Clasificación aprendizaje automático

dataset CIFAR-10

Profesor:
David Campoy Miñarro

Clustering

Vamos a utilizar el dataset CIFAR-10 para estudiar el clustering.

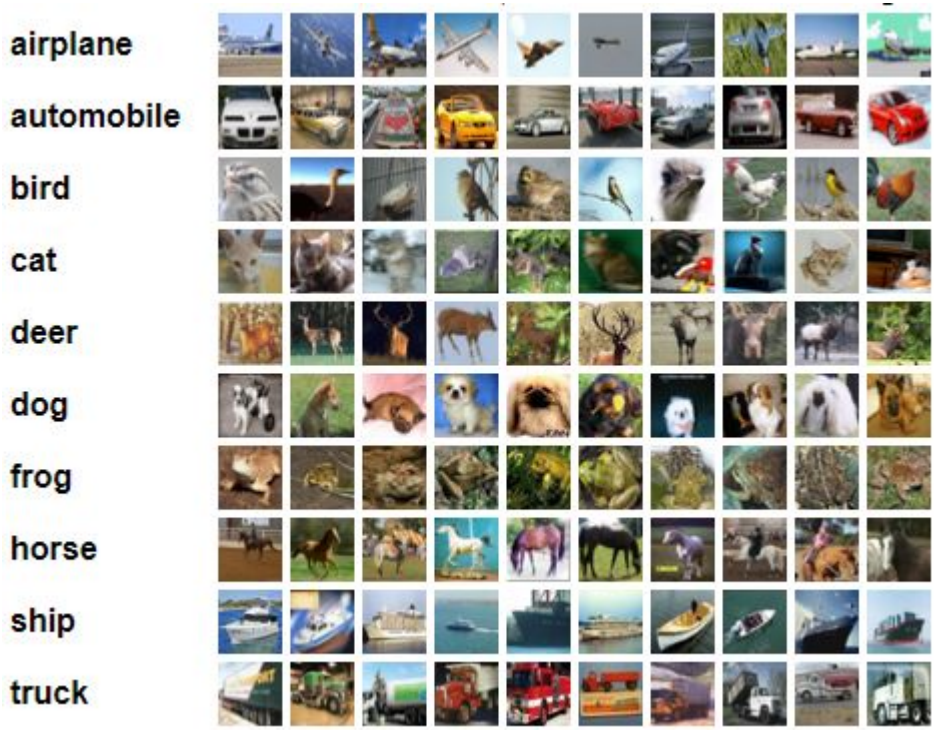
https://www.cs.toronto.edu/~kriz/cifar.html

¿Recuerdas lo que realiza el clustering? ¿es aprendizaje supervisado?

Modifica el código para realizar distintas pruebas de clustering:

- Coches y camiones
- Coches, ranas y camiones.
- Coches y aviones.
- … prueba con varias categorías.

