Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ingeniería Eléctrica

Entrega Final

Detección temprana de supernovas en ALeRCE

Integrantes: Diego Castillo
Nibaldo Foix
Profesor: Pablo Estévez
Auxiliares: Ignacio Reyes
Ayudantes: Esteban Reyes
Francisca Cona
Jhon Intriago
Mauricio Romero
Óscar Pimentel
Pablo Montero

Fecha de entrega: 31 de diciembre de 2020
Santiago, Chile

Índice de Contenidos

Lista de Figuras

Lista de Tablas

Lista de Códigos

1. Introducción

ALeRCE: *The Automatic Learning for the Rapid Classification of Events* es una iniciativa llevaba a acabo por Instituo Milenio de Astrofísica y el centro de modelamiento matemático de la universidad de Chile. Junto a Data observatory y otras universidades e instituciones se ha creado este proyecto con el fin de crear un sistema capaz de detectar, de forma automática, ciertos tipos de eventos astronómicos.

Alerce, en términos técnicos, es un *broker* de alertas que recibe y procesa observaciones astronómicas provenientes del survey ZTF en tiempo real. Dentro de sus tareas está la clasificación de objetos astronómicos, para lo cual cuenta con distintos algoritmos de *Machine Learning*.

Uno de ellos es el "clasificador temprano", el cual utiliza las imágenes de las primeras detecciones para entregar una clasificación rápida del objeto observado. Es justamente en esta tarea que se entrará en detalle en este proyecto.

Esta iniciativa nace por la gigantesca cantidad de datos producida cotidianamente en los observatorios astronómicos, la cual, sin técnicas de *Machine Learning* sería imposible de analizar. Al tener un clasificador de objetos astronómicos, como el que se pretende construir, se lograrían identificar fenómenos como las supernovas sin necesidad de que expertos tengan que observar cada imagen obtenida, lo que sería imposible dada la cantidad de datos que se generan.

Este proyecto nace como iniciativa del curso EL4106 - para profundizar todos los tópicos aprendidos a lo largo del semestre y efectos de este mismo, los objetivos se dividirán en 4 partes:

1. Familiarización y manipulación de gran cantidad de datos.
2. Manipulación inteligente de la información para ejecutar de forma óptima algoritmos de aprendizaje.
3. Familiarización e implementación con métodos de redes neuronales convolucionales.
4. Evaluación de la red y evaluación de una mejora en el diseño en vista de identificar 5 eventos astronómicos con una tasa de exactitud similar a la bibliografía.

Se espera obtener un modelo capaz de realizar predicción sobre la categoría de las diferentes clases con una exactitud mínima de 85 % en general. Para las supernovas en particular se espera poder tener para tener un *F2 Score* de al menos 80 %.

En términos de software se utilizará un código de programación *python* ya que es el lenguaje de programación utilizado a lo largo del curso. Asimismo, *python* posee varias librerías que servirán para hacer más ameno el trabajo esperado. Algunas de estas son:

- Pandas: librería para manipular gran cantidad de datos.
- numpy: librería para trabajar con tipos de datas array.
- matplotlib.pyplot: librería útil para graficar funciones e imágenes.

- tensorflow: librería útil para la construcción de redes neuronales.

Este trabajo está inspirado en [1], 'Enhanced Rotational Invariant Cinvolutional Neural Network for Supernovae Detection', trabajo llevado a cabo por el cuerpo docente de el curso. En lo particular se intentará replicar métodos explorados en el trabajo de investigación como el método rotacional y la entropía cruzada regulado.

El presente informe se dividirá en 5 secciones posterior a esta introducción. Se comenzará con Marco Teórico donde se abordará los temas importantes de redes convolucionales, se seguirá con Solución propuesta donde se tocarán los temas con los que se estuvo trabajando para alcanzar la meta planteada. Posteriormente se mostrarán resultados del algoritmo, para luego, abordar una discusión para comentar lo que se logró obtener y se terminará con las conclusiones del proyecto.

2. Marco Teórico

2.1. Preprocesamiento de Datos

Para comenzar es necesario e importante hablar sobre como hacer una limpieza de los datos a utilizar. El analizar como viene la data es por lo general de lo que menos se habla cuando al final de cuentas es parte importante del trabajo. Cualquier tipo de error en los datos conllevará a un error mayor en el proceso a posterior.

A fin de concretar lo dicho, se suele trabajar los siguientes aspectos:

1. Verificación de de las categorías para cada imagen
2. Verificación de formato de los datos
3. Normalización de los valores de todas las imágenes para lograr una estandarización en los valores de todas las componentes.

Haciendo este trabajo se alcanza la seguridad de poder trabajar con datos representativos del problema que se quiere abordar

Por otro lado, en algoritmos de aprendizaje, existe la necesidad de separar los datos entregados en grupos de entrenamiento y validación y test. La separación va a tener repercusiones en como aprenderá la red igual que como se podrá medir su rendimiento. Para efectos de código se suele usar distribución aleatoria , y la cantidad de datos en *training* es superior al de test.

2.2. Redes convolucionales

El clasificador de objetos astronómicos realizado en este trabajo, corresponde a una red neuronal convolucional (CNN), por lo que es indispensable escribir las características de estas.

Una **red neuronal convolucional** es una variación de un perceptrón multicapa, sin embargo, debido a que su aplicación es realizada en matrices bidimensionales. Son muy efectivas para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Las redes convolucionales suelen ser útiles a la hora de detectar eventos en imágenes, debido a su capacidad de identificar patrones. La naturaleza de convolución pareciera ser efectiva para el problema a resolver. Adicionalmente, es un método que ya se ha utilizado en tareas anteriores, aunque el grupo no ha diseñado una desde cero.

El principal fuerte de una red convolucional a la hora de detectar imágenes corresponde a su capacidad de aprender patrones, que en el caso de una imagen podrían ser, por ejemplo, la presencia de líneas, bordes, sombras, etc. Una red convolucional logra realizar esto requiriendo solo una pequeña fracción de los parámetros que requeriría una red MLP, ya que, desde su estructura genera menos pesos compartidos, lo que significa que cada filtro procesa toda la imagen.

