

Trabajo Práctico 5

Camussoni Francesco^{1,*}

¹Instituto Balseiro, Universidad Nacional de Cuyo

*camussonif@gmail.com

ABSTRACT

Se implementaron redes neuronales para aplicar el concepto de transferencia de aprendizaje sobre el set de datos de cat vs dogs con el aprendizaje sobre Imagenet como base y sobre MNIST con EMNIST como base. Finalmente, se utilizó la técnica de gradiente ascendente para maximizar la activación en ciertos filtros o clases de salida sobre una entrada ruidosa para observar lo que la red neuronal había aprendido.

Ejercicio 3

Se realizó una transferencia de aprendizaje de los datos de Imagenet (https://keras.io/guides/transfer_learning/) para clasificar gatos y perros del dataset cats vs dogs. Esta técnica se suele utilizar cuando se tiene un set de datos pequeños en donde entrenar una red neuronal de cero conlleva una baja precisión. De esta forma solo se utilizaron 2500 datos de training, validation y test.

Como base se utilizó la arquitectura de MobileNet provista por keras en donde es posible cargar los pesos aprendidos por Imagenet y quedarse solo con la arquitectura convolucional. De esta forma, se tiene las características mas generales aprendidas por dicha red. Luego se agregan capas densas, que serán las únicas que modificarán sus pesos durante el backpropagation para aprender a clasificar el dataset de cats vs dogs. Así se obtuvo la siguiente arquitectura:

Model: "Ejercicio3_catsvsdogs_learn_transfer"

Layer (type)	Output Shape	Param #
=====		
input (InputLayer)	[(None, 128, 128, 3)]	0
mobilenet_1.00_128 (Model)	(None, 4, 4, 1024)	3228864
global_average_pooling2d	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 1)	1025
=====		
Total params: 3,229,889		
Trainable params: 1,025		
Non-trainable params: 3,228,864		

En donde se observa que la mayoría de los parámetros no son entrenables. Se entrenó durante 50 épocas con un learning rate determinado y luego se entrenó por 10 épocas más modificando todos los parámetros de la red con un learning rate más chico. Este proceso se denomina fine tuning. Así se obtuvieron los resultados que se muestran en la [Figura 2](#).

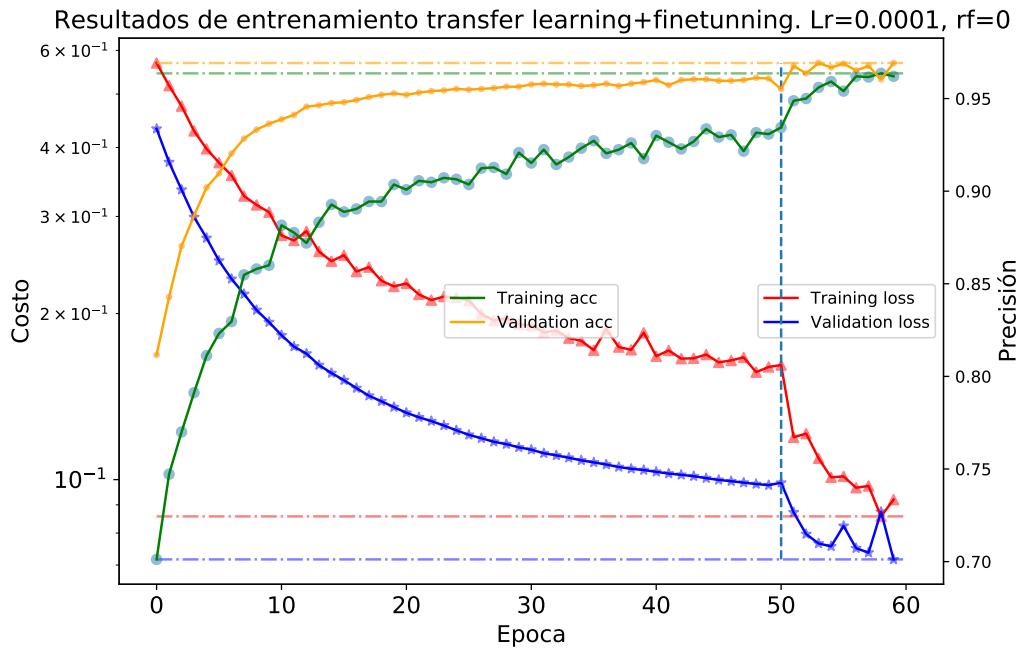


Figure 1. Resultados de entrenamiento para los datos de training y validation luego de la transferencia de aprendizaje.

