

# PRÁCTICA II

Pablo Chantada Saborido & José Romero Conde

---



## Introducción

### Clasificación Ship/No-ship

#### Modelo base

Describimos ahora nuestro modelo baseline, que nos valió para iterar y comparar resultados. El modelo se compone de los siguientes elementos:

- **EfficientNet.** La usamos como CNN de partida. Originalmente pensamos que tenía como requerimiento que las imágenes sean de tamaño (244, 244, 3), para ello implementamos un recortador automático de *el cuadrado más grande* que llamamos desde los `DataLoader` de entrenamiento y test. Mas tarde nos dimos cuenta que la red era realmente capaz de procesar imágenes de cualquier tamaño porque tiene al final de todo un `Global Pooling`. No obstante, como de cualquier forma los *mini-batches* tenían que ser todos del mismo tamaño decidimos quedarnos con la idea de recortar las imágenes de la forma mencionada.
- **Aumento de datos.** El aumento de datos en esta fase esencialmente consta de dos partes, por un lado, las transformaciones de `torchvision.transforms` y por otro imágenes recortadas a mano. Con respecto a lo primero, nosotros usamos: volteos horizontales, fluctuaciones leves en el color, también leves trasnformaciones afines y conversión a blanco y negro con baja probabilidad. El segundo punto es que hemos recortado manualmente las imágenes de barcos de forma que queden centrados y sin fondo que estorbe, se han añadido estas imágenes al conjunto de datos cuando se especificaba aumento de datos. De esta forma conseguimos dos cosas distintas:
  - Representar más variaciones que las presentes en el conjunto de entrenamiento, con la esperanza de que si sí ocurriesen en test, estaríamos preparados.
  - Representar con menos ruido adicional lo que sí queremos aprender. Porque aunque es beneficioso para la red someterse a mucha variación, también dificulta el entrenamiento (es un problema más difícil).
- Un **MLP** de tres capas, sobre la salida de la red. La salida de la última capa `nn.Softmax()` tiene 2 o 3 neuronas según si se quería clasificar en {no barco, barco} o en {no barco, barco no amarrado, barco amarrado}.

#### Entrenamiento y evaluación

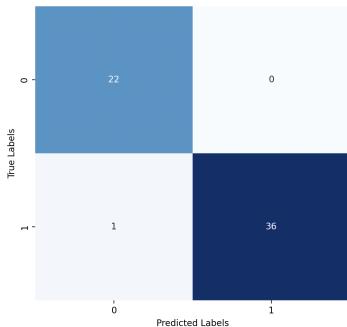
El entrenamiento fue difícil y además somos inexpertos pero hemos identificado algunos problemas para los que hemos encontrado solución. Aquí se presentan:

- **Datos no balanceados.** Observamos que la red, *hackeando* nuestras métricas decidía que todas las imágenes eran barcos (147 contra 88). Afrontamos esto alterando la probabilidad de ocurrencia de las clases de forma que si antes, al escoger una imagen de `DataLoader`,  $p(X = \text{barco}) = \frac{147}{147+88}$  ahora  $p(X = \text{barco}) = 0.5$ . Para ello usamos la utilidad `WeightedRandomSampler` de torch.
- **Capacidad del modelo.** Originalmente usamos la EfficientNet-B0, pero viendo malos resultados decidimos cambiar a la EfficientNet-B3, que es un poco más grande pero precisamente como explican en el artículo, sigue siendo muy eficiente.
- **Mínimos locales.** Probablemente si solucionásemos este problema de forma total estaríamos cerca de ganar un Nobel, naturalmente no lo hemos hecho. Pero para aliviar las consecuencias implementamos un `learning rate` que, con una cierta paciencia, mediaba su valor.<sup>1</sup>

---

<sup>1</sup>Esta paciencia mencionada y la del *earlystopping* pueden ambas ser ajustadas como `flags` en nuestro script.

