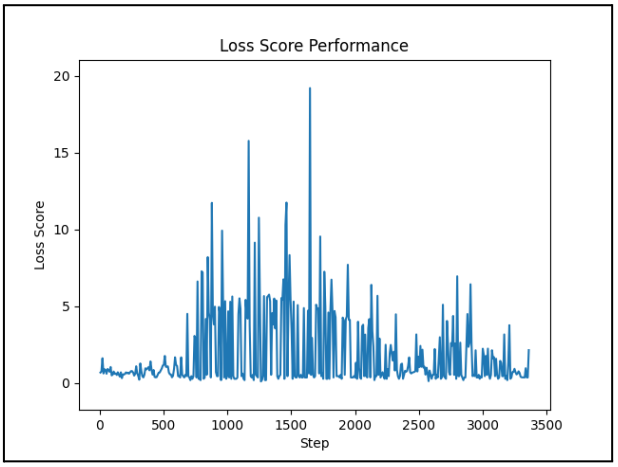


Figura No. 23 - Loss EfficientNet + GRU C2

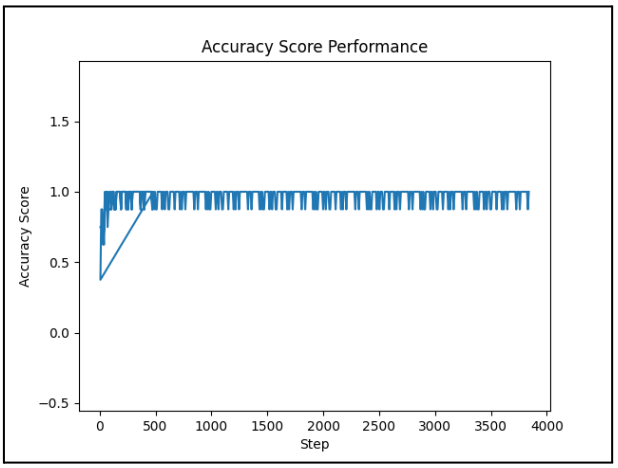


Figura No. 24 - Accuracy EfficientNet + GRU C3

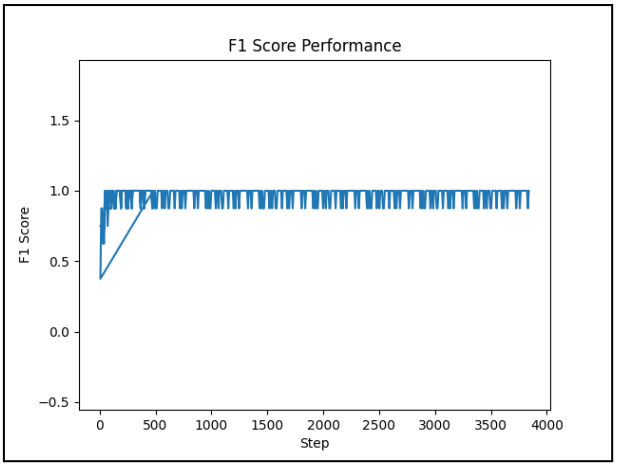


Figura No. 25 - F1 Score EfficientNet + GRU C3

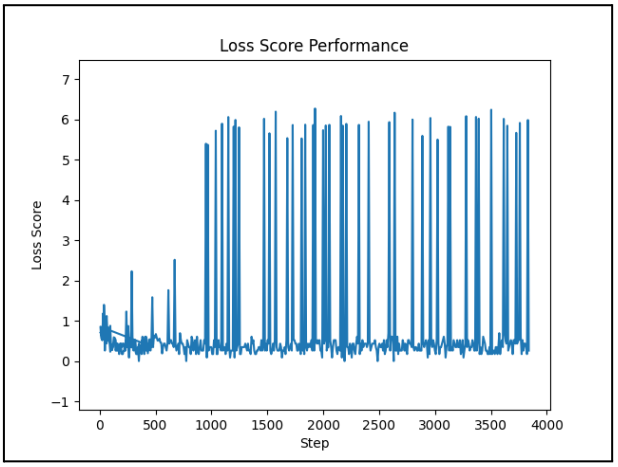


Figura No. 26 - Loss EfficientNet + GRU C3

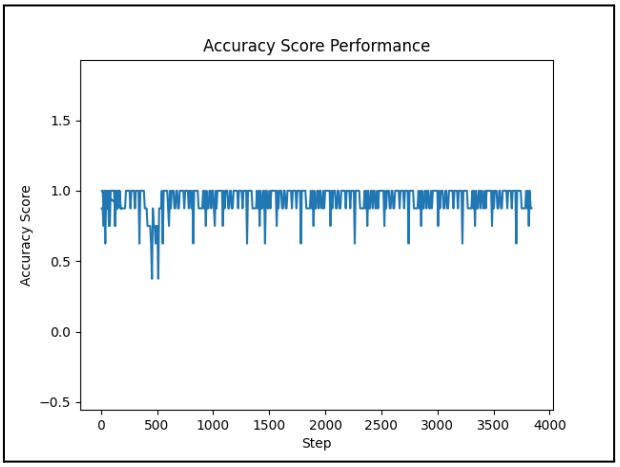


Figura No. 27 - Accuracy EfficientNet + GRU C4

