deep-learning-76

April 24, 2025

1 Paso 1: Configuración inicial y librerías

[19]:
```
# Paso 1: Importar librerías necesarias
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os
import time
from datetime import datetime

from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input,␣
↪BatchNormalization, LeakyReLU
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint,␣
↪TensorBoard, ReduceLROnPlateau
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import regularizers

# Configurar semilla para reproducibilidad
SEED = 42
np.random.seed(SEED)
tf.random.set_seed(SEED)

# Verificar versión de TensorFlow
print(f"Versión de TensorFlow: {tf.__version__}")

# Verificar si hay GPU disponible
print("GPU disponible:", tf.config.list_physical_devices('GPU'))
```

```
Versión de TensorFlow: 2.18.0
GPU disponible: [PhysicalDevice(name='/physical_device:GPU:0',
device_type='GPU')]
```

#Paso 2: Cargar y explorar el dataset CIFAR-10

[20]:
```
# Paso 2: Cargar el dataset CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Nombres de las clases
class_names = ['avión', 'automóvil', 'pájaro', 'gato', 'ciervo',
               'perro', 'rana', 'caballo', 'barco', 'camión']

# Verificar las dimensiones
print("Tamaño de x_train:", x_train.shape)
print("Tamaño de y_train:", y_train.shape)
print("Tamaño de x_test:", x_test.shape)
print("Tamaño de y_test:", y_test.shape)
print("Valores de píxeles: min =", x_train.min(), ", max =", x_train.max())
```

```
Tamaño de x_train: (50000, 32, 32, 3)
Tamaño de y_train: (50000, 1)
Tamaño de x_test: (10000, 32, 32, 3)
Tamaño de y_test: (10000, 1)
Valores de píxeles: min = 0 , max = 255
```

2 Paso 3: Visualización de ejemplos e inspección de clases

[21]:
```
# Función para visualizar imágenes del dataset
def plot_sample_images(X, y, class_names, n_samples=10):
    plt.figure(figsize=(15, 3))
    for i in range(n_samples):
        idx = np.random.randint(0, X.shape[0])
        plt.subplot(1, n_samples, i+1)
        plt.imshow(X[idx])
        plt.title(class_names[y[idx][0]])
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Mostrar imágenes de entrenamiento
plot_sample_images(x_train, y_train, class_names)
```

3 Paso 4: Preprocesamiento de datos (normalización y codificación)

[22]:
```
# Normalizar valores de píxeles a rango [0, 1]
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Convertir etiquetas a one-hot encoding
y_train_cat = to_categorical(y_train, 10)
y_test_cat = to_categorical(y_test, 10)

# Confirmar formas
print("x_train:", x_train.shape)
print("y_train_cat:", y_train_cat.shape)
```

```
x_train: (50000, 32, 32, 3)
y_train_cat: (50000, 10)
```

4 Paso 5: Data Augmentation con ImageDataGenerator

[23]:
```
# Crear generador de datos con aumentos
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    zoom_range=0.1,
    fill_mode='nearest'
)

# Ajustar el generador a los datos de entrenamiento
datagen.fit(x_train)

# Verificar una muestra aumentada
it = datagen.flow(x_train, y_train_cat, batch_size=1)
x_batch, y_batch = next(it)

plt.imshow(x_batch[0])
plt.title("Ejemplo con aumento")
plt.axis("off")
plt.show()
```

5 Paso 6: Crear el modelo denso (MLP) con regularización y activaciones modernas

[24]:
```
from tensorflow.keras import regularizers

# Aplanar las imágenes (32x32x3 = 3072)
x_train_flat = x_train.reshape(x_train.shape[0], -1)
x_test_flat = x_test.reshape(x_test.shape[0], -1)

# Regularización L1 + L2
regularizador = regularizers.L1L2(l1=1e-5, l2=1e-4)

# Definir el modelo secuencial
modelo = Sequential([
    Input(shape=(3072,)),

    Dense(2048, kernel_regularizer=regularizador),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    Dropout(0.5),
```

```
    Dense(1024, kernel_regularizer=regularizador),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    Dropout(0.5),

    Dense(512, kernel_regularizer=regularizador),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    Dropout(0.5),

    Dense(256, kernel_regularizer=regularizador),
    BatchNormalization(),
    LeakyReLU(negative_slope=0.01),
    Dropout(0.5),

    Dense(10, activation='softmax')  # CIFAR-10 tiene 10 clases
])

# Resumen del modelo
modelo.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 2048)	6,293,504
batch_normalization_4 (BatchNormalization)	(None, 2048)	8,192
leaky_re_lu_4 (LeakyReLU)	(None, 2048)	0
dropout_4 (Dropout)	(None, 2048)	0
dense_6 (Dense)	(None, 1024)	2,098,176
batch_normalization_5 (BatchNormalization)	(None, 1024)	4,096
leaky_re_lu_5 (LeakyReLU)	(None, 1024)	0
dropout_5 (Dropout)	(None, 1024)	0
dense_7 (Dense)	(None, 512)	524,800
batch_normalization_6	(None, 512)	2,048

