Universidad de sonora

Seminario de física computacional

Segunda evaluación:

Redes neuronales convolucionales

Minjares Neriz Victor Manuel

Abril 2021

Índice

1. Introducción

En este proyecto desarrollaré una red neuronal convolucional basado en el programa visto en clase de calificación de imágenes de flores. En este caso clasificaremos diversos tipos de animales marinos. Las imágenes las obtuve de un repositorio de kaggle:

https://www.kaggle.com/crowww/a-large-scale-fish-dataset.

Estas imágenes fueron recolectadas por O. Ulucan, D. Karakayay, M. Turkan del departamento de ingeniería electríca y electrónica de la universidad de Izmir, Turquía. En su paper: A Large-Scale Dataset for Fish Segmentation and Classification. El repositorio tiene 9 carpetas de los siguientes animales

- Glit head bream (Dorada)
- Red sea bream (Besugo)
- Sea bass (Lubina)
- Red mullet (Salmonete)
- Horse mackerel (Jurel)
- Black sea spart (Espadín de mar negro)
- Striped red mullet (Salmonete rayado)
- Trout (Trucha)
- Shrimp (Camarón)

imágenes tomadas por dos diferentes cámaras, Kodak Easyshare Z650 y Samsung ST60. Resolución de 2832x2128 y 1024x768 respectivamente. Los datos pasaron por un rescalamiento, reduciéndolas a 590x445. Además pasaron por un proceso de aumentación. Total de imágenes 18000, cada categoría tiene 2000 imágenes, la mitad esta en RGB y la otra mitad en pair-wise ground truth labels.

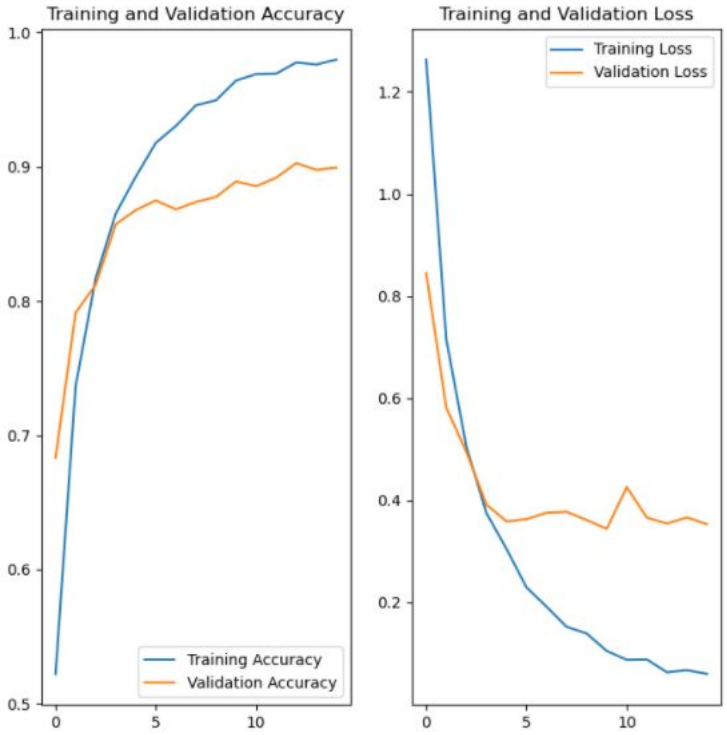
(a) Salmonete (b) Lubina en pair-wise ground (c) Camarón