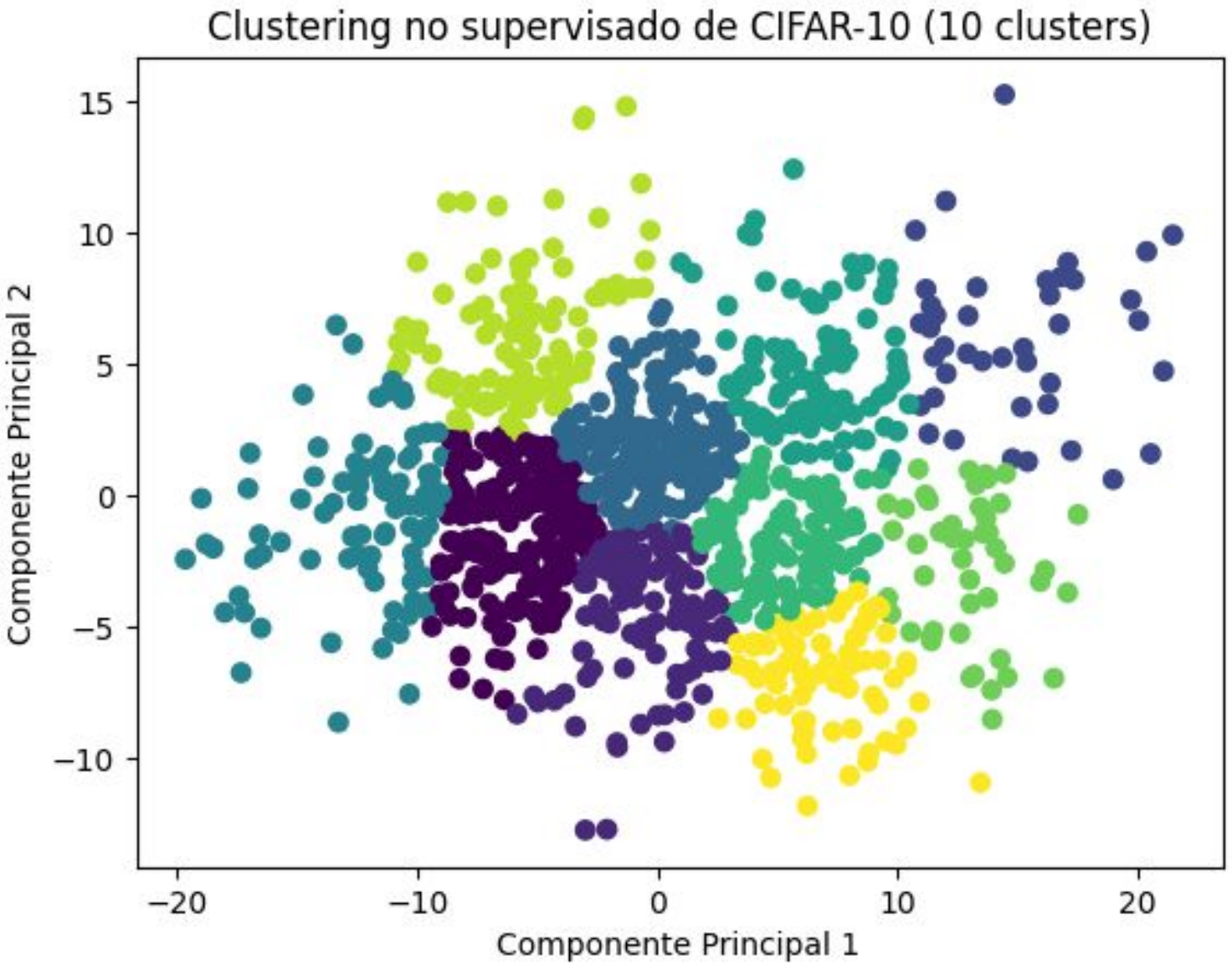
Documenta los resultados.



```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from keras.datasets import cifar10

# Cargar el conjunto de datos CIFAR-10
(train_images, train_labels), (_, _) = cifar10.load_data()

# Filtrar las imágenes de las categorías de "coches" (label 1) y
"aviones" (label 0)
car_indices = np.where((train_labels == 1).squeeze())[0][:5000]
airplane_indices = np.where((train_labels == 0).squeeze())[0][:5000]

selected_indices = np.concatenate((car_indices, airplane_indices))
selected_images = train_images[selected_indices]

# Redimensionar y normalizar las imágenes para el clustering
selected_images = selected_images.reshape(selected_images.shape[0], -1)
selected_images = selected_images / 255.0

# Reducir la dimensionalidad a 2 componentes con PCA
pca = PCA(n_components=2)
selected_images_pca = pca.fit_transform(selected_images)

# Realizar el clustering con K-Means
kmeans = KMeans(n_clusters=2, n_init=10)
kmeans.fit(selected_images_pca)
clusters = kmeans.predict(selected_images_pca)

# Visualización en 2D
plt.scatter(selected_images_pca[:, 0], selected_images_pca[:, 1],
c=clusters, cmap='viridis')
plt.title('Clustering de las categorías "coches" y "aviones" de
CIFAR-10')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.show()
```

