
# Mini Project No 1

CS 771A : INTRODUCTION TO MACHINE LEARNING

Team Members:
Lt Col Divya Sharma 241110085
Yugul Kishor Pandey 241110083
Rajender Gill 241110055
Deepak Soni 241110018

IIT Kanpur

October 22, 2024

# Machine Learning Models

## 1. Objective

The objective of this mini-project is to experiment with various machine learning (ML) models on three distinct binary classification datasets, each representing different feature sets derived from the same raw data. The primary goal is to determine the best classification models that balance both high accuracy and minimal training data usage. Additionally, we explore whether combining the three datasets leads to improved performance compared to using them individually.

## 2. Tasks

### 2.1. Task 1: Individual Dataset Model Training

For each of the three datasets, we train binary classifiers and evaluate them on the following criteria:

- High accuracy on the validation/test set.
- Minimal amount of training data required to achieve this accuracy.

### 2.2. Datasets

The three datasets used in this project are:

1. **Emoticon Dataset:** Contains 13 categorical features, where each feature corresponds to an emoti-con.
    - We assessed the accuracy of the following models on the given dataset:
        - LSTM
        - Bidirectional GRU
        - TCN
        - CNN
    - Evaluation: By training all these models on the given data set, we found that the best modelis **CNN** with its accuracy of 93.05% when we use 100% training data.
        - With Training set of 20%
          Accuracy Achieved is 78.94%
        - With Training set of 40%
          Accuracy Achieved is 85.28%
        - With Training set of 60%
          Accuracy Achieved is 86.91%
        - With Training set of 80%
          Accuracy Achieved is 90.59%
        - With Training set of 100%
          Accuracy Achieved is 93.05%
    - Summary of the Selected Model:
        - **Input:** Character-level sequences of emoticons (padded to length 20).
        - **Embedding Layer:** Maps characters to 32-dimensional dense vectors to capture relationships between them.
        - **Conv1D Layer:** Applies 16 filters with a kernel size of 3 to extract local patterns fromthe sequences.

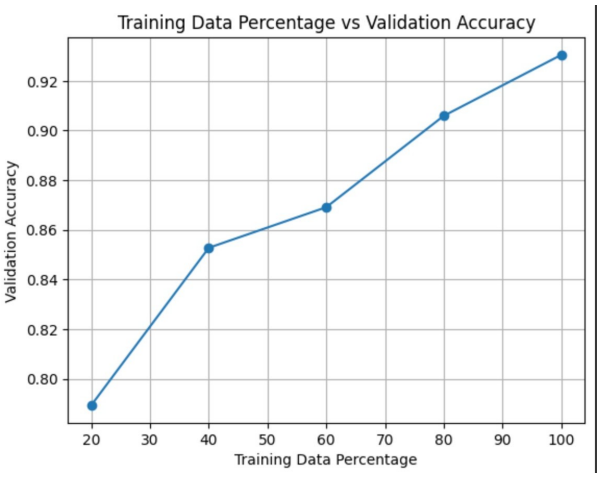


Figure 1: Trained Model: CNN

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_6 (Embedding) | (None, 20, 32) | 6,880 |
| conv1d_18 (Conv1D) | (None, 18, 16) | 1,552 |
| global_max_pooling1d (GlobalMaxPooling1D) | (None, 16) | 0 |
| dense_12 (Dense) | (None, 32) | 544 |
| dropout (Dropout) | (None, 32) | 0 |
| dense_13 (Dense) | (None, 1) | 33 |

Total params: 27,029 (105.59 KB)
Trainable params: 9,009 (35.19 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 18,020 (70.39 KB)

Figure 2: Summary: CNN

- **Global Max Pooling:** Selects the most significant feature from each filter, reducing dimensionality.
- **Dense Layer:** A fully connected layer with 32 units and ReLU activation to capturehigher-level patterns.
- **Dropout:** A 20% dropout rate to prevent overfitting.
- **Output Layer:** Uses a sigmoid function for binary classification, outputting a probabil-ity.
- **Optimizer:** Adam optimizer with a learning rate of 0.001.
- **Loss Function:** Binary crossentropy for measuring classification error.