```
 (BatchNormalization)

 leaky_re_lu_6 (LeakyReLU)        (None, 512)                    0

 dropout_6 (Dropout)              (None, 512)                    0

 dense_8 (Dense)                  (None, 256)              131,328

 batch_normalization_7            (None, 256)                1,024
 (BatchNormalization)

 leaky_re_lu_7 (LeakyReLU)        (None, 256)                    0

 dropout_7 (Dropout)              (None, 256)                    0

 dense_9 (Dense)                  (None, 10)                 2,570


Total params: 9,065,738 (34.58 MB)

Trainable params: 9,058,058 (34.55 MB)

Non-trainable params: 7,680 (30.00 KB)
```

6 Paso 7: Compilar el modelo (SGD con momentum + Nesterov)

[25]:
```python
# Definir el optimizador SGD
optimizador = SGD(
    learning_rate=0.01,
    momentum=0.9,
    nesterov=True
)

# Compilar el modelo
modelo.compile(
    optimizer=optimizador,
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

7 Paso 8: Entrenamiento del modelo con callbacks

[30]:
```
# Directorio para logs y checkpoints
log_dir = "logs/" + datetime.now().strftime("%Y%m%d-%H%M%S")
checkpoint_path = "mejor_modelo.keras"

# Callbacks
callbacks = [
    TensorBoard(log_dir=log_dir),
    ModelCheckpoint(filepath=checkpoint_path, save_best_only=True,␣
 ↪monitor='val_loss', mode='min', verbose=1),
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True,␣
 ↪verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-6,␣
 ↪verbose=1)
]

# Entrenamiento
historial = modelo.fit(
    x_train_flat, y_train_cat,
    validation_data=(x_test_flat, y_test_cat),
    epochs=100,
    batch_size=128,
    callbacks=callbacks
)
```

```
Epoch 1/100
388/391             0s 8ms/step -
accuracy: 0.7062 - loss: 1.5353
Epoch 1: val_loss improved from inf to 2.00539, saving model to
mejor_modelo.keras
391/391             4s 10ms/step -
accuracy: 0.7063 - loss: 1.5351 - val_accuracy: 0.5741 - val_loss: 2.0054 -
learning_rate: 6.2500e-04
Epoch 2/100
389/391             0s 6ms/step -
accuracy: 0.7224 - loss: 1.5031
Epoch 2: val_loss did not improve from 2.00539
391/391             4s 7ms/step -
accuracy: 0.7224 - loss: 1.5030 - val_accuracy: 0.5708 - val_loss: 2.0268 -
learning_rate: 6.2500e-04
Epoch 3/100
383/391             0s 6ms/step -
accuracy: 0.7210 - loss: 1.4911
Epoch 3: val_loss did not improve from 2.00539
391/391             3s 7ms/step -
accuracy: 0.7211 - loss: 1.4908 - val_accuracy: 0.5710 - val_loss: 2.0323 -
```

```
learning_rate: 6.2500e-04
Epoch 4/100
391/391             0s 6ms/step -
accuracy: 0.7258 - loss: 1.4794
Epoch 4: val_loss did not improve from 2.00539
391/391             3s 8ms/step -
accuracy: 0.7258 - loss: 1.4794 - val_accuracy: 0.5749 - val_loss: 2.0171 -
learning_rate: 6.2500e-04
Epoch 5/100
391/391             0s 6ms/step -
accuracy: 0.7316 - loss: 1.4601
Epoch 5: val_loss improved from 2.00539 to 1.99887, saving model to
mejor_modelo.keras
391/391             3s 8ms/step -
accuracy: 0.7316 - loss: 1.4601 - val_accuracy: 0.5766 - val_loss: 1.9989 -
learning_rate: 6.2500e-04
Epoch 6/100
389/391             0s 6ms/step -
accuracy: 0.7295 - loss: 1.4668
Epoch 6: val_loss did not improve from 1.99887
391/391             3s 7ms/step -
accuracy: 0.7295 - loss: 1.4667 - val_accuracy: 0.5663 - val_loss: 2.0735 -
learning_rate: 6.2500e-04
