**Report on Parkinson's Disease Detection using Deep Learning**

**1. Introduction**

This report details the development of a deep learning model for Parkinson's Disease (PD) detection from speech data. The project aims to classify individuals as either healthy or having PD, and additionally, to estimate the severity of the disease using the Unified Parkinson's Disease Rating Scale (UPDRS). The analysis covers data preprocessing, model building, training, and performance evaluation.

**2. Dataset Description**

The dataset is composed of voice recordings from 20 individuals with Parkinson's Disease (PWP) and 20 healthy controls. The training set includes 26 different types of voice samples per subject (sustained vowels, numbers, words, and short sentences). For each voice sample, 26 linear and time-frequency based features were extracted. The UPDRS score, a measure of disease severity, is also provided for each PD patient in the training set. The test set contains voice recordings from 28 PD patients, focusing on sustained vowels 'a' and 'o' repeated three times. The same 26 features are extracted from these test recordings.

Dataset Link: <https://archive.ics.uci.edu/dataset/301/parkinson+speech+dataset+with+multiple+types+of+Audio+recordings>

**3. Data Preprocessing**

* **Loading Data:** The training and testing datasets are loaded into pandas DataFrames from the text files using pd.read\_csv().
* **Column Naming:** Column names are assigned to the DataFrames. The training data columns include 'subject\_id', features 'f1' to 'f26', 'UPDRS', and 'class'. The test data columns are similar but lack the 'class' label.
* **Data Type Conversion:** The 'subject\_id' and 'class' columns are converted to integers, and the 'UPDRS' column is converted to float.
* **Data Trimming:** Since each subject has 26 samples, the DataFrames are trimmed to ensure that the number of rows is a multiple of 26. This is crucial for reshaping the data for sequence-based models.
* **Feature Scaling:** The features ('f1' to 'f26') are extracted from both training and testing sets. StandardScaler is used to standardize the features in the training set, and the same scaler is applied to the test set to ensure consistent scaling.
* **Data Reshaping:** The scaled features are reshaped into a 3D format ([number\_of\_subjects, samples\_per\_subject, number\_of\_features]) to be suitable for LSTM and other recurrent neural network models. For the training data, this results in a shape of (40, 26, 26), and for the test data, (6, 26, 26).
* **Target Preparation:** The target variables are extracted. For classification, y\_train\_class is created by taking the 'class' labels from the training data, selecting one label per subject. Similarly, y\_train\_updrs and y\_test\_updrs are created for the regression task, using the 'UPDRS' scores.

**4. Model Building**

We build five deep learning models using TensorFlow/Keras:

* **LSTM (Long Short-Term Memory):** A Bidirectional LSTM network is used, incorporating LayerNormalization, Dense layers with ReLU activation and L2 regularization, and Dropout for regularization.
* **GRU (Gated Recurrent Unit):** A Bidirectional GRU network with Dropout, BatchNormalization, and Dense layers.
* **BiLSTM (Bidirectional LSTM):** Similar to the LSTM model but potentially with slightly different hyperparameters.
* **SimpleRNN:** A basic Recurrent Neural Network with multiple SimpleRNN layers, BatchNormalization, Dropout, and Dense layers.
* **DenseNet1D:** A 1D Convolutional Neural Network inspired by DenseNet architecture, using Conv1D layers, GlobalAveragePooling1D, Dropout, and Dense layers.

Each model architecture is encapsulated in a build\_... function, allowing for easy running.

**5. Model Training and Compilation**

The notebook defines several compile\_and\_train\_... functions, one for each model type. These functions compile the models with the Adam optimizer, using 'sparse\_categorical\_crossentropy' loss for classification and 'mse' (Mean Squared Error) for regression. Accuracy is used as the metric for classification, and Mean Absolute Error (MAE) for regression. Callbacks including EarlyStopping and ReduceLROnPlateau are used to prevent overfitting and optimize learning rates during training.

The training data is split into training and validation sets using train\_test\_split for both classification and regression tasks. Each model is trained separately for classification (predicting 'class') and regression (predicting 'UPDRS').

**6. Results and Evaluation**

After training, each model is evaluated on the training data, and the loss and accuracy (for classification) or MAE (for regression) are recorded. The results are stored in a dictionary and printed.

**7. Results Summary**

The performance of each model for both classification and regression are:

* **Classification:** Loss and Accuracy
* **Regression:** Loss and MAE

The results show the SimpleRNN achieving the best accuracy (87.5%) in classification, while the GRU model has the lowest MAE (8.8119) in the regression task.

**8. Comparison and Discussion**

* The models vary in performance across the classification and regression tasks. Recurrent models (LSTM, GRU, BiLSTM, SimpleRNN) generally outperform the DenseNet1D for classification, suggesting that the sequential nature of the data is important for distinguishing between healthy and PD subjects.
* For regression (UPDRS prediction), GRU shows the best performance.
* The differences in performance highlight the importance of model selection and hyperparameter tuning for specific tasks.
* The project successfully demonstrates the feasibility of using deep learning for PD detection and severity estimation from speech data.

We have summarized the key aspects of the Parkinson's Disease detection project, providing insights into the data processing, model development, and results.