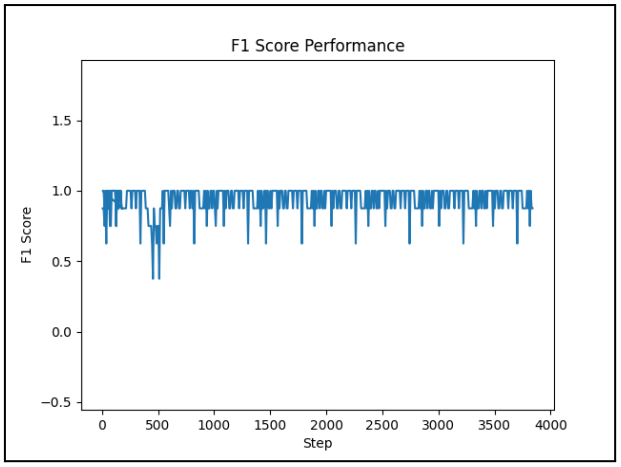


Figura No. 28 - F1 score EfficientNet + GRU C4

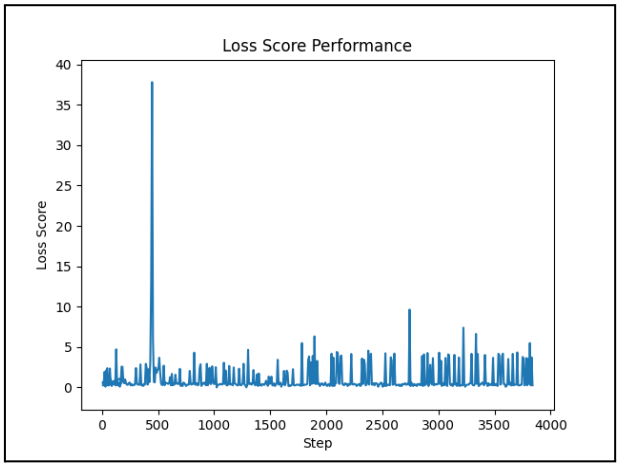


Figura No. 29 - Loss EfficientNet + GRU C4

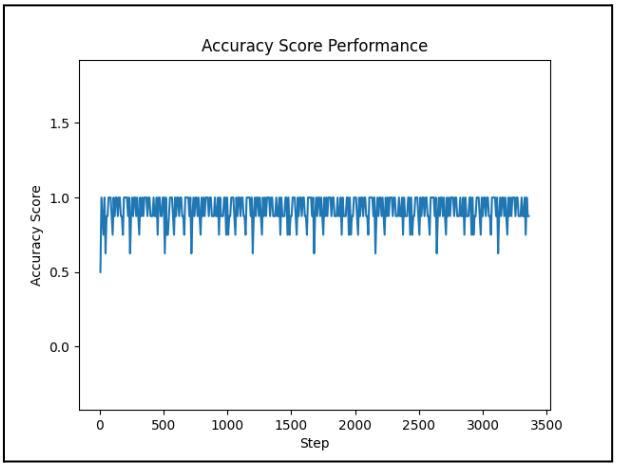


Figura No. 30 - AccuracyEfficientNet + GRU C5

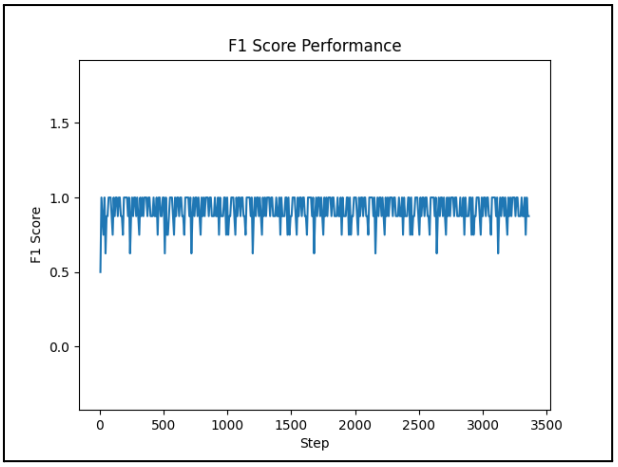


Figura No. 31 - F1 score EfficientNet + GRU C5

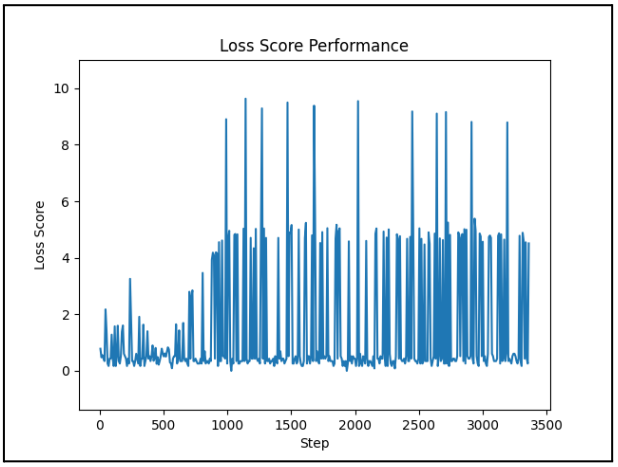


