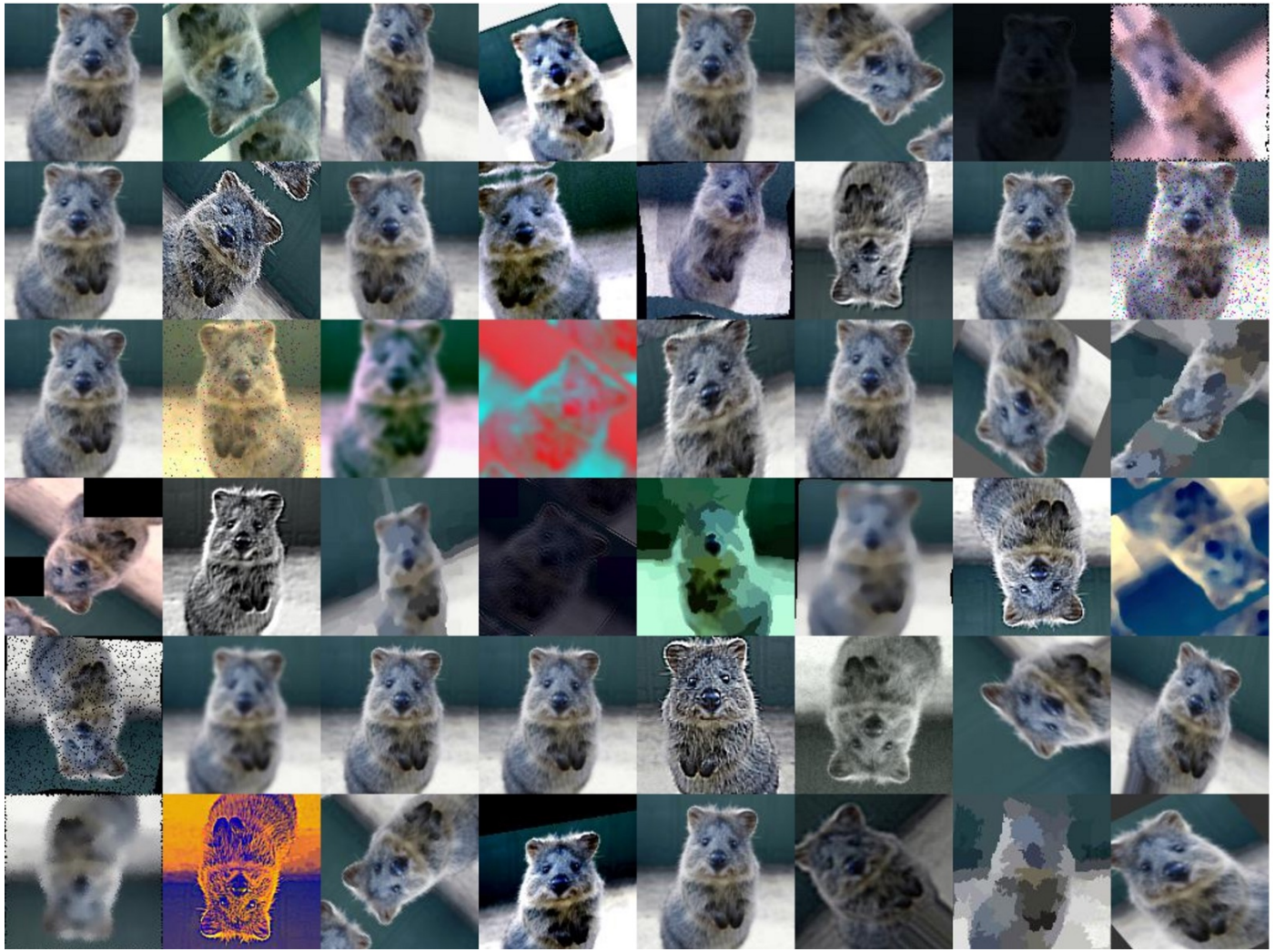
Tema 5. Aumento de datos

Ana Jiménez Pastor
anjipas@gmail.com

Aumento de datos

Conjunto de técnicas aplicadas a las imágenes con el objetivo de aumentar su tamaño creando muestras ligeramente **modificadas** o nuevas imágenes **sintéticas**. Ayudan a **reducir el sobreajuste** y a aumentar la **generalización** cuando se entrenan modelos basados en CNNs.