Versión 2

```
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from keras.datasets import cifar10

# Cargar el conjunto de datos CIFAR-10
(train_images, train_labels), (_, _) = cifar10.load_data()

# Filtrar 100 imágenes de cada categoría
category_indices = [np.where(train_labels == i)[0][:100] for i in
range(10)]
selected_indices = np.concatenate(category_indices)
selected_images = train_images[selected_indices]

# Redimensionar y normalizar las imágenes para el clustering
selected_images = selected_images.reshape(selected_images.shape[0], -1)
selected_images = selected_images / 255.0

# Reducir la dimensionalidad a 2 componentes con PCA
pca = PCA(n_components=2)
selected_images_pca = pca.fit_transform(selected_images)

# Realizar el clustering con K-Means (10 clusters)
kmeans = KMeans(n_clusters=10, n_init=10)
kmeans.fit(selected_images_pca)
clusters = kmeans.predict(selected_images_pca)

# Visualización en 2D
plt.scatter(selected_images_pca[:, 0], selected_images_pca[:, 1],
c=clusters, cmap='viridis')
plt.title('Clustering no supervisado de CIFAR-10 (10 clusters)')
plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.show()
```



Clasificación: Aprendizaje automático supervisado

Ahora vamos a clasificar las imágenes de CIFAR-10 pero con aprendizaje supervisado.

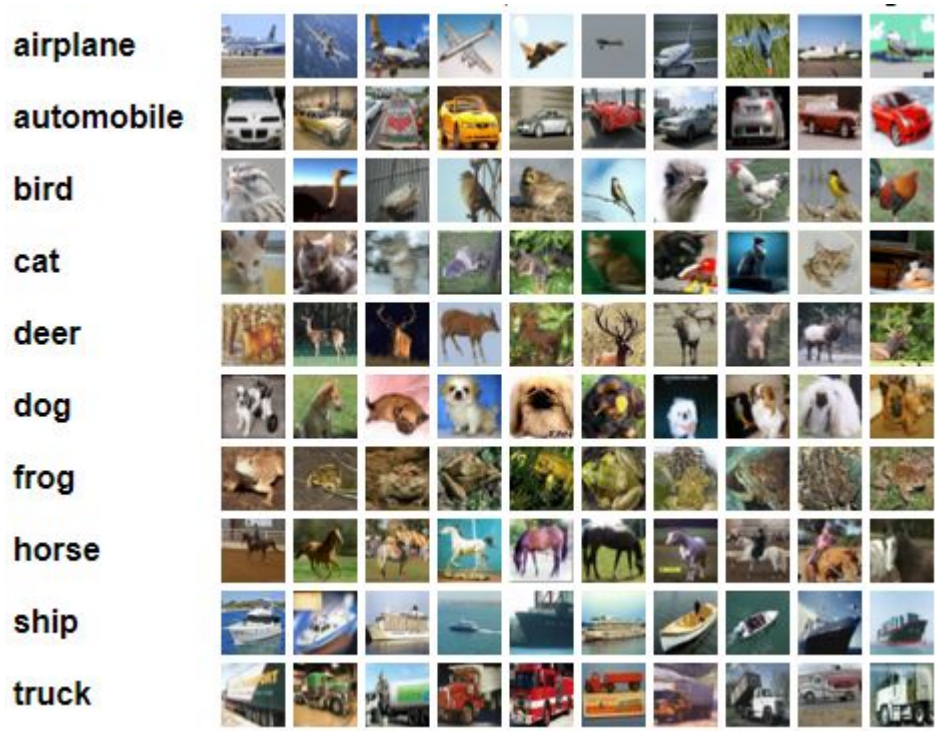
Para este ejemplo se ha utiliza un modelo de árboles de decisión.

¿Consideras que los resultados son buenos? ¿Qué harías si no lo son?

¿Sabes interpretar la matriz de confusión?