Figura No. 32 - Loss EfficientNet + GRU C5

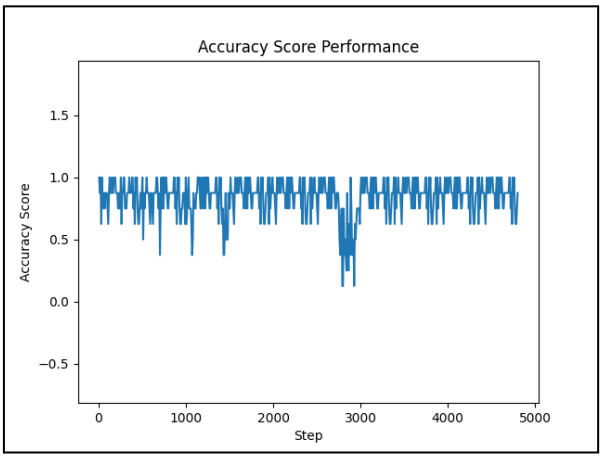


Figura No. 33 - Accuracy EfficientNet + GRU C6

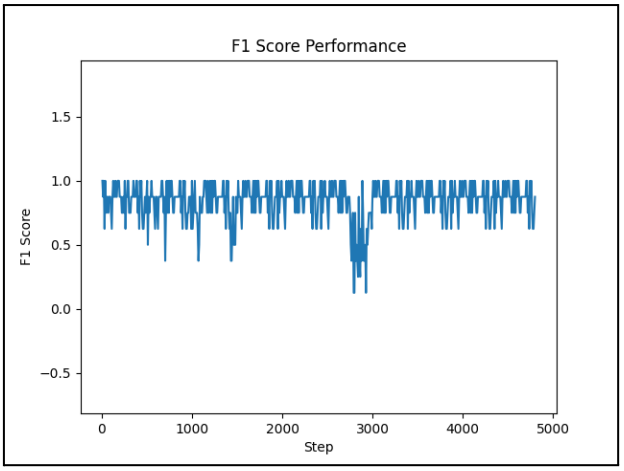


Figura No. 34 - F1 score EfficientNet + GRU C6

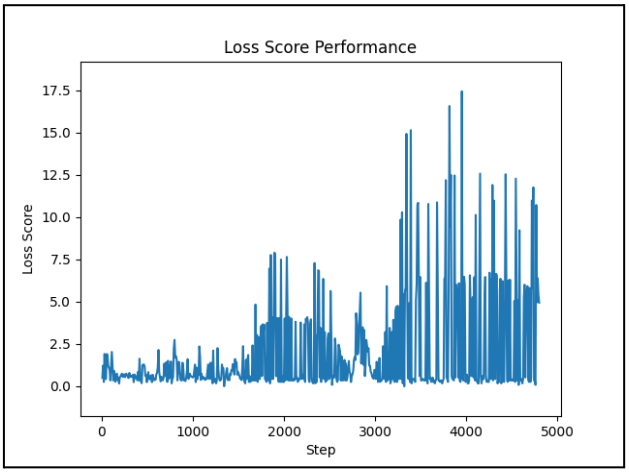


Figura No. 35 - Loss EfficientNet + GRU C6

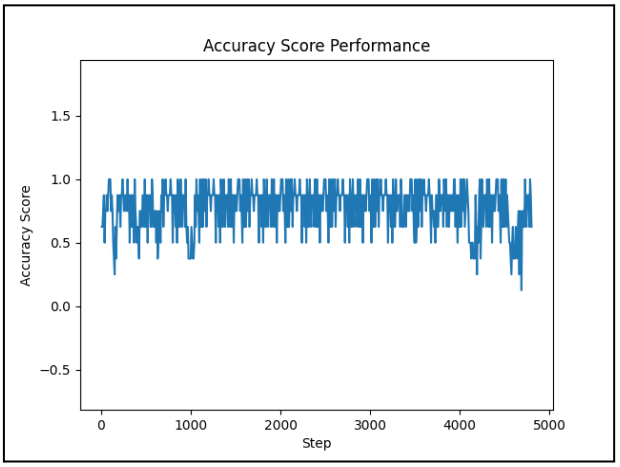


Figura No. 36 - AccuracyEfficientNet + GRU C7

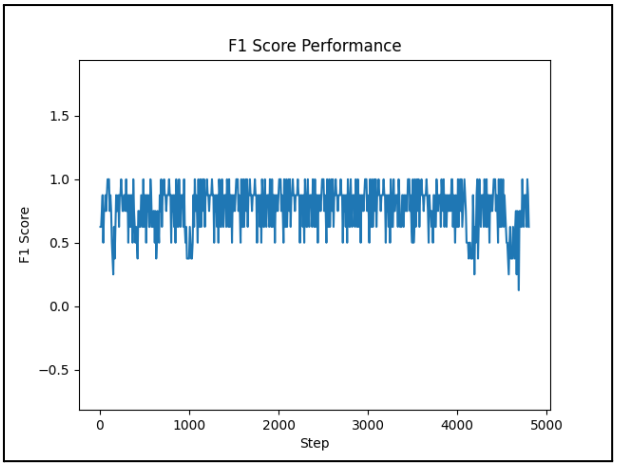