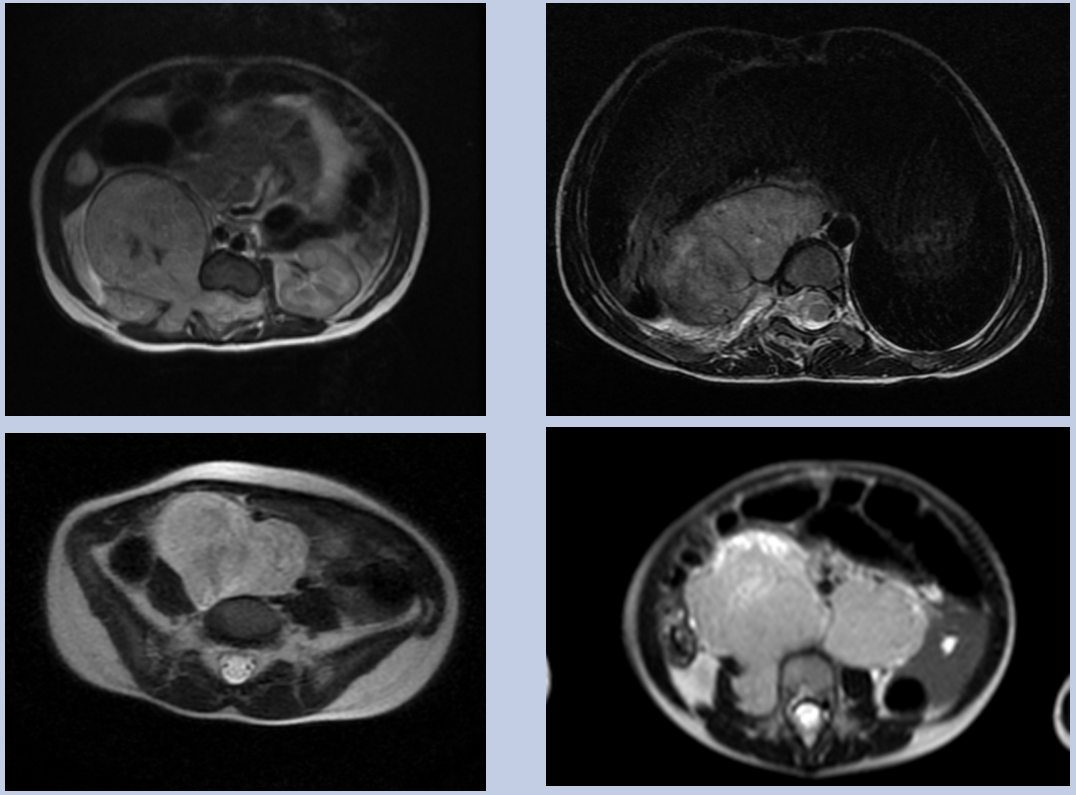


Aumento de datos

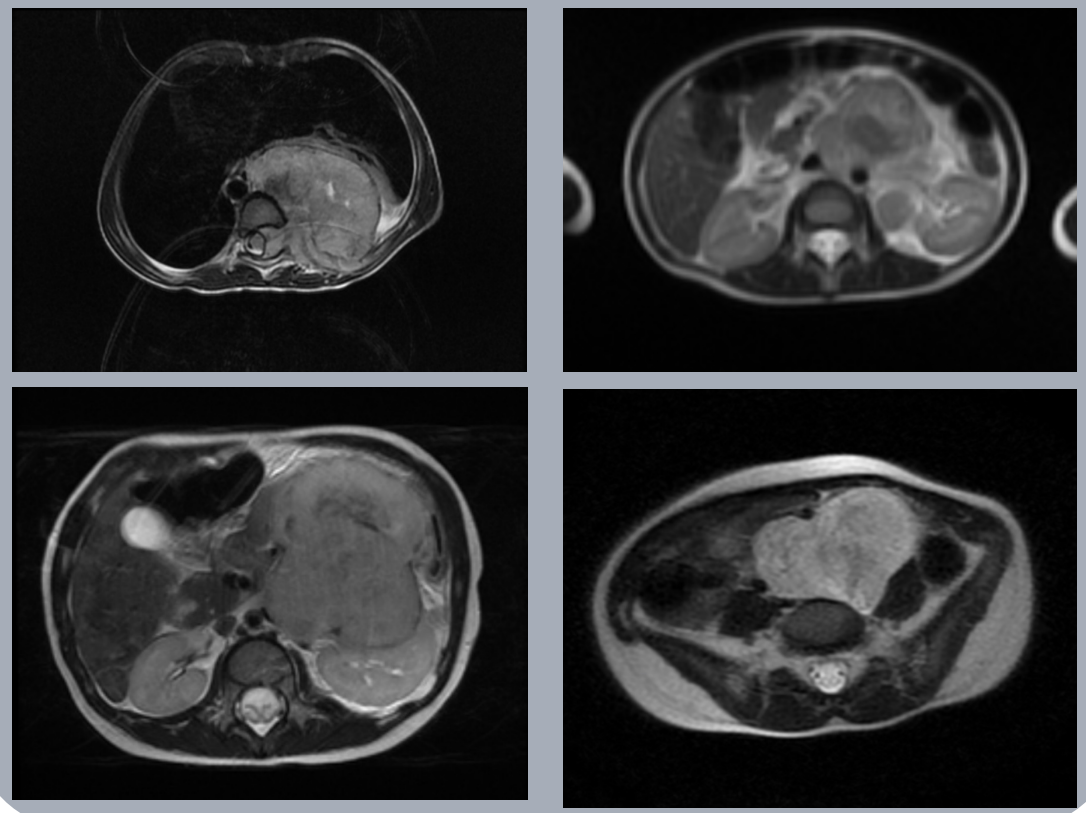
¿Por qué necesitamos aumentar los datos?

Imagina que queremos predecir el tiempo de supervivencia de un paciente empleando la siguiente base de datos…

Tiempo de supervivencia largo

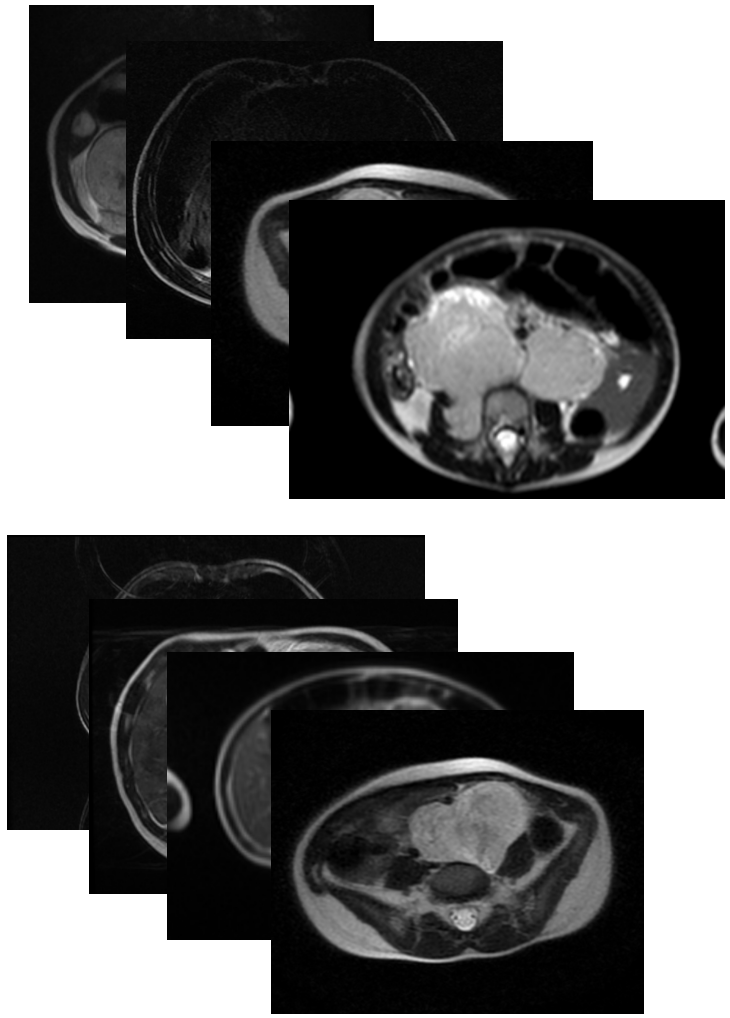


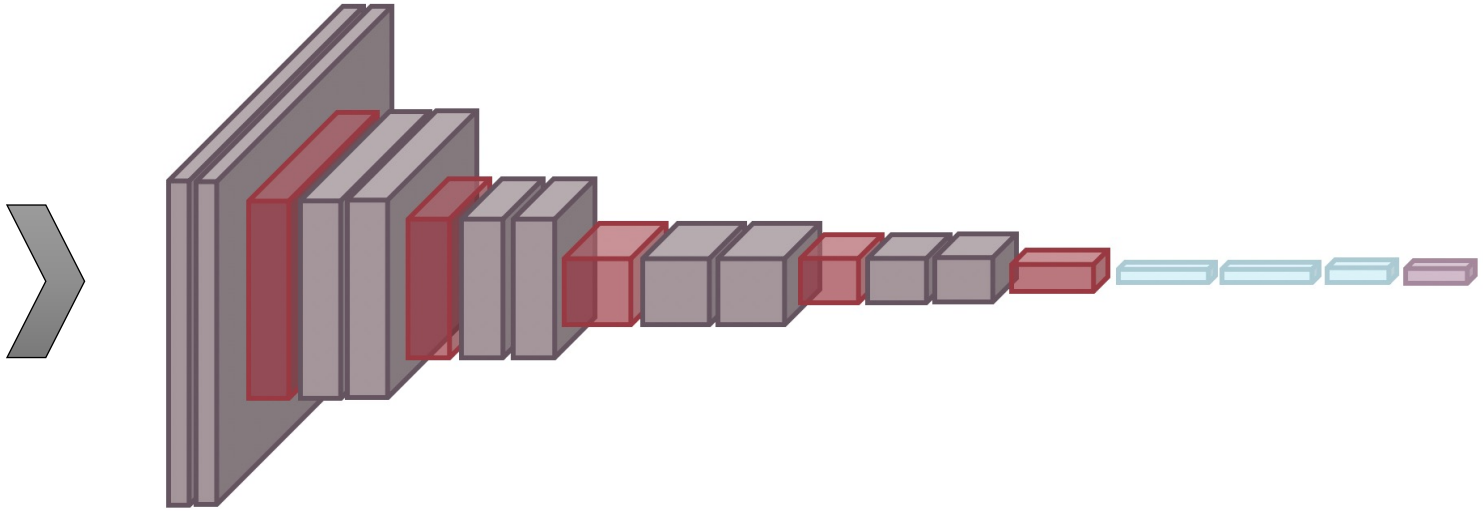
Tiempo de supervivencia corto



Aumento de datos

¿Por qué necesitamos aumentar los datos?

