dl-ex-8

May 23, 2023

[2]:
```
import numpy as np
from tensorflow import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten,␣
 ↪Dense, concatenate

# Define the architecture of the model
image_height, image_width, image_channels = 32, 32, 3
numeric_features = 5
num_classes = 10

# Input for the image data
image_input = Input(shape=(image_height, image_width, image_channels))
conv1 = Conv2D(32, kernel_size=(3, 3), activation='relu')(image_input)
maxpool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
conv2 = Conv2D(64, kernel_size=(3, 3), activation='relu')(maxpool1)
maxpool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
flatten1 = Flatten()(maxpool2)

# Input for the numerical data
numeric_input = Input(shape=(numeric_features,))

# Concatenate the flattened image and numeric input
concat = concatenate([flatten1, numeric_input])

# Output for image classification task
output1 = Dense(num_classes, activation='softmax')(concat)

# Output for regression task
output2 = Dense(1, activation='linear')(concat)

# Define the model
model = Model(inputs=[image_input, numeric_input], outputs=[output1, output2])

# Compile the model
model.compile(optimizer='adam',
              loss=['categorical_crossentropy', 'mean_squared_error'],
```

```
              metrics=['accuracy'])

# Prepare your data
# Example dataset for demonstration purposes
num_samples = 100
image_data = np.random.rand(num_samples, image_height, image_width,␣
 ↪image_channels)
numeric_data = np.random.rand(num_samples, numeric_features)
image_labels = keras.utils.to_categorical(np.random.randint(num_classes,␣
 ↪size=(num_samples, 1)), num_classes=num_classes)
numeric_labels = np.random.rand(num_samples, 1)

# Train the model
model.fit([image_data, numeric_data], [image_labels, numeric_labels],␣
 ↪epochs=10, batch_size=32)
```

```
Epoch 1/10
4/4 [==============================] - 1s 32ms/step - loss: 2.9960 -
dense_2_loss: 2.3841 - dense_3_loss: 0.6119 - dense_2_accuracy: 0.0900 -
dense_3_accuracy: 0.0000e+00
Epoch 2/10
4/4 [==============================] - 0s 32ms/step - loss: 2.5813 -
dense_2_loss: 2.3094 - dense_3_loss: 0.2720 - dense_2_accuracy: 0.1400 -
dense_3_accuracy: 0.0000e+00
Epoch 3/10
4/4 [==============================] - 0s 33ms/step - loss: 2.3748 -
dense_2_loss: 2.2942 - dense_3_loss: 0.0806 - dense_2_accuracy: 0.1300 -
dense_3_accuracy: 0.0000e+00
Epoch 4/10
4/4 [==============================] - 0s 32ms/step - loss: 2.4101 -
dense_2_loss: 2.2779 - dense_3_loss: 0.1323 - dense_2_accuracy: 0.1500 -
dense_3_accuracy: 0.0000e+00
Epoch 5/10
4/4 [==============================] - 0s 33ms/step - loss: 2.3831 -
dense_2_loss: 2.2735 - dense_3_loss: 0.1095 - dense_2_accuracy: 0.1200 -
dense_3_accuracy: 0.0000e+00
Epoch 6/10
4/4 [==============================] - 0s 32ms/step - loss: 2.3579 -
dense_2_loss: 2.2762 - dense_3_loss: 0.0817 - dense_2_accuracy: 0.1400 -
dense_3_accuracy: 0.0000e+00
Epoch 7/10
4/4 [==============================] - 0s 40ms/step - loss: 2.3643 -
dense_2_loss: 2.2688 - dense_3_loss: 0.0955 - dense_2_accuracy: 0.1400 -
dense_3_accuracy: 0.0000e+00
Epoch 8/10
4/4 [==============================] - 0s 31ms/step - loss: 2.3398 -
dense_2_loss: 2.2614 - dense_3_loss: 0.0784 - dense_2_accuracy: 0.1600 -
```

```
dense_3_accuracy: 0.0000e+00
Epoch 9/10
4/4 [==============================] - 0s 32ms/step - loss: 2.3321 -
dense_2_loss: 2.2537 - dense_3_loss: 0.0784 - dense_2_accuracy: 0.1600 -
dense_3_accuracy: 0.0000e+00
Epoch 10/10
4/4 [==============================] - 0s 32ms/step - loss: 2.3253 -
dense_2_loss: 2.2460 - dense_3_loss: 0.0793 - dense_2_accuracy: 0.1900 -
dense_3_accuracy: 0.0000e+00
```

[2]: <keras.callbacks.History at 0x7fab59be5f90>