Epoch 7/100
386/391             0s 6ms/step -
accuracy: 0.7324 - loss: 1.4567
Epoch 7: val_loss did not improve from 1.99887
391/391             3s 7ms/step -
accuracy: 0.7325 - loss: 1.4565 - val_accuracy: 0.5761 - val_loss: 2.0360 -
learning_rate: 6.2500e-04
Epoch 8/100
388/391             0s 6ms/step -
accuracy: 0.7301 - loss: 1.4510
Epoch 8: val_loss improved from 1.99887 to 1.98585, saving model to
mejor_modelo.keras
391/391             3s 8ms/step -
accuracy: 0.7302 - loss: 1.4509 - val_accuracy: 0.5812 - val_loss: 1.9858 -
learning_rate: 6.2500e-04
Epoch 9/100
388/391             0s 7ms/step -
accuracy: 0.7363 - loss: 1.4408
Epoch 9: val_loss did not improve from 1.98585
391/391             5s 7ms/step -
accuracy: 0.7363 - loss: 1.4407 - val_accuracy: 0.5625 - val_loss: 2.0839 -
learning_rate: 6.2500e-04
Epoch 10/100
388/391             0s 6ms/step -
accuracy: 0.7397 - loss: 1.4381
```

```
Epoch 10: val_loss did not improve from 1.98585
391/391            5s 7ms/step -
accuracy: 0.7398 - loss: 1.4380 - val_accuracy: 0.5634 - val_loss: 2.1013 -
learning_rate: 6.2500e-04
Epoch 11/100
385/391            0s 7ms/step -
accuracy: 0.7376 - loss: 1.4337
Epoch 11: val_loss did not improve from 1.98585
391/391            3s 8ms/step -
accuracy: 0.7376 - loss: 1.4336 - val_accuracy: 0.5645 - val_loss: 2.0739 -
learning_rate: 6.2500e-04
Epoch 12/100
387/391            0s 6ms/step -
accuracy: 0.7436 - loss: 1.4259
Epoch 12: val_loss did not improve from 1.98585
391/391            5s 7ms/step -
accuracy: 0.7436 - loss: 1.4258 - val_accuracy: 0.5828 - val_loss: 1.9959 -
learning_rate: 6.2500e-04
Epoch 13/100
386/391            0s 6ms/step -
accuracy: 0.7444 - loss: 1.4114
Epoch 13: val_loss did not improve from 1.98585

Epoch 13: ReduceLROnPlateau reducing learning rate to 0.0003124999930150807.
391/391            5s 7ms/step -
accuracy: 0.7444 - loss: 1.4114 - val_accuracy: 0.5756 - val_loss: 2.0152 -
learning_rate: 6.2500e-04
Epoch 14/100
388/391            0s 7ms/step -
accuracy: 0.7433 - loss: 1.4132
Epoch 14: val_loss improved from 1.98585 to 1.98095, saving model to
mejor_modelo.keras
391/391            4s 9ms/step -
accuracy: 0.7433 - loss: 1.4130 - val_accuracy: 0.5887 - val_loss: 1.9810 -
learning_rate: 3.1250e-04
Epoch 15/100
383/391            0s 6ms/step -
accuracy: 0.7454 - loss: 1.4057
Epoch 15: val_loss improved from 1.98095 to 1.96078, saving model to
mejor_modelo.keras
391/391            5s 8ms/step -
accuracy: 0.7455 - loss: 1.4055 - val_accuracy: 0.5929 - val_loss: 1.9608 -
learning_rate: 3.1250e-04
Epoch 16/100
387/391            0s 7ms/step -
accuracy: 0.7505 - loss: 1.3942
Epoch 16: val_loss did not improve from 1.96078
391/391            5s 8ms/step -
```

```
accuracy: 0.7505 - loss: 1.3941 - val_accuracy: 0.5864 - val_loss: 1.9937 -
learning_rate: 3.1250e-04
Epoch 17/100
383/391            0s 6ms/step -
accuracy: 0.7533 - loss: 1.3865
Epoch 17: val_loss did not improve from 1.96078
391/391            5s 7ms/step -
accuracy: 0.7534 - loss: 1.3862 - val_accuracy: 0.5869 - val_loss: 2.0064 -
learning_rate: 3.1250e-04
Epoch 18/100
383/391            0s 6ms/step -
accuracy: 0.7514 - loss: 1.3841
Epoch 18: val_loss did not improve from 1.96078
391/391            5s 7ms/step -
accuracy: 0.7515 - loss: 1.3839 - val_accuracy: 0.5873 - val_loss: 1.9844 -