Figura 1: Muestras de las imágenes a usar

2. Resultados

2.1. Configuración incial

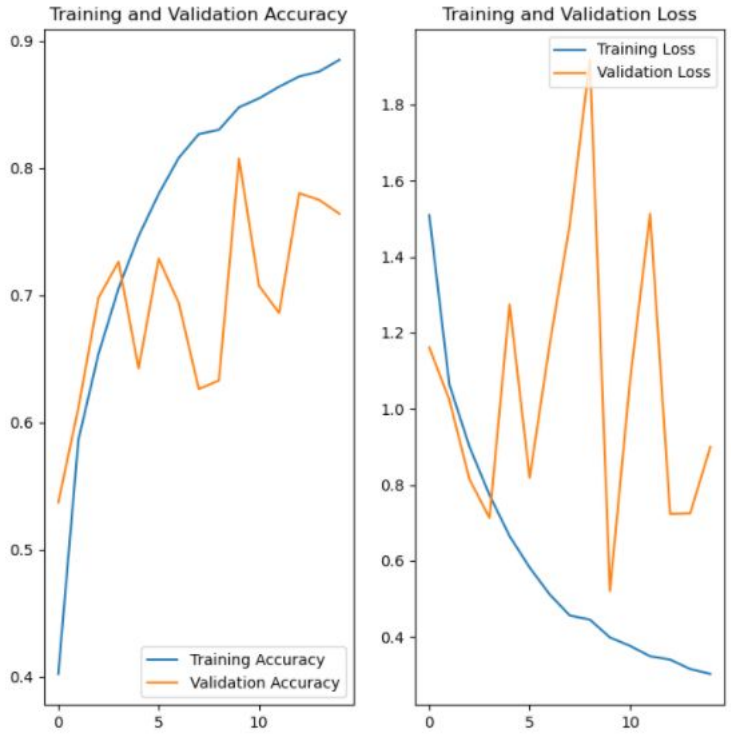
- Augmentation: no
- Dropout: no
- Optimizer: adam
- Activation function: relu
- Batch size: 40
- Dim. conv. layers: 3
- Dim. maxpool. layers: 3
- Num. layers: 3



Training		Validation	
accuracy:	98.35 %	accuracy:	89.94 %
loss:	0.0493	loss:	0.3535

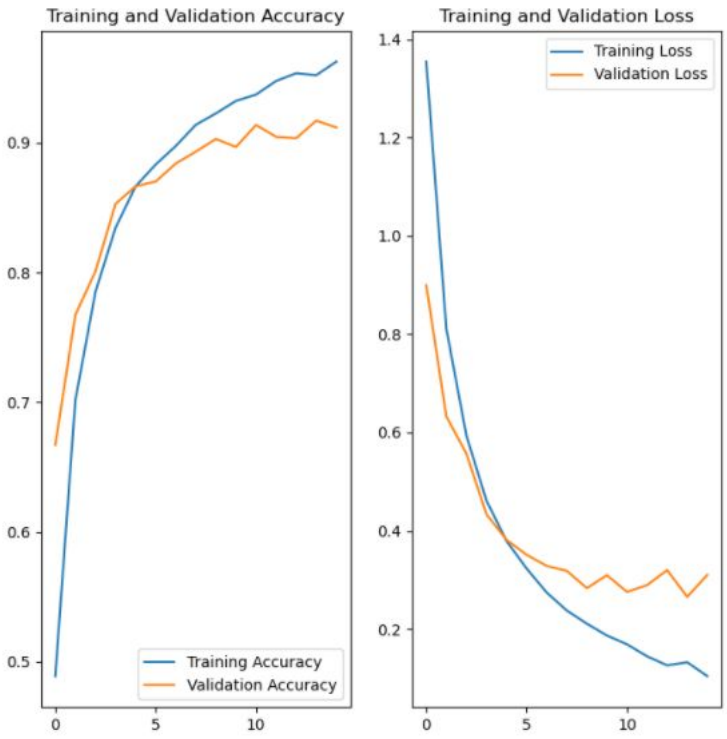
2.1.1. Augmentation y dropout

Augmentation sí, dropout no.



