Tarea 1: red neuronal convolucional

Fernando Antonio Quiroz Escobar [1]
Astronomía, Universidad de Concepción. Matrícula 2020048037. fquiroz2020@udec.cl

1 Introducción

El High Cadence Transient Survey (HiTS) observó el cielo durante 4 días cada 2 horas, obteniendo una película del firmamento. El objetivo de HiTS fue encontrar objetos que varían su brillo en corto tiempo, tales como supernovas, minutos después de su explosión. Para esto, HiTS consideró una imagen de ciencia (imagen en el momento de observación), y una imagen de template la cual fue restada de la imagen de ciencia, produciendo así la imagen de diferencia. Esta última imagen de diferencia, se puede escalar en función del ruido estimado por pixel produciendo la imagen de diferencia razón-señal-a-ruido (SNR difference). En el caso de que un objeto aparezca repentinamente en una observación, éste se debiese ver en la imagen de diferencia. Muchas veces aparecen artefactos en las imágenes de diferencia debido a malas restas, píxeles erróneos, o variación estadística de las imágenes.

Por lo que en este trabajo, se consideró clasificar las imágenes que eran de utilidad, por ende se planteo generar un algoritmo de clasificación binaria, en el cual usaremos diferentes técnicas, se ha de mencionar que se usará la librería de "Pytorch", la cual nos permitirá trabajar con esta red neuronal convolucional (CNN a partir de ahora).

2 Métodos y Materiales

2.1 Preprocesamiento de datos

Los 4026 datos de entrenamiento y los 1000 datos de prueba se entregaron en archivos ".pkl". Primero se inició la tarea de conocer los datos con los que estábamos trabajando, mostrando así las "tags" de los archivos, generando así **"dict_keys(['temp_images', 'SNR_images', 'diff_images', 'sci_images', 'labels', 'ID'])"** para el archivo de testeo, es importante destacar que en este caso disponemos de los "labels" los cuales nos indican si se trata de objetos reales (1) o artefactos (0). Otro punto a mencionar es que en este algoritmo solo se consideró usar la imágenes de ciencia, ya que el uso de múltiples tipos de imágenes concatenadas podría aumentar la complejidad del modelo y del proceso de entrenamiento sin agregar un beneficio significativo.

Luego se dividieron el conjunto de datos en dos conjuntos, el de entrenamiento, con un **83%** (3.342) y el de validación con un **17%** (684) con la intención de evaluar el modelo una vez finalizado. Más adelante, se convirtieron los datos en tensores de "Pytorch" para facilitar su procesamiento en la CNN

2.2 Entrenamiento del modelo CNN

En el proceso de entrenamiento, se consideraron varios hiperparámetros, incluyendo el "learning rate" (a partir de ahora lr) establecido en **0.001**, el tamaño del "batch" fijado en **32**, con **100** épocas, y un filtro de dimensiones 3x3 en la capa de convolución. Además, se aplicó un "dropout" del **50%**.

También, se utilizó el optimizador "Adaptive Moment Estimation" (adam a partir de ahora) ya que en términos de recursos de hardware, requiere menos memoria y es computacionalmente muy eficiente Alom (2021). Por ultimo se ha de mencionar que la función de pérdida de entropía cruzada como criterio de optimización.

2.3 Arquitectura

Se ha de mencionar que la idea de la arquitectura fue sacada de la ayuda de la tarea, no obstante, se logró modificar adecuadamente para conseguir mejores valores.

Capa de Convolución (Conv2d):

En la arquitectura de la CNN, se utiliza una capa de convolución (nn.Conv2d). Estas capas son fundamentales para extraer características relevantes de las imágenes de ciencia Lubinus (2021). En la capa de convolución

[1] 2020048037, fquiroz2020@udec.cl

Capa	Tamaño de Entrada	Tamaño de Salida	Filtro	Función de Activación
Input	$B\times(1\times21\times21)$	-	-	-
Conv1	$B\times(1\times21\times21)$	$B\times(32\times19\times19)$	3×3	ReLU
MaxPool1	$B\times(32\times19\times19)$	$B\times(32\times9\times9)$	2×2	-
Dropout	$B\times(32\times9\times9)$	$B\times(32\times9\times9)$	-	-
FC1	$B\times(32\times9\times9)$	$B\times90$	-	ReLU
FC2	$B\times90$	$B\times2$	-	-