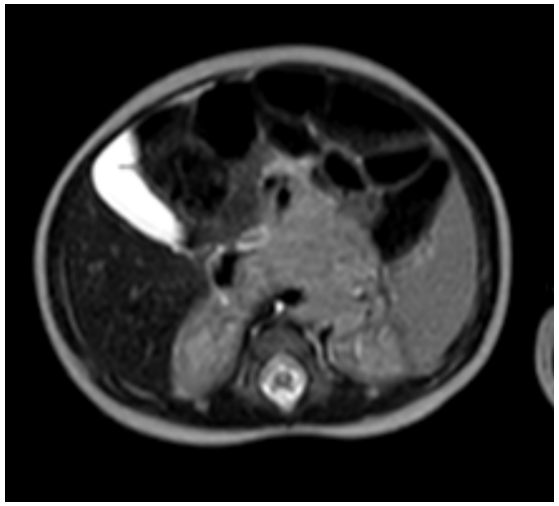




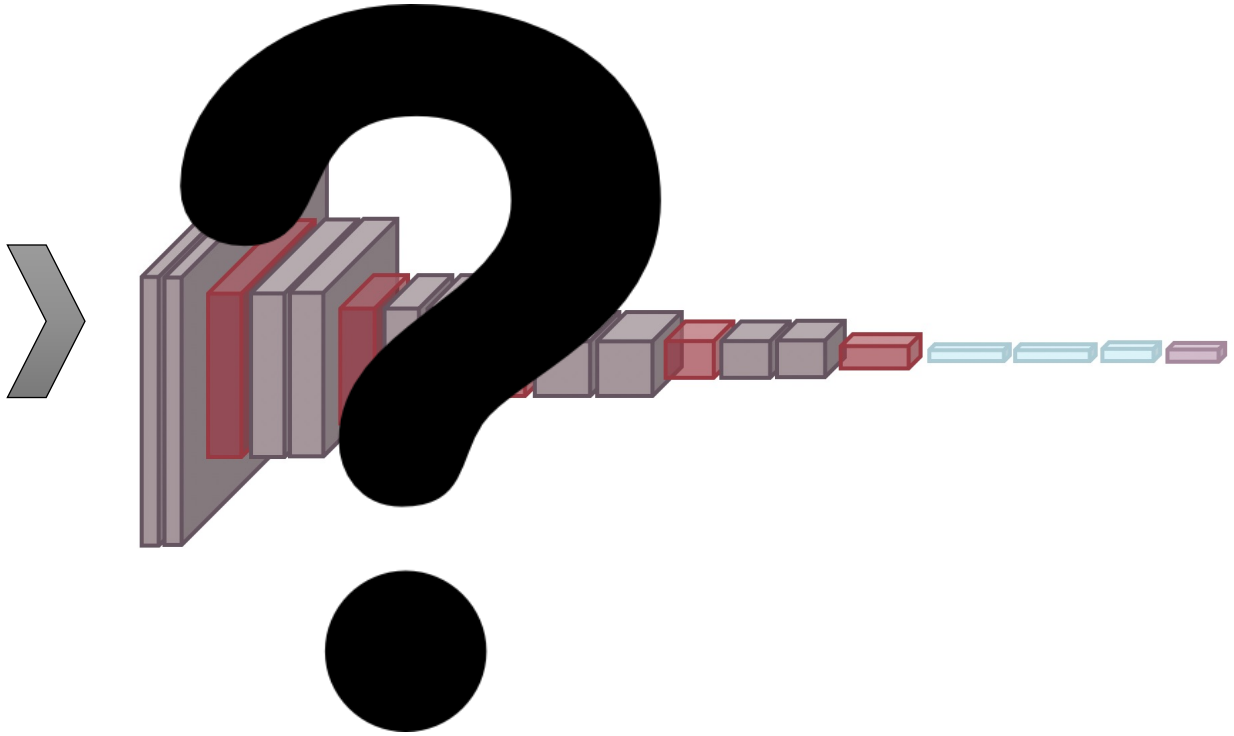
Precisión 99%

Aumento de datos

¿Por qué necesitamos aumentar los datos?



Tiempo de supervivencia largo

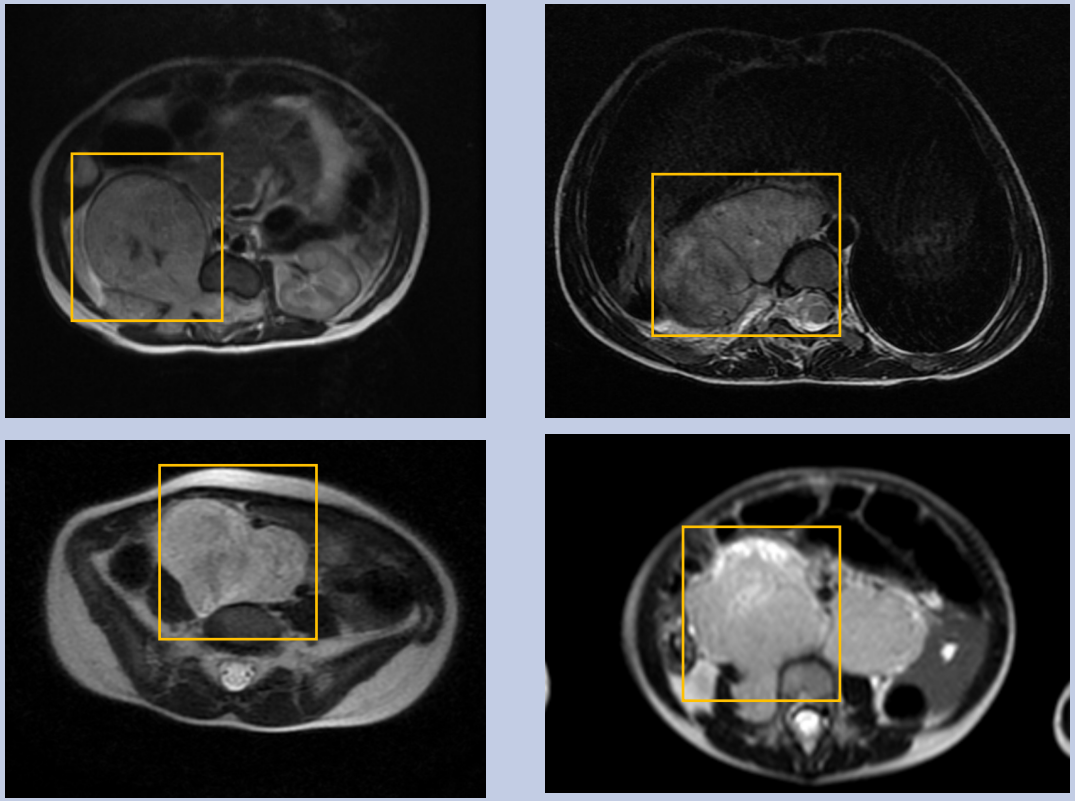


Tiempo de supervivencia corto

Aumento de datos

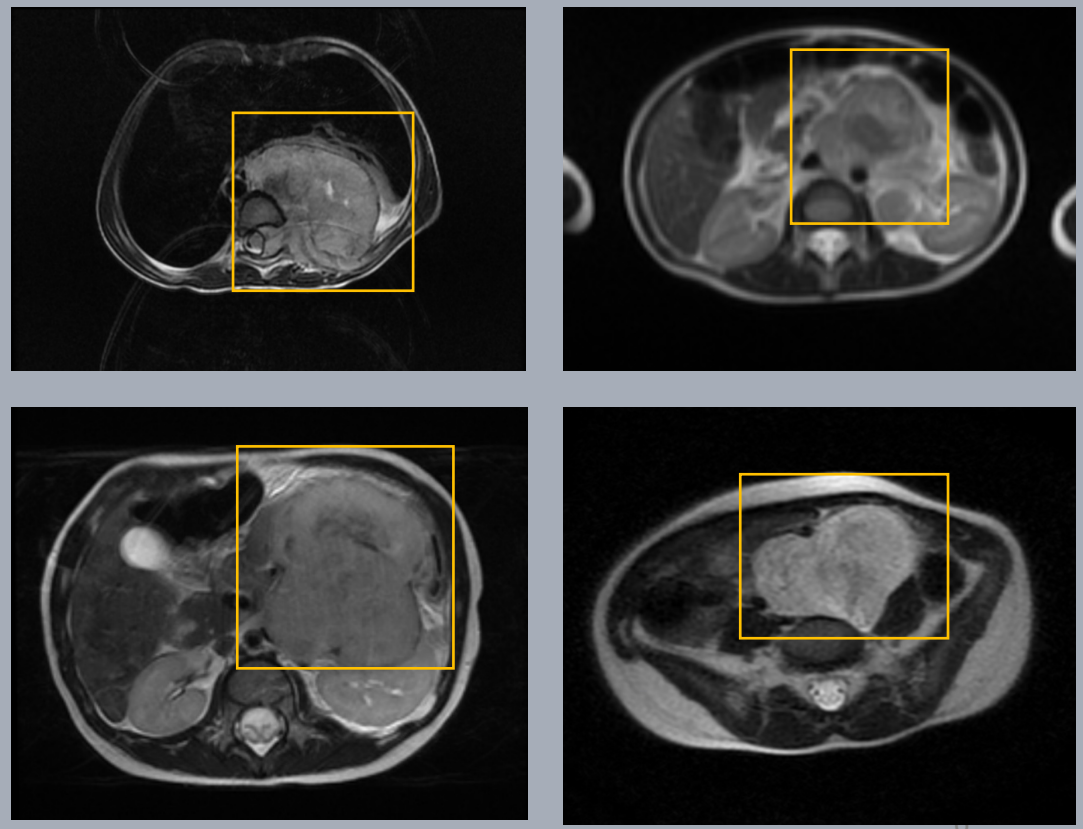
¿Por qué necesitamos aumentar los datos?

