Pneumony ChestXRay classification

Para realizar el clasificador de imágenes hemos decidido utilizar un dataset proveniente de Kaggle que contiene imágenes de casos de pneumonia y casos sin pneumonia(normales).

Como primera prueba decidimos probar el modelo para que clasificara entre casos normales y casos con pneumonia.

El modelo consiguió una accuracy muy alta, de 0.90 en adelante. Dados estos buenos resultados decidimos complicárselo un poco mas, haciendo que clasificara en 3 clases. Imágenes normales, imágenes con virus e imágenes con bacteria. Es decir, el conjunto de imágenes calificadas como pneumonia a su vez se podía clasificar en pneumonia producida por virus y pneumonia producida por bacterias. Para llevarlo a cabo modificamos las carpetas de nuestro set de datos y trabajamos con 3 carpetas, una para las imágenes normales, otra para las imágenes con virus y otra para las imágenes con bacterias.

La accuracy de nuestro modelo empeoro, lo que nos hizo pensar que pudo deberse a los datos de entrada que le indicamos al modelo. Las imagenes de pneumania por virus y pneumonia por bacterias son muy complicadas de distinguir para un humano medio (sin conocimientos del tema), y nosotros le estabamos reduciendo mucho la calidad de la imagen, la entrada del modelo era de 200×200, por lo que debía resultarle muy complicado distinguir entre virus y bacterias ya que para ello necesitaria que la calidad de las imagenes de entrada fuese mayor a la que era. La extracción de caracteristicas por grande que fuese no funcionaria si las resolucion de la imagen no es suficientemente buena. Pero debido a que trabajamos con nuestros portatiles tampoco podiamos aumentarle mucho mas dichas cifras.