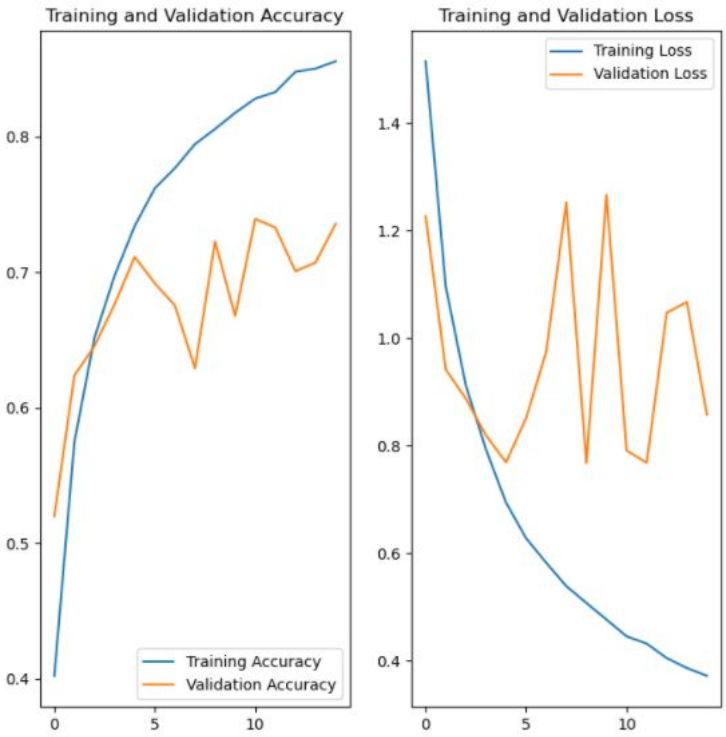
Training		Validation	
accuracy:	88.72 %	accuracy:	76.42 %
loss:	0.3062	loss:	0.8999

Augmentation no, dropout sí. El dropout es de 0.2 y se aplica solamente a la 'ultima capa de la red.



Training		Validation	
accuracy:	96.3 %	accuracy:	91.17 %
loss:	0.1016	loss:	0.3102

Augmentation sí, dropout sí. Dropout igual que el anterior.



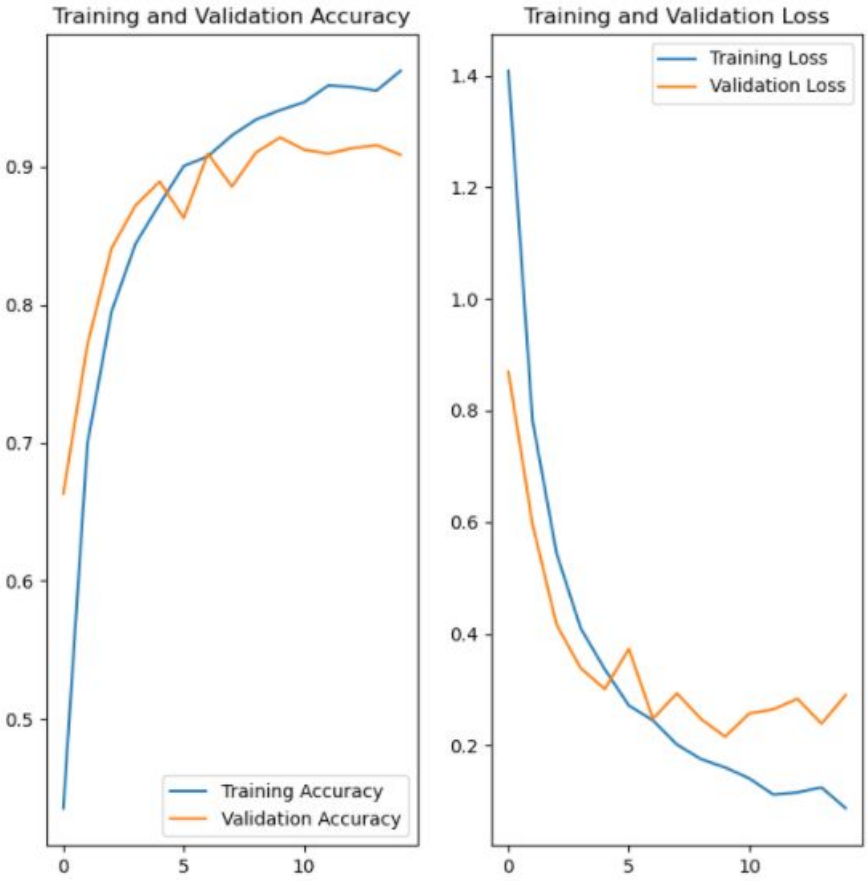
Training		Validation	
accuracy:	85.72 %	accuracy:	73.58 %
loss:	0.3653	loss:	0.8575

Como podemos ver nuestros resultados en la configuración incial son buenos, esto seguramente se debe a que los datos fueron con antelación procesados. Como vemos cuando aplicamos augmentation los resultados empeoran conciderablemente, me imagino que es porque los datos en su procesado ya había pasado por augmentation. Con respecto al dropout bajo la diferencia entre los resultados de los datos de entrenamiento y validación pero a su vez, bajo un poco la precisión. Sin embargo los resultados con dropout son los mejores de las 4 configuración, entonces usaremos esto cuando apliquemos las variaciones.