Tiempo de supervivencia largo



Derecha

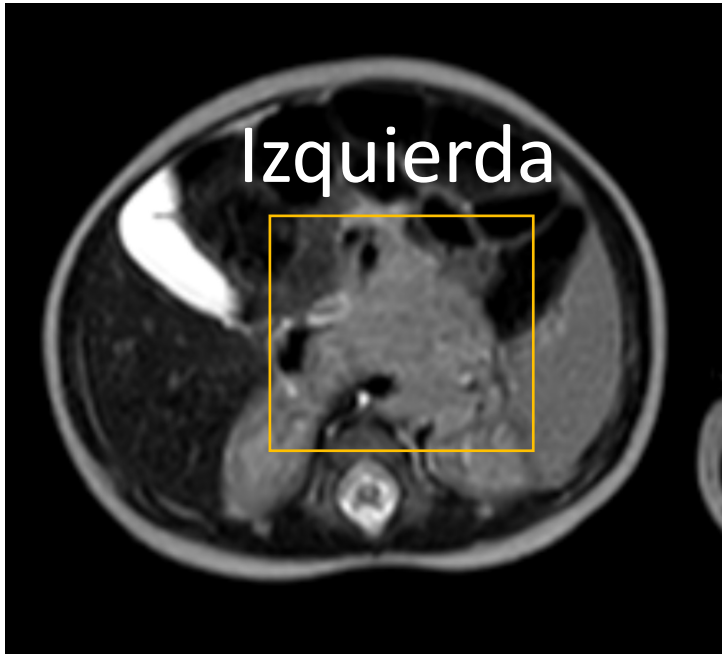
Tiempo de supervivencia corto



Izquierda

Aumento de datos

¿Por qué necesitamos aumentar los datos?



Tiempo de supervivencia largo

Tiempo de supervivencia corto

Aumento de datos

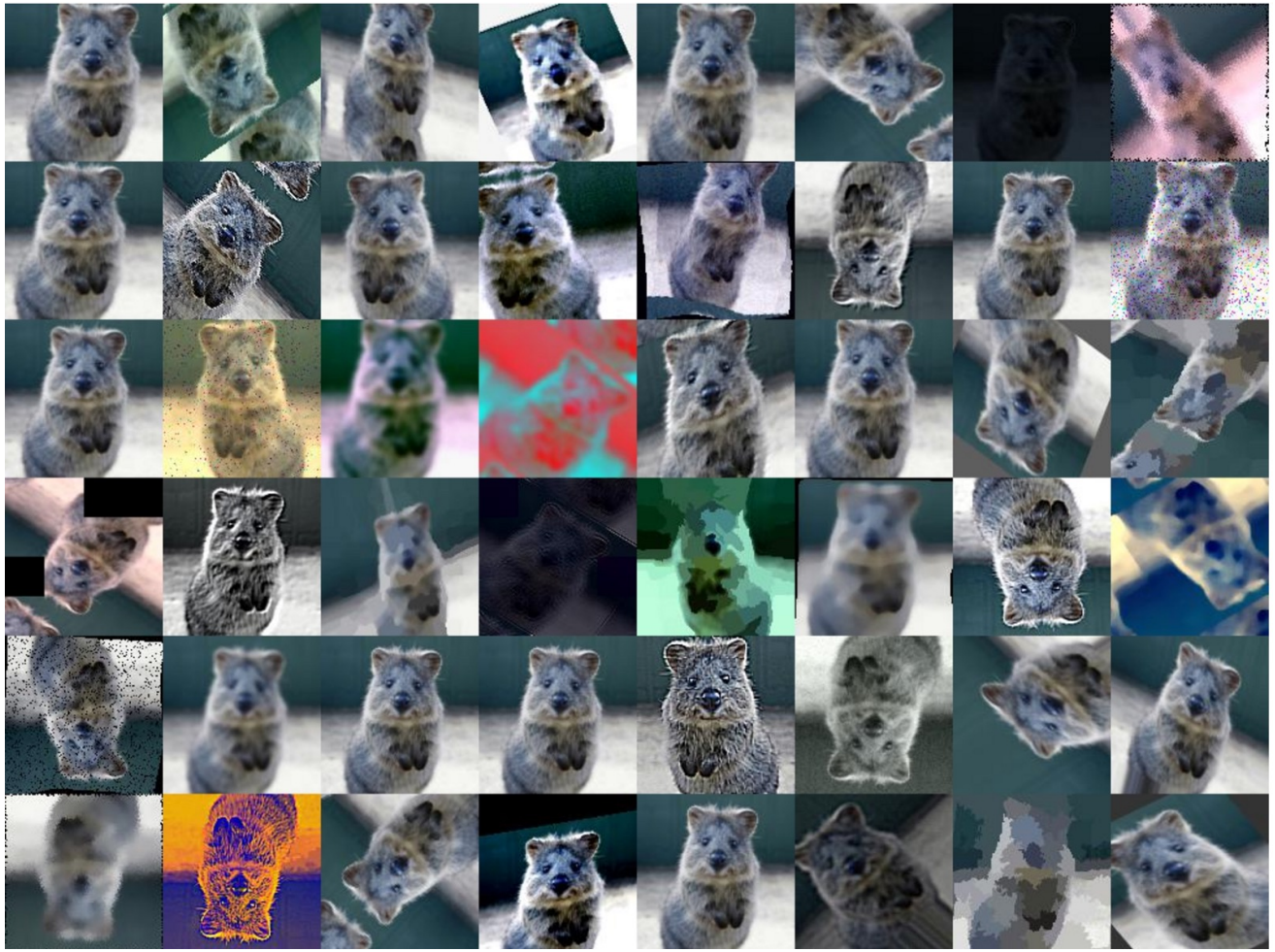
¿Por qué necesitamos aumentar los datos?

Los modelos de ML primero buscan las **características más obvias** a la hora de diferenciar entre clases. Por lo tanto, necesitamos reducir el número de **características irrelevantes** en nuestra base de datos.

Aumento de datos

¿Por qué necesitamos aumentar los datos?

- Rotaciones
- Traslaciones
- Cambios de contraste
- Añadido de ruido
- Filtrado
- Eliminado de zonas aleatorias de la imagen
- Cizallamientos
- Zoom
- …



Aumento de datos

Aumento de datos *offline*

- Se **incrementa la base de datos** un factor igual al número de transformaciones aplicadas.
- No se recomienda si trabajamos con **bases de datos grandes**.

Aumento de datos *online*

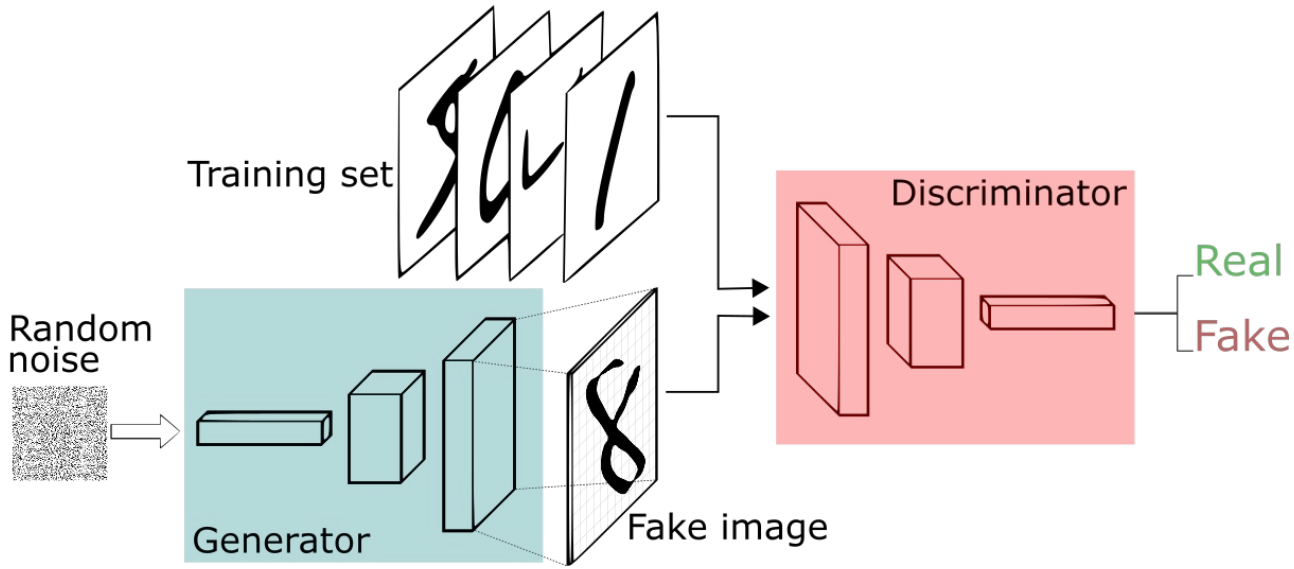
- Aumento de datos **al vuelo** durante el proceso de entrenamiento.
- Se aplican **transformaciones independientes** a cada batch.
- Se aplican **transformaciones aleatorias** a cada imagen. E.g: ruido + rotación; rotación + translación + cambio de contraste; zoom, etc.
- En cada época la misma imagen recibe transformaciones diferentes.