learning_rate: 3.1250e-04
Epoch 19/100
384/391            0s 6ms/step -
accuracy: 0.7580 - loss: 1.3755
Epoch 19: val_loss improved from 1.96078 to 1.95904, saving model to
mejor_modelo.keras
391/391            5s 8ms/step -
accuracy: 0.7581 - loss: 1.3754 - val_accuracy: 0.5935 - val_loss: 1.9590 -
learning_rate: 3.1250e-04
Epoch 20/100
384/391            0s 6ms/step -
accuracy: 0.7559 - loss: 1.3755
Epoch 20: val_loss did not improve from 1.95904
391/391            3s 7ms/step -
accuracy: 0.7559 - loss: 1.3754 - val_accuracy: 0.5887 - val_loss: 1.9984 -
learning_rate: 3.1250e-04
Epoch 21/100
388/391            0s 7ms/step -
accuracy: 0.7554 - loss: 1.3724
Epoch 21: val_loss did not improve from 1.95904
391/391            6s 9ms/step -
accuracy: 0.7554 - loss: 1.3723 - val_accuracy: 0.5927 - val_loss: 1.9807 -
learning_rate: 3.1250e-04
Epoch 22/100
386/391            0s 6ms/step -
accuracy: 0.7547 - loss: 1.3734
Epoch 22: val_loss did not improve from 1.95904
391/391            5s 7ms/step -
accuracy: 0.7548 - loss: 1.3733 - val_accuracy: 0.5934 - val_loss: 1.9975 -
learning_rate: 3.1250e-04
Epoch 23/100
391/391            0s 6ms/step -
accuracy: 0.7580 - loss: 1.3605
```

```
Epoch 23: val_loss did not improve from 1.95904
391/391            3s 7ms/step -
accuracy: 0.7580 - loss: 1.3605 - val_accuracy: 0.5890 - val_loss: 1.9882 -
learning_rate: 3.1250e-04
Epoch 24/100
390/391            0s 7ms/step -
accuracy: 0.7577 - loss: 1.3625
Epoch 24: val_loss did not improve from 1.95904

Epoch 24: ReduceLROnPlateau reducing learning rate to 0.00015624999650754035.
391/391            5s 8ms/step -
accuracy: 0.7577 - loss: 1.3625 - val_accuracy: 0.5889 - val_loss: 2.0048 -
learning_rate: 3.1250e-04
Epoch 25/100
387/391            0s 6ms/step -
accuracy: 0.7603 - loss: 1.3589
Epoch 25: val_loss improved from 1.95904 to 1.94940, saving model to
mejor_modelo.keras
391/391            3s 7ms/step -
accuracy: 0.7604 - loss: 1.3588 - val_accuracy: 0.5981 - val_loss: 1.9494 -
learning_rate: 1.5625e-04
Epoch 26/100
384/391            0s 6ms/step -
accuracy: 0.7610 - loss: 1.3528
Epoch 26: val_loss did not improve from 1.94940
391/391            3s 7ms/step -
accuracy: 0.7611 - loss: 1.3526 - val_accuracy: 0.5967 - val_loss: 1.9501 -
learning_rate: 1.5625e-04
Epoch 27/100
384/391            0s 6ms/step -
accuracy: 0.7593 - loss: 1.3550
Epoch 27: val_loss did not improve from 1.94940
391/391            3s 7ms/step -
accuracy: 0.7594 - loss: 1.3549 - val_accuracy: 0.5974 - val_loss: 1.9568 -
learning_rate: 1.5625e-04
Epoch 28/100
391/391            0s 6ms/step -
accuracy: 0.7660 - loss: 1.3372
Epoch 28: val_loss did not improve from 1.94940
391/391            5s 7ms/step -
accuracy: 0.7660 - loss: 1.3372 - val_accuracy: 0.5958 - val_loss: 1.9665 -
learning_rate: 1.5625e-04
Epoch 29/100
390/391            0s 6ms/step -
accuracy: 0.7624 - loss: 1.3456
Epoch 29: val_loss did not improve from 1.94940
391/391            3s 7ms/step -
accuracy: 0.7625 - loss: 1.3455 - val_accuracy: 0.5980 - val_loss: 1.9560 -
```

```
learning_rate: 1.5625e-04
Epoch 30/100
386/391            0s 6ms/step -
accuracy: 0.7657 - loss: 1.3431
Epoch 30: val_loss did not improve from 1.94940

Epoch 30: ReduceLROnPlateau reducing learning rate to 7.812499825377017e-05.
