# Clasificación de imágenes satélites con red neuronal convolucional (CNN), utilizando Keras en GPU con Colaboraty

**Tecnológico de Costa Rica**
Maestría en Electrónica
MP6104. Procesamiento Digital de Señales
Esteban Martínez Valverde,
email: estemarval@gmail.com

***Resumen*—Se presentan los resultados del proceso de clasificación binaria de imágenes satelitales utilizando una red neuronal convolucional / *Convolutional Neural Network* (CNN). El entrenamiento del modelo se realizó de manera secuencial utilizando el método de regularización *dropout*. Se implementaron 4 capas covolucionales, una capa de *max-pooling* y una capa de *dropout*, con un minimo de 20 neuronas y un maximo de 100 neuronas. Dividiendo el set de datos en 80 % y un EPOCH de 20, se obtuvo un precisión de 98 %.**

***Palabras Clave*—CNN, dropout, keras, Colaboraty, GPU,**

## I. Introducción

El uso de aplicaciones satelitales ha crecido en los últimos años, esto debido al mayor acceso que se dispone tanto de herramientas libres como oportunidades de lanzamiento para constelaciones de pequeños satélites. Como resultado de este aumento en el número de aplicaciones, el análisis de imágenes ha crecido de tal manera que ha despertado el interés de la comunidad de resolver los problemas mediante el uso de técnicas de aprendizaje automático. La metodología conocida como *deep learning* ocurre cuando se utilizan mas de 2 o más capas. Donde una capa es básicamente una unidad computacional. Se puede tener una capa llamada "multiplicar por 5"que multiplica cada 5 por 5. Podría alimentar a una capa "multiplicar 10"que toma la salida de la capa "multiplicar por 5$\tau$ la multiplica por 10.

Una CNN consiste en una capa de entrada y salida, así como múltiples capas ocultas. Las capas ocultas de una CNN consisten típicamente de de capas convolucionales, capas de agrupamiento, capas totalmente conectadas y capas de normalización [8]. A continuación se en listan estas capas y otras características importantes de las CNN [7].

- **Capas Convolucionales:**
  En esta capa se realiza el proceso de convolución, que matemáticamente, es una correlación cruzada en lugar de una convolución propiamente. Esto solo tiene importancia para los índices en la matriz y, por lo tanto, qué pesos se colocan en qué índice. Los parámetros de la capa consisten en un conjunto de filtros entrenables (*kernels*), que tienen un pequeño campo receptivo, pero se extienden a través de la profundidad total del volumen de entrada.
- **Capas de agrupación (*pooling*:)**
  Las redes convolucionales pueden incluir capas de agrupación locales o globales, las cuales combinan las salidas de las neuronas a una capa en una sola neurona en la siguiente capa. La capa de *pooling* sirve para reducir progresivamente el tamaño espacial de la representación, esto para reducir el número de parámetros y la cantidad de cálculos que requiere la red y, por lo tanto, también para controlar el *overfitting*. El *maxpooling*, por ejemplo, utiliza el valor máximo de cada un grupo de neuronas de la capa anterior, mientras que *average pooling* usa el promedio del grupo de neuronas de la capa anterior (MLP, *multi-layer perceptron neural network*).
- **Capas totalmente conectadas:**
  Las capas totalmente conectadas, conectan cada neurona de una capa con cada neurona de otra capa. En principio, es lo mismo que la red neuronal de perceptrón multicapa tradicional
- **Pesos (*Weights*):**
  Cada neurona en una red neuronal calcula un valor de salida aplicando alguna función a los valores de entrada provenientes del campo receptivo en la capa previa. La función que se aplica a los valores de entrada se especifica mediante un vector de pesos y un vector de *bias* (normalmente números reales). El aprendizaje en una red neuronal progresa haciendo ajustes incrementales a los bias y pesos.
- **Capa ReLU:**
  Por sus siglas en inglés, (*Rectified Linear Units*), está capa aplica la función de activación no-saturable. Aumenta las propiedades no lineales de la función de decisión y de la red global sin afectar los campos receptivos de la capa de convolución. También se usan otras funciones para aumentar la no linealidad, por ejemplo, la tangente hiperbólica (saturable) y la función *Sigmoid*
- **Capa de pérdidas:**
  Esta capa determina cómo el entrenamiento penaliza la desviación entre las etiquetas predichas y verdaderas y normalmente es la capa final. De esta manera se pueden usar diferentes funciones de pérdidas, apropiadas para diferentes tareas.