2. **Deep Features Dataset:** In this dataset, each input is represented using a 13 × 786 matrix which basically consists of 786-dimensional embeddings of each of the 13 emoticon features of an input.
   - We assessed the accuracy of the following models on the given dataset:
     - Temporal Convolutional Network(TCN)
     - Echo State Network
     - WaveNet
     - LSTM(Long Short Term Memory) Neural Network Model
   - Evaluation: By training all these models on the given data set, we found that the best modelis **LSTM** with its accuracy of 98.36% when we use 80% training data.
     - With Training set of 20%
       Accuracy Achieved is 94.68%
     - With Training set of 40%
       Accuracy Achieved is 96.73%
     - With Training set of 60%
       Accuracy Achieved is 97.34%

- With Training set of 80%
  Accuracy Achieved is 98.36%
- With Training set of 100%
  Accuracy Achieved is 97.96%

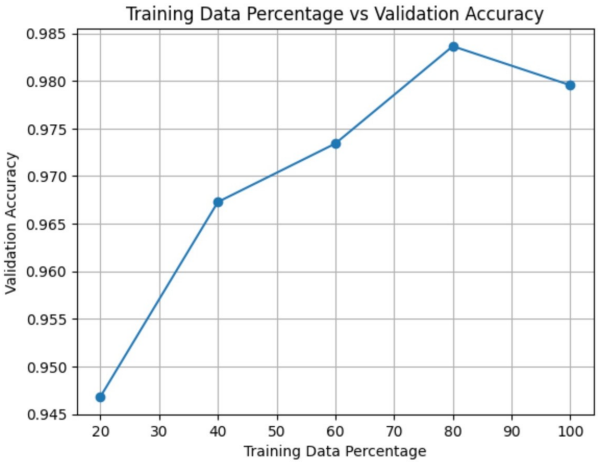


**Figure 3: Trained Model: LSTM**

- Summary of the Selected Model: The model used in this project is a Unidirectional Long Short-Term Memory (LSTM) neural network designed for binary classification on sequential data with a shape of (13, 768).
  - **Input Layer:** The input is a sequence of shape (13, 768), where 13 represents the time steps, and 768 represents the feature dimensions of the input data.
  - **LSTM Layer:** A unidirectional LSTM layer with 2 units.LSTMs are designed to capture temporal dependencies in the input data, making them suitable for sequential tasks.The return sequences=False argument ensures that the LSTM outputs only the final hidden state, summarizing the entire sequence.
  - **Fully Connected Layer:** A Dense layer with 64 units and ReLU activation. This layer helps in learning higher-level representations from the LSTM output and introduces non- linearity.
  - **Output Layer:** A Dense layer with a single unit and sigmoid activation for binary clas- sification. The output is a probability between 0 and 1, used to classify the input into oneof two categories.
  - **Optimization:** The model is optimized using the Adam optimizer with a binary crossen- tropy loss function, suitable for binary classification tasks.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| lstm_1 (LSTM) | (None, 2) | 6,168 |
| dense_20 (Dense) | (None, 64) | 192 |
| dense_21 (Dense) | (None, 1) | 65 |

Total params: 19,277 (75.30 KB)
Trainable params: 6,425 (25.10 KB)
Non-trainable params: 0 (0.00 B)

**Figure 4: Summary: LSTM**

3. **Text Sequence Dataset:** Dataset contains a 50-character long string representing each input.

- We assessed the accuracy of the following models on the given dataset:
  - GRU Model
  - CNN Model
  - CNN+LSTM Model

- CNN+ Bidirectional GRU Hybrid Model

- Evaluation: By training all these models on the given data set, we found that the best modelis **CNN+GRU Hybrid Model** with its accuracy of 83.84% when we use 100% training data.

