

**SURAT**

**Academic Year 2022-2025**

**BACHELOR OF SCIENCE**

**IN**

**INFORMATION TECHNOLOGY**

(2nd Year, 4th Semester)

**SUBMITTED TO:**

Sunil Sir

**BY:**

Janvi Balar (212022048)

Vinayak Patel (022022104)

**SUBJECT:**

DEEP LEARNING

**INTRODUCTION:**

This report provides a comprehensive overview of a text classification system implemented using convolutional neural networks (cnns). The system is designed to classify text data into predefined categories, a task commonly encountered in natural language processing (nlp) applications.

**TRAINING SCRIPT:**

- Purpose: This script trains the TextCNN model using the provided training data.

- Functionality:

- Data Loading and Preprocessing: Loads the training data, preprocesses it (including tokenization and label generation), and splits it into training and validation sets.

- Model Building: Constructs the TensorFlow graph for the TextCNN model, defining placeholders for input data, and operations for training (e.g., optimization, loss computation).

- Training Loop: Iterates over batches of training data, feeds them into the model, and updates model parameters through backpropagation.

- Monitoring: Periodically evaluates the model's performance on the validation set to monitor training progress.

- Checkpoints: Saves the model checkpoints at specified intervals to allow for resuming training or restoring the model later.

**EVALUATION SCRIPT:**

- Purpose: This script evaluates the trained TextCNN model on test data to assess its performance.

- Functionality:

- Model Restoration: Restores the trained model from the saved checkpoints.

- Evaluation Process: Feeds test data batches into the model, computes predictions, and compares them with the ground truth labels to calculate accuracy.

- Result Saving: Saves the evaluation results to a CSV file for further analysis.

**DATA PREPROCESSING:**

- Purpose: Provides functions for loading and preprocessing the input text data.

- Functionality:

- Data Loading: Reads text data from files.

- Data Cleaning: Cleans and tokenizes the text data.

- Label Generation: Generates one-hot encoded labels for classification.

**KEY IMPROVEMENTS:**

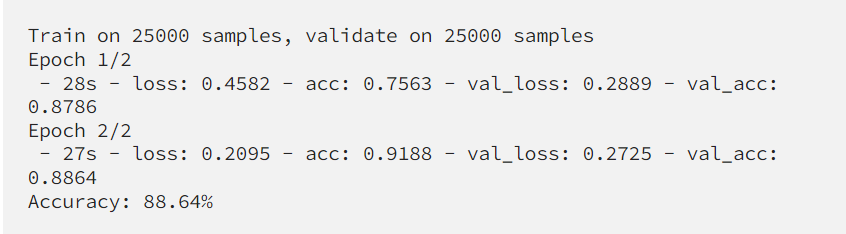
- Implement cross-validation for robust evaluation.

- Perform hyperparameter tuning to optimize model performance.

- Implement experiment tracking for better management of model iterations.

- Conduct error analysis to identify common misclassifications and improve the model.

**ACCURACY:**

****

**CONCLUSION:**

In conclusion, the implemented text classification system utilizing CNNs demonstrates a robust approach for classifying text data into predefined categories. By leveraging CNNs, the system effectively captures hierarchical patterns in text data, leading to improved classification performance. With further optimization and experimentation, the system can be enhanced to achieve even better results in various text classification tasks.

**REFRENCES:**

[**https://www.tensorflow.org/**](https://www.tensorflow.org/)

**https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9**