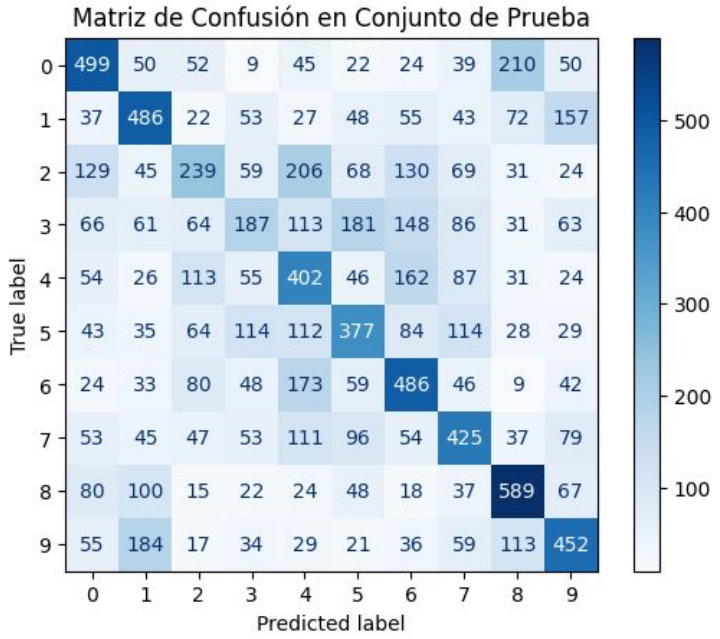
Documenta los resultados.

https://www.cs.toronto.edu/~kriz/cifar.html



```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from keras.datasets import cifar10
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# Cargar el conjunto de datos CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Tomar solo una fracción del conjunto de datos para reducir la memoria
frac = 0.2
x_train, _, y_train, _ = train_test_split(x_train, y_train,
test_size=1-frac, random_state=42)

# Aplanar las imágenes y normalizar los valores de píxeles a un rango de
0 a 1
x_train = x_train.reshape((x_train.shape[0], -1)) / 255.0
x_test = x_test.reshape((x_test.shape[0], -1)) / 255.0

# Convertir las etiquetas a un formato unidimensional
y_train = np.ravel(y_train)
y_test = np.ravel(y_test)

# Dividir el conjunto de entrenamiento en entrenamiento y validación
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.2, random_state=42)

# Crear y entrenar un clasificador de bosques aleatorios
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(x_train, y_train)

# Realizar predicciones en el conjunto de prueba
y_pred_test = clf.predict(x_test.reshape((x_test.shape[0], -1)))
```

```
# Evaluar la precisión en el conjunto de prueba
accuracy_test = accuracy_score(y_test, y_pred_test)

# Mostrar métricas adicionales
print("\nInforme de clasificación en el conjunto de prueba:")
print(classification_report(y_test, y_pred_test))

# Crear la matriz de confusión
cm = confusion_matrix(y_test, y_pred_test)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(y_test))
disp.plot(cmap=plt.cm.Blues, values_format='d')
plt.title('Matriz de Confusión en Conjunto de Prueba')
plt.show()
```



Clasificación: Aprendizaje automático NN

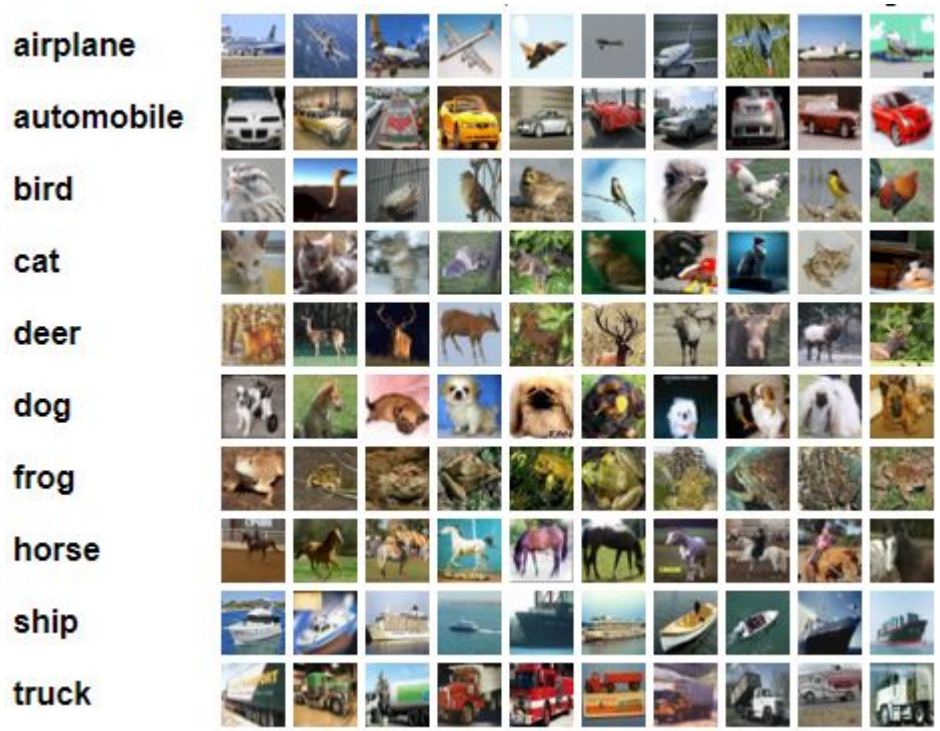
Ahora vamos a clasificar las imágenes de CIFAR-10 pero con aprendizaje supervisado aplicando una red neuronal sencilla.