Otro proceso importante en las CNN, es la regularización, el cual se encarga de introducir información adicional para resolver un problema mal planteado o para prevenir el *overfitting*. A continuación se describen los dos tipos de regularización utilizados en este proyecto:

- **Dropout:**
  Un una capa totalmente conectada tiende a ocupar la gran mayoría de los parámetros y producir *overfitting*. El *dropout*, resuelve esto ya que en cada etapa de entrenamiento, los nodos individuales son eliminados de la red con probabilidad $1-p$. Los valores para esta probabilidad se aconseja que sea en el rango de 20 % - 50 % [5].
- **Early stopping:**
  Es uno de los métodos más simples para evitar el *overfitting* de una red, el cuál se basa en simplemente detener el entrenamiento antes de que se haya producido el *overfitting*. Su desventaja es que el proceso de aprendizaje se detiene.

El uso de GPU para la implementación de CNN se ha venido haciendo aproximadamente desde el año 2005 [4], ya que la arquitectura e implementación de los GPU ha venido mejorando desde entonces. Recientemente han surgido nuevos servicios pago como AWS [1], en donde se pueden configurar maquinas virtuales, con los requerimientos que el usuario defina. Sin embargo la compañía Google ha creado una nueva herramienta, la cuál se basa en un *Jupyter Notebook*, con los paquetes y librerías más comunes para el aprendizaje automático y *deep learning*. Esta herramienta llamada Colaboraty, tiene la capacidad de ejecutarse en el computador local (del usuario) o en una maquina virtual que puede ser utilizada con un GPU. Colaboraty cuenta la gran ventaja que es totalmente gratis, es posible que varias personas trabajen en linea simultáneamente y es posible instalar otras librerías que no vienen pre-instaladas [2].

A continuación se presentará el uso de Colaboraty, ejecutándose con un GPU, para generar, entrenar y evaluar una CNN utilizando la librería Keras con TensorFlow como *backend*.

## II. Red Neuronal Convulucional

Utilizando la documentación de Keras [3] para la implementación de una CNN, se obtiene una un modelo secuencial que se compone de:

- 4 capas convolucionales con función de activación ReLu
- 1 capa de max-pooling con función de activación ReLu
- 2 de regularización con *dropout*.
- 1 capa de salida con función de activación *sigmoid*.

En la Figura 1, se muestra el resumen del modelo compilado con la distribución de capas final. Además se muestra que la cantidad de parámetros generados son 10440941, de los cuales su totalidad son entrenables.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_5 (Conv2D) | (None, 78, 78, 20) | 560 |
| activation_7 (Activation) | (None, 78, 78, 20) | 0 |
| conv2d_6 (Conv2D) | (None, 76, 76, 40) | 7240 |
| activation_8 (Activation) | (None, 76, 76, 40) | 0 |
| conv2d_7 (Conv2D) | (None, 74, 74, 60) | 21660 |
| activation_9 (Activation) | (None, 74, 74, 60) | 0 |
| conv2d_8 (Conv2D) | (None, 72, 72, 80) | 43280 |
| activation_10 (Activation) | (None, 72, 72, 80) | 0 |
| max_pooling2d_2 (MaxPooling2 | (None, 36, 36, 80) | 0 |
| flatten_2 (Flatten) | (None, 103680) | 0 |
| dense_3 (Dense) | (None, 100) | 10368100 |
| activation_11 (Activation) | (None, 100) | 0 |
| dropout_2 (Dropout) | (None, 100) | 0 |
| dense_4 (Dense) | (None, 1) | 101 |
| activation_12 (Activation) | (None, 1) | 0 |

Total params: 10,440,941
Trainable params: 10,440,941
Non-trainable params: 0

Figura 1: Resumen de la conformación del modelo para la CNN con *dropout*.

Inicialmente, se generaron rutinas para encontrar los mejores resultados de precisión en el entrenamiento, en función de los hiperparametros tanto del modelo como del entrenamiento. Luego de una exhaustiva búsqueda se determinó el siguiente conjunto de hiperparametros como aceptable:

- Número de capas = 4
- Mínimo de Neuronas = 20
- Máximo de Neuronas = 100
- Drop Rate = 0.5
- EPOCHS = 20
- Tamaño de BATCH = 20

Donde *Drop Rate* es la fracción de unidades de entrada que descartaran. El EPOCH es el número de veces que se utiliza el conjunto de datos por completo. El tamaño de BATCH es el numero de muestras de entrenamiento que se utilizan en un BATCH (el BATCH es una fracción del conjunto de datos que se utiliza para hacer más eficiente el proceso de entrenamiento). Se necesita un determinado número BATCHES para completar un EPOCH.

Se realizaron dos modelos, uno sin utilizar la capa de *dropout* y otro utilizando el *dropout*. Esto ya que esta capa no es considerada como una capa típica a utilizar en las CNN. Por lo que parte de los resultados que se verán mas adelante es la comparación entre estos dos modelos.