Figura No. 37 - F1 score EfficientNet + GRU C7

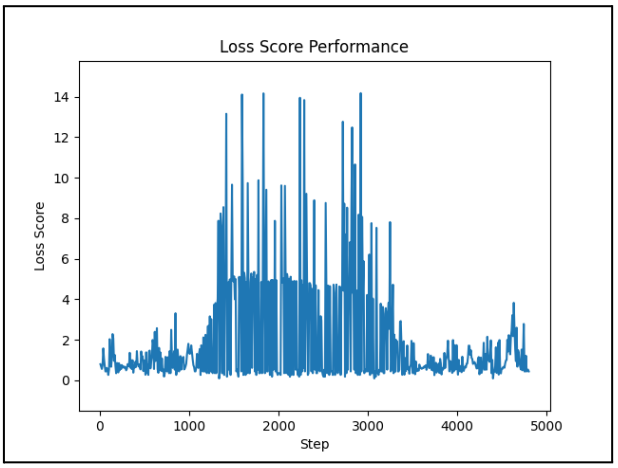


Figura No. 38 - Loss EfficientNet + GRU C7

Cervical	Accuracy	F1 Score
C1	0.91	NA
C2	0.81	NA
C3	0.958	NA
C4	0.833	NA
C5	0.933	NA

C6	0.833	NA
C7	0.175	0.29

Tabla No. 2 - Accuracy y F1 score 3DCNN

Cervical	Accuracy Test	F1 Score
C1	0.5	0.49
C2	0.5	0.48
C3	0.5	0.5
C4	0.498	0.49
C5	0.5	0.4935
C6	0.478	0.478
C7	0.5	0.46