Una red convolucional se separa en los siguientes componentes:

1. **Entrada:** Para cualquier tipo de imagen a color, se puede separar por los colores RGB creando así 3 dimensiones de imágenes con largo y ancho. Estas serán las imágenes que se querrán analizar.

2. **Capas convolucionales:** Su uso radica en el filtrado de imágenes, que consiste en la modificación de una imagen de manera que se mejoren o se realcen algunas de las características con vistas a obtener información relevante. Para esto se utilizan filtros o *kernels*. Las características importantes para lo que sirve el kernel son detectar bordes, enfoque, desenfoque, entre otros. Esto se logra al realizar la convolución entre la imagen y el kernel.

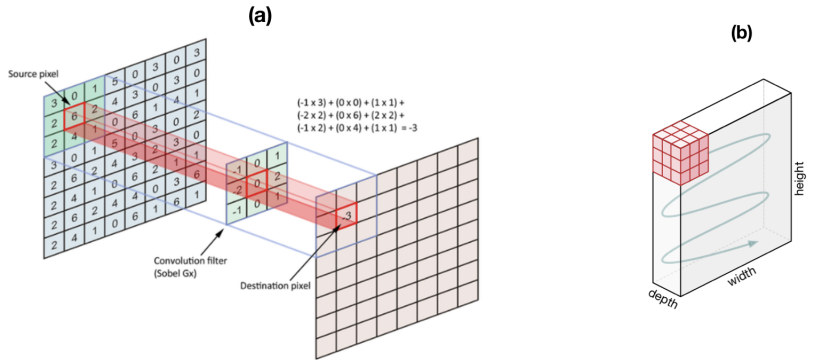

Figura 1: (a): Ejemplo de convolución con kernel de 3x3x1 para imagen de entrada cualquiera, (b): Trayectoria de un filtro o kernel de 3x3x3 para una imagen de entrada tipo RGB

3. **Capas de *pooling*:** Proporcionan herramientas para bajar dimensionalidad agrupando conjuntos de píxeles. Las dos técnicas mas comunes son *average pooling* y *max pooling*. La primera proporciona un promedio entre el grupo seleccionado y la segunda valor máximo.

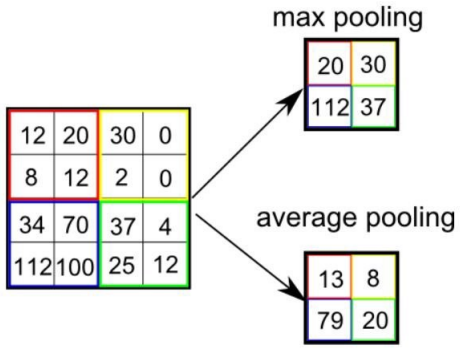

Figura 2: Ejemplo de capas de *pooling* para bajas dimensiones

4. **Capas *Fully Connected*:** Estas capas corresponden a grupos de células equivalentes a las presentes en una red MLP, en que todas las salidas de una capa corresponden a entradas de cada neuronal de la capa siguiente. La última capa de este tipo debe contener un número de células igual al número de categorías presentes en la base de datos a clasificar.

Juntando todos los conceptos anteriores, se puede armar una red convolucional completa como se muestra en el ejemplo a continuación:

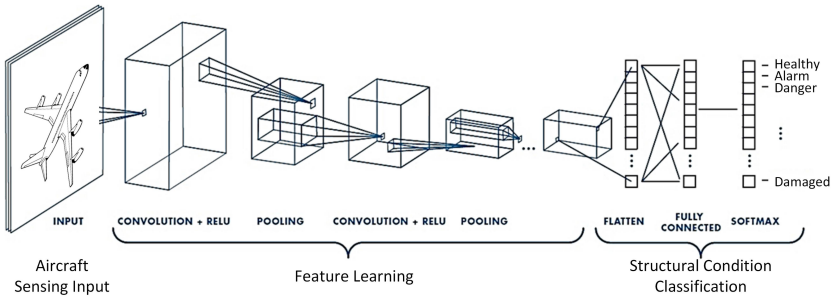

Figura 3: Ejemplo de red CNN completa

En cuanto a las capas de la red, a lo largo del proyecto se puede cambiar el número de capas para cada tipo de ellas, además de parámetros como la profundidad de cada capa convolucional, el tamaño de cada convolución, el tamaño de cada *pooling* y la cantidad de unidades de cada capa *fully-connected*.

5. **Hiperparámetros**

Por otro lado, los hiperparámetros son las variables disponibles para ir adaptando el modelo a un óptimo entrenamiento.

- **Número de épocas realizadas**: Durante un entrenamiento se tiene que entrenar el algoritmo en diferentes tandas. Cada tanda se define como una época. La incorrecta definición del número de épocas ejecutadas puede conllevar a la detención del algoritmo previamente a alcanzar un rendimiento del clasificador óptimo o a la detención de este una vez sobreajustado.
- **Número de *batches***: En una época se define el numero de *batches* como el número de imágenes de entrenamiento a evaluar (iteraciones) entre dos cómputos de métricas de validación.
- **Tasa de aprendizaje**: Dentro del problema general de minimización del error explicado más en detalle en la sección posterior, la tasa de aprendizaje se puede definir como el tamaño de actualización de datos. Es 'qué tan brusco' se van actualizando los valores de la red intentando llegar al mínimo global. Un valor muy bajo podría hacer converger a un mínimo local mientras que uno muy grande podría no converger

2.3. Función de Costo

Un punto relevante a considerar es la función de costo o error a minimizar en en el algoritmo. Esta función será la encargada de dictar si el algoritmo esta orientado hacia la buena dirección de predicción. Si la función dicta que no lo está, el algoritmo tendrá que cambiar ciertos parámetros para re-orientarse.