En donde se obtuvo una precisión para los datos de test de 96%.

Luego, se entrenó desde cero la misma red neuronal a modo de comparación. Así se obtuvieron los resultados que se muestran en la [Figura 2](#)

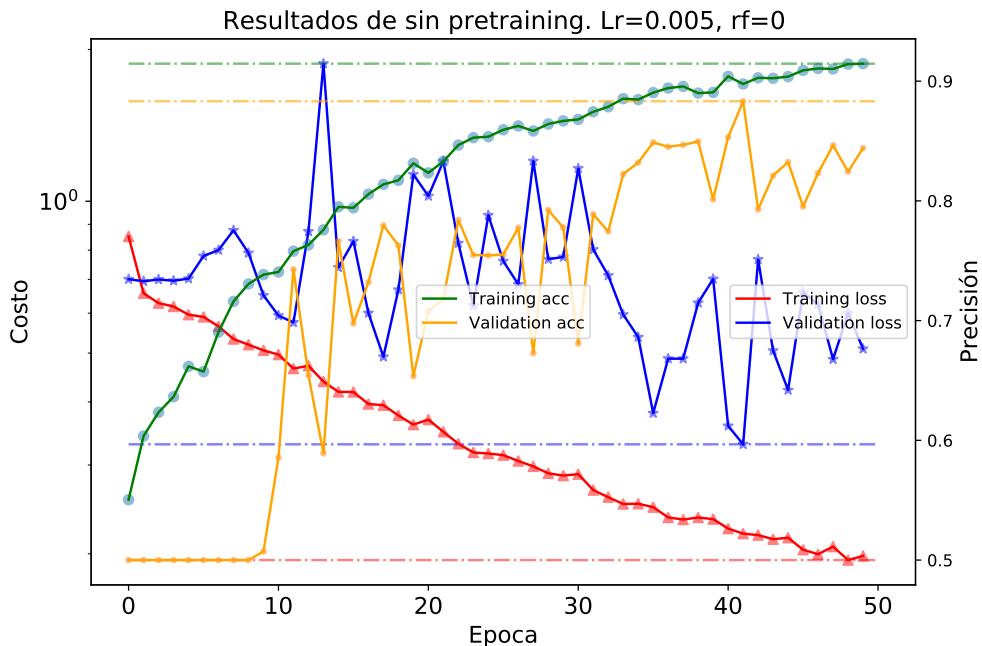


Figure 2. Resultados de entrenamiento para los datos de training y validation entrenando desde cero.

En donde se observa un comportamiento mucho mas errático para la generalización debido a los pocos datos de entre-

namiento y precisiones mucho menores, verificándose en este caso la ventaja de utilizar transferencia de aprendizaje.

Luego se realizó el mismo proceso para entrenar los datos de EMNIST que contiene caracteres alfabéticos con la siguiente red neuronal:

Model: "Ejercicio3_mnist_learn_transfer"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 28, 28, 10)	5770
=====		
max_pooling2d (MaxPooling2D)	(None, 14, 14, 10)	0
=====		
conv2d_1 (Conv2D)	(None, 14, 14, 10)	14410
=====		
flatten (Flatten)	(None, 1960)	0
=====		
dense (Dense)	(None, 10)	19610
=====		
Total params: 39,790		
Trainable params: 19,610		
Non-trainable params: 20,180		

Así, se obtuvieron los resultados de entrenamiento que se muestran en la **Figura 3**.

