image-object-recognition-version1

May 24, 2024

[3]:
```
import os
import pickle
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout,␣
 ↪MaxPooling2D
from tensorflow.keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt

# Function to load a batch of CIFAR-10 data
def load_cifar10_batch(file):
    with open(file, 'rb') as fo:
        dict = pickle.load(fo, encoding='bytes')
    images = dict[b'data']
    labels = dict[b'labels']
    images = images.reshape(len(images), 3, 32, 32).transpose(0, 2, 3, 1).
 ↪astype("float")
    return images, labels

# Function to load all CIFAR-10 data
def load_cifar10_data(data_dir):
    train_images = []
    train_labels = []
    for i in range(1, 6):
        file = os.path.join(data_dir, f'data_batch_{i}')
        images, labels = load_cifar10_batch(file)
        train_images.append(images)
        train_labels.append(labels)
    train_images = np.concatenate(train_images)
    train_labels = np.concatenate(train_labels)

    test_file = os.path.join(data_dir, 'test_batch')
    test_images, test_labels = load_cifar10_batch(test_file)

    return (train_images, train_labels), (test_images, test_labels)
```

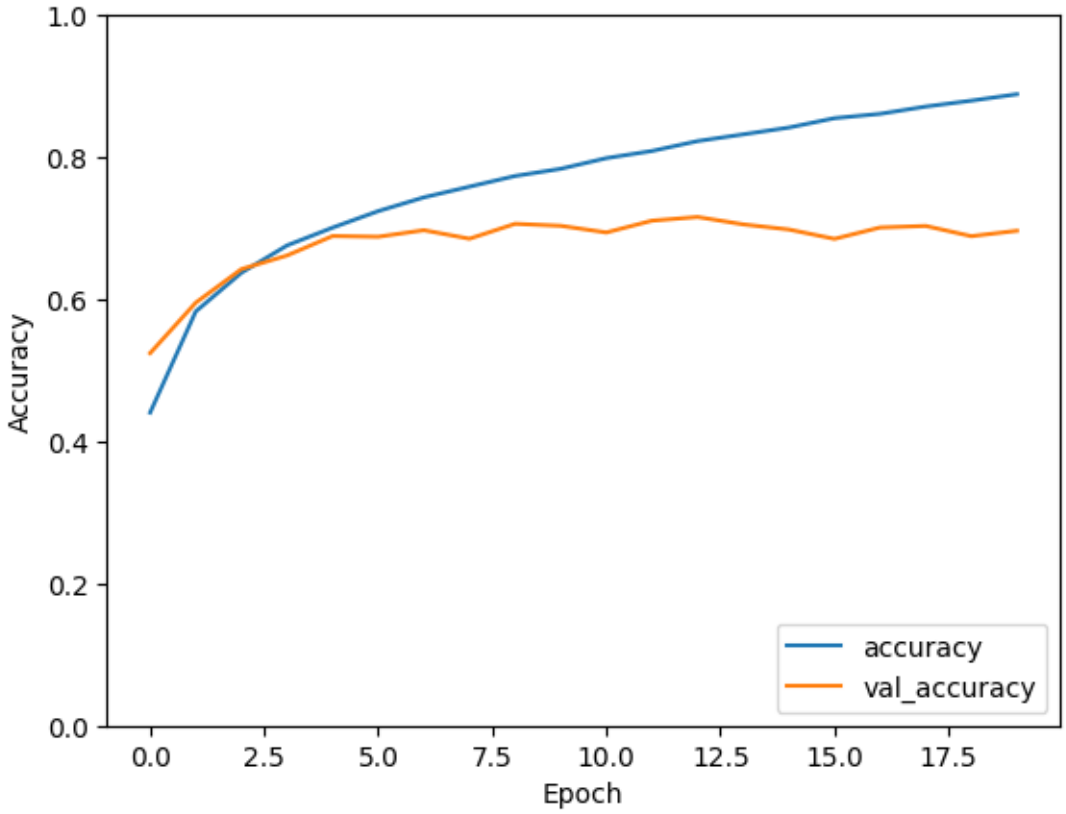
```
# Load CIFAR-10 data
data_dir = '/Users/williambolton/Downloads/cifar-10-batches-py'
(train_images, train_labels), (test_images, test_labels) =␣
 ↪load_cifar10_data(data_dir)

# Normalize the data
train_images, test_images = train_images / 255.0, test_images / 255.0

# Convert labels to numpy arrays
train_labels = np.array(train_labels)
test_labels = np.array(test_labels)

# Display first 5 images from the training dataset
plt.figure(figsize=(10, 2))
for i in range(5):
    plt.subplot(1, 5, i + 1)
    plt.imshow(train_images[i])
    plt.title(f"Label: {train_labels[i]}")
    plt.axis('off')
plt.show()

# Ensures representative splitting
def representative_split(X, y, test_size=0.2, random_state=42):
    np.random.seed(random_state)
    indices = np.arange(X.shape[0])
    np.random.shuffle(indices)
    split = int(X.shape[0] * (1 - test_size))
    train_idx, test_idx = indices[:split], indices[split:]
    return X[train_idx], X[test_idx], y[train_idx], y[test_idx]

# Partition the training data to create a validation set using representative␣
 ↪split
train_images, validation_images, train_labels, validation_labels =␣
 ↪representative_split(train_images, train_labels, test_size=0.1)

# Data specifics (metadata)
print(f"Training samples: {train_images.shape[0]}")
print(f"Validation samples: {validation_images.shape[0]}")
print(f"Test samples: {test_images.shape[0]}")

# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
```

```
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=20,
                    validation_data=(validation_images, validation_labels))

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")