Las funciones de costo son variadas y dependen mucho en que ámbito se esta trabajando. Se usan tanto en estadística como en economía como en matemática de optimizadora. En aprendizaje de maquina sirve enfocarse en la imprecisión de los clasificadores. Se puede definir de la siguiente manera: Sea $X \in R^d$ el espacio de todos los posibles inputs e Y la salida o output con los labels. En el caso de un clasificador binario, se definiría como $Y = \{-1, 1\}$. El objetivo se plantea como encontrar una función $f : X \to R$ que mejor pueda predecir la etiqueta y en función de una entrada $\vec{x}$. No obstante, dada la complejidad del problema, dado un mismo $\vec{x}$ se puede llegar a diferentes y. Dicho esto, el verdadero objetivo del aprendizaje es minimizar la perdida esperada o riesgo:

$$I[f] = \int_{X \times Y} V(f(\vec{x}, y))p(\vec{x}, y)d\vec{x}dy$$

Donde $V(f(\vec{x}), y)$ es la función de costos y $p(\vec{x}, y)$ la función densidad de probabilidad del proceso que genera la data

Si bien las más usadas son error cuadrático o función perdida 0-1, en en redes convolucionales se prefiere usar la función de Cross-Entropy o Entropía Cruzada, dada sus ventajas desde el marco de las teorías de la información.

$$H(f(\vec{x}), y) = -\sum_{x \in X} y(\vec{x}) log\ f(\vec{x})$$

Que solo es válida para procesos discretos. La evaluación del modelo, en una primera instancia, es realizada obteniendo las medidas de *accuracy* global y para las supernovas, considerando que se requiera un clasificador con las 5 categorías.

Dentro de lo que se busca es evitar al máximo que supernovas queden clasificadas como no-supernovas. Esto último debido a que se ve mucho más factible tener humanos que verifiquen la veracidad solo de las detecciones de supernovas, y descarten los objetos incorrectos a alguien que tenga que encontrar supernovas dentro de las clasificaciones de no-supernovas. Como *recall* mide mejor esto, se enfocará más en la medida F_2 score.

2.4. Métricas de Rendimiento

Si un algoritmo testea de forma binaria una clasificación (positivo=clase buscada) , se debe medir el desempeño de este mismo. Para lo anterior existe la matriz de contingencia que resume el éxito del procedimiento con las categorías: Verdadero Positivo (VP): clasificación positiva para clase positiva, Falso Positivo (FP): calorificación positiva para clase negativa, Verdadero Negativo (VN): clasificación negativa para clase negativa y Falso Negativo (FN): clasificación negativa para clase positiva.

Algunas concepto nacen de aqui para tener buenas herramientas a disposición para evaluar el desempeño:

1. *Accuracy:* porcentaje de clasificaciones correctas: $\dfrac{VP + VN}{VN + FP + VN + FN}$
2. *Precisión:* porcentaje de clasificaciones positivas como correcta: $\dfrac{VP}{VP + FP}$

3. *Recall:* porcentaje de clasificación correctas para clases positivas: $\frac{VP}{VP+FN}$

4. F_β *score:* es una ponderación entre recall (R) y precisión (P) para manejar los dos parámetros simultáneamente. Se relacionan como dos resistencias en paralelo: $(1+\beta^2)\frac{PR}{(\beta^2 P)+R}$

Para un caso no binario, es decir, más de dos clases los valores de VN y FN se extienden, pudiendo hacer la expansión a más dimensiones. Para estos casos, se puede sacar estos mismos parámetros como para una clase en particular como para el sistema completo.

3. Solución propuesta

3.1. Análisis y preprocesamiento de datos

La base de datos utilizada corresponde a *ALERCE_stamps_2020.pkl*, la cual se compone exclusivamente de un paquete de datos en formato *pkl* proveniente del modulo pickle de python. Este archivo contiene un diccionario con 3 atributos importantes:

- 'Images': Matrices, generalmente cuadradas de 63x63, que contienen 3 muestras para cada valor, representando 3 imágenes para cada objeto astronómico. El primer canal de cada imagen corresponde a "Science", el segundo a "Templatez el tercero a la diferencia entre los dos primeros.
- 'Labels': Número entero que representa la categorización de cada imagen. Se encuentran 5 clases: Active Galactic Nuclei (0), Supernova (1), Variable Star (2), Asteroid (3) y Artifacts (4).
- 'Metadata': Lista de valores que representan información adicional sobre cada imagen.

En términos de distribución se encontró que:

Tabla 1: Información sobre paquete de datos

Cantidad de Ïmages"	52.244
Cantidad de "Labels"	52.244
Cantidad de "Metadata"	52.244
Cantidad de Imagenes no cuadradas	650
Cantidad de valores "NaN"	9.743.759

La primera observación obtenida de la base de datos es el número de instancias presentes en esta, el cual es de 52.244 para sus tres atributos, por lo que se comprueba que no hay errores en las dimensiones de estos.

Los objetos astronómicos presentes en la base de datos siguen la distribución presentada en la figura 4, donde se observa un claro desbalance, sobre todo concerniente al número de supernovas de la base de datos.

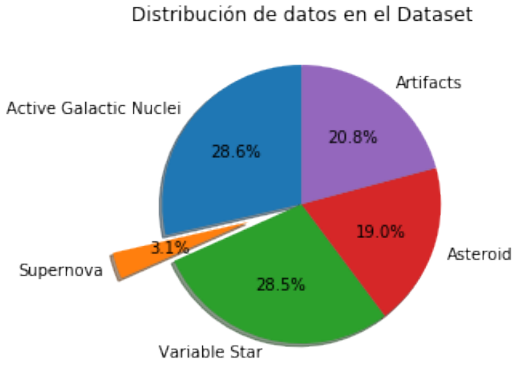

Figura 4: Distribución de Labels

A continuación, se observan ejemplos de las imágenes contenidas dentro de la base de datos.

