Birla Institute of Technology and Science Pilani, Hyderabad Campus

1nd Semester 2024-25, BITS F464: Machine Learning

Assign No:5 (Mini-Project), Max.Marks:15, Date given:16.11.2024, Date of sub:23.11.2024

Problem1: (Objective) Hand gesture recognition using Convolutional Neural Network (CNN).

Background:

American Sign Language (ASL) is a complete, complex language that employs signs made by moving the hands combined with facial expressions and postures of the body. It is the primary language of many deaf North Americans, and one of several communication options available to deaf people. ASL has its own grammar, syntax, and vocabulary, distinct from English. Below given are few hand gestures. The referred dataset format is patterned to match closely with the classic MNIST dataset. Each training and test case represents a label (0-25) as a one-to-one map for each alphabetic letter A-Z (and no cases for 9=J or 25=Z because of gesture motions).

Sign language is a vital mode of communication for individuals with hearing impairments. Automating the recognition of sign language gestures can aid in improving communication accessibility. Convolutional Neural Networks (CNNs) have demonstrated promising results in image classification tasks and are well-suited for recognizing hand gestures in sign language.

Problem Statement:

The objective of this mini-project is to develop CNN models capable of accurately classifying sign language letters based on image inputs. By training neural networks on a dataset of sign language gestures, the models will aim to recognize and classify hand gestures into their corresponding letters.

Dataset:

The dataset consists of grayscale images of hand gestures representing sign language letters. Each image is of size 28X28 pixels, and the dataset includes labeled examples for training and testing. You may download the dataset from the google class page (hand sign.zip) and use it for the task. For the project, consider the Features: Input Image: Grayscale images of hand gestures representing sign language letters. Target Variable: The target variable is the class label corresponding to the sign language letter depicted in the input image.

Build a CNN architecture with the following structure:

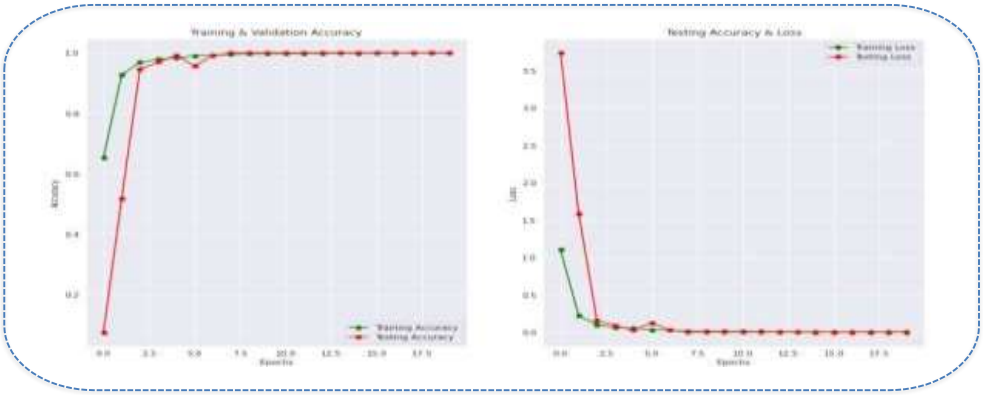
1. First Convolutional Layer: The first convolutional layer applies a set of learnable filters (kernels) to the input data. Each filter extracts certain features from the input, such as edges, textures, or patterns. The output of this layer is a set of feature maps representing the activation of each filter across the spatial dimensions of the input. Choose appropriate kernel/ filter to extract significant features.
2. Second Convolutional Layer: The output feature maps from the first convolutional layer serve as the input to the second convolutional layer. Similar to the first layer, the second convolutional layer applies another set of learnable filters to the feature maps generated by the first layer. These filters can capture more abstract and higher-level features by combining lower-level features learned in the first layer which could include features such as repetitive patterns, textures with varying scales, or more intricate structures within the images. Choose appropriate kernel/ filter.
3. Activation Function: After each convolutional layer, an activation function is typically applied element-wise to introduce non-linearity into the network, allowing it to learn more complex patterns. Use ReLU or Sigmoid for this task.
4. Max Pooling Layer: The max pooling layer down-samples the feature maps generated by the convolutional layers, reducing their spatial dimensions (width and height). Max pooling is a form of subsampling that retains the most important features while discarding less relevant information. It does this by selecting the maximum value from each pooling region. Use a pooling window size (e.g., 2x2 or 3x3).

Now, by stacking convolutional layers followed by max pooling layers, the network learns to extract increasingly abstract and hierarchical features from the input data. This hierarchical representation is then typically flattened and fed into one or more fully connected (dense) layers for final classification or regression. Use a dropout layer (with rate 0.28) after the fully connected layer that will help prevent overfitting by introducing randomness into the weights connecting the fully connected layer, thereby regularizing the network. Use an output layer of 26 nodes (for each class).

Model Visualization:

Provide a visual representation of the model architecture to illustrate the data flow through the layers. This visualization aids in understanding the model's structure and operation.

Model Training: Compilation: Compile the model using the Adam or AdamW optimizer. Training: Train the model for 20 epochs. Evaluation: Monitor changes in loss and testing accuracy for each epoch as plotted below. Results: Analyze the variations in loss and testing accuracy over epochs to assess model performance. Write your observations on Overfitting issues in this project. How does the CNNhandle this issue? Will the Overfitting further reduce if you use another dropout layer (say with rate0.4) after the Convolution layers? We have discussed Share structure property and Invariance property in the class for an image classification task. How has your project handled these? Put your observations into a text file and submit alongwith the implementation.



Success Metrics:

The success of the project will be evaluated based on the accuracy of sign language letter classification achieved by the CNN model. Key performance metrics include accuracy and loss, withthe goal of maximizing accuracy while minimizing loss.

Problem2: Time Series Forecasting (Energy Consumption Forecasting) using LSTMs.

LSTMs are widely used for time series forecasting because of their capability to capture complex temporal dependencies and effectively handle long-term memory. Time series datasets frequently display recurring patterns, spanning extended periods, such as annual cycles or weekly patterns. LSTMs excel at identifying and modeling both long-term and short-term seasonal variations in the data. LSTMs capture non-linear relationships for forecasting using several gates. You are given with (attached) individual household's electric power consumption data with a one-minute sampling rate over a period of almost 4 years (source: UCI Machine Learning Repository). The dataset contains features like date and time, power consumption (in kilowatts), voltage, current, power usage in different parts of the house etc. Your task in this part is to implement an LSTM-based deep learning model to forecast daily energy consumption for this household. Such forecasts could help in optimizing energy usage, reducing electric bills, and most importantly saving energy. Your implementation should include pre-processing, LSTM model development with TensorFlow, Keras or PyTorch, Evaluation using MAE, RMSE etc., and visualization (plotting forecasted vs. actual energy usage).

Mode of submission: Same as earlier submissions. You may send your queries if any, to I/C or to f20210564@hyderabad.bits-pilani.ac.in.

References: https://github.com/topics/sign-language-recognition-system; https://github.com/SheezaShabbir/Time-series-Analysis-using-LSTM-RNN-and-GRU