Aumento de datos

Redes Neuronales Adversarias (GANs) [Goodfellow et al., 2014]

Método para crear imágenes nuevas de manera sintéticas.
Las GANs están compuestas de dos redes:

- **Generador**: toma como entrada una variable aleatoria y devuelve, una vez entrenada, una variable aleatoria que sigue una distribución objetivo.
- **Discriminador**: toma como entrada una muestra y devuelve la probabilidad de ser una muestra "verdadera".



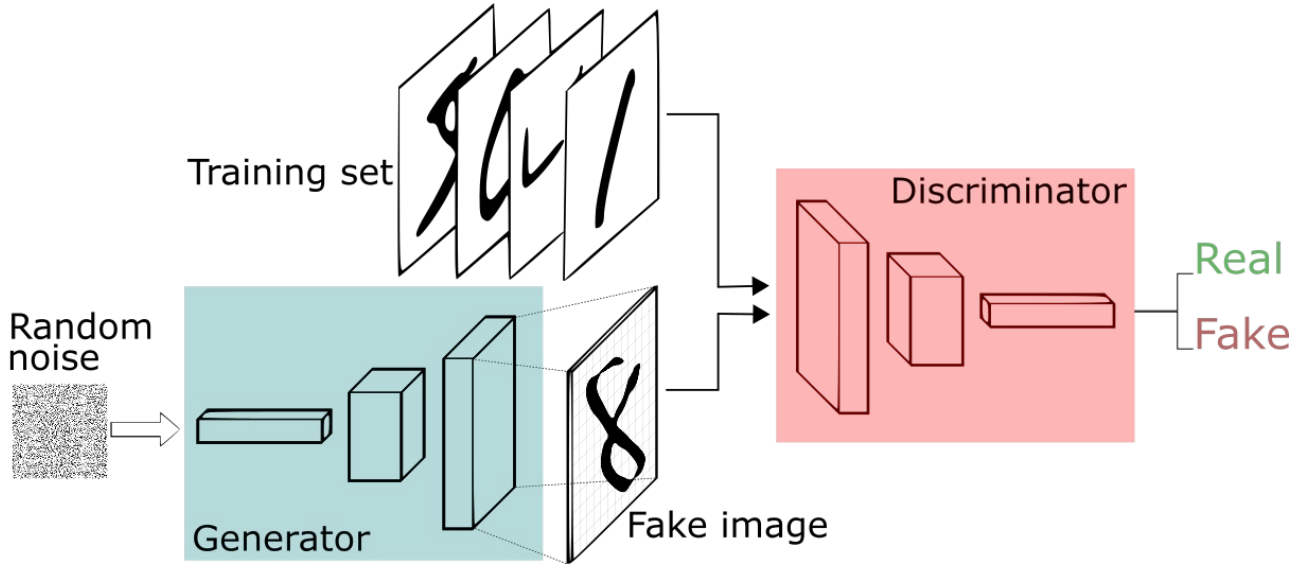
Aumento de datos

Redes Neuronales Adversarias (GANs) [Goodfellow et al., 2014]

Entrenamiento de una GAN:

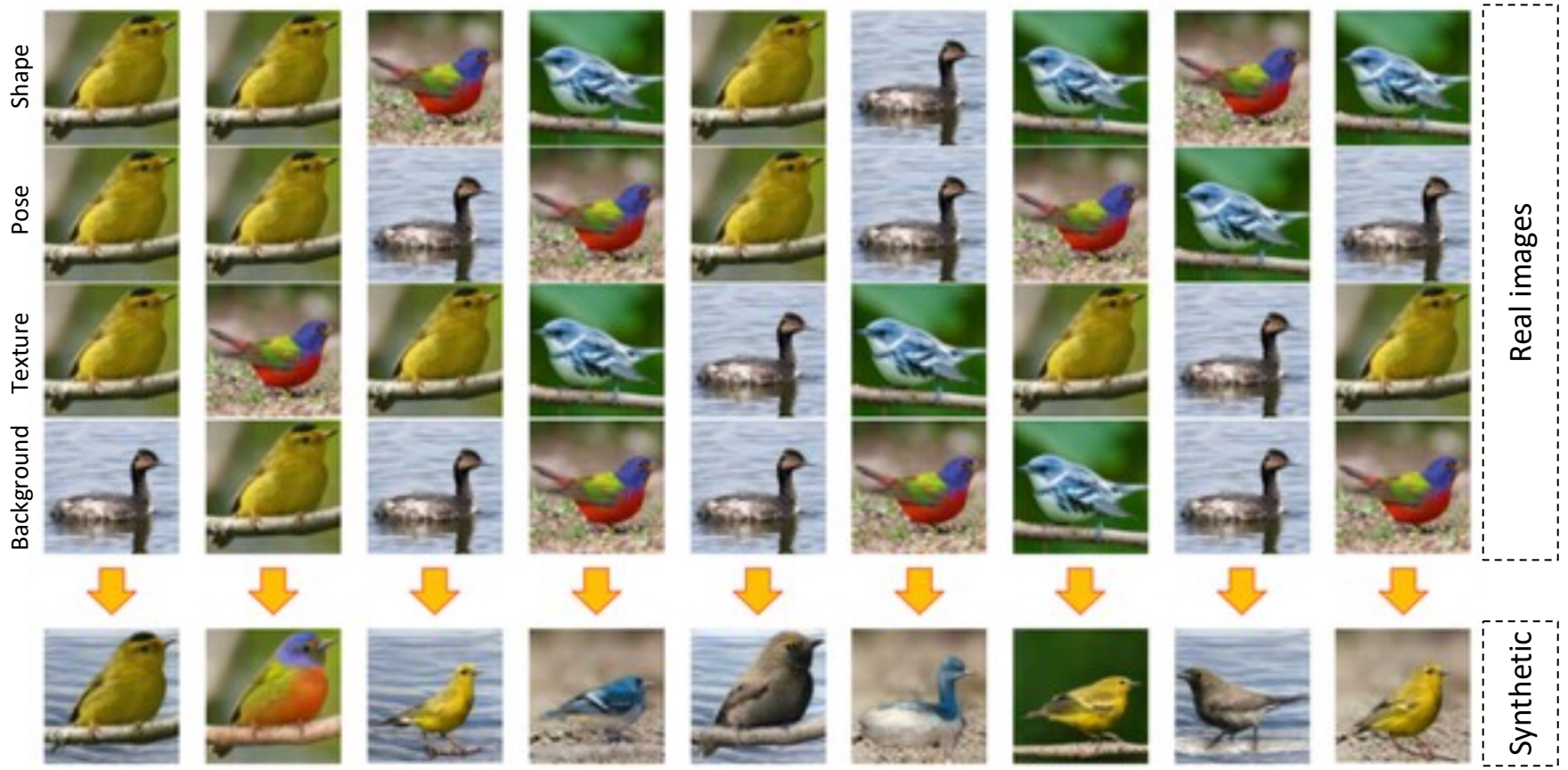
- El objetivo del **generador** es engañar al discriminador. Se entrena para **maximizar** el error de clasificación final.
- El objetivo del **discriminador** es detectar datos generados sintéticamente. Se entrena para **minimizar** el error en la clasificación final.

Uno minimiza el error y el otro lo maximiza → **Adversarias** → Compiten la una con la otra hasta alcanzar el equilibrio.



