Machine Learning and Optimization Methods (IT3071)

Labsheet 5 2024

Build a convolutional neural networks (CNNs) for image classification

Objective:

- To obtain basic knowledge about Convolutional Neural Network Architecture.
- Build a binary image classification model using the tensorflow library.

Materials Needed:

Tensorflow Lirary
Anaconda

1. Use Cat/Dog Image Dataset for the implementation of below CNN model

   a. Acquire Data

```
train_path = 'Training_data'
valid_path = 'Testing_data'
```

   b. Define the model

```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 300x300 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(filters=16,kernel_size=3, activation='relu', input_shape=(300, 300, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(filters=16,kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=2),
    # The third convolution
    tf.keras.layers.Conv2D(filters=32,kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=2),
    # The fourth convolution
    tf.keras.layers.Conv2D(filters=64,kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=2),
    # The fifth convolution
    tf.keras.layers.Conv2D(filters=64,kernel_size=3, activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    # Only 1 output neuron. It will contain a value from 0-1 where 0 for 1 class ('Cat') and 1 for the other ('Dog')
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

   i.

   c. Get model summary

   d. Compile the model

   e. Train the model from generators

f. Training

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# All train images will be rescaled by 1./255
train_datagen = ImageDataGenerator(rescale = 1./255)

# Flow training images in batches of 128 using train_datagen generator
train_generator = train_datagen.flow_from_directory(train_path,# This is the source directory for training images
                                                    target_size = (300, 300),# All images will be resized to 150x150
                                                    batch_size = 32,
                                                    # Since we use binary_crossentropy loss, we need binary labels
                                                    class_mode = 'binary')

# All test images will be rescaled by 1./255
test_datagen = ImageDataGenerator(rescale = 1./255)
# apply predefined specification to test dataset
test_set = test_datagen.flow_from_directory(valid_path,
                                            target_size = (300, 300),
                                            batch_size = 32,
                                            class_mode = 'binary')
```

g. Save weights for future prediction

h. Plot variation in loss and accuracy

```
import matplotlib.pyplot as plt
# plot the loss
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(history.history['accuracy'], label='train acc')
plt.plot(history.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```