Universidad Francisco Marroquín
Facultad de Ciencias Económicas
Curso: Machine Learning Models
Nombre: Juan Fernando González
Fecha: 05-05-2020

# Reporte Redes Neuronales Convolucionales

## Introducción

El propósito de este reporte es poder presentar mi proyecto de redes neuronales convolucionales aplicada a predecir una clase a partir de una imagen. En este proyecto se desarrolló y se aplicaron técnicas de normalización de datos para el rendimiento del cálculo computacional, el diseño de la arquitectura de una red convolucional, la cual puede tener capas convolucionales 2D, capas de *maxpooling,* capas de *dropout*, y capas densas, más adelante se explicará de mejor manera cada una de estas capas. Además, también se explicarán los resultados obtenidos, los parámetros escogidos y usados para el modelo, y se utilizarán gráficas y herramientas que sustenten y apoyen al análisis del modelo y su rendimiento. El dataset que escogí para realizar este proyecto es el CIFAR 10, el cual consiste en 60,000 imágenes a color de 32x32 pixeles, que están clasificadas en 10 clases, por lo que cada clase tiene 6,00 imágenes. Estas imágenes serán útiles para el entrenamiento y visualización del modelo. Algunas de las imágenes de este dataset se podrán ver en el apéndice. Las clases que mi modelo tratara de clasificar son las siguientes: Aviones, automóviles, pájaros, gatos, venados, perros, ranas, caballos, barcos y camiones.

## Arquitectura y resultados

Dicho lo anterior, puedo describir la arquitectura de mi red neuronal convolucional desarrollada en el proyecto. La primera capa de mi red es la del *input*, la cual tiene una dimensión de 32x32x3, y esto quiere decir que las imágenes de input son de 32 por 32 pixeles, y el 3, se refiere a las capas del RGB (Red, Green, Blue), ya que estas imágenes son a color. En total, mi arquitectura contiene cuatro capas convolucionales, el método de activación para todas las capas de convolución será la función RELU. Empecé a ingresar las capas convolucionales luego de mi capa de input, la primera capa contiene 32 filtros, y un *kernel size* de 3x3, que es el tamaño que cada filtro de convolución tendrá. La siguiente capa corresponde a una de convolución, la cual tiene la misma dimensión que la primera capa de convolución de 32 filtros y un *kernel size* de 3x3. Luego de estas 2 capas convolucionales, esta la capa de *MaxPooling,* con un *pool size* de 2x2, y después de esta última capa, esta la capa de *Dropout,* la cual tiene como parámetro desactivar el 25% de las neuronas en la capa.

Continuando con las capas de convolución, en mi arquitectura agregue 2 capas más, de 64 filtros y un *kernel size* igual a las capas convoluciones anteriores. Se aplico una capa más de *MaxPooling* y de *Dropout* de un 25% al igual que la anterior. Por último, se agregaron tres últimas capas, una densa de 256 neuronas, lo cual quiere decir que es una capa completamente conectada, lo que significa que todas las neuronas de una capa están conectadas a las de la capa siguiente, seguida de otra capa de *Dropout* del 30%, y la última capa con 10 neuronas, las

cuales corresponden al numero de clases de output del dataset, está a diferencia de las demás, tiene la función de activación *softmax*.

