Computer science department

2nd A.I MASTER

Lab N°=1 DL (CNN development)

Problem specification

Let us consider the code of our first CNN (developed in tensorflow) shown in page 2, the program classifies the brain tumors in MRI images. In this dataset, we have about 7000 MRI images with four classes: glioma tumors, pituitary tumors, meningioma tumors, and healthy images.

Questions

1) Execute the python code of the file (or page 2) and add evaluate the test set accuracy rate while modifying some parameters such as the size of the training set/test set, the size of the input images (if the RAM is weak, you should reduce the image height/width)

2) Change the code by replacing the RELU activation function by the ELU function and evaluate this change on the test performance

3) Add a function that search the best values for the dense layer ({64,128, 256}) that gives the highest accuracy on the test set, also evaluate the overfitting sensitivity by computing |TraingError-TestError| for each choice.

4) Using the function **classification_report ()** of scikit-learn, display the detailed performance of the best model on the test set; give your own comments about the results.

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
import PIL
from sklearn.metrics import confusion_matrix , classification_report
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
#dataset preparation
nl=112
nc=112
nch=3
data_dir= '.\\braincancerdataset'
import pathlib
data_dir = pathlib.Path(data_dir)
data_dir
list(data_dir.glob('*/*.jpg'))[:5]
image_count = len(list(data_dir.glob('*/*.jpg')))
print("dataset size",image_count)
imageglioma_count = len(list(data_dir.glob('glioma/*.jpg')))
print("glioma number",imageglioma_count)
flowers_images_dict = {
'glioma': list(data_dir.glob('glioma/*')),
'healthy': list(data_dir.glob('healthy/*')),
'meningioma': list(data_dir.glob('meningioma/*')),
'pituitary': list(data_dir.glob('pituitary/*')),
}
num_classes=len(flowers_images_dict)
flowers_labels_dict = {
'glioma': 0,
'healthy': 1,
'meningioma': 2,
'pituitary': 3,
}
print("example of irm image ",str(flowers_images_dict['pituitary'][1]))
img = cv2.imread(str(flowers_images_dict['pituitary'][1]))
plt.imshow(img)
plt.show()
X, Y = [], []
for flower_name, images in flowers_images_dict.items():
for image in images:
img = cv2.imread(str(image))
resized_img = cv2.resize(img,(nl,nc))
X.append(resized_img)
Y.append(flowers_labels_dict[flower_name])
X = np.array(X)
Y = np.array(Y)
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, shuffle=True,
random_state=0)
```

```
X_train= X_train / 255
X_test= X_test / 255
shape1=X_train.shape
shape2=X_test.shape
print("size of training set/ test set",shape1[0],shape2[0])
#model building
image_shape = (nl, nc, nch)
model = Sequential([
layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu', input_shape=(nl, nc, nch)),
layers.MaxPooling2D(),
layers.Conv2D(32, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Conv2D(64, 3, padding='same', activation='relu'),
layers.MaxPooling2D(),
layers.Flatten(),
layers.Dense(128, activation='relu'),
layers.Dense(num_classes) ])

model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy']
)
model.summary()
# training
epochs = 5
batchsize = 32
history = model.fit(
x = X_train,
y = Y_train,
validation_data= (X_test,Y_test),
batch_size = batchsize, epochs=epochs, verbose=(2) )

shape1=X_train.shape
shape2=X_test.shape
shape3=Y_test.shape
print("size of training set/test set",shape1[0],shape2[0])
print("shape of Y_test",shape1,shape3)
def plot_training_history(history):
# Plot training & validation accuracy values
plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
# Plot training & validation loss values
plt.subplot(122)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.tight_layout()
plt.show()
# Plot the training history
plot_training_history(history)
print(f"Final Training Accuracy: {history.history['accuracy'][-1]:.4f}")
print(f"Final Validation Accuracy: {history.history['val_accuracy'][-1]:.4f}")
```