Las variaciones se haran manteniendo la configuracón incial + dropout + la variación correspondiente solamente.

2.2. Variación 1

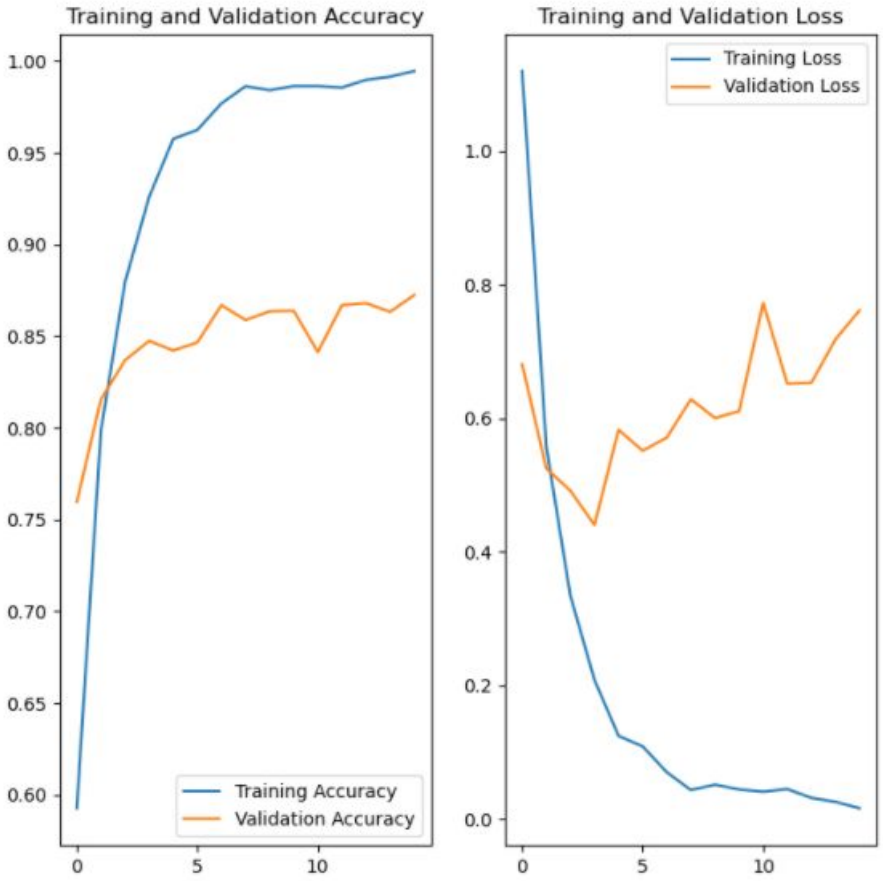
Variaremos el número de capas a 5 capas de pares convolución, maxpooling.



Training		Validation	
accuracy:	93.56 %	accuracy:	90.18 %
loss:	0.1065	loss:	0.3568