  - With Training set of 20%
    Accuracy Achieved is 70.96%
  - With Training set of 40%
    Accuracy Achieved is 76.07%
  - With Training set of 60%
    Accuracy Achieved is 74.44%
  - With Training set of 80%
    Accuracy Achieved is 80.78%
  - With Training set of 100%
    Accuracy Achieved is 83.84%

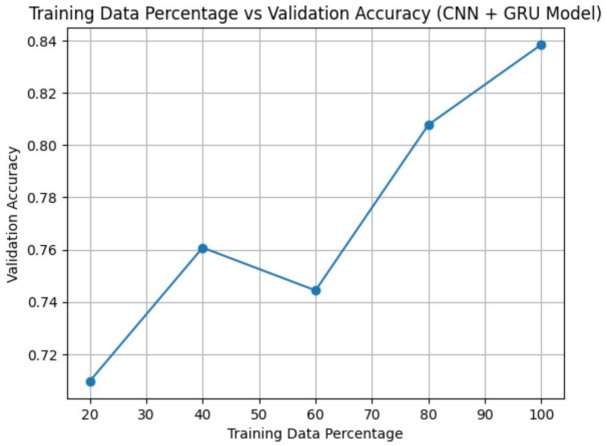


Figure 5: Trained Model: CNN+GRU Hybrid Model

- Summary of the Selected Model: The model used in this project is a CNN+ GRU HybridModel.

  - **CNN Layers:** The model starts with a Convolutional (Conv1D) layer, which acts as a feature extractor. It captures local patterns and important subsequences in the input by sliding filters over the sequence. This is followed by MaxPooling to reduce the dimensionality and retain the most important features.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| embedding_10 (Embedding) | (None, 50, 64) | 704 |
| conv1d_22 (Conv1D) | (None, 46, 16) | 5,136 |
| max_pooling1d_13 (MaxPooling1D) | (None, 23, 16) | 0 |
| bidirectional_1 (Bidirectional) | (None, 32) | 3,264 |
| dropout_5 (Dropout) | (None, 32) | 0 |
| dense_28 (Dense) | (None, 16) | 528 |
| dropout_6 (Dropout) | (None, 16) | 0 |
| dense_29 (Dense) | (None, 1) | 17 |

**Total params:** 28,949 (113.09 KB)
**Trainable params:** 9,649 (37.69 KB)
**Non-trainable params:** 0 (0.00 B)
**Optimizer params:** 19,300 (75.39 KB)

Figure 6: Summary: CNN+GRU Hybrid Model

- **Bidirectional GRU Layer:** The Bidirectional GRU is a type of Recurrent Neural Network (RNN) designed to capture both forward and backward temporal dependencies in the input sequence. By processing the input in both directions, it helps the model retain context from the past (previous characters) and future (next characters), making it well-suited for handling sequential dependencies.
- **Dense Layer:** A series of Dense layers with ReLU activation are used to further process thefeatures extracted by the CNN and GRU layers. Dropout is applied to prevent overfitting and improve generalization.
- **Output Layer:** The final output layer is a single neuron with a sigmoid activation function, designed for binary classification. It predicts the probability of the input belonging to oneof two classes.
- **Character-level Input:** The model is trained on sequences of individual characters, which allows it to handle textual data at a granular level.
- **Hybrid Structure:** The use of both CNNs and GRUs allows the model to capture local features (via CNN) and long-term dependencies (via GRU).
- **Bidirectional Processing:** The bidirectional nature of the GRU enables the model to understand context from both past and future, making it highly effective for tasks where context from both directions is important.

– **Optimization:** The model uses Adam optimizer with binary cross-entropy as the loss function, suitable for binary classification tasks. Dropout layers are used after the GRU and Dense layers to reduce overfitting during training. The model training includes early stopping, which halts the training process when the validation loss stops improving for a specified number of epochs, ensuring the model does not overfit.

## 2.3. Task 2: Combined Dataset Model Training

In this task, we have investigated that using all the 3 training datasets together, we have learnedan even better model that has a better accuracy than the models learned on each of the 3 datasets individually.