Tabla No. 2 - Accuracy y F1 score Efficient Net + GRU

Matrices de confusión Efficient Net + GRU

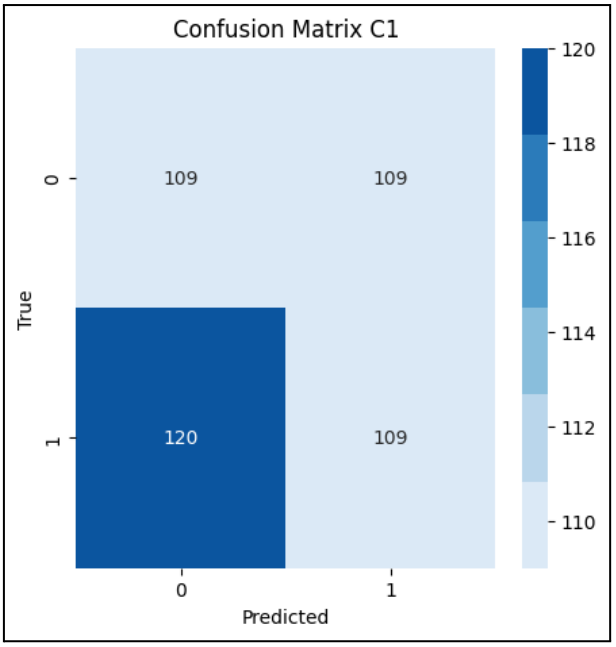


Figura No. 39 - Matriz de confusión Efficient Net + GRU C1

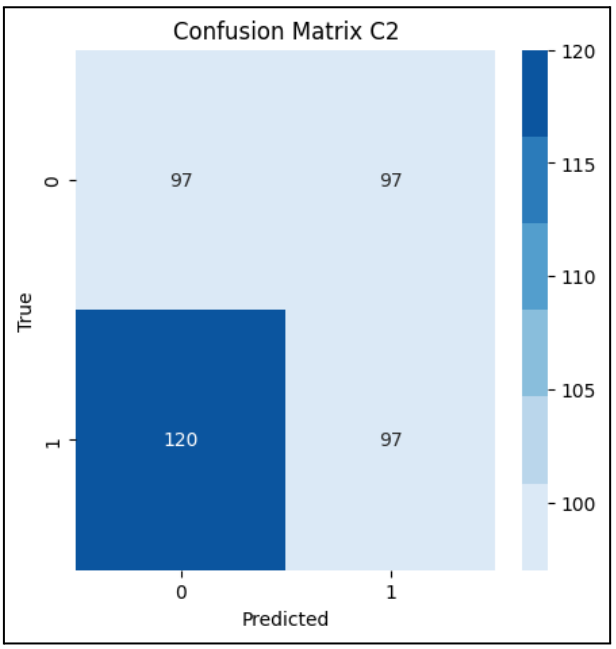


Figura No. 40 - Matriz de confusión Efficient Net + GRU C2

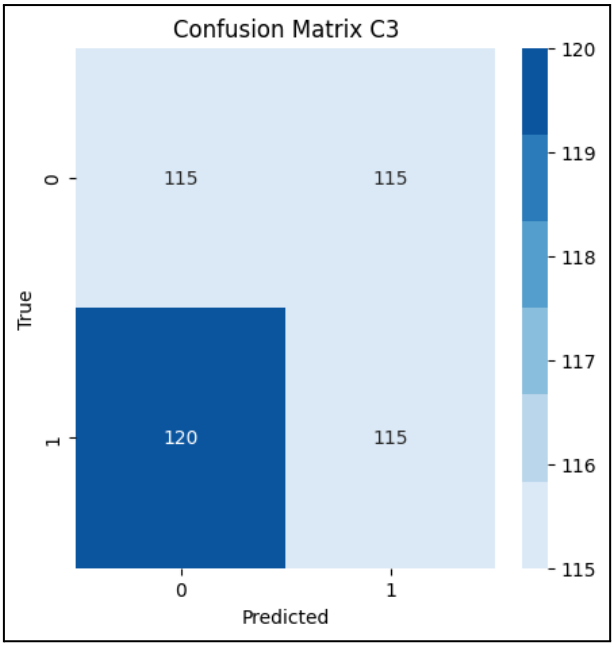


Figura No. 41 - Matriz de confusión Efficient Net + GRU C3

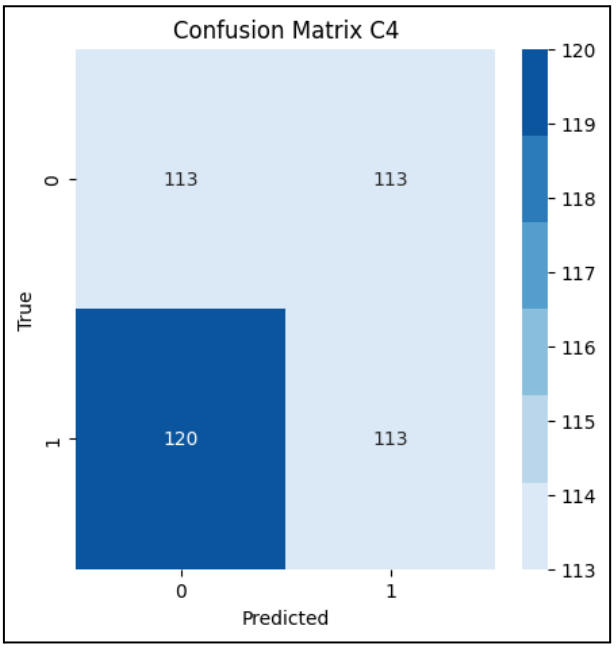


Figura No. 42 - Matriz de confusión Efficient Net + GRU C4

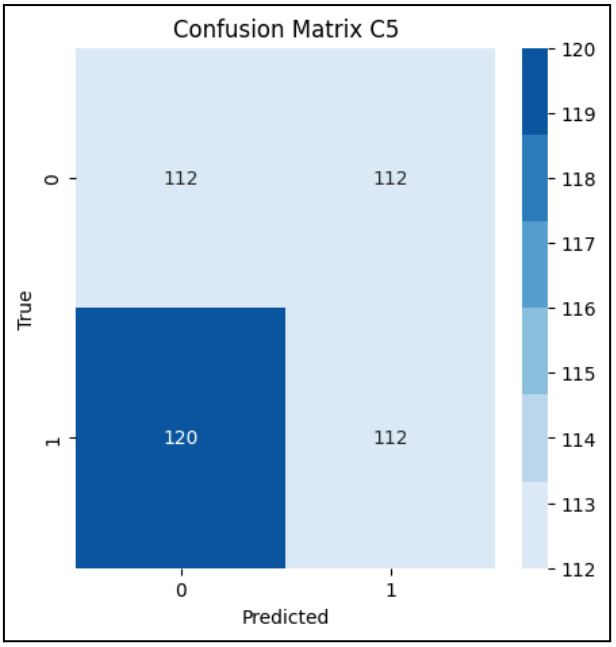


Figura No. 43 - Matriz de confusión Efficient Net + GRU C5

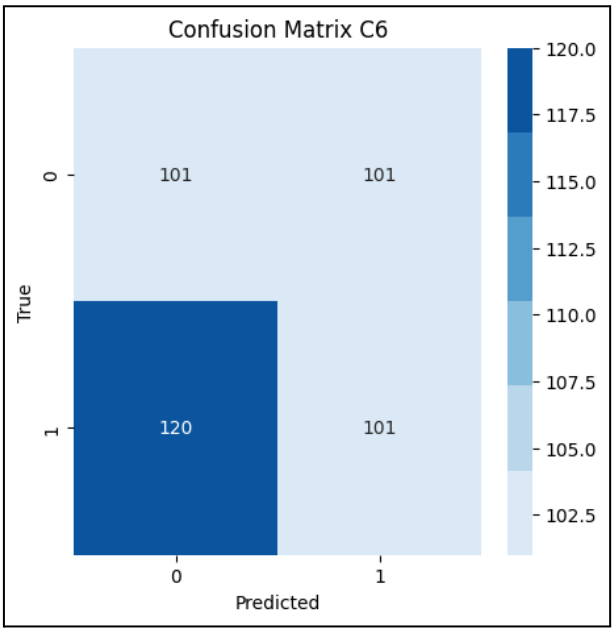


Figura No. 44 - Matriz de confusión Efficient Net + GRU C6

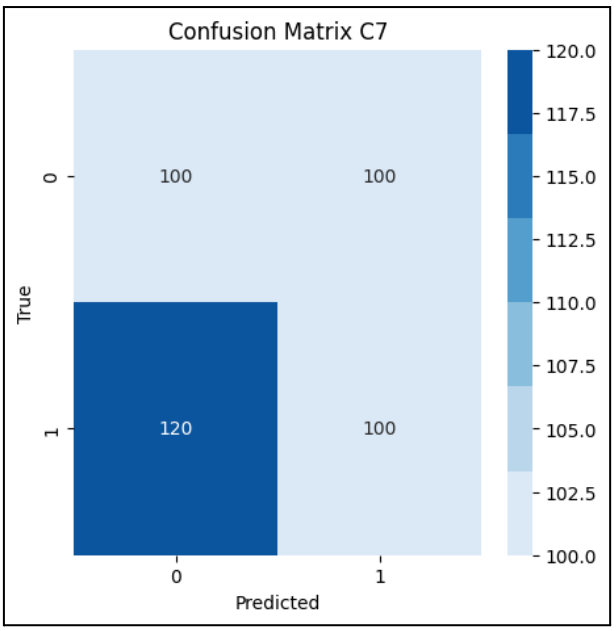


Figura No. 45 - Matriz de confusión Efficient Net + GRU C7

Discusión

Una de las características que denotaban la complejidad de este proyecto era la magnitud de datos. Se contaba con alrededor de 300 GB en imágenes de tomografías por computadora. La cantidad de imágenes por persona variaba desde 250 hasta 600 imágenes por paciente. Por lo tanto era necesario aplicar ciertas transformaciones que permitiera estandarizar la cantidad de imágenes por paciente, o una manera de representar todas las imágenes sin importar la cantidad.

