EXPERIMENTO 9

ANÁLISIS DE LA ARQUITECTURA PROPUESTA PARA SEGMENTACIÓN Y CLASIFICACIÓN

Por Antonio Jesús García Nieto

CONTROL DE VERSIONES

Versión	Descripción	Fecha de última modificación	Validado
1.0	Primera versión estable	11/04/21	No

INTRODUCCIÓN

En este experimento proponemos una solución compacta al problema de detección y segmentación de lesiones de piel. Esta solución es un intento de subsanar las debilidades de la arquitectura propuesta anteriormente.

La hipótesis que pretendemos validar en este estudio es:

1. *¿Podemos obtener unos resultados óptimos en la clasificación de lesiones de piel (95 %) a la vez que el modelos nos aporta unos resultados razonables en segmentación semántica sin superar en parámetros a una red de convencional que realice tan solo la tarea de clasificación?*

DESCRIPCIÓN DEL MODELO PROPUESTO

Esta propuesta surge del estudio de la idea fundamental de arquitectura de una CNN.

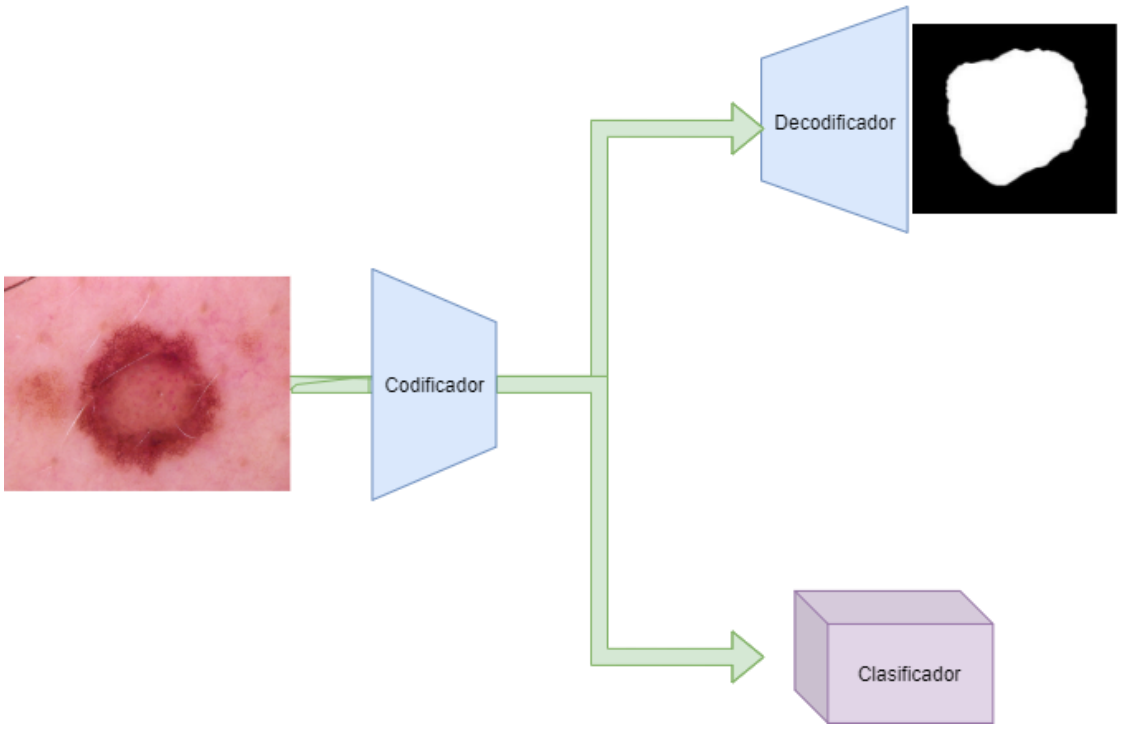
Reducido a lo mínimo, una CNN no es más que un reductor de dimensiones (codificador) y un clasificador.

Basándonos en esto y teniendo la mira puesta en reducir parámetros sin perder funcionalidad, surge la pregunta: ¿Podemos crear un modelo en el que un codificador sea compartido por dos modelos con distintos propósitos?