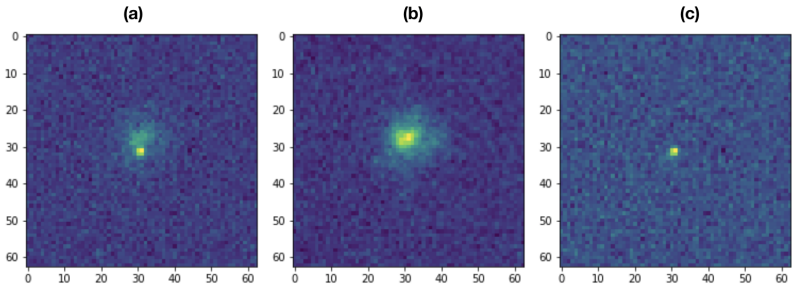

Figura 5: (a), (b) y (c) Muestras de ejemplo con categoría 'supernova' asociada

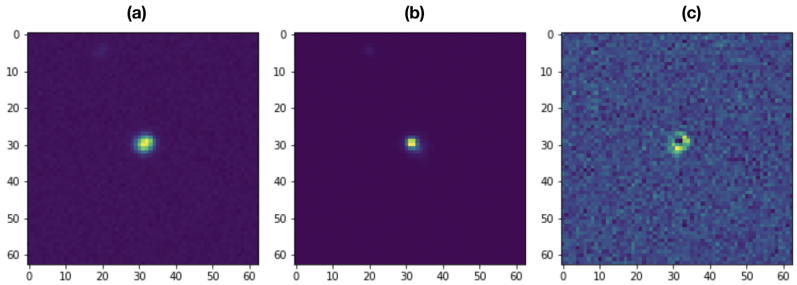

Figura 6: (a), (b) y (c) Muestras de ejemplo con categoría 'estrella variable' asociada

En cuanto a las imágenes no cuadradas, se observó solo 24 supernovas del total de 640 imágenes no cuadradas. Al no ser un número elevado, y por recomendación del cuerpo docente, se decidió eliminar todas las imágenes no cuadradas.

Posteriormente, se añade una nueva lista al diccionario que contiene los datos, donde se almacenan recortes de los 21x21 pixeles centrales de cada imagen.

Los tres canales tanto de las imágenes de 63x63 pixeles, como de aquellas de 21x21, son normalizados independientemente, obteniendo valores entre 0 y 1 para cada valor posible.

Adicionalmente, como se observa en la tabla 1, se ha encontrado una cantidad significativa de valores indeterminados (NaN), los cuales son satisfactoriamente reemplazados por ceros.

En la figura 8 se observan las imágenes recortadas y normalizadas de la supernova ilustrada previamente (figura 5).

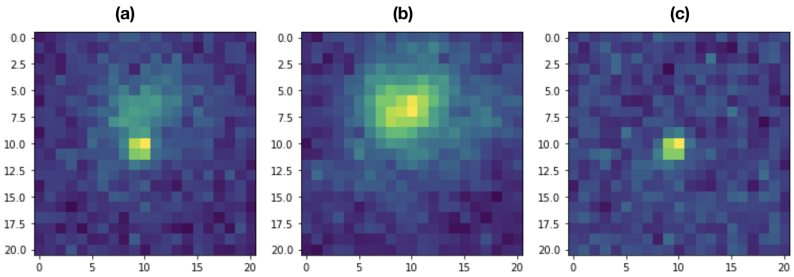

Figura 7: (a, (b) y (c) Muestras de ejemplo de imágenes de supernova de 21x21 pixeles

Tras la realización del preprocesamiento se obtiene un diccionario con 4 atributos: las imágenes completas normalizadas, las imágenes normalizadas y recortadas, los labels de los objetos y la metadata. Este diccionario se almacena exitosamente en un archivo *pkl*, obteniendo la base de datos preprocesada que se utilizará en adelante.

3.2. Creación de sets

Se observa una gran diferencia en la cantidad de total entre los diferentes eventos astronómicos. En lo particular se busca trabajar con supernovas que representan menos de un 4 % del total de los datos.

Para la creación de los sets de validación y testing, se seleccionan aleatoriamente imágenes de cada uno de las 5 categorías, obteniendo conjuntos perfectamente balanceados. De esta forma, el conjunto de validación se compone de 500 instancias, mientras el de testing de 1000.

Por otro lado, para la creación del conjunto de entrenamiento, se realiza oversampling de las 4 categorías menos abundantes, repitiendo imágenes hasta obtener un conjunto balanceado. La repetición de dichas imágenes se realiza de una forma en que para cada categoría, toda imagen está repetida a lo más una vez más que cualquier otra. De esta forma, el conjunto de entrenamiento se compone de 72210 imágenes.

3.3. Invarianza Rotacional

Ciertas Redes Convencionales CNN incluyen capas rotacionales que ayudan en cierta medida a obtener invarianza rotacional. Esto promediando los resultados obtenidos para distintas rotaciones de cada imagen. Más aún, se ha demostrado que este efecto ayuda a mejorar la variable de *accuracy*.

Tomando este concepto se intenta entrenar el algoritmo con imágenes ya rotadas. Para el caso de eventos astronómicos hace sentido decir que no existe una orientación correcta. Varios autores han intentado replicar esta técnica jugando con la simetría y entregando como input la imagen rotada por k,90ž, $k \in \{0, 1, 2, 3\}$. De esta forma cada *batch* de imágenes aumenta a un tamaño $4N$.

En la última parte del algoritmo, antes de las capas *fully-conected*, se usa un *pooling* cíclico que calcula los promedios de *pooling* sobre la representación dense de cada ejemplo rotado. De esta forma las capas *fully connected* consideran la información de cada una de las rotaciones evaluadas.

La idea de lo anterior se puede visualizar en la figura 8. Cabe destacar que los números de parámetros y conexiones no son las que se utilizan pero es la estructura en general la que interesa.

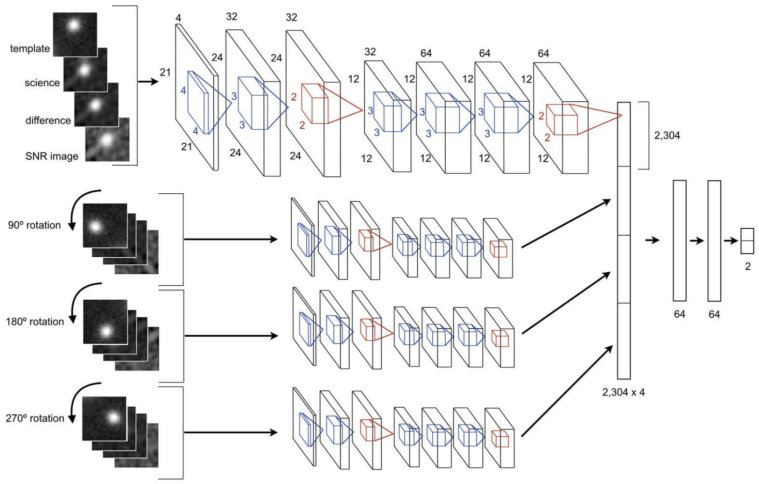

Figura 8: Ejemplo de red convolucional usando técnica de invarianza rotacional [2].