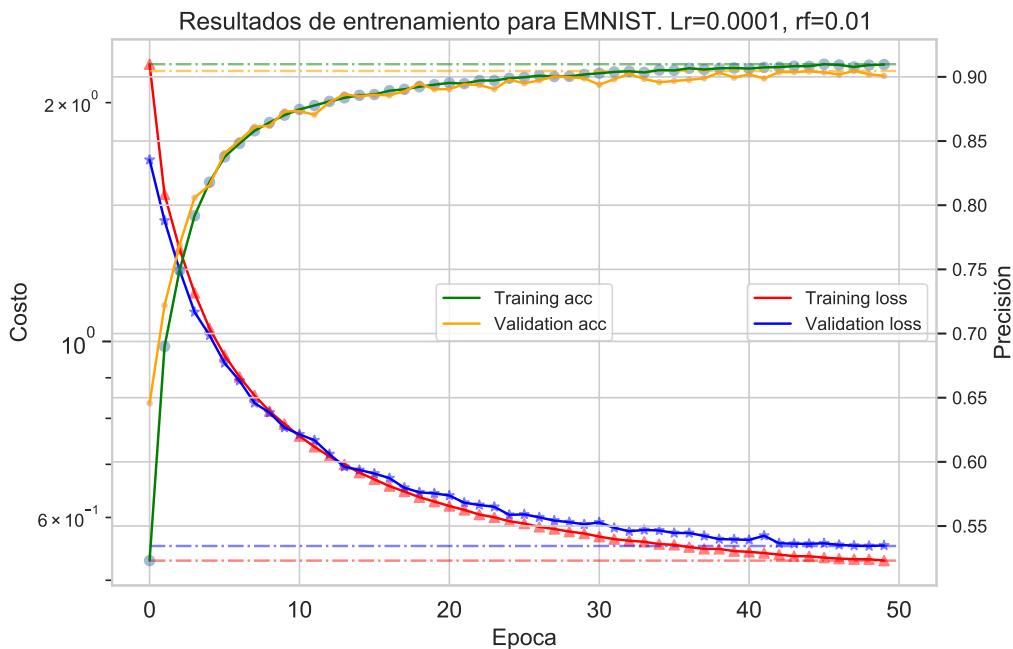


Figura 3. Resultados de entrenamiento para los datos de training y validation sobre los datos de EMNIST.

En donde se obtuvo una precisión de 91% para los datos de validación. Luego se realizó una transferencia de aprendizaje de la misma forma que se describió anteriormente obteniendo los resultados que se muestran en la **Figura ??**.