- **Sobreajuste y generalización.** Para cerciorarnos de que la red aprendía correctamente, añadimos una penalización de la norma eucídea en la función de perdida, es decir, regularización Tikhonov.



Otras consideraciones sobre el entrenamiento que consideramos menos profundas y más ordinarias son: como optimizador hemos usado AdamW con un `learning_rate` de  $10^{-4}$ . Usamos un `batch_size` de 16, que aunque es relativamente pequeño, no mucho más pueden hacer nuestros portátiles.

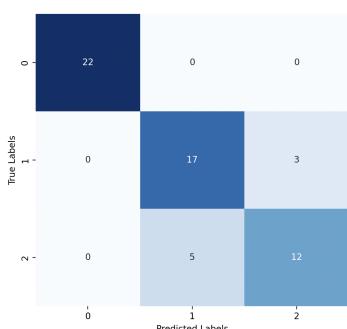
Con respecto a los resultados obtenidos en esta primera iteración, pueden observarse a grandes rasgos en la matriz de confusión de la izquierda, la cual nosotros prudentemente podemos decir que estamos orgullosos porque las predicciones de la red se concentran en las etiquetas verdaderas, fallando en sólo una imagen de 59. Figuras de las curvas de pérdida y precisión en aprendizaje o de las etiquetas predichas por la red para algunas imágenes en concreto pueden encontrarse para este ejercicio y el siguiente en el Apéndice al final del documento.

## Clasificación Docked/Undocked

Nosotros concebimos este ejercicio como una extensión del ejercicio anterior, de esta forma pensamos en reutilizar todas las representaciones aprendidas.

- Cuando se especifica `pretrained=True`, nuestra red no toma los pesos de Imagenet, sino del ejercicio anterior. De esta forma ya tiene mucho aprendido y el proceso de optimización será más sencillo.
- Como ahora la cabeza de la red tiene 3 neuronas pero antes tenía 2, en vez de inicializar aleatoriamente las tres neuronas<sup>2</sup>, dos se copian y sólo una se inicializa aleatoriamente. De nuevo, de esta forma tiene menos trabajo por delante.

## Resultados



Aunque a priori parece que son peores resultados que en el ejercicio 1, queremos señalar las siguientes observaciones:

- Ahora el problema original (barco contra no barco) está totalmente resuelto (no barco tiene la etiqueta de 0). A esto le encontramos la siguiente intuición: que la red se esfuerce en un problema más complejo hace que sea más capaz de resolver el problema sencillo.
- Las diferencias entre las clases atracado contra no atracado son realmente sutiles [poner foto] y a un humano (nosotros probamos con amigos) que no ha sido *entrenado* con las etiquetas también le puede resultar complejo resolver.
- No obstante en general podríamos decir que aunque la clasificación no es perfecta por lo menos es bastante diagonal.

<sup>2</sup>Como nuestro MLP tiene una única capa, cada neurona es un único vector del tamaño de salida de la red (1536 en el caso de la EfficientNet-B3), y esto significa que estamos asumiendo que el problema es linealmente separable en el espacio  $\mathbb{R}^{1536}$ . Esto me parece interesante chanti pero borralo siquieres