Table 2: Tabla de Capas de la red neuronal convolucional utilizada para la tarea de clasificación binaria

(self.conv1), se especifica un canal de entrada de tamaño 1 (escala de grises), 32 canales de salida y un kernel (filtro) de tamaño 3x3. Esto implica que la capa convolucional realiza 32 operaciones de convolución con un kernel de 3x3 en cada canal de la imagen de entrada.

Función de Activación ReLU:

Después de la capa de convolución, se aplica una función de activación ReLU (self.relu). La función ReLU (Rectified Linear Unit) introduce no linealidades en la red, lo que permite que la CNN aprenda características no lineales en los datos Goodfellow et al. (2016). Esto es crucial para capturar la complejidad de las imágenes científicas, ya que muchas relaciones en los datos visuales no son lineales.

Capa de Max Pooling (MaxPool2d):

Después de aplicar la función ReLU, se utiliza una capa de max pooling (nn.MaxPool2d) con un kernel de tamaño 2x2 (self.maxpool). El max pooling reduce la dimensionalidad de la salida de las capas de convolución, conservando las características más importantes y reduciendo el número de parámetros en la red Zhou and Chellappa (1988). Esto ayuda a evitar el sobreajuste y mejora la eficiencia computacional durante el entrenamiento.

Capa de Dropout:

Para regularizar la red y prevenir el sobreajuste, se aplica una capa de dropout (nn.Dropout) después de la capa de max pooling (self.dropout). El dropout apaga aleatoriamente unidades (neuronas) durante el entrenamiento, lo que obliga a la red a aprender características más robustas y generalizables Srivastava et al. (2014).

Capas Completamente Conectadas (Linear):

Finalmente, después de aplanar la salida de la capa de dropout, se utilizan dos capas completamente conectadas (nn.Linear) para la clasificación binaria. La primera capa completamente conectada (self.fc1) tiene 32x9x9 neuronas de entrada (resultado de la capa de max pooling) y 90 neuronas de salida. La segunda capa completamente conectada (self.fc2) tiene 90 neuronas de entrada y 2 neuronas de salida, que representan las clases binarias a predecir.

3 Resultados

Una vez ya definido y entrenado el modelo (CNN), podemos graficar las curvas de aprendizaje y la matriz de confusión.

3.1 curvas de aprendizaje

Las curvas de aprendizaje muestran la evolución de la pérdida durante el entrenamiento y la validación en función del número de épocas. Estas curvas son importantes para evaluar el proceso de aprendizaje y detectar posibles problemas como el sobreajuste o subajuste del modelo.

Podemos ver en 1 que la curva de perdida de validación se acopla de buena manera a la curva de perdida de entrenamiento, no obstante, vemos que a partir de la época 50 aproximadamente, la curva de perdida de validación empieza a desfasarse, no obstante, tampoco son valores exagerados, quizás se pudo hacer un ”earlystopping”, sin embargo, si nos fijamos en 2 podemos ver que es completamente al revés, la curva de precisión de validación se va ajustando a medida que pasan las epocas

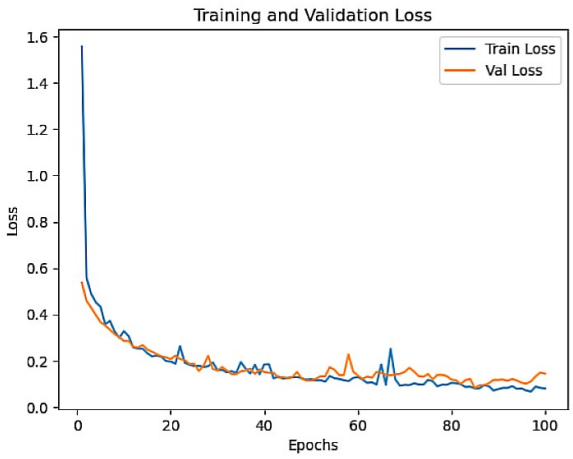

Figure 1: Gráfico de perdida de validación y de entrenamiento, ejecutada con 100 épocas

Figure 2: Gráfico de precisión de validación y entrenamiento, ejecutada con 100 épocas

3.2 Matriz de confusión

Se presenta una matriz de confusión que visualiza la distribución de las predicciones del modelo en comparación con las etiquetas verdaderas, en el caso de 3 podemos ver que se clasificaron 112 objetos reales de manera correcta y 105 artefactos de manera correcta, no obstante, también podemos ver las erróneas clasificaciones, 2 en el caso de objetos reales clasificados como artefactos y 5 artefactos clasificados como objetos reales

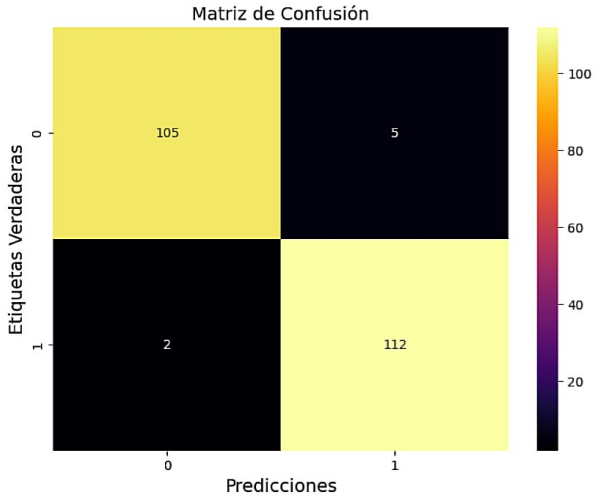

Figure 3: Matriz de confusión, en la cual podemos ver las clasificaciones predecidas por el modelo en comparación a las etiquetas reales, la "colorbar" representa las cantidades de elementos en cada etiqueta, siendo amarillo valores altos y negro valores bajos

4 Métricas significativas

```
Accuracy:  0.96875
Precision:  0.9692866842399552
Recall:  0.9685007974481659
F1 Score:  0.9687194525904204
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 19, 19]             320
              ReLU-2          [-1, 32, 19, 19]               0
         MaxPool2d-3            [-1, 32, 9, 9]               0
           Dropout-4            [-1, 32, 9, 9]               0
            Linear-5                  [-1, 90]         233,370
              ReLU-6                  [-1, 90]               0
            Linear-7                   [-1, 2]             182
================================================================
Total params: 233,872
Trainable params: 233,872
Non-trainable params: 0
```

Figure 4: Métricas obtenidas después de correr el algoritmo con 100 épocas, en el cual podemos ver tanto precisión, recall y f1-score, tanto como sus capas y parámetros totales

4.1 Precisión

Es la medida que denota la proporción de predicciones de verdaderos positivos respecto al total de predicciones hechas por el modelo, es decir:

$$\text{Precisión} = \frac{\text{verdaderos positivos}}{\text{verdaderos positivos} + \text{falsos positivos}} \quad (1)$$

una precisión alta proporciona información respecto a la capacidad de realizar predicciones correctas de forma consistente, en este caso, como podemos ver en 4, el valor de nuestra precisión es de 0.969 aproximadamente

4.2 Recall

Básicamente hace referencia a la sensibilidad del algoritmo o a la tasa de verdaderos positivos, se calcula de la siguiente manera

$$\text{Recall} = \frac{\text{verdaderos positivos}}{\text{verdaderos positivos} + \text{falsos negativos}} \quad (2)$$

En nuestro modelos se obtuvo 0.968 aproximadamente como se puede observar en 4

4.3 F1 score

Esta métrica es definida a través de las dos anteriores, la cual se calcula como la media armónica de la precisión y el "recall"

$$\text{F1} = 2 \times \frac{\text{precisión} \times \text{recall}}{\text{precisión} + \text{recall}} \quad (3)$$

y como podemos ver en nuestro algoritmo nuestro "F1-score" es de aproximadamente 0.969 (véase imagen 4)