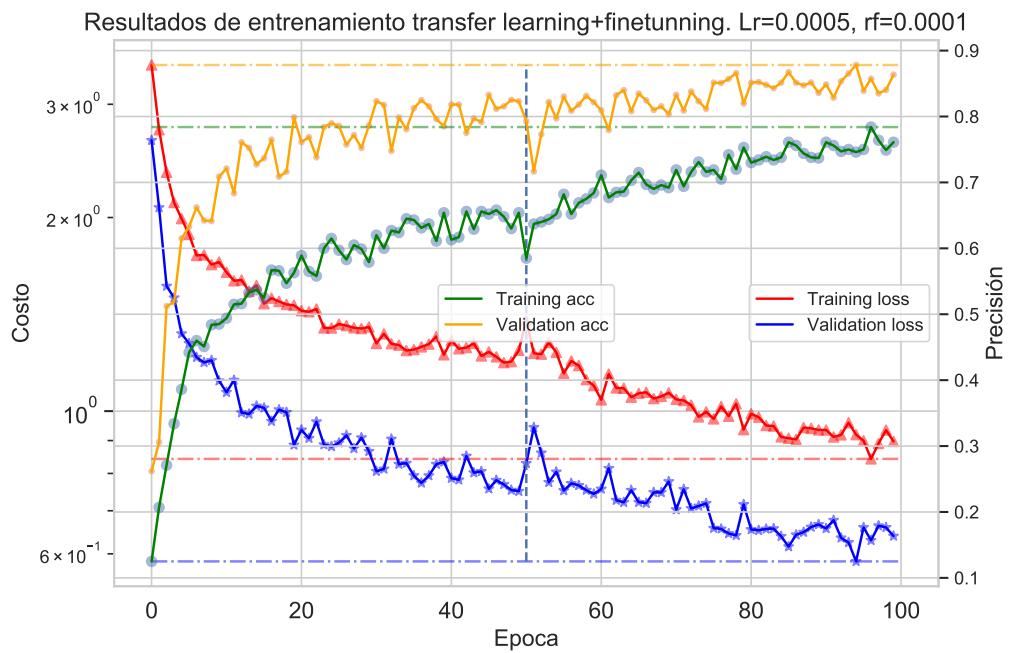


Figure 4. Resultados de entrenamiento para los datos de training y validation sobre los datos de MNIST utilizando trasnferencia de aprendizaje.

En donde se obtuvo una precisión de los datos de test del 87%, siendo una precisión bastante elevada para utilizar solamente 1000 datos. Sin embargo, al volver a entrenar la red neuronal desde cero con la misma cantidad de datos se obtuvo un rendimiento similar como se muestra en la **Figura 5**.

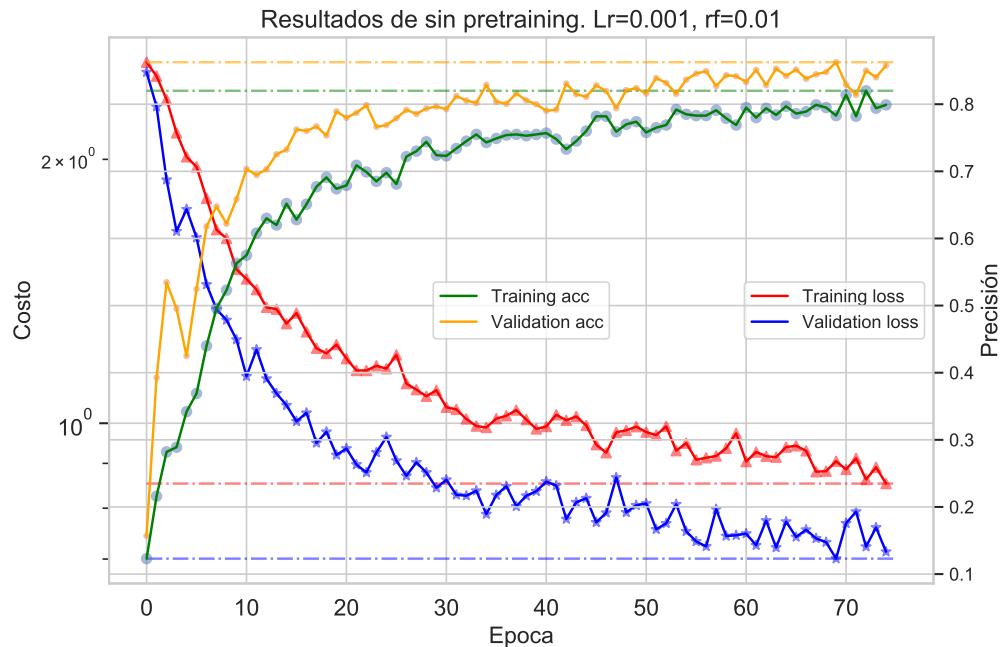


Figure 5. Resultados de entrenamiento para los datos de training y validation sobre los datos de MNIST.

Esto puede deberse a que la complejidad de los datos de MNIST es mucho menor a la de cat vs dogs, en donde todos los números están bastante centrados y solo se tiene un canal de colores (escala de grises). Además la cantidad de datos a aprender es del mismo orden de magnitud que los ya aprendidos. Es probable que al disminuir la cantidad de datos de training, 100 por ejemplo, la ventaja de la transferencia de aprendizaje se haga notoria.

En ambos casos, catsvsdogs y MNIST se utilizó aumentación de datos.