2.3. Variación 2

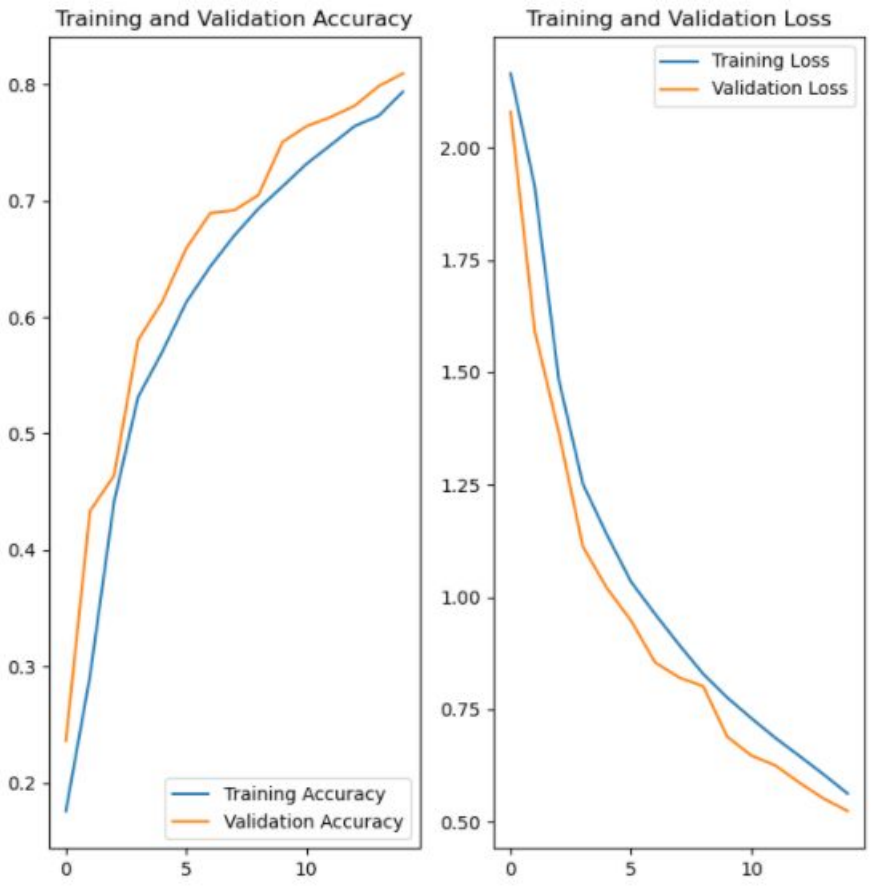
Usaremos capas de convolución con dimensión 4x4 y 2x2 para maxpolling.



Training		Validation	
accuracy:	98.652 %	accuracy:	87.88 %
loss:	0.0253	loss:	0.7675