- High accuracy on the validation/test set.
- Minimal amount of training data required to achieve this accuracy.
- We assessed the accuracy of the following models on the given dataset:
  - CNN Model
  - LSTM Model
  - CNN+GRU Hybrid Model
  - SVM Model
  - Random Forest Model
- Evaluation: By training all these models on the given data set, we found that the best modelis **Random Forest Model** with its accuracy of 93.05% when we use 100% training data and it is found to be the most efficient model amongst all the tried models.
  - With Training set of 20%
    Accuracy Achieved is 87.73%
  - With Training set of 40%
    Accuracy Achieved is 91.21%

- With Training set of 60%
  Accuracy Achieved is 92.02%
- With Training set of 80%
  Accuracy Achieved is 92.23%
- With Training set of 100%
  Accuracy Achieved is 93.05%

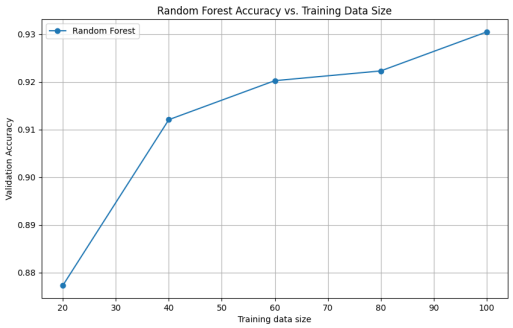


Figure 7: Trained Model: Random Forest

- Summary of the Selected Model: The model used in the combined dataset is a Random Forest.
  - **Random Forest Classifier:**
    - Ensemble Method: Combines multiple decision trees to improve predictive accuracy andcontrol overfitting.
    - Hyperparameter Tuning: Utilizes Grid Search to optimize parameters like:
    - n estimators: Number of trees in the forest.
    - max depth: Maximum depth of each tree to prevent overfitting.
    - min samples split: Minimum samples required to split an internal node.
    - min samples leaf: Minimum samples required to be at a leaf node.
  - **Loading and Processing Emoticon Data:** The training dataset containing emoticon strings and labels is loaded from a CSV file. Then, converting Emoticons to Integers: Emoticon strings are transformed into unique integer representations, which are necessary for machine learning algorithms that cannot process string data directly. Sequence Conversion: Emoticon strings are converted into padded integer sequences to standardize the input size.
  - **Extracting Labels:**Labels corresponding to each emoticon string are extracted for training the classification model.
  - **Loading Deep Feature Data:** Deep features are loaded from a .npz file. These features are crucial for providing a numerical representation of the emoticon data, capturing important information for classification.
  - **Loading Text Sequence Data:** Text sequence data is loaded from a CSV file, excluding label columns. This data offers additional numerical features that can improve model performance.
  - **Handling Missing Values:** Missing values in the deep features and text sequence data are replaced with zeroes to ensure the dataset is clean for training.
  - **Data Scaling and PCA:** Scaling: Data is standardized using StandardScaler to ensure all features

contribute equally to the model's training.PCA (Principal Component Analysis): PCA is applied to the deep features to reduce dimensionality, preserving the most significant information and improving computational efficiency.

- **Combining Features:** All processed features (emoticons, deep features, text sequences) are concatenated into a single feature matrix for training.
- **Validation Data Processing:** TA similar processing pipeline is applied to the validation dataset, including loading emoticons, deep features, and text sequences, followed by integer encoding, PCA transformation, and scaling.
- **Loading and Processing Test Data:** The test dataset undergoes the same preprocessing steps as the validation dataset to ensure compatibility with the model.
- **Model Training and Hyperparameter Tuning:** A RandomForestClassifier is instantiated, and hyperparameters are tuned using GridSearchCV based on validation accuracy. This helpsin finding the best-performing model configuration.
- **Final Model Training** The best Random Forest model is retrained on the entire combined training dataset.

- **Final Model Training** The best Random Forest model is retrained on the entire combinedtraining dataset.

## Deliverables:

- Predictions for each test set: `pred_emoticon.txt`, `pred_deepfeat.txt`, `pred_textseq.txt`, and `pred_combined.txt`.
- A detailed report (6-8 pages) discussing the models and their performances.
- Code (Python files) including a `README` file, which can generate the final predictions.

## Conclusion

We analyzed the results of the individual models trained on each dataset, comparing their per- formance. Additionally, we experimented with combining the datasets by merging the feature representations. While combining datasets may lead to improved performance, it is possible that some individual models outperform the combined model.