## III. Conjunto de imágenes satelitales

El conjunto de imágenes (*dataset*) se obtuvo del repositorio llamado Kaggle, donde un usuario de esta plataforma utilizó varias imágenes tomadas en la bahía de San Francisco, utilizando los satélites de la compañía Planet [6], y se dio la tarea de generar 4000 imágenes mas pequeñas (tamaño de 80x80p) cada una. Las imágenes del satélite fue considerado como escenarios, ya que contaba con varios barcos por imagen, como se muestra en la Figura 2.

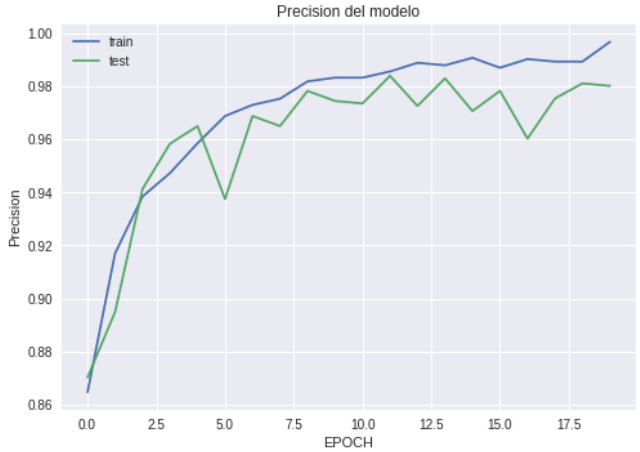
Además, colocó una etiqueta (*label*) en el nombre de cada archivo, indicando si hay o no un barco dentro de la imagen. Se parte de este conveniente conjunto de datos ya que se dispone de todo lo necesario para empezar entrenar modelos y evaluar sus predicciones.

Figura 2: Escenario utilizado en la generación del conjunto de datos, obtenido en Kaggle [6].

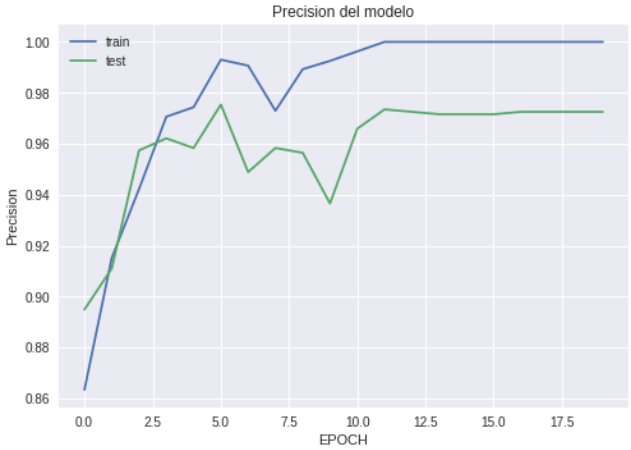
Los imágenes fueron pre-procesadas utilizando la librería *Numpy*, con el cuál se obtiene las dimensiones y realiza un escalamiento entre un valor de 512, con el objetivo de disminuir la cantidad de procesamiento requerido. Además se obtienen los vectores de entrada y el vector de salida dividido en 80 % para el entrenamiento (*training*) y 20 % para la evaluación (*testing*).

## IV. Resultados y análisis

Con los dos modelos compilados y los hiperparámetros seleccionados, se procede a realizar ciertas evaluaciones para determinar el comportamiento y diferencias en el uso de una capa con *dropout*. Inicialmente, se utilizó el historial que genera la función de entrenamiento creada, para visualizar el comportamiento del entrenamiento con respecto a la cantidad de EPOCHS utilizados. Específicamente, este comportamiento fue medido en términos de la precisión y las pérdidas durante el entrenamiento. En las Figuras 3a y 3b, se muestran las gráficas de las precisíones obtenidas para el modelo con y sin *dropout*, respectivamente. De donde observa la eliminación del *overfitting* al implementarse el *dropout*.




(a) Con *Dropout*.




(b) Sin *dropout*.

Figura 3: Gráficas de las precisiones contra la cantidad de EPOCH, obtenido durante el entrenamiento (*train*) y validación (*test*).

De forma similar, en las Figuras 4a y 4b, se muestran las gráficas de las pérdidas obtenidas para el modelo con y sin *dropout*, respectivamente. De donde se evidencia el efecto de regularización, al presentarse menos fluctuaciones tanto en el *train* como en le *test*.


(a) Con *Dropout*.


(b) Sin *dropout*.