391/391            3s 7ms/step -
accuracy: 0.7657 - loss: 1.3430 - val_accuracy: 0.6010 - val_loss: 1.9573 -
learning_rate: 1.5625e-04
Epoch 31/100
387/391            0s 7ms/step -
accuracy: 0.7683 - loss: 1.3319
Epoch 31: val_loss improved from 1.94940 to 1.94078, saving model to
mejor_modelo.keras
391/391            6s 9ms/step -
accuracy: 0.7683 - loss: 1.3318 - val_accuracy: 0.6007 - val_loss: 1.9408 -
learning_rate: 7.8125e-05
Epoch 32/100
384/391            0s 6ms/step -
accuracy: 0.7676 - loss: 1.3278
Epoch 32: val_loss did not improve from 1.94078
391/391            4s 7ms/step -
accuracy: 0.7677 - loss: 1.3277 - val_accuracy: 0.6006 - val_loss: 1.9465 -
learning_rate: 7.8125e-05
Epoch 33/100
383/391            0s 6ms/step -
accuracy: 0.7674 - loss: 1.3291
Epoch 33: val_loss did not improve from 1.94078
391/391            3s 7ms/step -
accuracy: 0.7675 - loss: 1.3290 - val_accuracy: 0.6001 - val_loss: 1.9473 -
learning_rate: 7.8125e-05
Epoch 34/100
386/391            0s 7ms/step -
accuracy: 0.7666 - loss: 1.3287
Epoch 34: val_loss did not improve from 1.94078
391/391            3s 8ms/step -
accuracy: 0.7667 - loss: 1.3287 - val_accuracy: 0.5993 - val_loss: 1.9415 -
learning_rate: 7.8125e-05
Epoch 35/100
391/391            0s 6ms/step -
accuracy: 0.7655 - loss: 1.3351
Epoch 35: val_loss did not improve from 1.94078
391/391            3s 7ms/step -
accuracy: 0.7655 - loss: 1.3351 - val_accuracy: 0.6003 - val_loss: 1.9436 -
learning_rate: 7.8125e-05
Epoch 36/100
385/391            0s 6ms/step -
```

```
accuracy: 0.7709 - loss: 1.3304
Epoch 36: val_loss did not improve from 1.94078

Epoch 36: ReduceLROnPlateau reducing learning rate to 3.9062499126885086e-05.
391/391            5s 7ms/step -
accuracy: 0.7710 - loss: 1.3303 - val_accuracy: 0.6007 - val_loss: 1.9458 -
learning_rate: 7.8125e-05
Epoch 37/100
385/391            0s 7ms/step -
accuracy: 0.7674 - loss: 1.3260
Epoch 37: val_loss improved from 1.94078 to 1.93696, saving model to
mejor_modelo.keras
391/391            6s 9ms/step -
accuracy: 0.7675 - loss: 1.3260 - val_accuracy: 0.6024 - val_loss: 1.9370 -
learning_rate: 3.9062e-05
Epoch 38/100
383/391            0s 6ms/step -
accuracy: 0.7711 - loss: 1.3217
Epoch 38: val_loss did not improve from 1.93696
391/391            3s 7ms/step -
accuracy: 0.7711 - loss: 1.3216 - val_accuracy: 0.6017 - val_loss: 1.9420 -
learning_rate: 3.9062e-05
Epoch 39/100
391/391            0s 6ms/step -
accuracy: 0.7715 - loss: 1.3220
Epoch 39: val_loss did not improve from 1.93696
391/391            3s 7ms/step -
accuracy: 0.7715 - loss: 1.3220 - val_accuracy: 0.6011 - val_loss: 1.9418 -
learning_rate: 3.9062e-05
Epoch 40/100
386/391            0s 6ms/step -
accuracy: 0.7657 - loss: 1.3307
Epoch 40: val_loss did not improve from 1.93696
391/391            3s 7ms/step -
accuracy: 0.7658 - loss: 1.3306 - val_accuracy: 0.6017 - val_loss: 1.9409 -
learning_rate: 3.9062e-05
Epoch 41/100
390/391            0s 7ms/step -
accuracy: 0.7721 - loss: 1.3153
Epoch 41: val_loss did not improve from 1.93696
391/391            5s 7ms/step -
accuracy: 0.7721 - loss: 1.3153 - val_accuracy: 0.6026 - val_loss: 1.9376 -
learning_rate: 3.9062e-05
Epoch 42/100
385/391            0s 6ms/step -
accuracy: 0.7674 - loss: 1.3293
Epoch 42: val_loss did not improve from 1.93696
```