Ejercicio 4

Mediante el proceso de gradiente ascendente (<https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>) se obtuvo la entrada que maximizaba la activación de cierto filtro de la red o de cierta clase de salida actualizando los datos de entrada.

Primero se utilizó la arquitectura de ResNet50V2 provista por keras con los pesos entrenados sobre Imagenet. Se visualizaron 64 filtros de la salida de las capas convolucionales 3 y 5 obteniendo los resultados que se muestran en las **Figuras 6 y 7**.

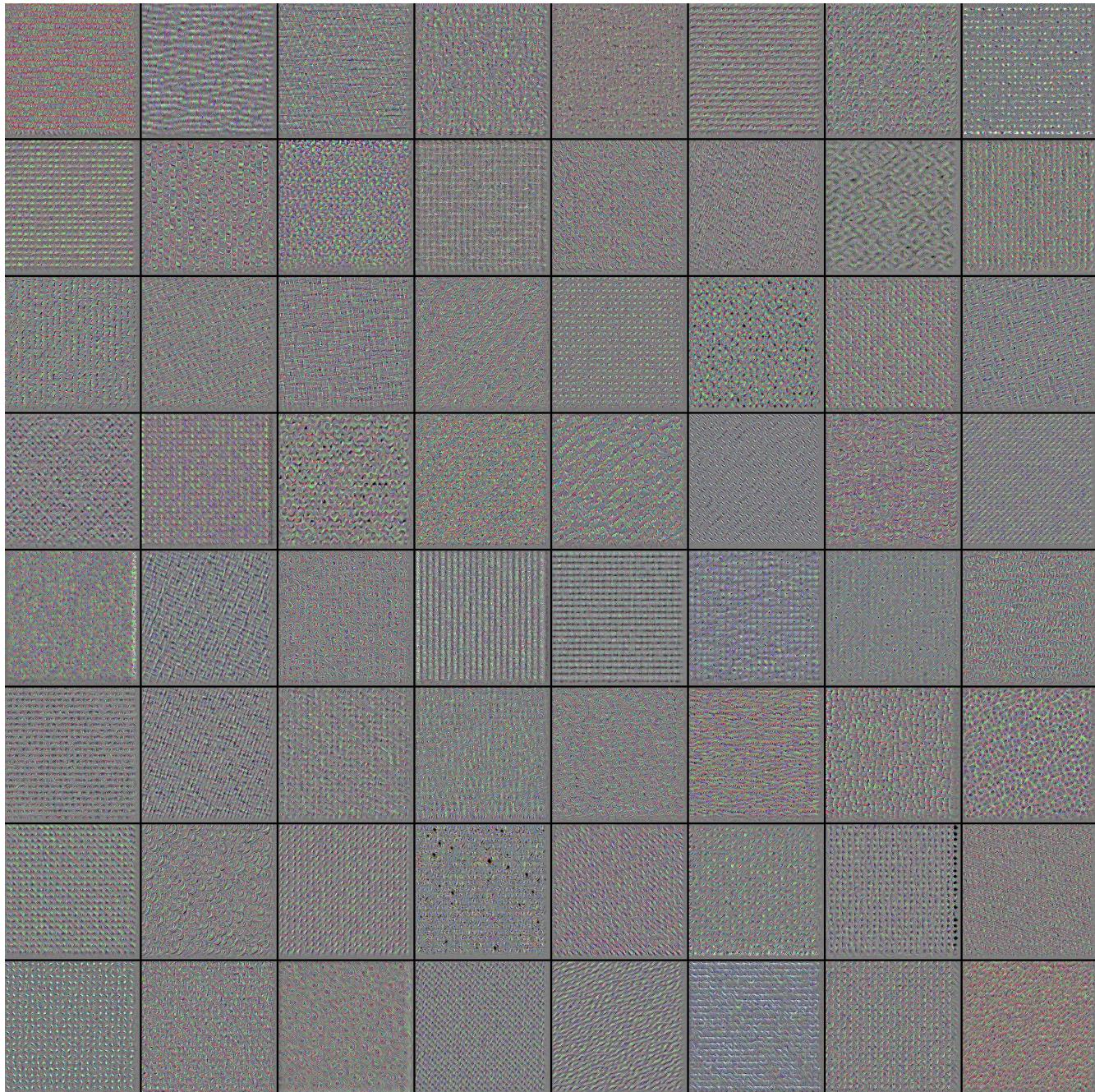


Figure 6. Visualizacion de 64 filtros para la layer conv3_block4_out.