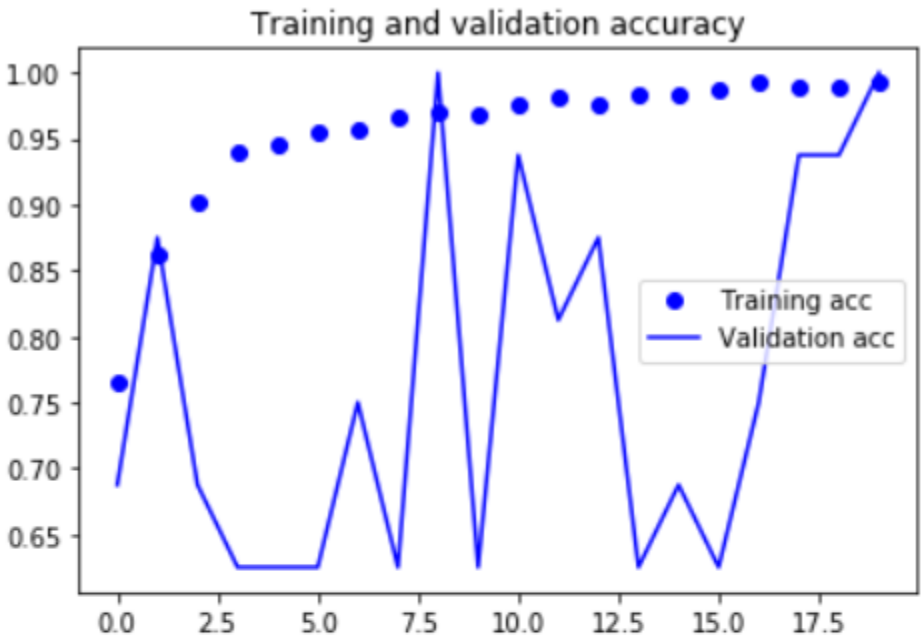
```
input_shape=(200, 200, 3)))
```

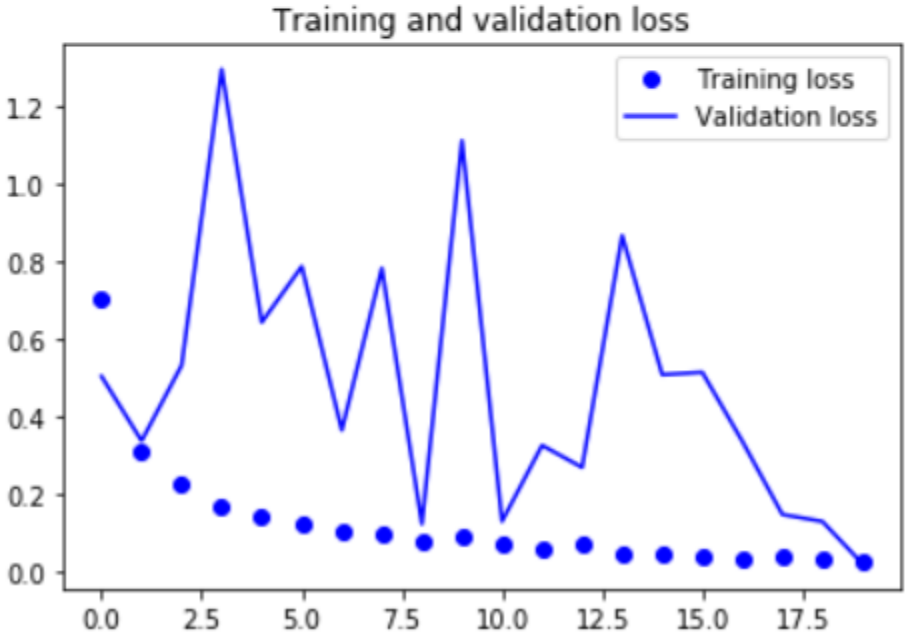
Además otro problema que consideramos que pudo encontrarse el modelo fue que las clases no estaban balanceadas, la cantidad de imágenes de pneumonia por bacterias era mayor a la cantidad de imágenes de pneumonia por virus, lo que unido a que la calidad no era muy alta hacía mas difícil al modelo aprender a diferenciar mejor entre ambos y por ello conseguir una accuracy mas alta.