Aumento de datos

Redes Neuronales Adversarias (GANs) [Goodfellow et al., 2014]



Aumento de datos

Para aplicar aumento de datos vamos a emplear la librería **albumentations** y, para facilitar su implementación, de **ImageDataAugmentor.**

```
!pip install -q -U albumentations
!pip install git+https://github.com/mjkvaak/ImageDataAugmentor
```

```
import albumentations as A
from ImageDataAugmentor.image_data_augmentor import *
```

```
# Cargamos la imagen
img = Image.open('perro.jpeg')
img = np.array(img)

plt.figure(figsize=(10,10))
plt.imshow(img)
plt.axis('off')
```

Aumento de datos

Ejemplo: Giro horizontal

```
transform = A.Compose([
        A.HorizontalFlip(p=1.0)
])
augmented_image = transform(image=img)['image']

plt.figure(figsize=(10,10))
plt.imshow(augmented_image)
plt.axis('off')
```

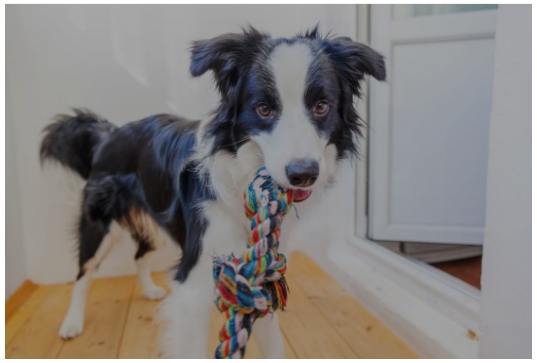
En la mayoría de las transformaciones encontramos el argumento “p”. Este define la probabilidad con la que se aplica la transformación durante el entrenamiento. p=0 hace que no se aplique nunca y p=1 hace que se aplique siempre.

Aumento de datos

Ejemplo: Cambio de brillo y contraste

Hay otras transformaciones que podemos establecer el límite de otros parámetros. Esto permite crear, de una misma imagen, una gran cantidad de variaciones que harán que aumente la generalización del modelo y disminuya el sobreajuste ya que en cada época la red "ve" imágenes diferentes.

```
transform = A.Compose([
        A.RandomBrightnessContrast(brightness_limit=0.5, contrast_limit=1, p=1.0),
])
augmented_image = transform(image=img)['image']

plt.figure(figsize=(10,10))
plt.imshow(augmented_image)
plt.axis('off')
```



Aumento de datos

Ejemplo: Giro horizontal + Cambio de brillo y contraste

```
transform = A.Compose([
        A.HorizontalFlip(p=0.6),
        A.RandomBrightnessContrast(brightness_limit=0.5, contrast_limit=0.5, p=1.0),
])
augmented_image = transform(image=img)['image']

plt.figure(figsize=(10,10))
plt.imshow(augmented_image)
plt.axis('off')
```

Aumento de datos

Ejercicio 1

Implementa una rutina de aumento de datos que haga lo siguiente:

- Giro horizontal con una probabilidad de 0.5
- Giro vertical con una probabilidad de 0.2
- Rotación de la imagen con un ángulo límite de 45º y con una probabilidad de 0.8
- Cambio de brillo y contraste con un contraste máximo de 0.8 y un brillo máximo de 0.8 y una probabilidad de 0.5
- Transformación CLAHE con una probabilidad de 0.5

Una vez implementada visualiza varias transformaciones.

NOTA: Ayúdate de la documentación de albumentations. Los parámetros que no se han especificado toman sus valores por defecto.

Aumento de datos

Aumento de datos durante el entrenamiento

Vamos a realizar el mismo entrenamiento que en la sesión anterior (clasificación de tipos de flores) aplicando transferencia de conocimiento y aumento de datos.

En primer lugar, vamos a entrenar un modelo sin aumento de datos para, posteriormente, realizar la comparativa:

Aumento de datos

Entrenamiento sin aumento de datos

```
IMG_SIZE = (224,224)
BATCH_SIZE = 32

train_dataset = image_dataset_from_directory(data_root,
        validation_split=0.2,
        subset="training",
        seed=123,
        image_size=IMG_SIZE,
        batch_size=BATCH_SIZE)

validation_dataset = image_dataset_from_directory(data_root,
        validation_split=0.2,
        subset="validation",
        seed=123,
        image_size=IMG_SIZE,
        batch_size=BATCH_SIZE)

AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_dataset.prefetch(buffer_size=AUTOTUNE)
validation_ds = validation_dataset.prefetch(buffer_size=AUTOTUNE)
```

Aumento de datos

Entrenamiento sin aumento de datos

```
IMG_SHAPE = IMG_SIZE + (3,)

def get_model():
    input = layers.Input(shape=IMG_SHAPE)
    preprocessing = layers.experimental.preprocessing.Rescaling(1./255)(input)
    x = layers.Conv2D(16, 3, padding='same', activation='relu')(preprocessing)
    x = layers.MaxPooling2D()(x)
    x = layers.Conv2D(32, 3, padding='same', activation='relu')(x)
    x = layers.MaxPooling2D()(x)
    x = layers.Conv2D(64, 3, padding='same', activation='relu')(x)
    x = layers.MaxPooling2D()(x)
    x = layers.Flatten()(x)
    x = layers.Dense(128, activation='relu')(x)
    output = layers.Dense(5, 'softmax')(x)

    model = Model(inputs=[input], outputs=[output])

    return model
```

Aumento de datos

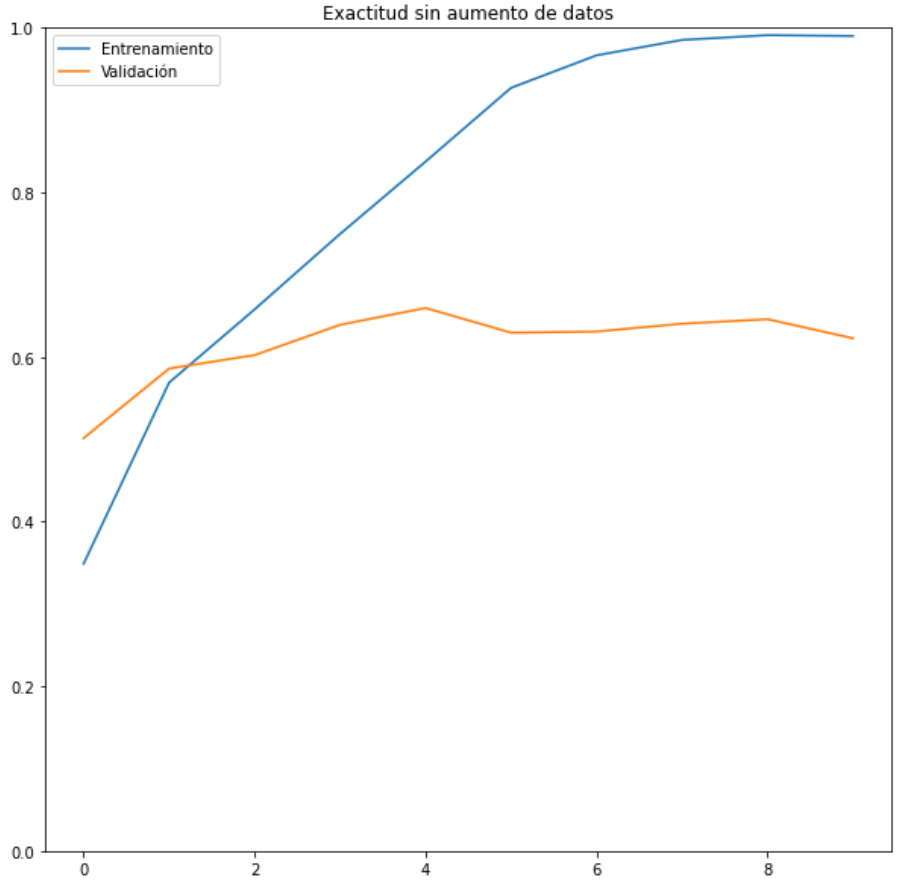
Entrenamiento sin aumento de datos

```
model_no_data_aug = get_model()

model_no_data_aug.compile(optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(
    filepath= path_experiment / 'flowers_no_data_aug.h5',
    monitor='val_accuracy',
    mode='max',
    save_best_only=True,
    verbose=1)

history_no_data_aug = model_no_data_aug.fit(train_ds,
    epochs=10,
    validation_data=validation_ds,
    callbacks=[model_checkpoint],
    verbose=1)

np.save(path_experiment / 'history_no_aug.npy', history_no_data_aug.history)
```

Aumento de datos

Entrenamiento sin aumento de datos

Visualización curvas aprendizaje entrenamiento y validación:



Aumento de datos

Entrenamiento con aumento de datos

En este caso los datos se van a cargar de manera ligeramente diferente ya que, primero, vamos a crear un generador de datos y, emplearemos este para aplicar las transformaciones cuando se carguen las imágenes del sistema de archivos.