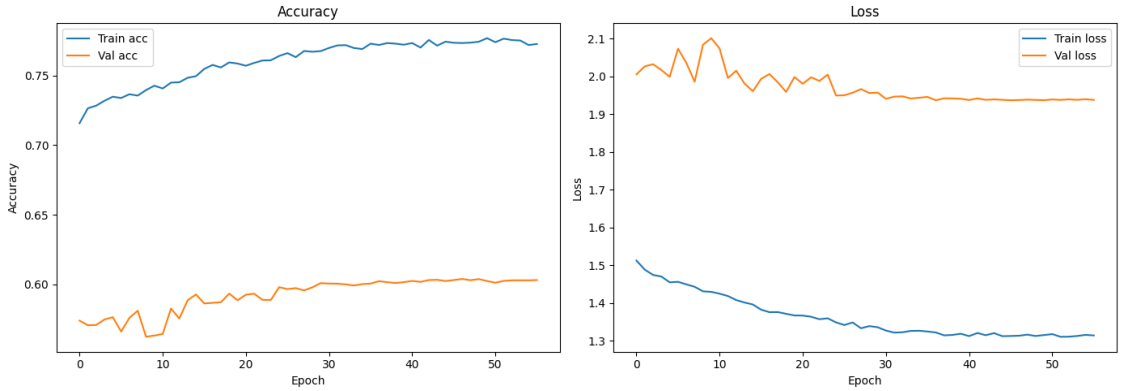
```
Epoch 42: ReduceLROnPlateau reducing learning rate to 1.9531249563442543e-05.
391/391            5s 7ms/step -
accuracy: 0.7675 - loss: 1.3292 - val_accuracy: 0.6019 - val_loss: 1.9418 -
learning_rate: 3.9062e-05
Epoch 43/100
391/391            0s 6ms/step -
accuracy: 0.7750 - loss: 1.3207
Epoch 43: val_loss did not improve from 1.93696
391/391            5s 7ms/step -
accuracy: 0.7750 - loss: 1.3207 - val_accuracy: 0.6032 - val_loss: 1.9380 -
learning_rate: 1.9531e-05
Epoch 44/100
391/391            0s 6ms/step -
accuracy: 0.7677 - loss: 1.3253
Epoch 44: val_loss did not improve from 1.93696
391/391            5s 7ms/step -
accuracy: 0.7677 - loss: 1.3252 - val_accuracy: 0.6034 - val_loss: 1.9393 -
learning_rate: 1.9531e-05
Epoch 45/100
384/391            0s 6ms/step -
accuracy: 0.7698 - loss: 1.3212
Epoch 45: val_loss did not improve from 1.93696
391/391            3s 8ms/step -
accuracy: 0.7698 - loss: 1.3210 - val_accuracy: 0.6025 - val_loss: 1.9381 -
learning_rate: 1.9531e-05
Epoch 46/100
383/391            0s 6ms/step -
accuracy: 0.7695 - loss: 1.3207
Epoch 46: val_loss improved from 1.93696 to 1.93692, saving model to
mejor_modelo.keras
391/391            3s 8ms/step -
accuracy: 0.7696 - loss: 1.3205 - val_accuracy: 0.6032 - val_loss: 1.9369 -
learning_rate: 1.9531e-05
Epoch 47/100
384/391            0s 6ms/step -
accuracy: 0.7715 - loss: 1.3203
Epoch 47: val_loss did not improve from 1.93692

Epoch 47: ReduceLROnPlateau reducing learning rate to 9.765624781721272e-06.
391/391            5s 7ms/step -
accuracy: 0.7716 - loss: 1.3201 - val_accuracy: 0.6040 - val_loss: 1.9374 -
learning_rate: 1.9531e-05
Epoch 48/100
386/391            0s 7ms/step -
accuracy: 0.7707 - loss: 1.3213
Epoch 48: val_loss did not improve from 1.93692
391/391            5s 8ms/step -
accuracy: 0.7707 - loss: 1.3212 - val_accuracy: 0.6031 - val_loss: 1.9386 -
```

```
learning_rate: 9.7656e-06
Epoch 49/100
385/391             0s 6ms/step -
accuracy: 0.7701 - loss: 1.3183
Epoch 49: val_loss did not improve from 1.93692
391/391             3s 7ms/step -