Luego de configurar la arquitectura de mi red convolucional, procedí a compilar mi modelo para poder pasar a entrenarlo. Realice 5 modelos, todos con diferentes parámetros y arquitecturas, pero me quede con el que me dio un mejor *accuracy* y el que fue mejor que el resto en cuestión de rendimiento en general. Al principio probé un modelo con 50 epochs y un batch size de 32, ese me dio un accuracy de 73%, el cual estaba por debajo del mínimo requerido, por lo que seguí probando con otros, modificando los parámetros y probando modificar mi arquitectura, hasta que di con la arquitectura ya antes mencionada y con los parámetros de entrenamiento para mi modelo. El entrenamiento de este modelo duro 5 horas y 40 minutos, se llevaron a cabo 100 *epochs*, se asignó un *batch size* de 32, fue entrenado con 50,000 datos de entrenamiento, y validado con 10,000 datos de validación. El *test accuracy* obtenido después del entrenamiento fue del 80%, y un 0.66 de *test loss.* Las gráficas del *test accuracy* y del *test loss* presentan un overfitting en el modelo, ya que el *accuracy* en los datos de entrenamiento va en incremento a medida que los *epochs* avanzan, sin embargo, no lo es así con los datos de validación, ya que a partir del *epoch* 40 se empieza a aplanar la curva, y tiene una tendencia constante al avanzar los *epochs*. Por otro lado, en la matriz de confusión, se muestra un traslape o una “confusión” entre las predicciones de las clases de los gatos con las aves, más adelante en el apéndice se adjuntará una imagen de un ejemplo de este traslape al graficar unas predicciones de ejemplo que realizo el modelo. En este se ve claramente los errores que comete el modelo al tratar de predecir un pájaro, ya que los predice como gato, uno más evidente que otro, pero contiene ese sesgo en esas 2 clases en específico. Otro *insight* que se obtuvo con la matriz de confusión fue que el modelo también tenia cierto problema en clasificar los perros, ya que en casos los calificaba como gatos, y en modelos anteriores, en los cuales tuve un menor *accuracy* había varias veces que el modelo confundía a los perros con gatos, en el apéndice adjuntare las imágenes para demostrar estos *insights.*

## Feature maps

Luego de entrenar mi modelo, comencé a visualizar los filtros por las cuatro capas de convolución que configuré en mi arquitectura, así como también los *feature maps* que se fueron generando a partir de estos filtros en la imagen *input.* Como ya había dicho en el párrafo anterior, mi arquitectura posee 4 capas de convolución, dos capas con 32 filtros y otras dos capas de 64 filtros. Los índices de mis capas son los siguientes: {0, 1, 4, 5}. Utilice la imagen de un caballo que se encontraba en mi data set para poder visualizar y analizar de mejor manera como iba mi red aprendiendo de estos *feature maps*. Al visualizar el *feature map* de la primera capa de convolución, obtuve un resultado de 32 filtros, y cada uno de ellos se enfocaba en diferentes aspectos de la imagen. En los primeros 4 filtros del *feature map*, pude observar como se enfocaban en la textura o contorno de la figura del caballo, otro filtro en solo la figura del caballo sin el fondo, otro en los colores fuertes o colores brillantes, y otro en solo el fondo, sin la figura del caballo. En el filtro número 28 del feature map de esta capa, fue uno de los que más me llamo la atención, ya que en ese filtro se logra apreciar la profundidad y de cierta manera, se puede ver al caballo como en una especie de *gradiente*, se puede ver bien la figura de los músculos

resultados de las patas y la cabeza. Por otro lado, se empezó a analizar el *feature map* de la segunda capa de convolución. En esta capa se empieza a dificultar un poco más el análisis por cada filtro. En los primeros 3 filtros se puede seguir observando que las filtros contienen los contornos de la figura del caballo, como también, los enfoques en tonos brillantes. En esta capa, uno de los filtros que me llamaron la atención fue el numero 19, ya que parece enfocar las partes frontales del caballo, y parece dar la impresión de que la luz se refleja sobre él. Sin embargo, no todos los filtros van de acuerdo con lo que se vio en el primer *feature map*.

A medida que nos acercamos a la capa output del modelo, se hace más y más difícil la interpretación de los *feature maps.* En las ultimas dos capas de convolución de mi modelo, pude analizar lo siguiente, los filtros dentro de la capa número tres de convolución, muestran detalles más específicos, por ejemplo, en el filtro numero 0 se puede visualizar la pata del caballo, en el filtro numero 28 parece visualizarse la cara del caballo. Cabe recordar que estos feature maps ya son de 64 filtros, por lo que hay más filtros que pueden detectar nuevas figuras o contornos de las imágenes. Luego, en la última capa de convolución de 64 filtros, ya solo parecen verse pixeles y el análisis se vuelve demasiado complejo, sin embargo, aún se logra apreciar ciertas características especificas de la imagen. En el filtro numero 2 del *feature map* parece enfocar el ojo y la oreja del caballo, en el filtro 0 un ojo, y en el filtro 29 una pata. Una cosa a tomar en cuenta también en este análisis es que ya de por sí, las imágenes que trae el dataset están pixeladas, por lo que incluso a simple vista, la imagen original a veces cuesta un poco entenderla, por lo que esto puede también influir en el análisis de los feature maps por filtros en las capas de convolución. Y esto es el resultado esperado de una red convolucional, ya que, al principio, entre más cerca de la capa de input, los filtros y los feature maps contienen una información mas entendible y detallada de la imagen, y a medida que se va acercando a la capa de output, es más difícil para el humano interpretar cada filtro en el *feature map*, pero para la computadora no lo es. Al finalizar mi análisis con esa imagen, hice lo mismo con la imagen de un pájaro, y los insights de los *feature maps* por capa fueron los mismo o similares, por lo que logré consolidar mejor mi análisis.

