UNIVERSIDAD DE CONCEPCIÓN
FACULTADE INGENIERÍA

503649-1 DEEP LEARNING (S1-2022)
Tarea 1

Profesores
Guillermo Cabrera - Julio Godoy - Pedro Pinacho
Alumno
Iván Elías Montti Davison

Introducción

En la siguiente tarea se creó un modelo de red convolucional con el objetivo de clasificar estampillas HiTS (High Cadence Transient Survey) como objetos reales o artefactos.

Materiales y Métodos

Manipulación y Augmentación de Dataset

La forma de las estampillas de data_train.pkl es transformada a un matriz y son concatenadas, luego se augmentan los datos. Se rotan las estampillas concatenadas en 90, 180 y 270 grados, estas estampillas rotadas se concatenan con las originales. Así se logra pasar de 4096 elementos a 16384.

El mismo proceso, sin incluir la rotación, ocurre para los datos de unlab_test.pkl.

Modelo

Layer (type)	Output Shape	Param #
conv2d_36 (Conv2D)	(None, 21, 21, 4)	20
conv2d_37 (Conv2D)	(None, 20, 20, 8)	136
max_pooling2d_12 (MaxPoolin g2D)	(None, 10, 10, 8)	0
conv2d_38 (Conv2D)	(None, 9, 9, 16)	528
conv2d_39 (Conv2D)	(None, 8, 8, 16)	1040
max_pooling2d_13 (MaxPoolin g2D)	(None, 4, 4, 16)	0
conv2d_40 (Conv2D)	(None, 3, 3, 32)	2080
conv2d_41 (Conv2D)	(None, 2, 2, 32)	4128
flatten_6 (Flatten)	(None, 128)	0
dense_18 (Dense)	(None, 16)	2064
dropout_12 (Dropout)	(None, 16)	0
dense_19 (Dense)	(None, 8)	136
dropout_13 (Dropout)	(None, 8)	0
dense_20 (Dense)	(None, 2)	18

```
model.add(Conv2D(filters = 4,kernel_size = 1,input_shape = (21,21,4)))
model.add(Conv2D(8, 2, activation='relu'))
model.add(MaxPool2D(2, 2))
model.add(Conv2D(16, 2, activation='relu'))
model.add(Conv2D(16, 2, activation='relu'))
model.add(MaxPool2D(2, 2))
model.add(Conv2D(32, 2, activation='relu'))
model.add(Conv2D(32, 2, activation='relu'))
model.add(Flatten())
model.add(Dense(units=16,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=8,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units = 2,activation = 'softmax'))
```

Solo se utiliza la como función de activación el ReLU (Rectified Linear Unit), exceptuando la última capa, en donde se utiliza softmax con el objetivo de normalizar el output.

Componentes e hiperparametros

- Loss Function: MSE (Mean Square Error)
- Optimizer: RMSProp (Root Mean Squared Propagation)
- Batch Size: 128
- Epoch: 12

No se utilizaron callbacks[1] como Early Stopping.

Detalles de uso

Lo único que se debe hacer aparte correr las celdas secuencialmente, es indicar el camino a la carpeta con los archivos data_train.pkl y unlab_test.pkl en path. En el caso que los archivos sean importados por la interfaz gráfica se debe dejar la variable path vacía.

Resultados y Discusión

Se obtuvieron los siguientes resultados:

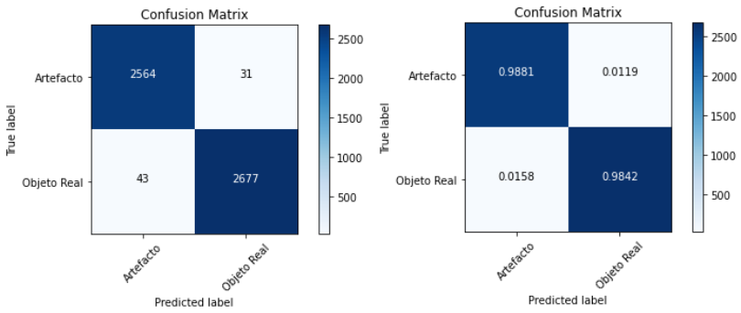

Fig 1, 2: Matrices de confusión, no regularizada y regularizada correspondientemente

accuracy	0.98607714
precision	0.988552437
recall	0.984191176
f1	0.986366986

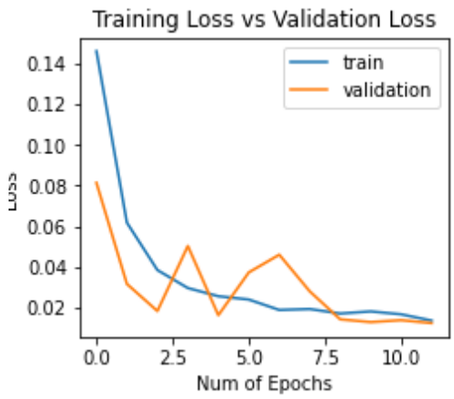

Fig. 3: Loss de training y validación.

El valor de loss de validación es inestable, lo que significa que el modelo completo es inestable. Esto se noto claramente al intentar replicar el primer valor de f1, pues se necesito correr el modelo múltiples veces para lograrlo. En la mayoría de los casos el valor de f1 rondaba 0.983.

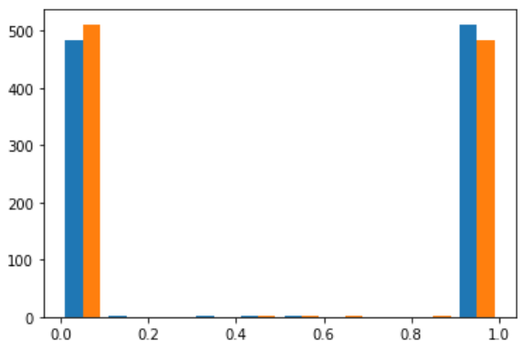

Fig . 4: Histograma con las probabilidades asignadas a los datos de prueba

El modelo discrimina fuertemente pues hay pocos valores entre 0 y 1. Igualmente se necesita aplicar un argmax() para limitar los valores a 0 o 1.

Conclusiones

Lamentablemente los resultados no son satisfactorios, la efectividad del modelo podría ser mayor y su principal problema es su inestabilidad. La razón de lo último probablemente se debe al tamaño exagerado del modelo comparado con el problema.

En el futuro debemos considerar una actitud más metódica al crear el modelo y evitar agregar capas innecesariamente.

Bibliografía

[1] “Keras API reference / Callbacks API.” n.d. Keras. Accessed May 6, 2022. https://keras.io/api/callbacks/.