Figura 4: Gráficas de las pérdidas contra la cantidad de EPOCH, obtenido durante el entrenamiento (*train*) y validación (*test*).

Seguidamente, se realiza una predicción con el modelo entrando, utilizando *dropout* y se obtiene la matriz de confusión, la cuál es una herramienta que permite la visualización del desempeño de un algoritmo. Esta muestra la precisión que se tiene en las predicciones con respecto a las etiquetas involucradas. En la Figura 5, se muestra la matriz de confusión, donde se obtienen un 99 % para las predicciones verdaderas y el valor real era verdadero. Mientras que para las predicciones falsas y valor real falso, se obtiene un 97 %.

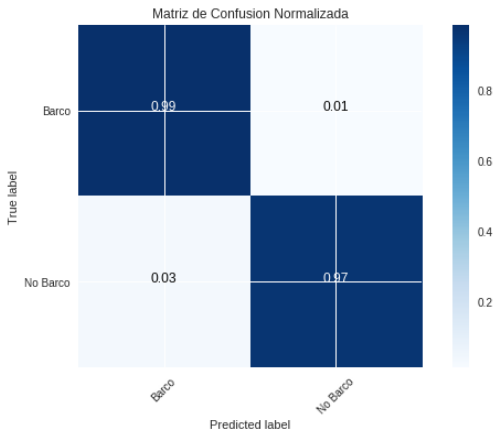



Figura 5: Matriz de confusión para la predicción, modelo con *dropout*.

Se realiza una visualización de los errores cometidos por la predicción de validación. En la Figura 6, se muestran las imágenes

con predicciones incorrectas, donde cada etiqueta contiene una ”P”(predicción) y una R”(real). En total hubo 14 errores.

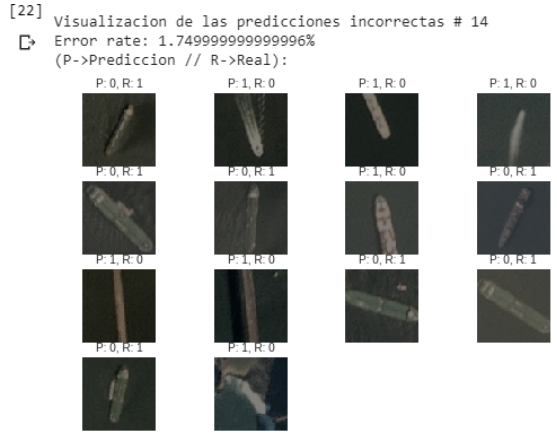



Figura 6: Visualización de los errores para la predicción, modelo con *dropout*.

Finalmente se obtiene un resumen del comportamiento de ambos modelos. En la Tabla I, se muestran las mediciones del tiempo de entrenamiento, precisión y valor F1 (macro). Se observa en todas las mediciones se obtiene una mejora cuando se utiliza la capa de *dropout*.

| Modelo | T. Entrenamiento (s) | Precisión | F1_macro |
|---|---|---|---|
| Con Dropout | 197.64 | 98.2 | 97.7 |
| Sin Dropout | 199.70 | 97.5 | 96.6 |

Tabla I: Mediciones de precisión y f1 para los dos modelos propuestos.

Cabe destacar, que las mediciones en el tiempo de entrenamiento, se realizaron utilizando el GPU disponible en la herramienta. Para efectos de comparación, se realizó una medición en el tiempo entrenamiento del modelo con *dropout* y se obtuvo un valor de 4347 segundos, casi 22 veces más lento que con GPU.

## V. Conclusiones

Se realizaron pruebas con dos modelos diferentes de CNN (con y sin *dropout*), utilizando la herramienta Colaboraty y un acelerador GPU, demostrando la gran ventaja de utilizar nuevas herramientas libres en el desarrollo y evaluación de redes neuronales.

Al utilizar el modelo con *dropout* se obtuvo:

- Una precisión del 98 %.
- Menor tiempo de entrenamiento.
- Un entrenamiento sin *overfitting*.

## Referencias


[1] Amazon web services. Disponible en: https://aws.amazon.com/.

[2] Colaboraty google. Disponible en: https://colab.research.google.com/.

[3] Docs convolutional layers - keras. Disponible en: https://keras.io/layers/convolutional/.

[4] matecconf_se2018_01008 @ www.matec-conferences.org.

[5] Need help with deep learning? take the free mini-course dropout regularization in deep learning models with keras. Disponible en: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/.

[6] Ships in satellite imagery. Disponible en: https://www.kaggle.com/rhammell/ships-in-satellite-imagery/home.

[7] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1994.

[8] Http://deeplearning.net/. Lenet @ Deeplearning.Net. Disponible en: http://deeplearning.net/tutorial/lenet.html.