Como no teníamos capacidad para entrenar un modelo con mayor complejidad para que clasificara en mas de 2 clases decidimos volver a probar nuestro modelo para que clasificara entre normal y pneumonia para examinar mejor como había entrenado ya que la primera prueba que hicimos con estas mismas condiciones fue mas bien informativa para conocer la accuracy con la que partía el modelo al clasificar en 2 clases.

```
Epoch 1/20
100/100 [==============================] - 246s 2s/step - loss: 0.7028 - acc: 0.7645 - val_loss: 0.5027 - val_acc: 0.6875
Epoch 2/20
100/100 [==============================] - 220s 2s/step - loss: 0.3095 - acc: 0.8625 - val_loss: 0.3363 - val_acc: 0.8750
Epoch 3/20
100/100 [==============================] - 218s 2s/step - loss: 0.2268 - acc: 0.9018 - val_loss: 0.5312 - val_acc: 0.6875
Epoch 4/20
100/100 [==============================] - 215s 2s/step - loss: 0.1656 - acc: 0.9395 - val_loss: 1.2949 - val_acc: 0.6250
Epoch 5/20
100/100 [==============================] - 218s 2s/step - loss: 0.1437 - acc: 0.9450 - val_loss: 0.6421 - val_acc: 0.6250
Epoch 6/20
100/100 [==============================] - 220s 2s/step - loss: 0.1207 - acc: 0.9545 - val_loss: 0.7861 - val_acc: 0.6250
Epoch 7/20
100/100 [==============================] - 218s 2s/step - loss: 0.1039 - acc: 0.9564 - val_loss: 0.3640 - val_acc: 0.7500
Epoch 8/20
100/100 [==============================] - 221s 2s/step - loss: 0.0976 - acc: 0.9655 - val_loss: 0.7816 - val_acc: 0.6250
Epoch 9/20
100/100 [==============================] - 218s 2s/step - loss: 0.0736 - acc: 0.9709 - val_loss: 0.1218 - val_acc: 1.0000
Epoch 10/20
100/100 [==============================] - 214s 2s/step - loss: 0.0871 - acc: 0.9690 - val_loss: 1.1108 - val_acc: 0.6250
Epoch 11/20
100/100 [==============================] - 212s 2s/step - loss: 0.0679 - acc: 0.9760 - val_loss: 0.1289 - val_acc: 0.9375
Epoch 12/20
100/100 [==============================] - 211s 2s/step - loss: 0.0566 - acc: 0.9810 - val_loss: 0.3247 - val_acc: 0.8125
Epoch 13/20
100/100 [==============================] - 212s 2s/step - loss: 0.0676 - acc: 0.9750 - val_loss: 0.2679 - val_acc: 0.8750
Epoch 14/20
100/100 [==============================] - 211s 2s/step - loss: 0.0463 - acc: 0.9835 - val_loss: 0.8659 - val_acc: 0.6250
Epoch 15/20
100/100 [==============================] - 211s 2s/step - loss: 0.0424 - acc: 0.9840 - val_loss: 0.5069 - val_acc: 0.6875
Epoch 16/20
100/100 [==============================] - 218s 2s/step - loss: 0.0374 - acc: 0.9870 - val_loss: 0.5132 - val_acc: 0.6250
Epoch 17/20
100/100 [==============================] - 216s 2s/step - loss: 0.0290 - acc: 0.9920 - val_loss: 0.3354 - val_acc: 0.7500
Epoch 18/20
100/100 [==============================] - 215s 2s/step - loss: 0.0393 - acc: 0.9895 - val_loss: 0.1465 - val_acc: 0.9375
Epoch 19/20
```

Al examinar las gráficas nos damos cuenta de que cuando el modelo solo tiene que clasificar en 2 grupos parece que en las primeras iteraciones esta conociendo los datos y vemos que a medida que aumenta el numero de iteraciones la accuracy empieza a crecer de manera constante y no genera los picos como los que se estaban generando en las primeras iteraciones. A partir de la iteración 15 el modelo no vuelve a decaer en términos de accuracy, lo que nos hace pensar si probásemos mas de 20 iteraciones el modelo daría muy buenos resultados. esta mejora con el aumento de las iteraciones también se observa en la grafica de perdida, ya que a partir de la iteración 15 la perdida del modelo no vuelve a aumentar, solo se mantiene o decrece.