# Plot the training and validation accuracy
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0, 1])
plt.legend(loc='lower right')
plt.show()
```


```
Training samples: 45000
Validation samples: 5000
Test samples: 10000
Epoch 1/20
1407/1407             38s 26ms/step -
accuracy: 0.3521 - loss: 1.7547 - val_accuracy: 0.5242 - val_loss: 1.3429
Epoch 2/20
1407/1407             38s 27ms/step -
accuracy: 0.5689 - loss: 1.2066 - val_accuracy: 0.5952 - val_loss: 1.1521
```

```
Epoch 3/20
1407/1407            38s 27ms/step -
accuracy: 0.6270 - loss: 1.0498 - val_accuracy: 0.6424 - val_loss: 1.0163
Epoch 4/20
1407/1407            38s 27ms/step -
accuracy: 0.6705 - loss: 0.9373 - val_accuracy: 0.6616 - val_loss: 0.9558
Epoch 5/20
1407/1407            38s 27ms/step -
accuracy: 0.7058 - loss: 0.8470 - val_accuracy: 0.6890 - val_loss: 0.9084
Epoch 6/20
1407/1407            36s 25ms/step -
accuracy: 0.7270 - loss: 0.7797 - val_accuracy: 0.6880 - val_loss: 0.9114
Epoch 7/20
1407/1407            37s 26ms/step -
accuracy: 0.7470 - loss: 0.7295 - val_accuracy: 0.6972 - val_loss: 0.8698
Epoch 8/20
1407/1407            38s 27ms/step -
accuracy: 0.7612 - loss: 0.6862 - val_accuracy: 0.6852 - val_loss: 0.9369
Epoch 9/20
1407/1407            38s 27ms/step -
accuracy: 0.7774 - loss: 0.6412 - val_accuracy: 0.7062 - val_loss: 0.8824
Epoch 10/20
1407/1407            38s 27ms/step -
accuracy: 0.7874 - loss: 0.6001 - val_accuracy: 0.7034 - val_loss: 0.8983
Epoch 11/20
1407/1407            37s 26ms/step -
accuracy: 0.8041 - loss: 0.5547 - val_accuracy: 0.6940 - val_loss: 0.9442
Epoch 12/20
1407/1407            36s 25ms/step -
accuracy: 0.8135 - loss: 0.5256 - val_accuracy: 0.7106 - val_loss: 0.9187
Epoch 13/20
1407/1407            38s 27ms/step -
accuracy: 0.8309 - loss: 0.4776 - val_accuracy: 0.7158 - val_loss: 0.9188
Epoch 14/20
1407/1407            40s 28ms/step -
accuracy: 0.8346 - loss: 0.4612 - val_accuracy: 0.7054 - val_loss: 1.0050
Epoch 15/20
1407/1407            38s 27ms/step -
accuracy: 0.8484 - loss: 0.4287 - val_accuracy: 0.6982 - val_loss: 1.0048
Epoch 16/20
1407/1407            37s 26ms/step -
accuracy: 0.8647 - loss: 0.3873 - val_accuracy: 0.6850 - val_loss: 1.0875
Epoch 17/20
1407/1407            39s 27ms/step -
accuracy: 0.8688 - loss: 0.3638 - val_accuracy: 0.7010 - val_loss: 1.1354
Epoch 18/20
1407/1407            38s 27ms/step -
accuracy: 0.8806 - loss: 0.3365 - val_accuracy: 0.7032 - val_loss: 1.1065
```

```
Epoch 19/20
1407/1407              38s 27ms/step -
accuracy: 0.8860 - loss: 0.3112 - val_accuracy: 0.6888 - val_loss: 1.2218
Epoch 20/20
1407/1407              39s 27ms/step -
accuracy: 0.8999 - loss: 0.2805 - val_accuracy: 0.6964 - val_loss: 1.2318
313/313 - 3s - 8ms/step - accuracy: 0.6987 - loss: 1.1908
Test accuracy: 0.6987000107765198
```



[ ]: