dl-ex-6

May 1, 2023

[12]:
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
```

[13]:
```
# Load the CIFAR-10 dataset
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

[14]:
```
# Normalize the pixel values
X_train = X_train / 255.0
X_test = X_test / 255.0
```

[15]:
```
# Convert the labels to one-hot encoded vectors
y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
```

[16]:
```
# Define the CNN model with different activation functions
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu',␣
 ↪input_shape=(32, 32, 3)),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
    tf.keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='sigmoid'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
    tf.keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='tanh'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units=256, activation='relu'),
    tf.keras.layers.Dense(units=10, activation='softmax')])
```

[17]:
```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[18]:
```
# Train the model
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test,␣
 ↪y_test))
```

```
Epoch 1/10
1563/1563 [==============================] - 99s 62ms/step - loss: 1.7032 -
```

```
accuracy: 0.3759 - val_loss: 1.4279 - val_accuracy: 0.4778
Epoch 2/10
1563/1563 [==============================] - 97s 62ms/step - loss: 1.3110 -
accuracy: 0.5293 - val_loss: 1.2034 - val_accuracy: 0.5704
Epoch 3/10
1563/1563 [==============================] - 94s 60ms/step - loss: 1.1495 -
accuracy: 0.5916 - val_loss: 1.1504 - val_accuracy: 0.5931
Epoch 4/10
1563/1563 [==============================] - 92s 59ms/step - loss: 1.0157 -
accuracy: 0.6406 - val_loss: 1.0547 - val_accuracy: 0.6290
Epoch 5/10
1563/1563 [==============================] - 93s 59ms/step - loss: 0.9076 -
accuracy: 0.6797 - val_loss: 0.9732 - val_accuracy: 0.6563
Epoch 6/10
1563/1563 [==============================] - 94s 60ms/step - loss: 0.8136 -
accuracy: 0.7149 - val_loss: 0.9637 - val_accuracy: 0.6690
Epoch 7/10
1563/1563 [==============================] - 92s 59ms/step - loss: 0.7302 -
accuracy: 0.7442 - val_loss: 0.9489 - val_accuracy: 0.6756
Epoch 8/10
1563/1563 [==============================] - 99s 63ms/step - loss: 0.6504 -
accuracy: 0.7720 - val_loss: 0.9490 - val_accuracy: 0.6815
Epoch 9/10
1563/1563 [==============================] - 97s 62ms/step - loss: 0.5826 -
accuracy: 0.7956 - val_loss: 0.9572 - val_accuracy: 0.6952
Epoch 10/10
1563/1563 [==============================] - 99s 64ms/step - loss: 0.5128 -
accuracy: 0.8190 - val_loss: 0.9552 - val_accuracy: 0.7023
```

[18]:
```
<keras.callbacks.History at 0x7f6c28fe1e70>
```

[19]:
```
# Extract the activations of the convolutional layers
activation_model = tf.keras.models.Model(inputs=model.input, outputs=[layer.
↪output for layer in model.layers if isinstance(layer, tf.keras.layers.
↪Conv2D)])
```

[20]:
```
# Choose an image from the test set to visualize
image = X_test[0]
image = np.expand_dims(image, axis=0)
```

[21]:
```
# Generate heatmaps for each activation function
activations = activation_model.predict(image)
relu_activation = activations[0]
sigmoid_activation = activations[1]
tanh_activation = activations[2]
```

```
1/1 [==============================] - 0s 103ms/step
```

[22]:
```
# Plot the heatmaps for each activation function
fig, axs = plt.subplots(nrows=1, ncols=3, figsize=(10, 5))
axs[0].imshow(relu_activation[0,:,:,0], cmap='jet')
axs[0].set_title('ReLU activation')
axs[1].imshow(sigmoid_activation[0,:,:,0], cmap='jet')
axs[1].set_title('Sigmoid activation')
axs[2].imshow(tanh_activation[0,:,:,0], cmap='jet')
axs[2].set_title('Tanh activation')
plt.show()
```