Para implementar esta solución, se utilizó capas *lambda* de tensorflow, las que permiten aplicar una función cualquiera a la entrada, usando *image.rot*, que da la posibilidad de rotar una imagen en $90k^{o}$. Así, se pasará la entrada por funciones rotación como la siguiente:

Código 1: Implementación Rotación

```
rot_90 = tf.keras.layers.Lambda(lambda x: tf.image.rot90(x, k=1))(img_inputs)
```

La siguiente imagen ilustra las cuatro entradas de las capas convolucionales:

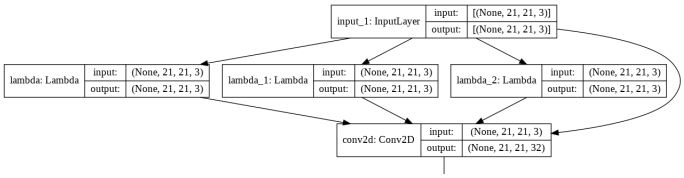

Figura 9: Nuevo modelo tensorflow de CNN con la entrada original y las tres capas lambda de rotaciones

3.4. Modificación de la función Loss

Para efectos prácticos del proyecto, la función de costos *cross entropy* tiende a saturar en los valores de salida 0 y 1 sin tomar en cuenta los valores de entremedio. En este caso, no se poseen valores entremedio dado que los valores de predicción $f(\vec{x})$ y verdadera etiqueta $y(\vec{x})$ solo pueden tomar valores de 1 si es la clase o 0 si no lo es.

Se cree que crear una escala mas gradual permitiría poder tener mejor criterio para mejorar *recall* y *precision*. La idea es penalizar la alta confianza del algoritmo cambiando la función de costos a una entropía regulada:

$$H_r(f(\vec{x}), y) = -\sum_{x \in X} y(\vec{x}) log\ f(\vec{x}) + \beta \sum_{x \in X} f(\vec{x}) log\ f(\vec{x})$$

La primera parte de la ecuación corresponde a la entropía cruzada, mientras la segunda parte es el termino que permite ir regulando, multiplicado por un valor de β arbitrario. Para los experimentos se utilizó $\beta = 0{,}5$.

Una nueva función de costo representaba un desafío, teniendo que editar funciones de la estructura *keras* mismo. Finalmente, para su creación se implementó el siguiente código:

Código 2: Implementación Entropia Regulada

```
@tf.function
def crossentropy_regulated(y_true: TensorLike, y_pred: TensorLike, beta: FloatTensorLike = 0.5,
    ↪ from_logits: bool = False,
) -> tf.Tensor:
   y_pred = tf.convert_to_tensor(y_pred)
   y_true = tf.convert_to_tensor(y_true, dtype=y_pred.dtype)

    # Entropia cruzada por defecto
   ce = tf.keras.losses.sparse_categorical_crossentropy(y_true, y_pred, from_logits=from_logits)

   #se cambia formato para que calcen las dimensiones con la nueva fórmula
   y_pred2 = math_ops.cast(y_true, y_true.dtype)

   #Entropia Regulada
   ce2=K.sparse_categorical_crossentropy(y_pred2, y_pred, from_logits=from_logits)

  # se computa el final loss
   return tf.reduce_mean(ce)+beta*tf.reduce_mean(ce2)
```

3.5. Detalles algoritmo utilizado

La estructura utilizada, se inspira en aquella utilizada en la bibliografía ([1], [3]). La entrada de la red corresponde a las imágenes de 21 x 21 pixeles, agrupadas en batches. Tras la entrada se añaden 3 capas lambda, las cuales, se encargan de realizar las tres rotaciones deseadas: 90°, 180° y 270°. La salida de cada una de estas capas, al igual que la entrada original, ingresan a la parte convolucional de la red.

La parte covolucional de la red se compone de 5 capas convolucionales con un *padding "same"* y kernels de 3x3, además de dos capas *maxpooling*, ubicadas entre la segunda y tercera capa convolucional y tras la última capa convolucional.

Luego de la parte convolucional de la red, se incluyen dos capas propias de *tensorflow*, las cuales se encargan de juntar y promediar las 4 versiones rotadas de cada imagen.

Finalmente la red cuenta con tres capas *fully-connected*, donde la salida de la última es de 5, coincidiendo con el número de categorías posibles para los objetos astronómicos.

En la tabla 2 se observa el detalle capa a capa de la red, donde se aprecia que esta utiliza un total de 210.117 parámetros.

Tabla 2: Estructura del modelo

Layer	**Salida output**	**Número de parámetros**
InputLayer	(None, 21, 21, 3)	0
3 x Lambda	(None, 21, 21, 3)	0
Conv32-3	(None, 21, 21, 32)	1568
Conv32-3	(None, 21, 21, 32)	9248
Maxpooling	(None, 10, 10, 32)	0
Conv64-3	(None, 10, 10, 64)	18496
Conv64-3	(None, 10, 10, 64)	36928
Conv64-3	(None, 10, 10, 64)	36928
Maxpooling	(None, 5, 5, 64)	0
Stack	(4, None, 5, 5, 64)	0
Promedio	(None, 5, 5, 64)	0
Flatten	(None, 1600)	0
Dense64	(None, 64)	102464
Dense64	(None, 64)	4160
Dense5	(None, 5)	325

Se utiliza un optimizador Adam, disponible en la librería *keras* de *tensorflow*. Tras experimentar con diversos valores en el parámetro *learning rate*, se decide dejar el valor por defecto de este $(0,001)$. Los demás parámetros de dicho optimizador no son alterados.