¿Consideras que los resultados son buenos?

Compara la matriz de confusión con la anterior.

Documenta los resultados.

https://www.cs.toronto.edu/~kriz/cifar.html



```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# Cargar el conjunto de datos CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Tomar solo una fracción del conjunto de datos para reducir la memoria
frac = 0.2
x_train, _, y_train, _ = train_test_split(x_train, y_train,
test_size=1-frac, random_state=42)

# Aplanar las imágenes y normalizar los valores de píxeles a un rango de
0 a 1
x_train = x_train.reshape((x_train.shape[0], -1)) / 255.0
x_test = x_test.reshape((x_test.shape[0], -1)) / 255.0

# Convertir las etiquetas a un formato unidimensional
y_train = np.ravel(y_train)
y_test = np.ravel(y_test)

# Dividir el conjunto de entrenamiento en entrenamiento y validación
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.2, random_state=42)
```

```
# Construir una red neuronal simple
model = Sequential()
model.add(Dense(128, input_dim=x_train.shape[1], activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(x_train, y_train, epochs=50, batch_size=32,
validation_data=(x_val, y_val), verbose=1)

y_pred_test_probs = model.predict(x_test)
y_pred_test = np.argmax(y_pred_test_probs, axis=1)

accuracy_test = accuracy_score(y_test, y_pred_test)

# Mostrar métricas adicionales
print("\nInforme de clasificación en el conjunto de prueba:")
print(classification_report(y_test, y_pred_test))

# Crear la matriz de confusión
cm = confusion_matrix(y_test, y_pred_test)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(y_test))
disp.plot(cmap=plt.cm.Blues, values_format='d')
plt.title('Matriz de Confusión en Conjunto de Prueba')
plt.show()

# Mostrar la curva de aprendizaje
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Curva de Aprendizaje')
plt.show()
```

Clasificación: Aprendizaje automático CNN

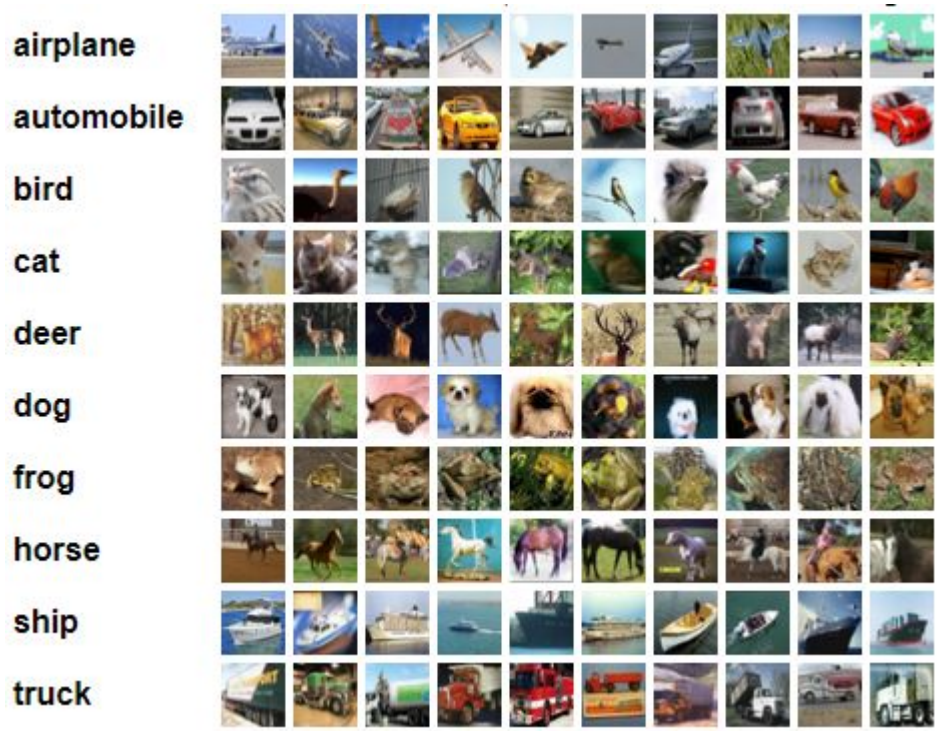
Ahora vamos a clasificar las imágenes de CIFAR-10 pero con aprendizaje supervisado aplicando una red neuronal convolucional.