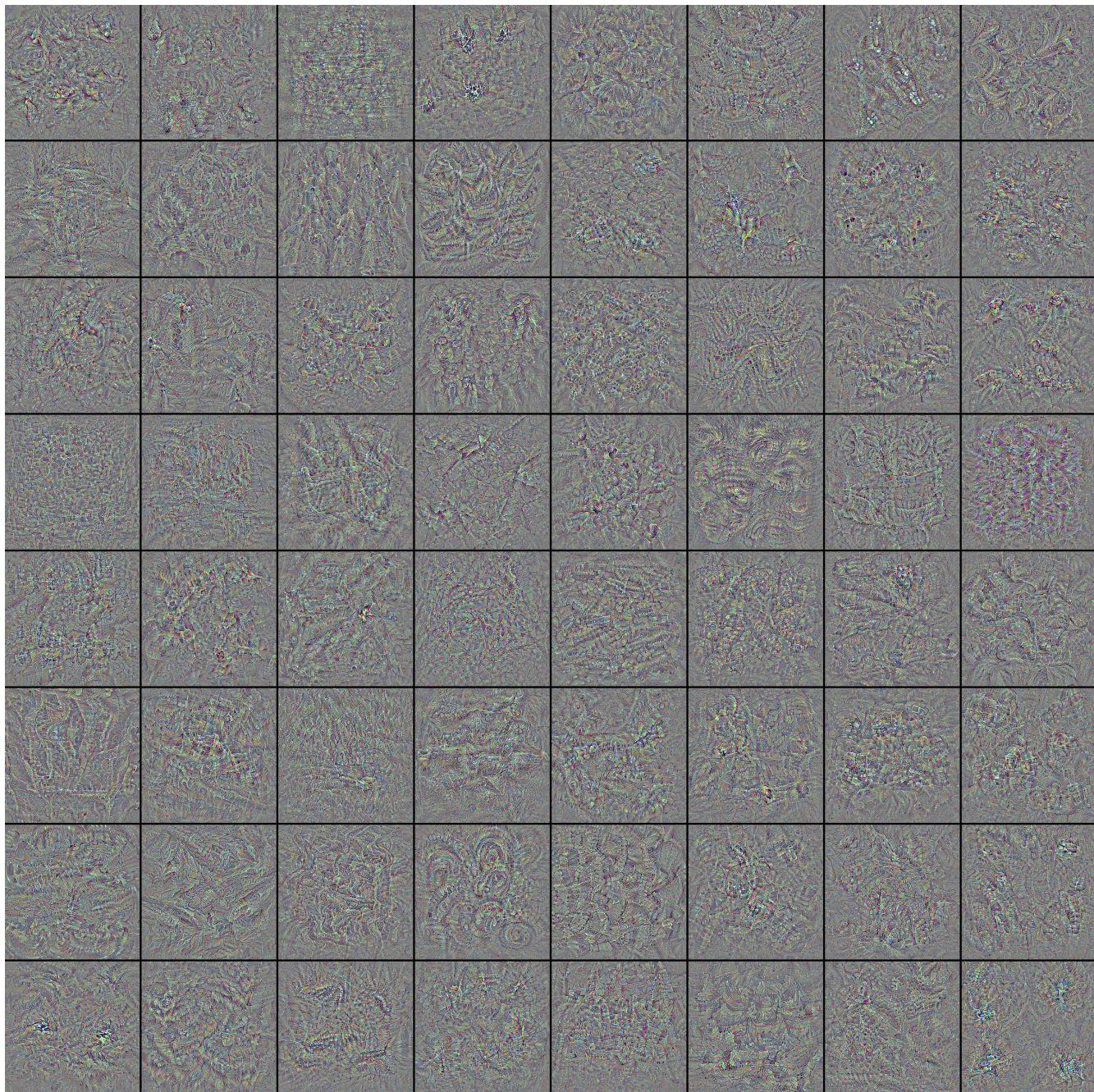
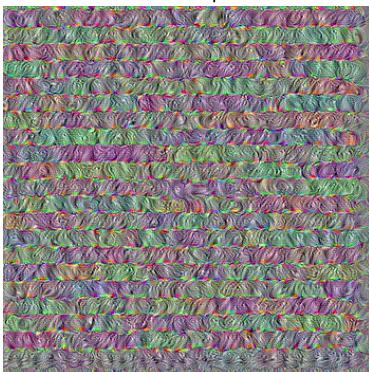