Como métricas a maximizar, se considera en primera instancia el *accuracy* global de la detección de los 5 tipos de objetos astronómicos presentes en la base de datos. Adicionalmente, debido a que se considera que las supernovas son el objeto astronómico más importante de detectar, debido a su

escasez en la base, se toma un enfoque especial para dicho objeto.

Se considera que, para el caso de las supernovas, es preferible un alto *recall* por sobre una alta métrica de *precision*, por lo que se estudia también el F_2 score resultante en cada ensayo. Esto ya que dicha métrica considera un balance entre *recall* y *precision*, donde la primera métrica es más importante.

Se realizan 3 ensayos, los cuales son repetidos varias veces. El primero de estos utiliza la red sin la invarianza rotacional, el segundo con invarianza rotacional y entropía cruzada, mientras que el tercero, utiliza la invarianza rotacional y entropía regulada como función de costos.

En cuanto al criterio de detención, pese a que originalmente se pensaba implementar un sistema de detención temprana basado en la detección de una disminución constante del *accuracy* de validación, finalmente no se utiliza dicha solución. Esto debido a que, con la tasa de aprendizaje utilizada, se observa que, en caso de esperar, por ejemplo, 5 iteraciones en que el *accuracy* de validación descienda, la caída producida en dicha métrica es de más del 5 %, por lo que dicha solución no parece óptima.

Por lo tanto, en caso de querer utilizar este sistema de detención temprana habría que disminuir fuertemente la tasa de aprendizaje, lo que aumentaría de sobremanera el número de iteraciones previas al sobreajuste, y con esto el tiempo necesario del algoritmo.

De esta manera, antes de definir el número de iteraciones de cada ensayo, se realiza dicho ensayo con 40 iteraciones 4 veces, identificando un número de épocas que, en promedio, maximicen el *accuracy* de validación. Así, para el ensayo en que no se utiliza invarianza rotacional, el algoritmo se detiene en 19 épocas, para aquel con invarianza rotacional y entropía cruzada en 22, y para aquel con invarianza rotacional y entropía regulada en 26.

Los parámetros definidos por el grupo son la tasa de aprendizaje (aunque esta coincide con el valor por defecto), el número de épocas a realizar en cada ensayo, el número de imágenes por *batch* y el parámetro β de la entropía regulada. Por otro lado, los demás parámetros del optimizador son definidos automáticamente.

4. Resultados

En la presente sección se explican los resultados obtenidos para cada uno de los tres ensayos realizados. Cada uno de dichos ensayos es repetido 5 veces, obteniéndose métricas promediadas.

Para comprender los resultados obtenidos en las matrices de confusión es importante recordar la categorización de los objetos: 0: Active Galactic Nuclei, 1: Supernova, 2: Variable Star, 3: Asteroid, 4: Artifacts.

4.1. Primer ensayo: CNN sin invarianza rotacional, uso de cross-entropy

Para este caso, no se utilizan las capas lambda utilizadas para la invarianza rotacional, por lo que las imágenes de entradas son directamente el input de la primera capa convolucional. Este ensayo es replicable ejecutando la subsección "Modelo paper sin rotaciones"de la sección "Modelos sin Invarianza Rotacional"del archivo *Çlasificador.ipynb"* del repositorio.

Para este caso se obtiene un accuracy global promedio de $86,52 \pm 0,24\,\%$ Además, para el caso de las supernovas, se obtiene un precision de $94,4 \pm 1,9\,\%$ y un recall de $73,6 \pm 1,95\,\%$, lo que da valores de $82,69 \pm 0,96$ y $76,98 \pm 1,56$ para F-1 score y F-2 score respectivamente.

El tiempo promedio que el algoritmo tarda en finalizar una época es de 3,3 segundos, utilizando una GPU de Google Colab.

En la figura 10, se puede ver un ejemplo de matriz de confusión obtenida para este caso. Dicho ensayo obtiene un $86\,\%$ de accuracy global. Las filas de esta matriz están normalizadas, representando cada número la fracción de detección de cada objeto respecto a su *true label*.

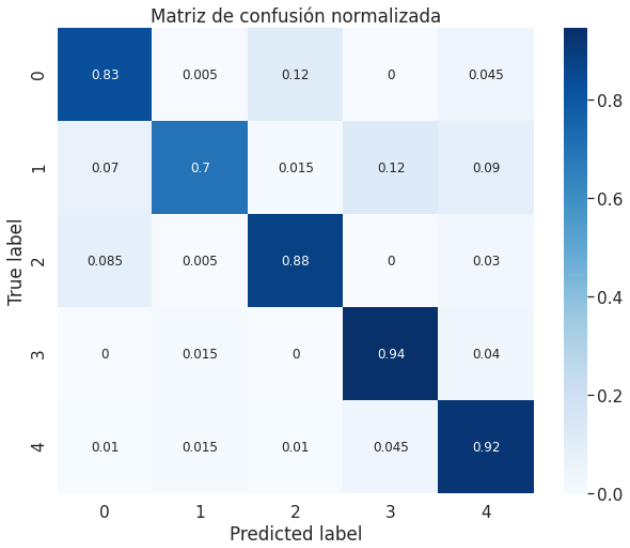

Figura 10: Ejemplo de matriz de confusión sin implementación de invarianza Rotacional

4.2. Segundo ensayo: CNN con invarianza rotacional, uso de cross-entropy

En este caso, se aplica la invarianza rotacional, introduciendo las 3 capas lambdas correspondientes. Este ensayo es replicable ejecutando las subsecciones Çonstrucción Redz Çross Entropy"de la sección "Modelo con Invarianza Rotacional"del archivo *Çlasificador.ipynb"* del repositorio.

El tiempo de ejecución de cada época aumenta a 11,2 segundos, utilizando una GPU de Google Colab.

Se obtienen métricas promedio de $87,68 \pm 0,87\,\%$, $90,20 \pm 2,85\,\%$ y $82,40 \pm 1,29\,\%$ para el accuracy global, precision para supernovas y recall para supernovas respectivamente.

En la figura 11 se puede observar un ejemplo de matriz de confusión generada en este caso, donde se ha obtenido un accuracy de $88,2\,\%$, un poco más alto de la medio de este ensayo. Nuevamente, las filas de esta matriz están normalizadas.