accuracy: 0.7702 - loss: 1.3182 - val_accuracy: 0.6039 - val_loss: 1.9377 -
learning_rate: 9.7656e-06
Epoch 50/100
383/391             0s 6ms/step -
accuracy: 0.7736 - loss: 1.3233
Epoch 50: val_loss did not improve from 1.93692
391/391             5s 7ms/step -
accuracy: 0.7737 - loss: 1.3231 - val_accuracy: 0.6025 - val_loss: 1.9370 -
learning_rate: 9.7656e-06
Epoch 51/100
387/391             0s 7ms/step -
accuracy: 0.7671 - loss: 1.3319
Epoch 51: val_loss did not improve from 1.93692
391/391             5s 8ms/step -
accuracy: 0.7672 - loss: 1.3317 - val_accuracy: 0.6013 - val_loss: 1.9390 -
learning_rate: 9.7656e-06
Epoch 52/100
387/391             0s 6ms/step -
accuracy: 0.7744 - loss: 1.3139
Epoch 52: val_loss did not improve from 1.93692

Epoch 52: ReduceLROnPlateau reducing learning rate to 4.882812390860636e-06.
391/391             5s 7ms/step -
accuracy: 0.7744 - loss: 1.3139 - val_accuracy: 0.6026 - val_loss: 1.9379 -
learning_rate: 9.7656e-06
Epoch 53/100
387/391             0s 6ms/step -
accuracy: 0.7713 - loss: 1.3202
Epoch 53: val_loss did not improve from 1.93692
391/391             3s 7ms/step -
accuracy: 0.7713 - loss: 1.3200 - val_accuracy: 0.6030 - val_loss: 1.9393 -
learning_rate: 4.8828e-06
Epoch 54/100
389/391             0s 7ms/step -
accuracy: 0.7694 - loss: 1.3260
Epoch 54: val_loss did not improve from 1.93692
391/391             3s 8ms/step -
accuracy: 0.7694 - loss: 1.3259 - val_accuracy: 0.6030 - val_loss: 1.9380 -
learning_rate: 4.8828e-06
Epoch 55/100
390/391             0s 6ms/step -
accuracy: 0.7711 - loss: 1.3191
```

```
Epoch 55: val_loss did not improve from 1.93692
391/391             5s 7ms/step -
accuracy: 0.7711 - loss: 1.3191 - val_accuracy: 0.6030 - val_loss: 1.9396 -
learning_rate: 4.8828e-06
Epoch 56/100
386/391             0s 6ms/step -
accuracy: 0.7692 - loss: 1.3198
Epoch 56: val_loss did not improve from 1.93692
391/391             3s 7ms/step -
accuracy: 0.7693 - loss: 1.3197 - val_accuracy: 0.6032 - val_loss: 1.9379 -
learning_rate: 4.8828e-06
Epoch 56: early stopping
Restoring model weights from the end of the best epoch: 46.
```