En este estudio pretendemos crear un modelo inspirado en esta idea, para ello se ha decidido utilizar RSM_BAJ por dos motivos:

- es compacta (8M de parámetros para imágenes de 128x128)
- es sencilla de adaptar para transformarla en una FCNN

El esquema de funcionamiento de la red es el siguiente:



La arquitectura resultante sería la siguiente:

Layer (type)	Output Shape	Param #	Connected to
input_6 (InputLayer)	[(None, 128, 128, 3)	0	
conv2d_45 (Conv2D)	(None, 128, 128, 32)	896	input_6[0][0]
dropout_30 (Dropout)	(None, 128, 128, 32)	0	conv2d_45[0][0]
conv2d_46 (Conv2D)	(None, 128, 128, 32)	9248	dropout_30[0][0]
dropout_31 (Dropout)	(None, 128, 128, 32)	0	conv2d_46[0][0]
max_pooling2d_10 (MaxPooling2D)	(None, 64, 64, 32)	0	dropout_31[0][0]
conv2d_47 (Conv2D)	(None, 64, 64, 64)	18496	max_pooling2d_10[0][0]
max_pooling2d_11 (MaxPooling2D)	(None, 32, 32, 64)	0	conv2d_47[0][0]
conv2d_48 (Conv2D)	(None, 32, 32, 128)	73856	max_pooling2d_11[0][0]
dropout_32 (Dropout)	(None, 32, 32, 128)	0	conv2d_48[0][0]
conv2d_49 (Conv2D)	(None, 32, 32, 128)	147584	dropout_32[0][0]
dropout_33 (Dropout)	(None, 32, 32, 128)	0	conv2d_49[0][0]
up_sampling2d_10 (UpSampling2D)	(None, 64, 64, 128)	0	dropout_33[0][0]
concatenate_10 (Concatenate)	(None, 64, 64, 192)	0	up_sampling2d_10[0][0] conv2d_47[0][0]
conv2d_50 (Conv2D)	(None, 64, 64, 64)	110656	concatenate_10[0][0]
flatten_5 (Flatten)	(None, 65536)	0	max_pooling2d_11[0][0]
dropout_34 (Dropout)	(None, 64, 64, 64)	0	conv2d_50[0][0]
dense_10 (Dense)	(None, 128)	8388736	flatten_5[0][0]
conv2d_51 (Conv2D)	(None, 64, 64, 64)	36928	dropout_34[0][0]
p_re_lu_10 (PReLU)	(None, 128)	128	dense_10[0][0]
up_sampling2d_11 (UpSampling2D)	(None, 128, 128, 64)	0	conv2d_51[0][0]
dropout_36 (Dropout)	(None, 128)	0	p_re_lu_10[0][0]
concatenate_11 (Concatenate)	(None, 128, 128, 96)	0	up_sampling2d_11[0][0] dropout_31[0][0]
dense_11 (Dense)	(None, 128)	16512	dropout_36[0][0]
conv2d_52 (Conv2D)	(None, 128, 128, 32)	27680	concatenate_11[0][0]
p_re_lu_11 (PReLU)	(None, 128)	128	dense_11[0][0]
dropout_35 (Dropout)	(None, 128, 128, 32)	0	conv2d_52[0][0]
dropout_37 (Dropout)	(None, 128)	0	p_re_lu_11[0][0]
conv2d_53 (Conv2D)	(None, 128, 128, 32)	9248	dropout_35[0][0]
softmax_output (Dense)	(None, 7)	903	dropout_37[0][0]
decoder_output (Conv2D)	(None, 128, 128, 1)	33	conv2d_53[0][0]

Esta cuenta con un total de 8,841,032 parámetros entrenables.

Se han trabajado con dos modificaciones de esta arquitectura, una estándar con 8,8 millones de parámetros (la descrita anteriormente) y le versión ligera, con 4 M de parámetros. La única modificación entre la versión estándar y la ligera es que en el clasificador las capas de densidad pasan de tener 128 neuronas a 64.

Además de estas dos versiones se probará una versión hiper ligera de tan solo cuatrocientos mil parámetros. Esta consiste en una modificación de la arquitectura ligera donde se sustituye la capa de aplanamiento por una de agrupamiento medio global.

PLANTEAMIENTO DEL ESTUDIO

Se trabajará con imágenes de 128 x 128.