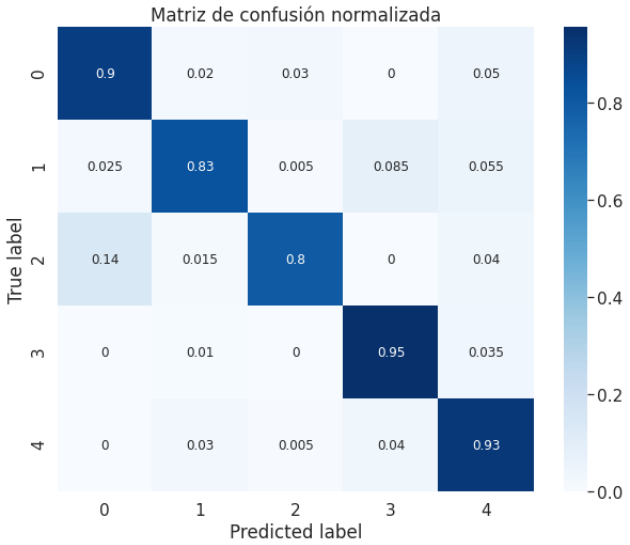

Figura 11: Ejemplo de matriz de confusión con implementación de invarianza Rotacional

4.3. Tercer ensayo: CNN con invarianza rotacional, uso de entropía regulada

Este ensayo es replicable ejecutando las subsecciones Çonstrucción Redz .Entropía Regulada"de la sección "Modelo con Invarianza Rotacional"del archivo *Çlasificador.ipynb"* del repositorio.

Como resultados de este ensayo se obtienen métricas de $88,32 \pm 0,66\,\%$ en *accuracy* global, además de $92,15 \pm 2,49\,\%$ y $82,10 \pm 2,38\,\%$ para *precision* y *recall* respectivamente para supernovas, resultando en F_1 y F_2 scores de $86,78 \pm 0,33\,\%$ y $83,90 \pm 1,59\,\%$ respectivamente.

Al igual que para el segundo ensayo, el tiempo promedio de cada época es de 11,2 segundos.

La figura 12 muestra un ejemplo de matriz de confusión resultante tras la realización de este ensayo.

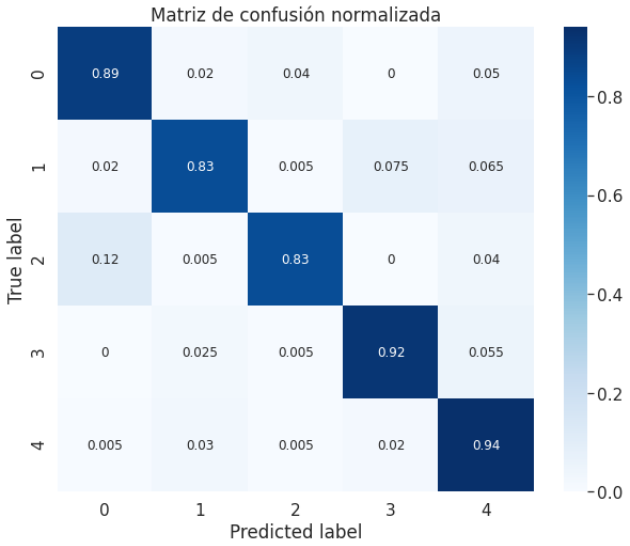

Figura 12: Ejemplo de matriz de confusión para el ensayo 3

La tabla 3 resume los resultados obtenidos para cada uno de los tres ensayos realizados, donde se observa que el ensayo 3 supera tanto en *accuracy* como en F_2 score a los dos primeros ensayos.

Tabla 3: Métricas de rendimiento promedio para cada ensayo enfocados en Supernovas (S.N)

Característica/Métrica	Ensayo 1	Ensayo 2	Ensayo 3
Invarianza Rotacional	No	Sí	Sí
Función de costos	Cross-entropy	Cross-entropy	Entropía Regulada
Épocas realizadas	19	22	26
Tiempo promedio por época [s]	3,3	11,2	11,2
Accuracy Global (%)	$86,52 \pm 0,24$	$87,68 \pm 0,87$	$88,32 \pm 0,66$
Precision S.N (%)	$94,40 \pm 1,90$	$90,20 \pm 2,85$	$92,15 \pm 2,49$
Recall S.N (%)	$73,60 \pm 1,95$	$82,40 \pm 1,29$	$82,10 \pm 2,38$
F-1 S.N (%)	$82,69 \pm 0,96$	$86,11 \pm 1,82$	$86,78 \pm 0,33$
F-2 S.N (%)	$76,98 \pm 1,56$	$83,84 \pm 1,43$	$83,90 \pm 1,59$

5. Discusión

Observando la tabla 3, se puede notar que la aplicación de la invarianza rotacional produce un aumento en todas las métricas utilizadas, salvo *precision*, que se considera la métrica menos importante. Esto demuestra que la utilización de dicha técnica mejora los resultados obtenidos.

Sin embargo, el aumento del *accuracy* global es de solo un 1 %, mientras que el tiempo de ejecución del algoritmo aumenta a casi el triple.

Por otro lado, la utilización de la entropía regulada como función de costos igualmente mejora las métricas de *accuracy* global y *precision* para supernovas, sin embargo, el *recall* de supernovas cae ligeramente. Pese a lo anterior tanto F1 como F2 score aumentan, debido a que la caída del *recall* es muy pequeña comparado al aumento de *precision*.

Se puede notar que, pese a que el aumento en *accuracy* al utilizar entropía regulada pueda parecer pequeño ($\sim 1\,\%$), este aumente es similar a aquel que se gana con la utilización de invarianza rotacional. Esto es bastante importante considerando que el uso de entropía regulada no aumenta el tiempo de cada época (aunque sí la cantidad de épocas para alcanzar el máximo rendimiento), mientras que el uso de invarianza rotacional casi triplica el tiempo de ejecución de cada época.