## Conclusiones y recomendaciones

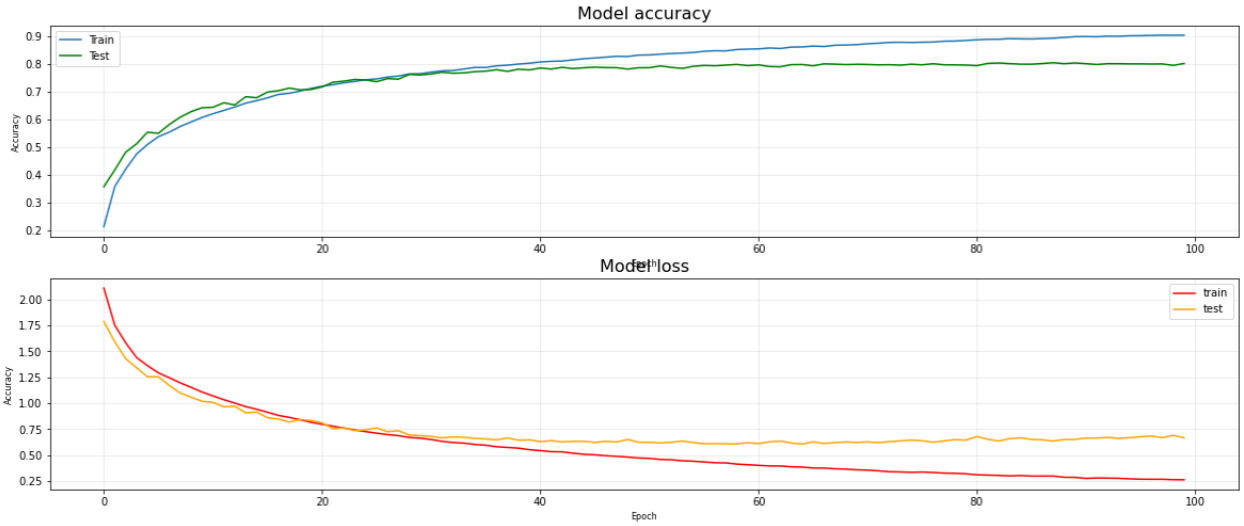
Luego del proceso de exploración y análisis de las redes neuronales convolucionales, y de los feature maps que hay por cada capa de convolución puedo concluir que la red aprende de las imágenes basándose en los diferentes filtros que captan características diferentes de la misma, características como la profundidad, detección de bordes en la figura que se encuentra en la imagen, los fondos o el contraste, la figura que se encuentra en la imagen. Además, como ya había mencionado antes, los feature maps que se encuentren más cerca a la capa de input, tendrán detalles más visibles e interpretables para nosotros los humanos, podremos entender de mejor manera esos detalles que cada filtro destaca en el feature map de esas capas. Por otro lado, las capas más lejanas a la capa de input y más cercanas a la capa final de output, serán menos interpretables, y es así debido a las capas de *maxpooling,* las cuales reducen las dimensiones de las imágenes, y los diferentes filtros que se encuentran en cada feature map por cada capa de convolución. Es importante también mencionar, el procesamiento de las imágenes, como también las funciones de activación, los parámetros y el método de optimización que se utilizara.