Figure 7. Visualizacion de 64 filtros para la layer conv5_block3_out.

En donde se observa que a medida que se avanza en la red neuronal convolucional los filtros distinguen cosas mas complejas. Luego se maximizó y minimizó la activación de un filtro particular de la layer conv5_block3_out para comparar resultados que se muestran en las **Figuras 8 y 9**.

Filtro con activación máxima para ruido como entrada



Filtro con activación mínima para ruido como entrada

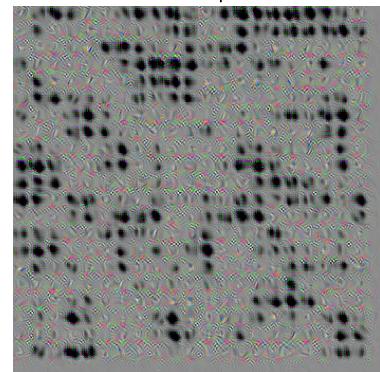


Figure 8. Activación máxima del filtro 0 de la layer conv3_block4_out.

Figure 9. Activación mínima del filtro 0 de la layer conv3_block4_out.

A continuación se maximizo la salida de la clase 238, correspondiente a gato siamés para ver que interpretaba la red neuronal como dicho animal. Se obtuvo el resultado que se muestra en la **Figura ??**.

Clase 283 (gato siames) con activación máxima para ruido como entrada

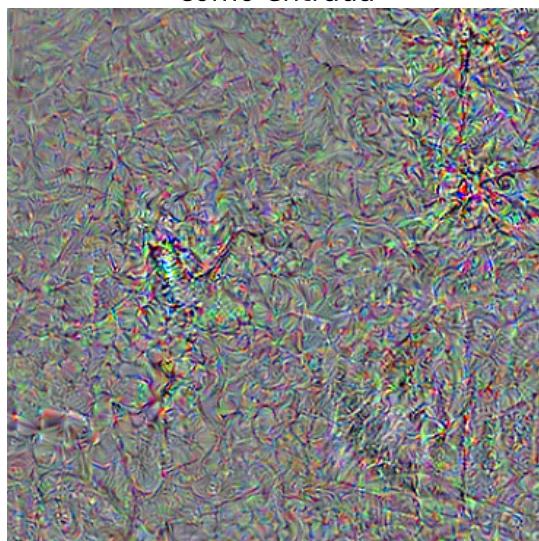


Figure 10. Visualizacion de la entrada que maximiza la salida de gato siamés.

En donde no se encuentra ninguna similitud con algún gato o felino.

Para obtener un ejemplo concreto más visible de que interpreta una red neuronal como clase, se utilizó la arquitectura utilizada en el ejercicio 3 para aprender MNIST con todos sus datos y maximizar cada una de las salidas. Así se obtuvieron los resultados que se muestra en la **Figura 11**.