Capa	Tamaño de Entrada	Tamaño de Salida	Filtro	Función de Activación
Input	4x21x21	-	-	-
Conv1	4x21x21	25x20x20	2x2	ReLU
MaxPool1	25x20x20	25x10x10	2x2	-
Conv2	25x10x10	50x9x9	2x2	ReLU
MaxPool2	50x9x9	50x4x4	2x2	-
Dropout	50x4x4	50x4x4	-	-
FC1	50x4x4	25	-	ReLU
FC2	25	2	-	Sigmoid

Table 3: Intento de la CNN en la cual se consideró concatenar las 4 tipos de imágenes (temp images, SNR image, diff images y sci images), además de considerar 2 capas convolucionales con sus respectivos maxpool, además considerando una función sigmoide como función de activación

5 Discusión

Se ha de mencionar que pese a no conseguir el mejor f1 posible, debido posiblemente a la simpleza de la CNN, se han probado múltiples técnicas para hacer mejorar estas métricas, desde variar los hiperparámetros a través del método de búsqueda de grillas, lo cual no dio muy buenos resultados, ya que las curvas de aprendizaje no terminaban de estabilizarse, por lo que se decidió no agregar esa parte, decidiendo así hiperparámetros bastante estándar. Volviendo al algoritmo principal, posiblemente estos resultados pudieron haberse visto completamente beneficiados si se hubiese agregado un "early-stopping" para parar el código en una época más prudente, como lo pudo haber sido en la época 70.

Por otro lado se probaron distintas arquitecturas, por ejemplo 3, no obstante era demasiado lenta, además de que en comparativa no me entregaba métricas mejores, ya que proporcionaba "f1-score" de 0.80 aproximadamente, a pesar de que entregaba unos curvas de aprendizaje superiores.

Se probaron múltiples hiperparametros diferentes, entre ellos diferentes lr (0.1, 0.01, 0.001 y 0.0001) en los cuales los primeros dos son los que entregaban los peores valores de los cuatro, ya que generaban "F1-scores" que rozaban el 0.7. Finalmente el lr que fue seleccionado fue 0.001 debido a que era mas consistente con sus métricas, también se probó la función Stochastic Gradient Descent (SGD), que en base a lo que se vio convergía más lento que Adam, por lo tanto se seleccionó este ultimo, el tamaño del conjunto de validación fue comparado con 0.17, 0.19, 0.2, 0.21 y 0.23 y aquí la decisión si que fue mucho mas rápida debido a que 0.17 entregaba unas métricas bastante mas altas que el resto, generando curvas de aprendizaje bastantes similares, por ultimo el tamaño del batch solo fue comparado entre 2 valores, 32 y 64, debido a que no eran demasiados datos, con 32 se comportaba de mejor manera en las métricas. Se ha de mencionar que cada una de estos configuraciones de hiperparametros fueron probada de 20 a 35 veces cada una, debido a que el código corre a una excelente velocidad.

6 Conclusiones

- Queda en evidencia que el uso de deep learning en astronomía es un recurso muy valioso que podría llegar a facilitar y optimizar tiempos en el área de clasificación de eventos, tal cual como lo hace ALeRCE.
- En este caso particular funcionó de mejor manera una arquitectura bastante simple, no obstante, no se descarta la posibilidad de que sea causa por falta de conocimiento
- Se destaca la importancia de técnicas como el dropout para regularizar el modelo y prevenir el sobreajuste, así como el uso de métricas como la matriz de confusión para evaluar el desempeño del algoritmo de manera más detallada.

References

Alom, M. (2021). Adam optimization algorithm.

Goodfellow, I. J., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, MA, USA. http://www.deeplearningbook.org.

Lubinus, L. (2021). Redes neuronales convolucionales: un modelo de deep learning en imágenes diagnósticas. revisión de tema. *Revista Colombiana De radiología*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Zhou, Y.-T. and Chellappa, R. (1988). Ieee 1988 international conference on neural networks. 2:71–8.