Estandarizar la cantidad de imágenes por paciente tenía ciertas desventajas, por ejemplo, si se reducía la cantidad de imágenes se podía perder información importante, utilizar data augmentation elevaría mucho la complejidad del modelo. Por lo tanto la mejor opción era utilizar las imágenes de cada paciente sin importar la cantidad. Para el primer acercamiento el cual fue el modelo 3DCNN, se logró crear una representación intermedia la cual sería el input del modelo. Para esta representación intermedia se le aplicó un procesamiento a las N imágenes del paciente y todas juntas se escribieron en un objeto de numpy. Este objeto redujo la profundidad, es decir, la cantidad de imágenes por paciente a una cantidad estándar, pero esto se realizó por medio de transformaciones propias de numpy.

Con esta representación de numpy ya era posible entrenar al modelo, a pesar de que se redujo significativamente el tamaño de la entrada, el tiempo que demoraba en procesar los 2019 pacientes sobrepasaba las capacidades computacionales de todos los recursos con los que se contaban. Por está razón se seleccionó una muestra de 600 para el entrenamiento y 120 para el test. Una buena métrica de desempeño para este modelo era crucial para no dejarse engañar por los resultados. Esto se debe a que el modelo podía predecir correctamente 5 cervicales no rotas y equivocar la etiqueta de las dos últimas, donde claramente es más importante poder identificar las que sí están rotas. Por está razón se utilizó una métrica en la que se premiaba al identificar de forma correcta una cervical rota o predecir correctamente un paciente que no sufría de ninguna fractura.