# Apéndice

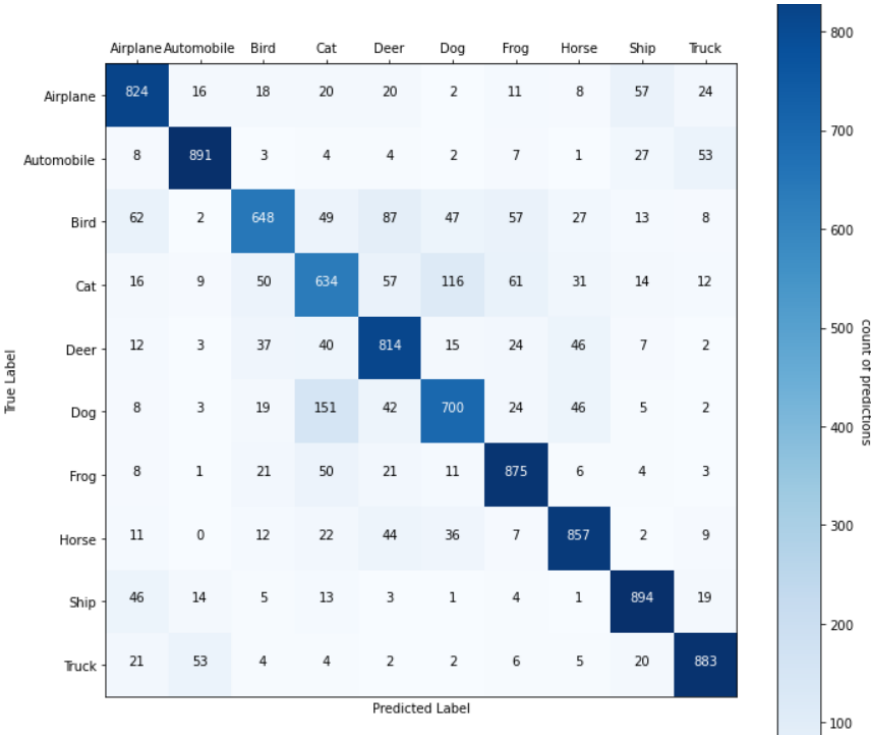
## Arquitectura del modelo:

```
Model: "Cool CNN No. 3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 32)        896
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 10, 10, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 5, 5, 64)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               409856
_________________________________________________________________
dropout_6 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 10)                2570
=================================================================
Total params: 477,994
Trainable params: 477,994
Non-trainable params: 0
```