¿Consideras que los resultados son mejores?

Compara la matriz de confusión con las anterior.

Documenta los resultados.

https://www.cs.toronto.edu/~kriz/cifar.html



```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from keras.utils import to_categorical
from keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# Cargar el conjunto de datos CIFAR-10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Tomar solo una fracción del conjunto de datos para reducir la memoria
frac = 0.2
x_train, _, y_train, _ = train_test_split(x_train, y_train,
test_size=1-frac, random_state=42)

# Normalizar los valores de píxeles a un rango de 0 a 1
x_train = x_train / 255.0
x_test = x_test / 255.0

# Convertir las etiquetas a un formato unidimensional
y_train = np.ravel(y_train)
y_test = np.ravel(y_test)

# Convertir las etiquetas a un formato categórico (one-hot encoding)
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

# Dividir el conjunto de entrenamiento en entrenamiento y validación
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
test_size=0.2, random_state=42)
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(32, 32, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

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

# Utilizar ImageDataGenerator para aumentar los datos y reducir el tamaño del lote
datagen = ImageDataGenerator(rotation_range=20, width_shift_range=0.2,
height_shift_range=0.2, horizontal_flip=True)
datagen.fit(x_train)

# Entrenar la red neuronal convolucional con el generador de datos
history = model.fit(datagen.flow(x_train, y_train, batch_size=32),
steps_per_epoch=len(x_train) // 32, epochs=10, validation_data=(x_val, y_val),
verbose=1)

# Realizar predicciones en el conjunto de prueba
y_pred_test_probs = model.predict(x_test)
y_pred_test = np.argmax(y_pred_test_probs, axis=1)

# Evaluar la precisión en el conjunto de prueba
accuracy_test = accuracy_score(np.argmax(y_test, axis=1), y_pred_test)

# Mostrar métricas adicionales
print("\nInforme de clasificación en el conjunto de prueba:")
print(classification_report(np.argmax(y_test, axis=1), y_pred_test))

# Crear la matriz de confusión
cm = confusion_matrix(np.argmax(y_test, axis=1), y_pred_test)

# Mostrar la matriz de confusión
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=np.unique(np.argmax(y_test, axis=1)))
disp.plot(cmap=plt.cm.Blues, values_format='d')
plt.title('Matriz de Confusión en Conjunto de Prueba')
plt.show()

# Mostrar la curva de aprendizaje
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Curva de Aprendizaje')
plt.show()
```

¿Qué has aprendido?

Aprendizaje automático no supervisado

- Clustering con imágenes

Aprendizaje automático supervisado

- Categorización con una red neuronal simple
- Categorización con una red neuronal convolucional
- Categorización con un árbol de decisión.

Practicar con la matriz de confusión.
Optimización de hiperparámetros.

"Algunas personas llaman a esto inteligencia artificial, pero la realidad es que esta tecnología nos mejorará. Entonces, en lugar de inteligencia artificial, creo que aumentaremos nuestra inteligencia."

Ginni Rometty, empresaria estadounidense, actual presidenta y CEO de la compañía IBM