El modelo presentó una accuracy de 80% aproximadamente entre todas las cervicales en la fase de entrenamiento, esta métrica se alcanzó luego de un entrenamiento de 10 horas. Dada la complejidad del proyecto y el poco tiempo que se entrenó se consideró un buen resultado. Pero era necesario realizar la fase de prueba, para evaluar si el modelo no sufría de sobreajuste. El modelo presentó resultados muy “buenos” de C1 a C5 la media se encontraba en 92% de Accuracy, pero al revisar las matrices de confusión, se pudo observar que el modelo clasificó muy bien aquellas cervicales que no estaban rotas. En cambio C7 que muchos de los pacientes tenían esa cervical rota el desempeño fue de 17.5%. Estos resultados

indican que el modelo sabe detectar cuando una cervical no está rota, pero no es capaz de extraer las características y entenderlas para poder predecir las fracturas.

Si observamos de la figura 1 a la 7 podremos comprobar que de la cervical C1 al C5 los Verdaderos negativos son predominantes, el modelo es bueno para detectar una no fractura. Pero el siguiente valor alto en las matrices de confusión son los falsos negativos, esto indica que el modelo al ver fracturas las categoriza como inexistentes. Claramente esto es un problema pues la principal causa de elaborar este modelo era prevenir y ayudar a los diagnósticos erróneos. Dado que no se logró el objetivo fue necesario evaluar las fases de desarrollo del modelo. Este análisis nos llevó a las siguientes conclusiones. Debido a la falta de potencia computacional no fue posible entrenar con los 2019 pacientes, ni por largos periodos de tiempo. Estos dos factores fueron críticos en las capacidades del modelo para el feature extraction. Por lo que si se contara con mejores equipos hubiese sido posible evaluar y entrenar de una mejor manera el acercamiento de la 3DCNN

Para el modelo híbrido de EfficientNet y GRU, también se debieron realizaron transformaciones a los datos. Esto consistió en pasar de imágenes en formato dicom a imágenes en formato jpg, perdiendo cierta información de los datos que viene en el formato original, debido a que dicom es especializado para tomografías computacionales. Esto se realizó debido a los requerimientos de EfficientNet en el formato de entrada. Aparte de esto, también se realizó una modificación de las dimensiones de las imágenes, haciendo que fueran de 224 x 224 píxeles, en lugar de 512 x 512. Además, se realizó una normalización de estas, sin perder RGB, porque igualmente EfficientNet lo requería.

Inicialmente, el modelo se utilizó de forma que el input eran todas las imágenes de cada tomografía por paciente. Sin embargo, debido a la profundidad de la CNN utilizada, se agotaba la memoria de la GPU, por lo que no fue posible entrenar de esta forma. Entonces, lo que se hizo fue realizar modelos especializados por cada cervical, los cuales solo predijeron si dicha cervical estaba rota o no. Para poder hacer esto, se empleó un dataset que indicaba que imágenes de cada CT correspondían a cada cervical. De forma que se obtenían las imágenes con la cervical en la que se estaba especializando el modelo, por paciente, y estas se usaban como input. Para esto se debió hacer un generador de datos, que fuera leyendo las imágenes conforme las fuera necesitando. Si se hubiera intentado cargar todas las imágenes al mismo tiempo, la computadora se hubiera quedado sin RAM.

De las figuras 18 a la 38, principalmente las que están relacionadas con la pérdida, se puede observar que esta es muy irregular. Tiene múltiples picos y valles pero no

sigue una tendencia clara. No tienen una forma exponencial invertida, como se espera de la pérdida. Esto quiere decir que los modelos realizados por cervical no están capturando adecuadamente las características y relaciones entre los datos. Las figuras relacionadas a la accuracy durante el entrenamiento y al F1 score, igualmente son irregulares y oscilan entre valores, pero no se ve un aumento claro como se espera. Esto se confirma al observar la tabla 2, donde la mayoría de accuracies están en 0.5, lo cuál quiere decir que el modelo clasifica de una forma aleatoria. Lo mismo sucede con el F1 score.

Estos resultados tienen diversas explicaciones. La primera y más importante, es la poca cantidad de datos con las que se entrenó cada modelo. Se usaron 600 pacientes de 2019 que se tenían en total, para entrenar cada modelo. Se puso un límite de cuántas imágenes por cervical se podían usar, debido a que si se usaban todas, la GPU se quedaba sin memoria. Para C1 se usaron 100 imágenes por cervical, para C2 se usaron 25, para C3 se emplearon 50, C4 usó 55, C5 usó 25, C6 usó 80 y C7 usó 75. Esto se hizo por la capacidad de los distintos equipos de cómputo con los que se contaban. Debido a la capacidad de la GPU y el tiempo de entrenamiento. Esta restricción de imágenes por cervical, anula de cierta forma el uso de una GRU en la arquitectura, debido a que esta se usa para captar relaciones temporales entre imágenes, que determinen si la cervical está rota o no. Pero al tener pocas imágenes, no se puede tener una visión más profunda de la vértebra, porque es posible que el modelo vea imágenes muy cercanas entre sí en la CT.