Primero implementamos la rutina de aumento de datos con las transformaciones a aplicar:

```
# Implementamos la rutina de transformaciones
transforms = A.Compose([
        A.Rotate(limit=40, p=0.5),
        A.RandomBrightnessContrast(p=0.5),
        A.HorizontalFlip(p=0.5)
])
```

Aumento de datos

Entrenamiento con aumento de datos

A continuación, creamos los generadores de datos de entrenamiento y validación.

IMPORTANTE: Solo aplicamos aumento de datos en entrenamiento

```
# Generador de datos de entrenamiento
train_datagen = ImageDataAugmentor(
        augment=transforms,
        validation_split=0.2,
        seed=123)

# Generador de datos de validación --> ¡¡¡ No aplicamos aumento de datos !!!
val_datagen = ImageDataAugmentor(
        validation_split=0.2,
        seed=123)
```

Aumento de datos

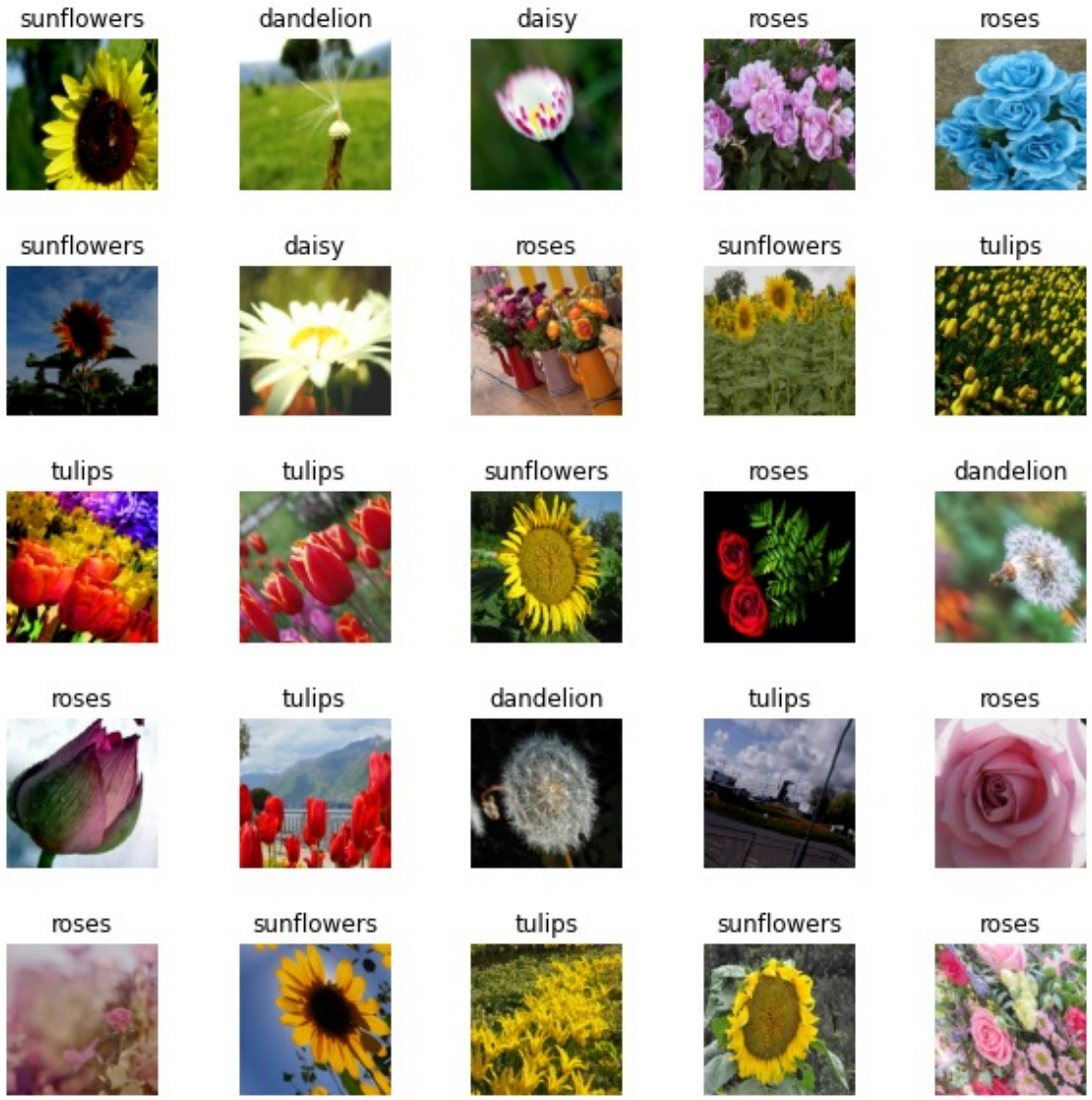
Entrenamiento con aumento de datos

```
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
# Dataset de entrenamiento
train_dataset = train_datagen.flow_from_directory(
        data_root,
        subset="training",
        target_size=IMG_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='sparse',
        shuffle=True)

# Dataset validación
validation_dataset = val_datagen.flow_from_directory(
        data_root,
        subset="validation",
        target_size=IMG_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='sparse',
        shuffle=True)
```

Aumento de datos

Entrenamiento con aumento de datos

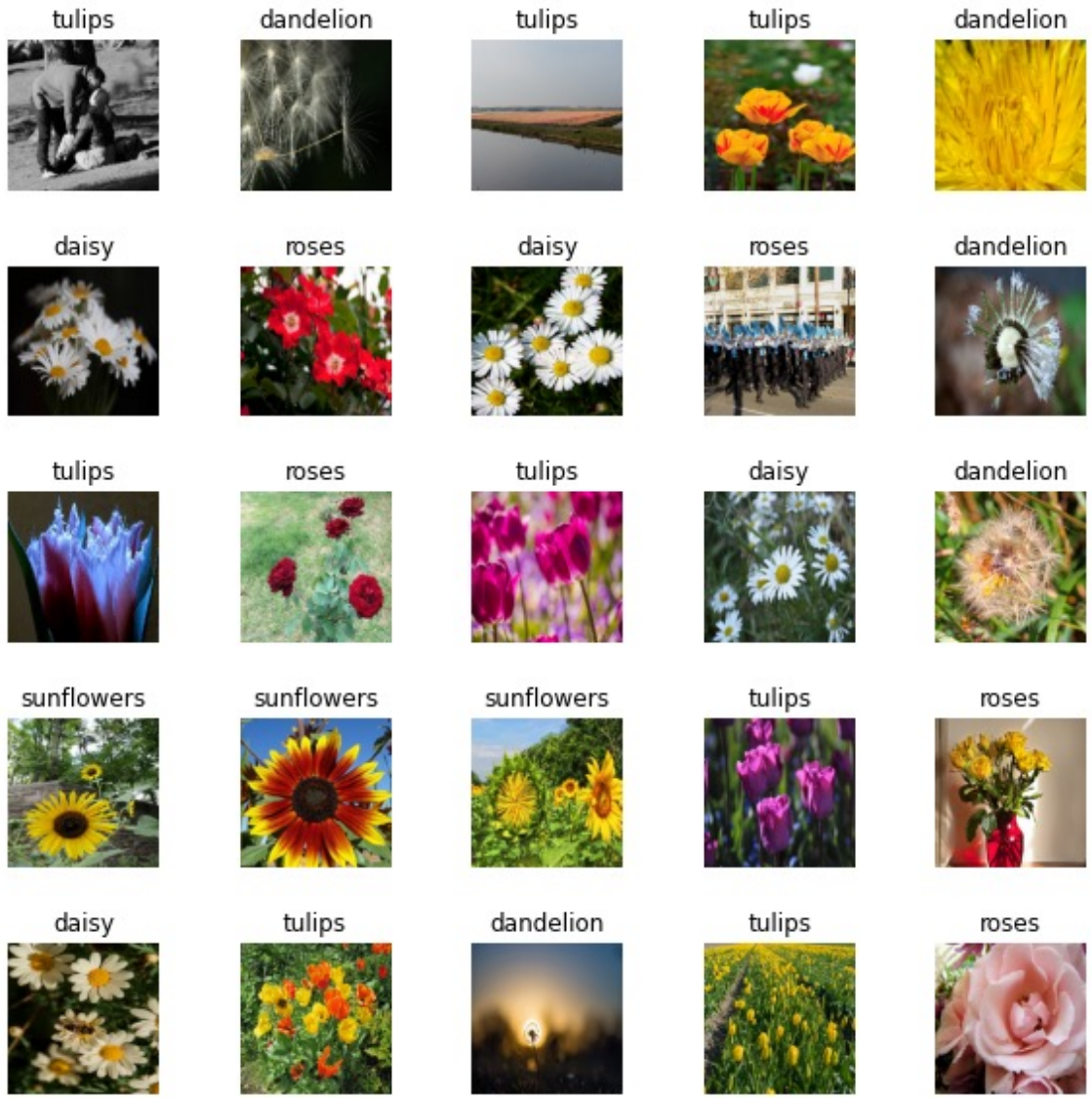
```
# Visualizamos imágenes entrenamiento:
# con aumento de datos

train_dataset.show_data()
```