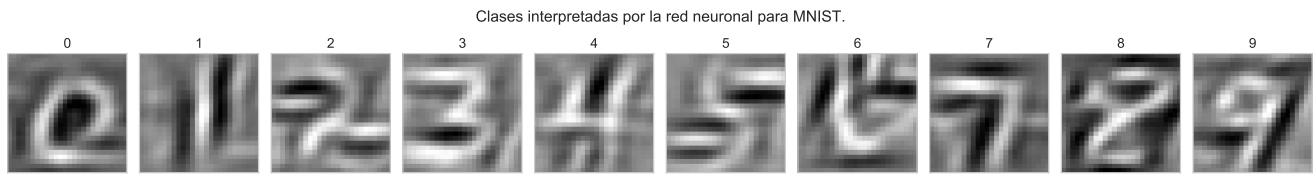


Figure 11. Visualizacion de las entrada que maximiza cada número de salida.

En donde se observa una mejor representación de cada clase.

Por completitud se presenta la activación máxima de cada filtro (10 para cada capa convolucional) de las dos capas convolucionales de la red propuesta en las **Figuras 12 y 13**.

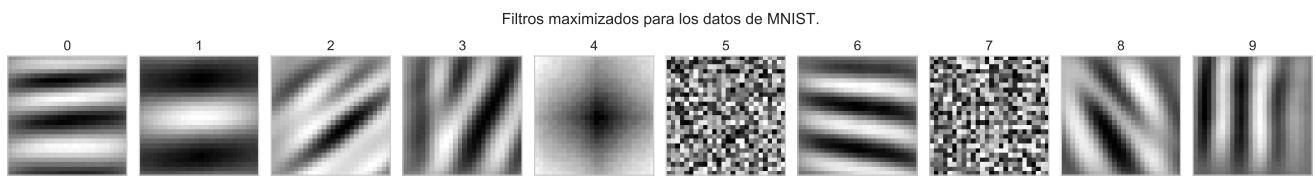


Figure 12. Visualizacion de las entrada que maximiza cada uno de los filtros de la primer capa convolucional.



Figure 13. Visualizacion de las entrada que maximiza cada uno de los filtros de la segunda capa convolucional.

En donde parece haber redundancia de filtros.

Finalmente se utilizó la arquitectura del ejercicio 3 para clasificar gatos y perros y se observó que interpretaba dicha red como estos animales. Los resultados se muestran en la **Figura ??**

Clases interpretadas por la red neuronal para cat vs dogs.

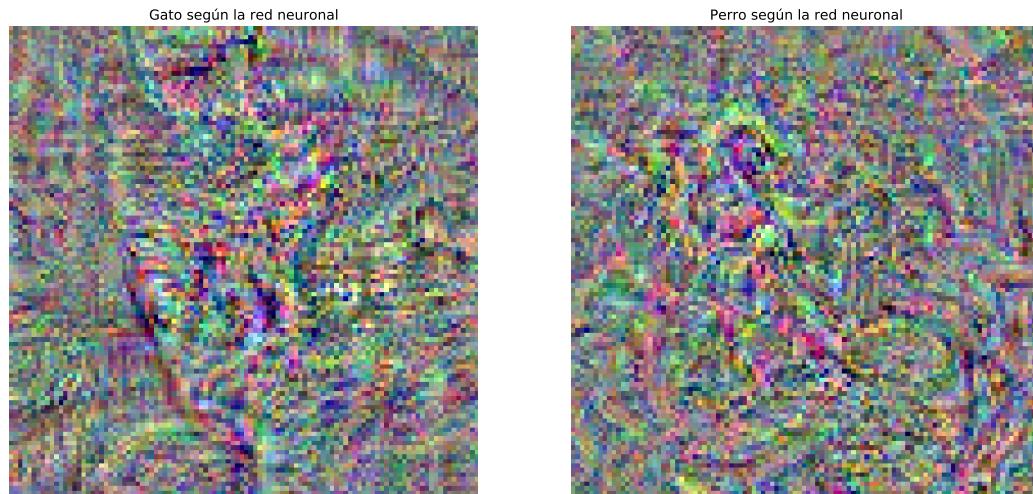


Figure 14. Visualizacion de las entradas que maximiza y minimiza la salida de la red de catvsdog.