## Model Accuracy y Model Loss:



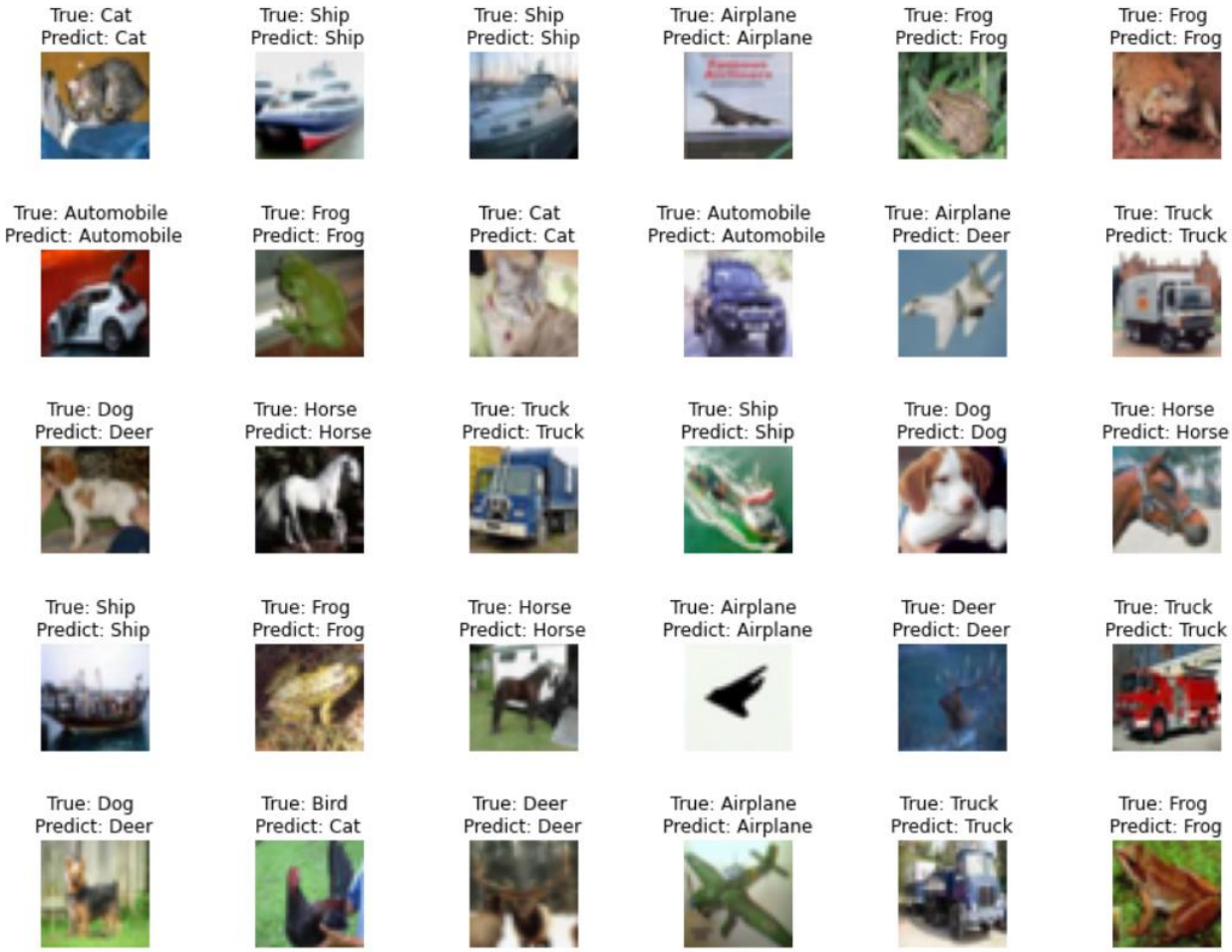

## Matriz de confusión:




**Traslape o confusión entre la clase de pájaros con gatos:**


**Traslape o confusión entre la clase de pájaros con gatos:**


**Visualización de predicciones (*true vs predict value):***

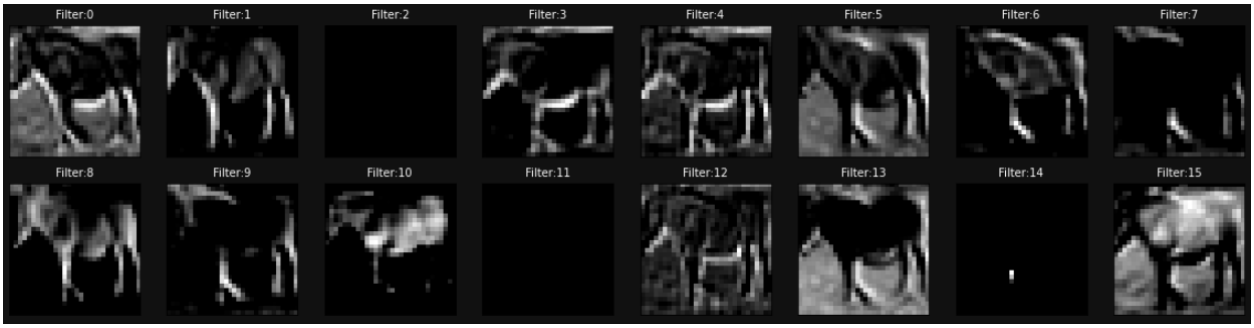



**índices y *shapes* de capas convolucionales:**

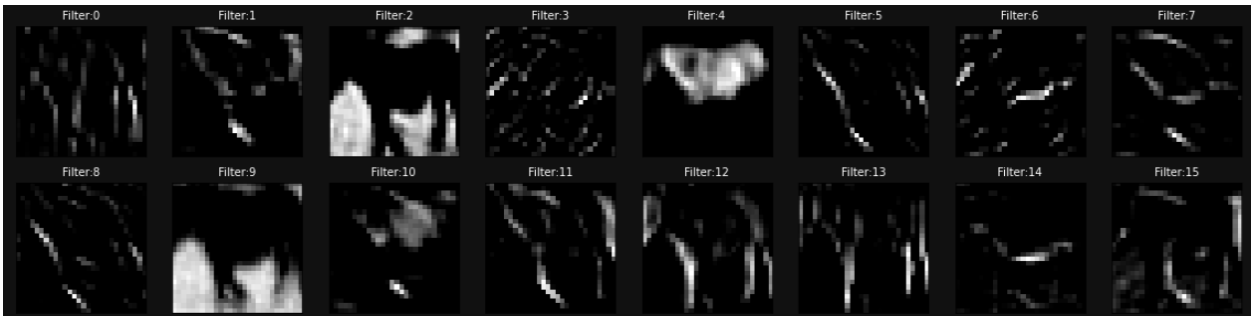
```
0 conv2d_1 (None, 30, 30, 32)
1 conv2d_2 (None, 28, 28, 32)
4 conv2d_3 (None, 12, 12, 64)
5 conv2d_4 (None, 10, 10, 64)
```