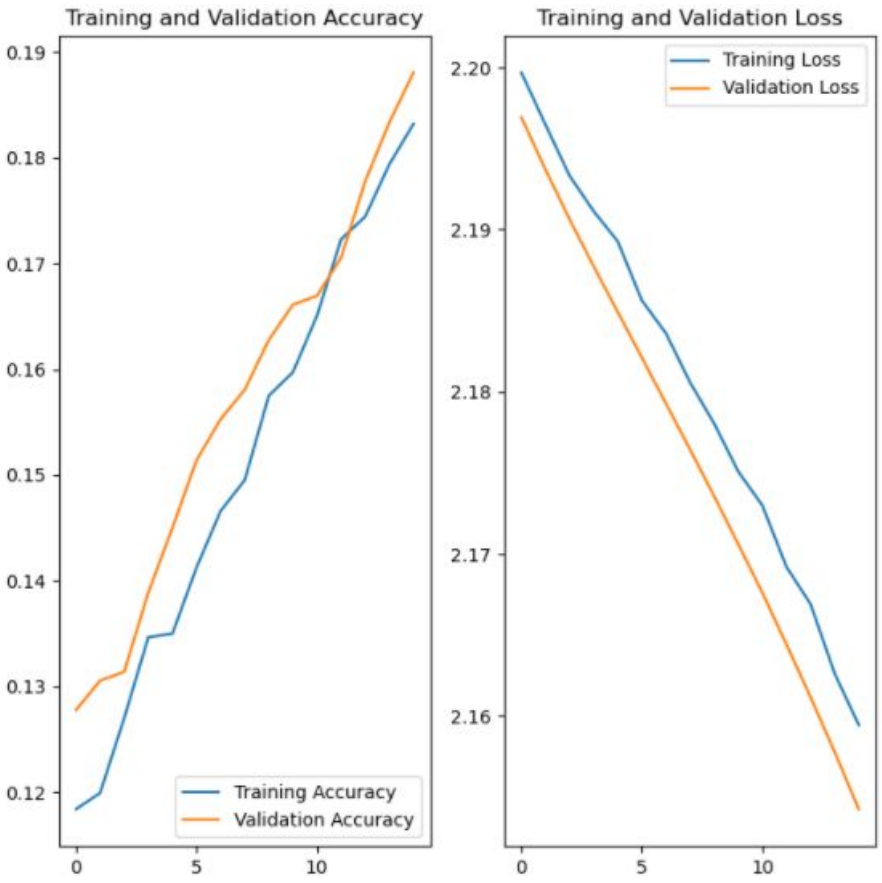
2.4. Variación 3

2.4.1. Optimizer SGD



Training		Validation	
accuracy:	79.72 %	accuracy:	81.48 %
loss:	0.626	loss:	0.575

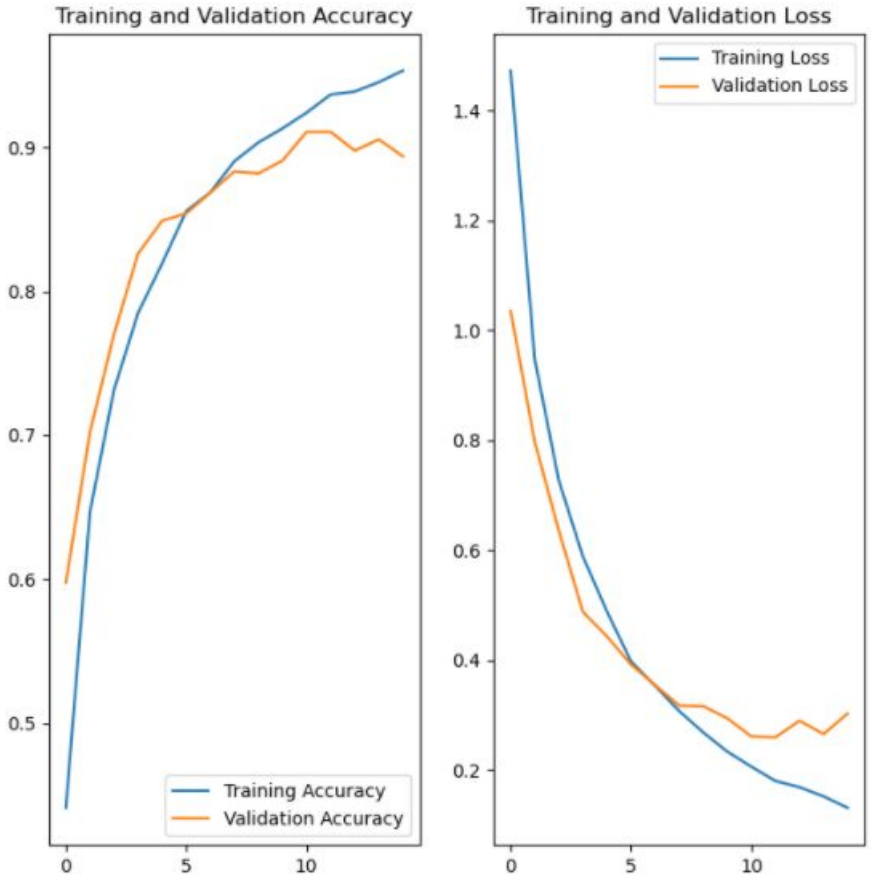
2.4.2. Optimizer Adadelta



Training		Validation	
accuracy:	18.0235 %	accuracy:	18.756 %
loss:	2.165	loss:	2.1553