8 Paso 9: Visualizar el accuracy y loss

[33]:
```python
# Graficar precisión y pérdida
plt.figure(figsize=(14, 5))

# Accuracy
plt.subplot(1, 2, 1)
plt.plot(historial.history['accuracy'], label='Train acc')
plt.plot(historial.history['val_accuracy'], label='Val acc')
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# Loss
plt.subplot(1, 2, 2)
plt.plot(historial.history['loss'], label='Train loss')
plt.plot(historial.history['val_loss'], label='Val loss')
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()
```

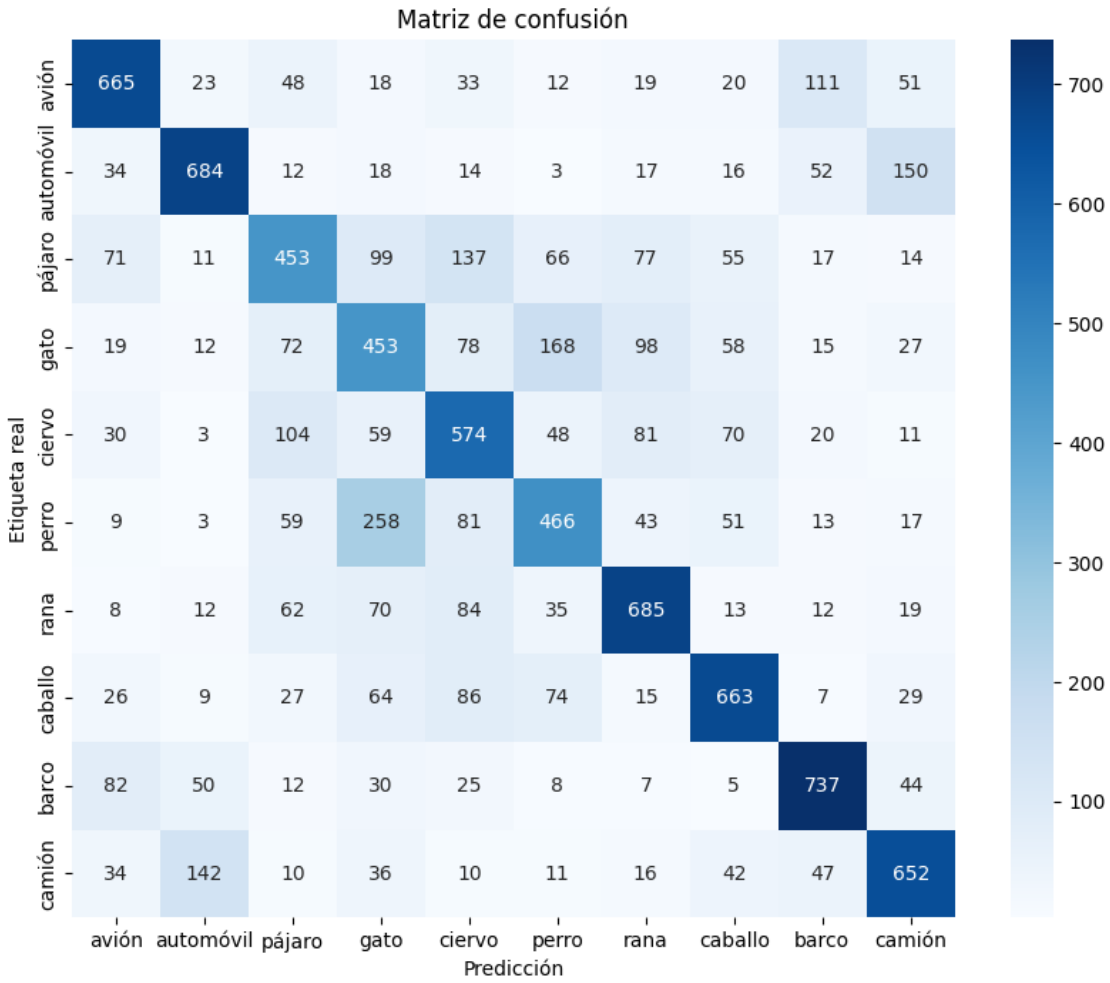


9 Paso 10: Evaluación del modelo y reporte por clase

[34]:
```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Predecir en el set de test
y_pred_probs = modelo.predict(x_test_flat)
y_pred = np.argmax(y_pred_probs, axis=1)
y_true = np.argmax(y_test_cat, axis=1)

# Mostrar clasificación por clase
print(classification_report(y_true, y_pred, target_names=class_names))

# Matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred)

plt.figure(figsize=(10, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names,
            yticklabels=class_names)
plt.xlabel('Predicción')
plt.ylabel('Etiqueta real')
plt.title('Matriz de confusión')
plt.show()
```

```
313/313             1s 2ms/step
              precision    recall  f1-score   support

       avión       0.68      0.67      0.67      1000
  automóvil        0.72      0.68      0.70      1000
     pájaro        0.53      0.45      0.49      1000
       gato        0.41      0.45      0.43      1000
```

ciervo	0.51	0.57	0.54	1000
perro	0.52	0.47	0.49	1000
rana	0.65	0.69	0.67	1000
caballo	0.67	0.66	0.67	1000
barco	0.71	0.74	0.73	1000
camión	0.64	0.65	0.65	1000
accuracy			0.60	10000
macro avg	0.60	0.60	0.60	10000
weighted avg	0.60	0.60	0.60	10000



10 Paso 11: Probar el modelo con imágenes individuales

[35]:
```
# Mostrar 40 imágenes aleatorias con predicciones
num_images = 40
indices = np.random.choice(len(x_test_flat), num_images, replace=False)

plt.figure(figsize=(20, 10))
for i, idx in enumerate(indices):
    img = x_test[idx]
    true_label = class_names[y_true[idx]]
    pred_label = class_names[y_pred[idx]]

    plt.subplot(5, 8, i + 1)
    plt.imshow(img)
    plt.title(f"R: {true_label}\nP: {pred_label}", fontsize=8)
    plt.axis("off")

plt.tight_layout()
plt.show()
```