**Imagen input para feature maps:**

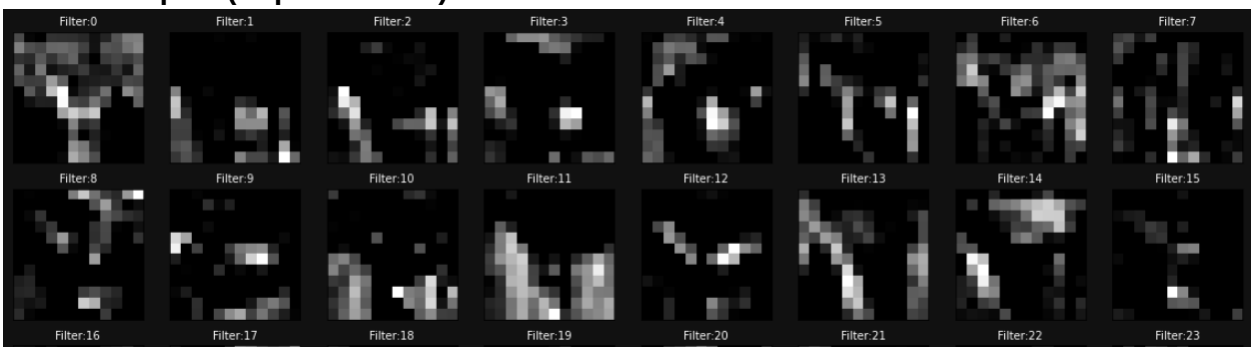
**Feature map #1 (Capa Conv. #1):**



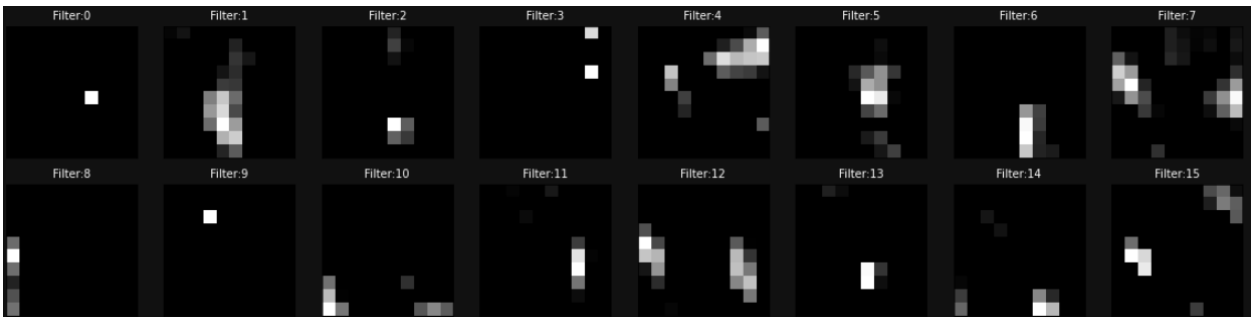

**Feature map #2 (Capa Conv. #2):**




**Feature map #3 (Capa Conv. #3):**




**Feature map #4 (Capa Conv. #4):**




(Brownlee, Machine Leaning Mastery, 2019)

(Pokharna, 2016)
(Kevin, 2018)
(Keras, s.f.)

# Referencias


Brownlee, J. (2019, May 06). *Machine Leaning Mastery*. From https://machinelearningmastery.com/how-to-visualize-filters-and-feature-maps-in-convolutional-neural-networks/

Keras. (n.d.). *Keras Doc.* From https://keras.io/layers/convolutional/

Kevin, C. (2018, May 11). *Feature maps*. From Medium: https://medium.com/@chriskevin_80184/feature-maps-ee8e11a71f9e

Pokharna, H. (2016, July 28). *The best explanation of Convolutional Neural Networks on the Internet!* From Medium : https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8