2.5. Variación 4

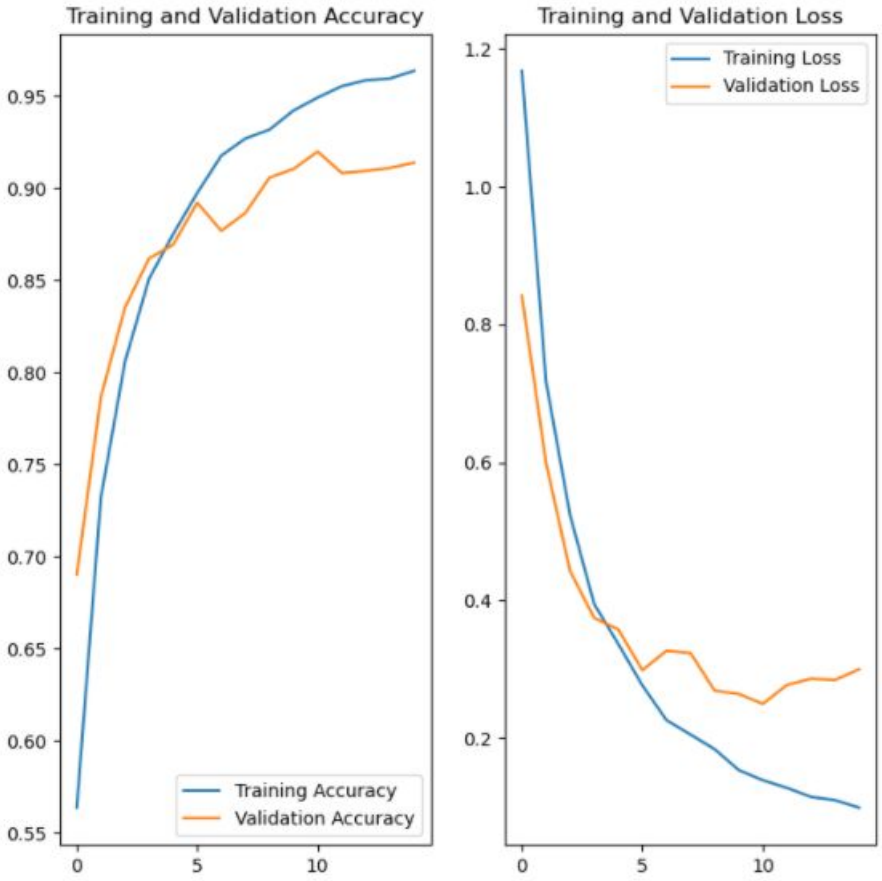
Aumentaremos el tamaño del batch a 110.



Training		Validation	
accuracy:	95.65 %	accuracy:	89.863 %
loss:	0.1654	loss:	0.3678

2.6. Variación 5

En esta variación cambiaremos la función de activación, usaremos elu.



Training		Validation	
accuracy:	96.657 %	accuracy:	91.256 %
loss:	0.1556	loss:	0.3763

3. Conclusiones

Comparando las gráficas y resultados finales del accuracy y el loss vemos que la mayoría de las variaciones, configuraciones tienen un rendimiento malo, destacando el optimizer Adadelta, el cual no llega ni al 20 % de acurracy y su entrenamiento sube de forma lineal a diferencia de los demás que aumenta parecido a los logaritmos.
Los que presentaron mejores resultados, ya que en general tuvieron un accuracy alto, loss bajo y además una diferencia entre los resultados de validación y entrenamiento pequeña fueron la configuración incial, con solo dropout, y con dropout y variación 5. De las cuales las dos últimas mencionadas tienen resultado casi idénticos y todas configuraciones son las mejores que se obtuvieron.
Los resultados fueron muy buenos con los datos elegidos, ya que fueron procesados por los publicadores de los mismos. También porque el repositorio contaba con imágenes tanto en color, fotos normales, como solo las formas de los animales marinos, sin color, solo fondo negro y la forma en blanco. Esto ayudo bastante al momento de entrenar la red neuronal.
No encontré como hacer que mi código leyera URL de imágenes de google para hacer predicciones, por lo que mi solución fue; antes de entrenar la CNN quitarle unas pocas imágenes al repositorio de kaggle y ponerlas en una carpeta de test, después de entrenar la red la testeé con estas imágenes, con las cuales obtuve un 100 % de presición usando las configuraciones que me dieron un accuracy alto, esto se ha de deber en gran parte que son imágenes que tienen el mismo formato que las imágenes con las que se entreno la CNN.

Ruta del programa:
/LUSTRE/ccd/home/Victor_Minjares/evaluacion2/ev2_CNN_VictorMinjares.py