## Evaluación de modelos

### Análisis de resultados

### Conclusión

### Apéndice

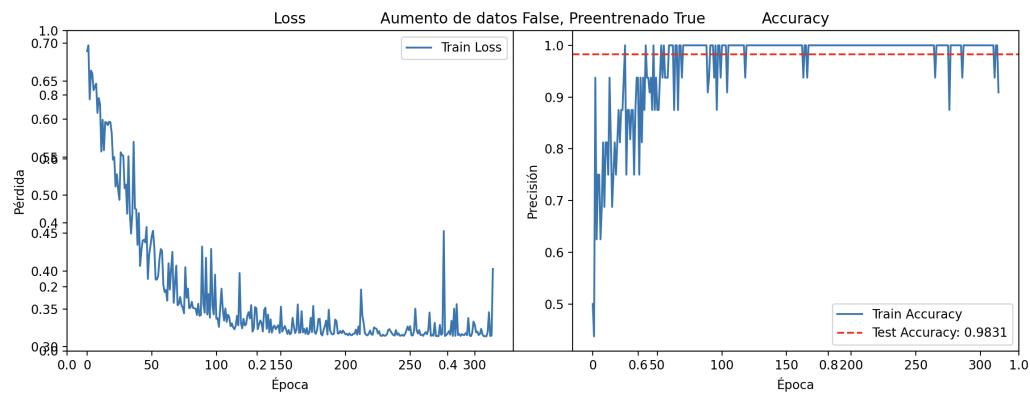


Figure 1: Loss y Accuracy para el primer ejercicio

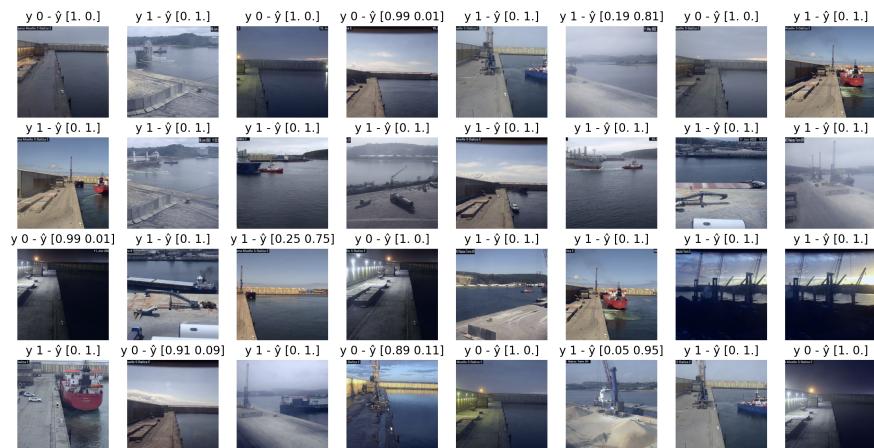


Figure 2: Imágenes con su etiqueta correcta y el vector SoftMax predicha por la red para el primer ejercicio

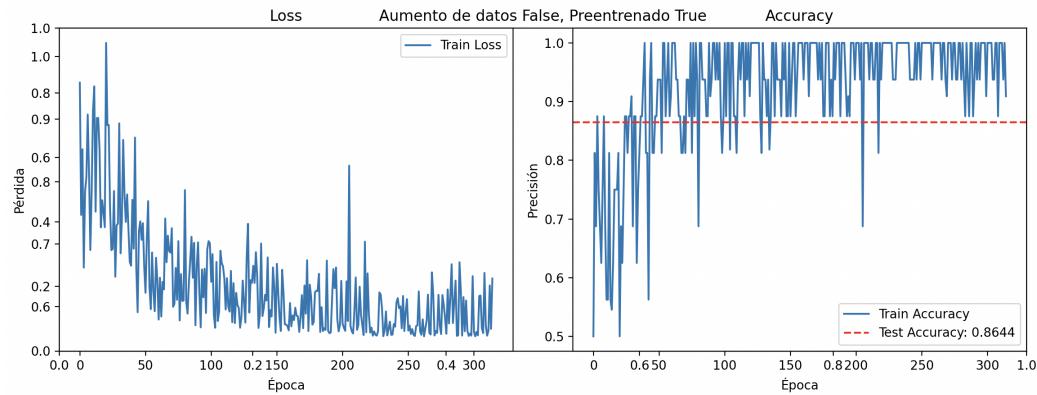


Figure 3: Loss y Accuracy para el segundo ejercicio



Figure 4: Imágenes con su etiqueta correcta y el vector SoftMax predicha por la red para el segundo ejercicio