ILLINOIS INSTITUTE OF TECHNOLOGY

CS 584: MACHINE LEARNING

Theoretical Derivation

MUSIC RECOMMENDATION SYSTEM USING EMOTION RECOGNITION

Author:
Dikshitha Varman – A20521565
Jayasurya R – A20513480
Kiran Gopi – A20520561

Supervisor:
Professor Yan Yan

THEORETICAL DERIVATION OF ALGORITHM:

Convolutional neural network (CNN) model emo_model, with 16 layers, is defined by our team's code. With certain alterations, the model is based on the VGG16 architecture. The approach aims to categorize face expressions into 7 groups. Convolutional layers with various filter sizes are the foundation of the model architecture, which is then followed by batch normalization and max-pooling layers. L2 regularization with a 0.01 coefficient is used to regularize the convolutional layers. The final layer of the model, which has seven output units with a SoftMax activation function, has three fully connected layers. The stochastic gradient descent (SGD) optimizer is used to train the model, using a learning rate of 0.0001, momentum of 0.85, and Nesterov momentum. Accuracy is employed as the evaluation metric, while categorical cross-entropy is the applied loss function. The ImageDataGenerator class of TensorFlow is used to create the training data, and it employs data augmentation methods like zooming, shifting, and flipping to expand the training dataset. With a batch size of 113 and 400 epochs, the model is trained using the fit () method, with early termination depending on validation accuracy. Theoretically, this code develops a CNN model that is based on VGG16 and is capable of accurately identifying facial expressions. The generalization performance of the model is enhanced using batch normalization and L2 regularization. The strategies used for data augmentation during training assist in reducing overfitting and enhancing the model's capacity to generalize to new data. The SGD optimizer with momentum and Nesterov acceleration helps to speed up the convergence of the model during training. Convolutional layers with batch normalization, layers with maximum pooling, and fully linked (dense) layers make up most of the model. The stochastic gradient descent optimizer (tf.keras.optimizers.SGD) is then used to assemble the model with a learning rate of 0.0001, momentum of 0.85, and Nesterov momentum enabled. The accuracy metric is used to assess the model's performance, and the categorical cross-entropy loss (tf.keras.losses.CategoricalCrossentropy) is the chosen loss function.

TIME COMPLEXITY:

We must examine the number of operations carried out by the algorithm as a function of the volume of the input data in order to determine the temporal complexity of the given code. Images are used as the input data in this case and are trained using a deep convolutional neural network model. The number of layers, the size of each layer, the number of parameters, the batch size, the optimizer employed, and the number of epochs all affect how time-consuming it is to train a deep neural network. The model has 13 convolutional layers in the code provided, followed by 3 fully connected layers and a SoftMax activation function. Each convolutional layer increasingly uses more filters, going from 64 to 512. Each filter is 3x3 in size, with "same" padding applied. Each layer is subjected to batch normalization and L2 regularization in order to enhance performance and decrease overfitting. A stochastic gradient descent (SGD) optimizer with Nesterov momentum and a learning rate of 0.0001 is used to train the model.

The temporal complexity of training the model can be calculated as O (N * E * P * L2), where N is the total number of training examples, E is the total number of epochs, P is the total number of model parameters, and L is the largest size that each layer can be. The number of steps per epoch in the provided code is equal to the number of training instances divided by the batch size, which in this case is 113. The number of images in the training set, x_train.shape[0], serves as the value of N for the training set. The value of E is 400, which represents the quantity of training epochs. The number of parameters in each layer of the model can be added together to determine the value of P. The greatest size of any feature map in the model, which in this case is 28x28, can be used to estimate the value of L. As a result, it is possible to estimate the time complexity of training the model as O (N * E * P * L2) = O (28709 * 400 * 13491947 * 784) = O (2.28 * 1020), which is a very huge number. This is simply a rough estimate, and the real time complexity may vary based on several variables, including the hardware used for training, the model's specific configuration, and the training optimization methods.