Aumento de datos

Entrenamiento con aumento de datos

```
# Visualizamos imágenes validación:
# son aumento de datos

validation_dataset.show_data()
```



Aumento de datos

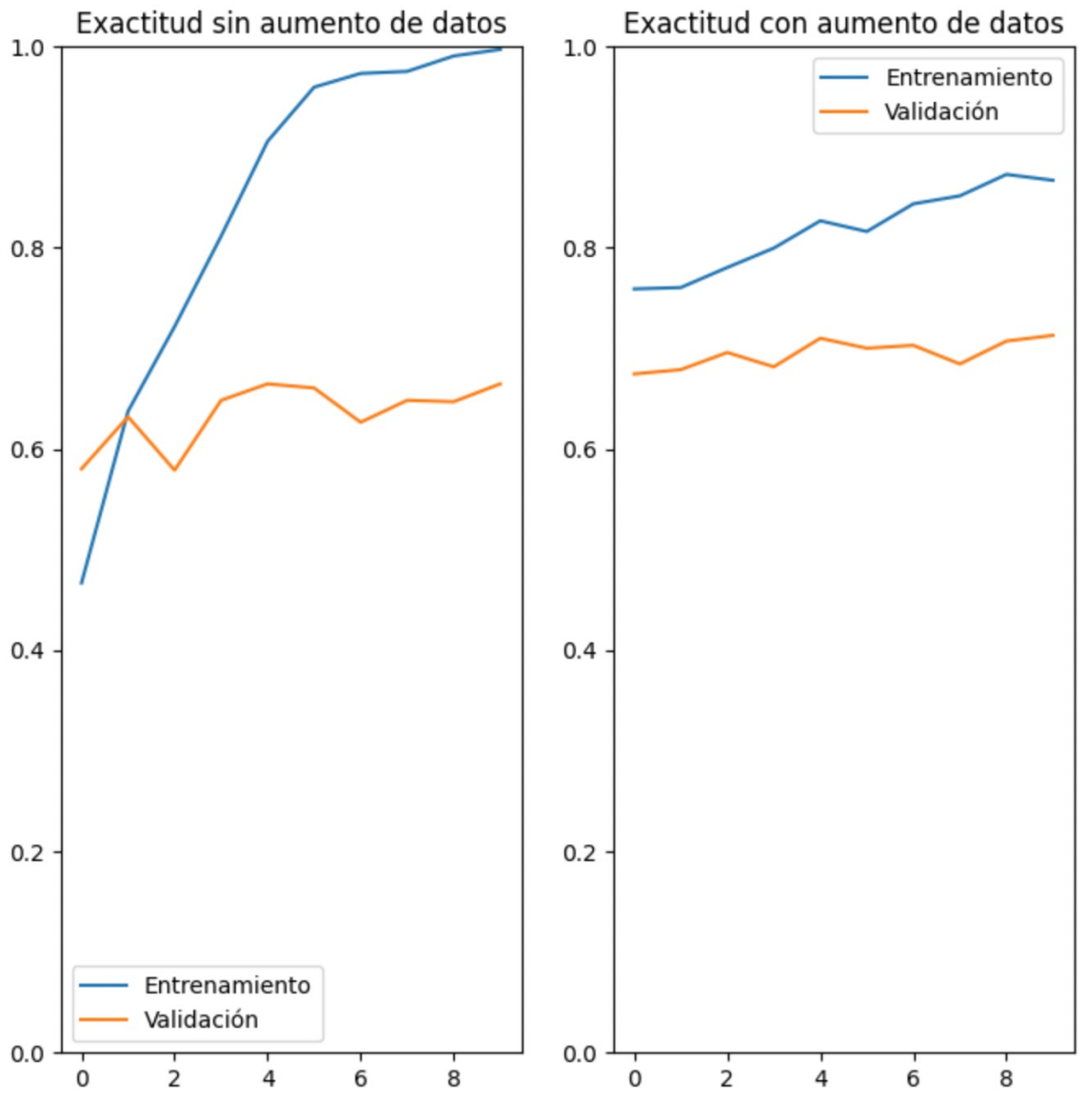
Entrenamiento con aumento de datos

```
model_data_aug = get_model()

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
        filepath=path_experiment / 'flowers.h5',
        monitor='val_accuracy',
        mode='max',
        save_best_only=True,
        verbose=1)

model_data_aug.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
                       metrics=['accuracy'])

train_samples = train_dataset.n
validation_samples = validation_dataset.n

history_data_aug = model_data_aug .fit(train_dataset,
                                       steps_per_epoch = train_samples//BATCH_SIZE,
                                       epochs=10,
                                       validation_data=validation_dataset,
                                       validation_steps = validation_samples//BATCH_SIZE,
                                       callbacks=[model_checkpoint_callback],
                                       verbose=1)

np.save(path_experiment / 'history_aug.npy', history_data_aug.history)
```

Aumento de datos

Entrenamiento con aumento de datos

Visualización curvas aprendizaje entrenamiento y validación: sin vs. con aumento de datos



Óptimización entrenamiento - EarlyStopping

Finalmente, vamos a ver otro callback que puede ser de gran interés a la hora de realizar el entrenamiento.
El número de épocas es un parámetro importante para evitar el **subajuste** (no entrenar durante el tiempo suficiente y, por tanto, el entrenamiento no llega a converger), y el **sobreajuste** (entrenar durante demasiado tiempo que los pesos se adaptan al conjunto de datos de entrenamiento y no es capaz de generalizar a nuevos datos).
Para evitar esto existe el callback **EarlyStopping**, que permite detener el entrenamiento tras un número determinado de épocas sin mejoras en el conjunto de datos de validación.

```
earlystopping = tf.keras.callbacks.EarlyStopping(monitor="val_loss",
                                                 mode="min", patience=5,
                                                 restore_best_weights=True,
                                                 verbose=1)
```

```
history_flowers = model_flowers.fit(train_dataset,
                                    epochs=10,
                                    validation_data=validation_dataset,
                                    callbacks=[model_checkpoint_callback, earlystopping],
                                    verbose=1)
```

Aumento de datos

Ejercicio 2

Realiza el mismo entrenamiento, pero esta vez aplica tanto aumento de datos como transferencia de conocimiento.

Práctica 2. Clasificación razas de perro

Vamos a experimentar con una nueva base de datos. En este caso, hemos cogido un subconjunto de imágenes de la base de datos *Stanford Dogs Dataset*. Esta es una base de datos que cuenta con 20.580 imágenes, divididas en 120 categorías. Sin embargo, en esta práctica, para simplificar el problema, hemos seleccionado, de estas 120 categorías, únicamente 10. Para la selección se han escogido aquellas clases que tienen un mayor número de imágenes.

Class	Files
n02111129-Leonberg	210
n02111500-Great_Pyrenees	213
n02086240-Shih-Tzu	214
n02111889-Samoyed	218
n02090721-Irish_wolfhound	218
n02107683-Bernese_mountain_dog	218
n02112018-Pomeranian	219
n02092002-Scottish_deerhound	232
n02088094-Afghan_hound	239
n02085936-Maltese_dog	252

Tema 5. Aumento de datos

Ana Jiménez Pastor
anjipas@gmail.com