El tiempo de entrenamiento fue un factor que influyó bastante en la reducción de las imágenes usadas para entrenar. Si se usaba el dataset completo eran cerca de 6 días de entrenamiento. Solo se utilizaron 10 épocas de entrenamiento por modelo, lo cual causó que los modelos no pudieran captar las relaciones entre los datos de una forma adecuada. Esto también se evidencia en las figuras de pérdida previamente mencionadas, donde no hay una convergencia clara de la tendencia, lo cual indica que se requieren más épocas para el modelo.

Otro factor que influyó en los resultados, fueron las imágenes de las cervicales. Estas tienen bastante ruido debido a cómo se observan las tomografías en sí. Es posible que cada imagen tenga otros elementos del cuerpo humano dentro de ella, no solo cervicales. Por lo cuál, es posible que el modelo se esté enfocando en features de la imagen que no tienen relación con las fracturas de las cervicales, sobre todo al ser estas tan pequeñas y complejas de visualizar. Por lo tanto, se recomienda utilizar modelos de segmentación semántica que aislen las vértebras de todos los demás elementos.

Conclusiones

- La capacidad computacional fue la mayor limitante para desarrollar el proyecto, debido a la cantidad de datos y complejidad de los modelos.
- El uso de imágenes que tienen mucho ruido causa que las redes neuronales convolucionales no puedan captar las relaciones adecuadamente, por lo que es necesario realizar un preprocesamiento más profundo de las imágenes.
- Si no se usa el formato original de las imágenes médicas, se puede perder información valiosa, por lo se deben usar modelos que se ajusten a estos formatos y no viceversa.
- Si se realiza una reducción de la dimensionalidad de las tomografías computarizadas, es importante entender la forma en la cuál se puede mantener la información más importante de estas, de forma que se tenga más eficiencia y no se pierdan relaciones.

Referencias y Bibliografía

Pathak, A. (2022, August 30). *Redes neuronales convolucionales (CNN): una introducción*. Geekflare. https://geekflare.com/es/convolutional-neural-networks/

KeepCoding, R. (2023, March 2). *LSTM bidireccionales: cómo implementarlas*. Keepcoding.io. https://keepcoding.io/blog/lstm-bidireccionales-como-implementarlas/#:~:text=Una%20LSTM%20bidireccional%20(memoria%20a

Garzón, J. I. (2018, November 6). *Cómo usar redes neuronales (LSTM) en la predicción de averías en las máquinas*. Blog.gft.com. https://blog.gft.com/es/2018/11/06/como-usar-redes-neuronales-lstm-en-la-prediccion-de-averias-en-las-maquinas/

Telecomunicación, E., Saúl, D., & Raneros, R. (2021). *UNIVERSIDAD DE VALLADOLID Estudio de la arquitectura YOLO para la detección de objetos mediante deep learning*. https://uvadoc.uva.es/bitstream/handle/10324/45359/TFM-G1316.pdf

Lis Data Solutions. (2020, May 25). Deep Learning: clasificando imágenes con redes neuronales | LIS Data Solutions. Retrieved September 21, 2023, from LIS Data Solutions website: https://www.lisdatasolutions.com/es/blog/deep-learning-clasificando-imagenes-con-redes-neuronales/

Ardila-Rey, D., Martínez-Martínez, E., & Muñoz-Ávila, J. A. (2018). Deep learning for brain tumor segmentation in MRI: A systematic review. Nature Medicine, 24(1), 17-25.

Wang, Z., Liu, J., Chen, X., Lu, Y., & Sun, J. (2019). Liver segmentation in contrast-enhanced CT using deep learning with attention mechanism. IEEE Transactions on Medical Imaging, 38(1), 223-232.

Xu, Y., Li, J., Li, Q., & Liu, Y. (2020). Lung segmentation in CT images using a 3D fully convolutional network with residual connections. Radiology, 296(1), 176-183.

Majurski, M., Manescu, P., Padi, S., Schaub, N., Hotaling, N., Simon, C., & Bajcsy, P. (n.d.). Cell Image Segmentation using Generative Adversarial Networks, Transfer Learning, and Augmentations. Retrieved September 21, 2023, from https://openaccess.thecvf.com/content_CVPRW_2019/papers/CVMI/Majurski_Cell_Image_Segmentation_Using_Generative_Adversarial_Networks_Transfer_Learning_and_CVPRW_2019_paper.pdf