Se seguirá el mismo esquema de entrenamiento que en el experimento 1. Recordemos que el este es:

- El conjunto de entrenamiento = source (60% del total del conjunto de datos),
- El conjunto de reentrenamiento = train (70% del 40% restante del conjunto de datos),
- El conjunto de test = test (21% del 40% restante del conjunto de datos)
- El conjunto de validación = validación (9% del 40% restante del conjunto de datos)

Los hiperparámetros serán los siguiente:

- Ratio de aprendizaje de 0.001 y optimizador Amsgrad.
- Épocas: 60 para entrenamiento. No obstante se espera que el modelo pare entre la iteración 20 y 30.

Se incluirán algunas llamadas como:

- El ratio de aprendizaje se reducirá en un factor de 0.1 cada vez que el modelo de dos resultados negativos consecutivos.
- En caso de dar 6 resultados negativos continuados se parará el entrenamiento del modelo.
- Se guardarán los mejores pesos del modelo.

Se realizarán 10 entrenamientos con cada propuesta de forma que nos aseguremos que el modelo es robusto.

RESULTADOS

Los resultados obtenidos por el modelo estándar han sido los siguientes:

Para el **modelo estándar**:

	Clasificación	Segmentación
Iteraciones	0.9346428513526917 0.9307143092155457 0.9282143115997314 0.9282143115997314 0.9053571224212646 0.9292857050895691 0.9253571629524231 0.9303571581840515 0.928928554058075 0.9285714030265808	0.888629496097564 0.890869498252868 0.8843020796775818 0.883222997188568 0.884669303894043 0.890204191207885 0.885354101657867 0.886958718299865 0.884135365486145 0.881459355354309
Media	0.9269642889499664	0.8859805107116699

Para el **modelo ligero**:

	Clasificación	Segmentación
Iteraciones	0.9285714030265808 0.925000011920929 0.9235714077949524 0.931071400642395 0.9232142567634583 0.9253571629524231 0.927142858505249 0.9235714077949524 0.931071400642395 0.9267857074737549	0.8829624056816101 0.8853650093078613 0.8876053094863892 0.8906531929969788 0.8782516717910767 0.8849279880523682 0.8855441808700562 0.882202684879303 0.8903136253356934 0.8881378173828125
Media	0.926535701751709	0.8855963885784149

Para el modelo **hiper ligero**:

	Clasificación	Segmentación
Iteraciones	0.5453571677207947 0.6221428513526917 0.6585714221000671 0.6067857146263123 0.6389285922050476 0.708214282989502 0.6775000095367432 0.6214285492897034 0.6285714507102966 0.574999988079071	0.8924560546875 0.9011145234107971 0.9035531878471375 0.8970199227333069 0.8952728509902954 0.9096527099609375 0.9047389626502991 0.8978752493858337 0.8994259834289551 0.895118772983551
Media	0.628250002861023	0.8996228218078614

CONCLUSIONES

Antes de hablar de los resultados obtenidos es importante recalcar que esta arquitectura soluciona dos problemas (segmentación y clasificación), y que los pesos para cada modelo y sus precisiones son:

Red	Precisión en clasificación	Precisión en segmentación	Número de parámetros
Propuesta estándar	92.70%	88.60%	8.7M
Propuesta ligera	92.65%	88.56%	4M
Propuesta hiper ligera	62.83%	89.96%	400k
VGG-16	95.89%	-	15,7M
ResNet50	95.82%	-	27,7 M
InceptionV3	93.46%	-	23,8M
RSM_BAJ	95,33%	-	8.6 M

Con base en lo anterior, podemos ver que la propuesta estándar disminuye en 2.63% la precisión en la tarea de clasificación respecto al mismo modelo especializado en la clasificación (RSM_BAJ). A cambio, nos aporta una funcionalidad extra con el mismo número de parámetros.

Por otra parte, nuestra propuesta ligera tiene la mitad de los parámetros de RSM_BAJ y la propuesta estándar y solo reduce su precisión en un 0.05% lo cual es prácticamente despreciable.

La arquitectura hiper ligera es la que peores resultados da, no obstante, es una precisión aceptable teniendo en cuenta de que el modelo es 20 veces más ligero que la versión estándar.

Como conclusión de este estudio, queda en el aire si la hipótesis se valida o no. Por una parte, hemos encontrado un modelo un 50 % más pequeño el cual realiza la segmentación y la clasificación de la imagen. No obstante, un modelo especifico para la clasificación añade al modelo aproximadamente un 2.6 % de precisión.

Personalmente, se considera que esta pedida de precisión es asumible teniendo en cuenta la gran simplificación del modelo.