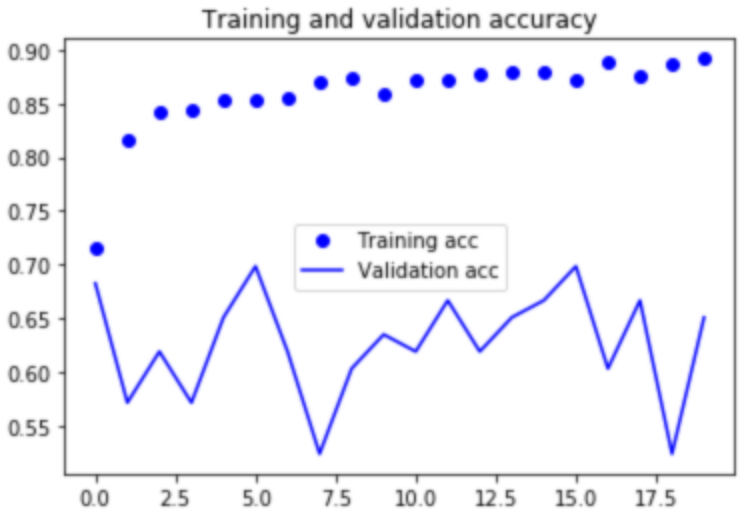




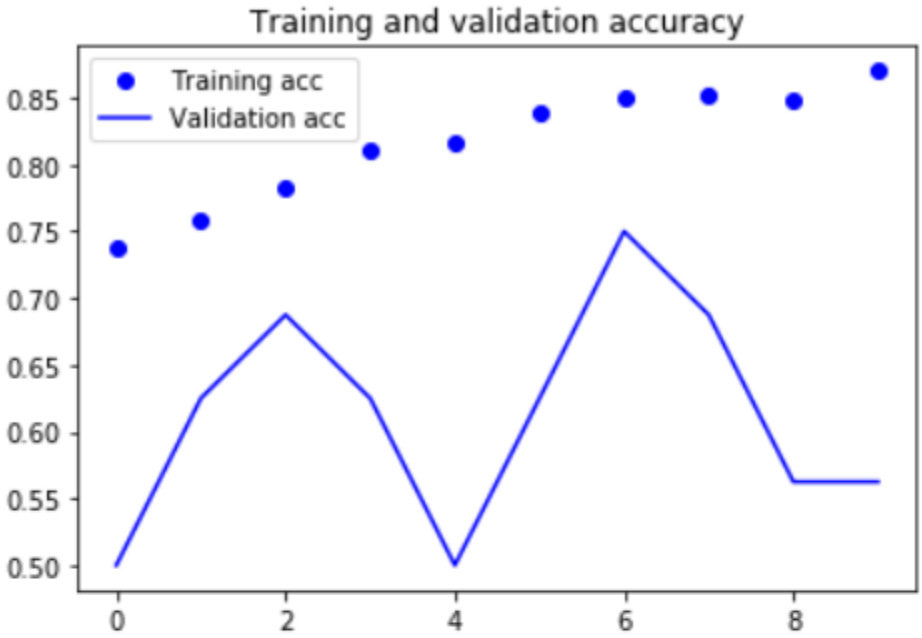
Después, decidimos examinar las gráficas generadas cuando el modelo tenía que clasificar en 3 grupos para compararlas con los resultados de la clasificación en 2 grupos.

Tal y como habíamos dicho, la accuracy es mucho menor y no parece mostrar una mejoraría con el número de iteraciones, pero si lo pensamos bien, nos damos

cuenta de que para que el modelo mejorase cuando tiene que clasificar 3 clases necesitaría que el numero de iteraciones fuera mayor. Para clasificar 2 clases vemos mejoría a partir de la iteración 15 y sobre todo con la iteración 20 por lo que, si queremos que con tres clases mejore tendríamos que trabajar con un numero de iteraciones como mínimo mayor de 30



Por último, decidimos pintar las gráficas de los resultados en validación con 10 y 15 iteraciones. Y lo que podemos observar es que la accuracy del modelo en training es mucho mas alta que en validación, lo que nos da a entender que el modelo podría estar sobreentrenado y por ello con las imágenes de validación saca peores resultados. Pero al igual que en las gráficas anteriores puede que con aumentar el numero de iteraciones empecemos a conseguir mejores resultados. Ya que si nos fijamos, con 8 iteraciones parece que el modelo esta un poco descontrolado, pero a partir de la iteración 8, aun que la accuracy no sea alta, el modelo parece que se estabiliza un poco y hay menos diferencia entre una iteración y la siguiente. Por esta razón, concluimos diciendo que para que nuestro modelo mejore notablemente deberíamos mandarle un numero más alto de iteraciones. Y cuanto mayor sea la cantidad de grupos a clasificar, mayor deberían ser las iteraciones.


Training and validation accuracy
Training acc
Validation acc
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
2
4
6
8


Training and validation loss
Training loss
Validation loss
1.0
0.8
0.6
0.4
0
2
4
6
8


Training and validation accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0
2
4
6
8
10
12
14
Training acc
Validation acc


Training and validation loss
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12
14
Training loss
Validation loss