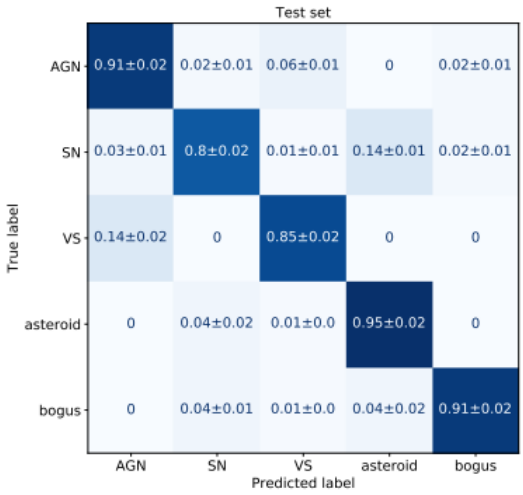

Figura 13: Matriz de confusión obtenida al no usar metadata [3]

Como se observa en la figura 13, los resultados obtenidos en este proyecto son casi idénticos a los obtenidos en la bibliografía al no utilizar la metadata, que fue el caso estudiado. Las pequeñas diferencias obtenidas probablemente son causadas por un algoritmo de detención más preciso y, probablemente, por el uso de una menor tasa de aprendizaje, pero de más iteraciones necesarias.

En una primera instancia se intentó la clasificación de los objetos utilizando imágenes de 63x63 pixeles, lo que permitiría comparar cuánto se pierde al recortar las imágenes. Sin embargo, esto no pudo ser realizado, ya que dichos datos utilizaban toda la memoria RAM disponible en la plataforma Google Colab, y localmente una sola época con dichos datos tardó 15 minutos, resultando imposible

esperar hasta las 20 épocas deseadas.

Finalmente, se puede notar que con los hiperparámetros utilizados, se realizan como máximo 26 épocas de 50 iteraciones cada una, utilizando apenas 1300 datos de entrenamiento, siendo que tras el *oversampling* hecho, se tienen casi 73000 imágenes, dejando la sensación de no usar a cabalidad la base de datos. Sin embargo, al utilizar más datos gracias a la disminución de la tasa de aprendizaje y al cambio del número de épocas realizadas, no se obtienen mejores métricas.

Como alternativa, se propone hacer *oversampling* de las categorías más escasas, pero también *undersampling* de las más numerosas, obteniendo un número de datos de entrenamiento bastante superior a las 1300 imágenes, evitando un sesgo en los datos utilizados y permitiendo el estudio del algoritmo con otros hiperparámetros; pero muy inferior a las 73000. Esta solución reduciría enormemente los recursos computacionales utilizados, tanto en almacenamiento de datos como en tiempo utilizado en cargar la base de datos.

En cuanto a otras alternativas para aumentar el rendimiento del clasificador, se considera, primero que nada, la inclusión de la metadata, ya que es una información que podría ser incluso más valiosa que las mismas imágenes. También, se podrían considerar modificaciones a la estructura de la red, como un aumento en su profundidad. Esto no fue realizado por el grupo, debido a que el presente trabajo se basó en el modelo de la bibliografía.

Debido a que la utilización de la entropía regulada ha logrado incrementar las métricas de rendimiento, otra forma de hacerlo podría ser la utilización de otras funciones de costos que el grupo no probó.

6. Conclusión

Como se ha dicho en la sección de discusión del presente trabajo, los resultados obtenidos en este trabajo son muy cercanos a aquellos obtenidos en la bibliografía en que se basó este proyecto, siendo un primer punto importante a rescatar.

Pese a lo anterior, se considera que los resultados obtenidos son aún mejorables, sobre todo si se utiliza la metadata disponible en la base de datos utilizada. Esto permitiría obtener los resultados finales de la bibliografía, donde se alcanza un 94 % de *accuracy* global.

Se considera que cada uno de los objetivos de este proyecto han sido satisfechos en las distintas etapas de este, logrando la familiarización y manipulación de grandes cantidades de datos, sobre todo durante el preprocesamiento de datos, y la construcción de una red neuronal convolucional capaz de clasificar los 5 tipos de objetos astronómicos disponible, alcanzando un rendimiento similar a aquel alcanzado en la bibligrafía.

Es importante destacar que a lo largo del trabajo el grupo se vio enfrentado a diversos problemas y desafíos, pese a lo cual siempre se supo sacar un buen trabajo adelante, gracias a la perseverancia y las numerosas horas de trabajo dedicadas a este proyecto.

Las principales complicaciones del grupo se debieron principalmente a que para ambos integrantes era la primera vez en que se construía un modelo propio con las librerías utilizadas, por lo que se destinó bastante tiempo a la lectura y prueba de las diferentes funciones utilizadas. Además, se incurrió en pequeños errores durante el preprocesamiento que fueron muy difíciles de identificar a posteriori. Estos errores se pudieron evitar con una aún más ardua revisión de cada línea del código de preprocesamiento, o con la utilización de alguna función ya diseñada para estos propósitos.

Finalmente, destaca la ayuda brindada por el equipo docente, sin la cual hubiera sido bastante difícil cumplir todos los objetivos de este proyecto.

Referencias

[1] Reyes, E., Estévez, P. A., Reyes, I., Cabrera-Vives, G., Huijse, P., Carrasco, R., Forster, F. (2018, July). Enhanced Rotational Invariant Convolutional Neural Network for Supernovae Detection. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). IEEE.

[2] Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., Maureira, J. C. (2017). Deep-hits: Rotation invariant convolutional neural network for transient detection. The Astrophysical Journal, 836(1), 97.

[3] Carrasco-Davis, R., Reyes, E., Valenzuela, C., Förster, F., Estévez, P y otros (2020). Alert Classification for the ALeRCE Broker System: The Real-time Stamp Classifier.

[4] Alerce Science, consultado en Noviembre 2020 http://alerce.science/

[5] Fei-Fei Li: 'CS231n: Convolutional Neural Networks for Visual Recognition', 2016, Standford University, https://cs231n.github.io/convolutional-networks/

[6] Yutaka Sasaki: 'The truth of the F-measure' , 2007, School of Computer Science, University of Manchester, https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf

[7] Jason Brownlee: ' A Gentle Introduction to Pooling Layers for Convolutional Neural Networks' , 2019, Deep Learning for Computer Vision https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/