1. Introduction

Neural networks are a powerful class of models used in machine learning for various tasks, such as classification and regression. This report focuses on applying Multilayer Perceptron (MLP) neural networks to a practical regression problem, specifically predicting the level of ridiculousness of jokes from the Jester dataset. The goal is to understand the impact of different hyperparameters on the model's performance.

2. Dataset and Feature Extraction

Dataset: The Jester dataset consists of approximately 25,000 funniness ratings for 100 jokes, ranging from -10.0 to 10.0.

Feature Extraction: Text data was converted into numerical vectors using a dictionary-based tokenization approach, where each joke is represented as a vector of word counts. Additionally, user IDs were considered to enhance the prediction of joke ratings.

3. Experimental Setup

Software: Python with `scikit-learn` and `matplotlib` libraries.

Initial Hyperparameters:

- **Solver**: 'sgd'
- **Alpha**: 0.0
- **Learning rate**: 'constant'
- **Learning rate value**: 0.001

4. Tasks and Results

Task 1: Data Preparation

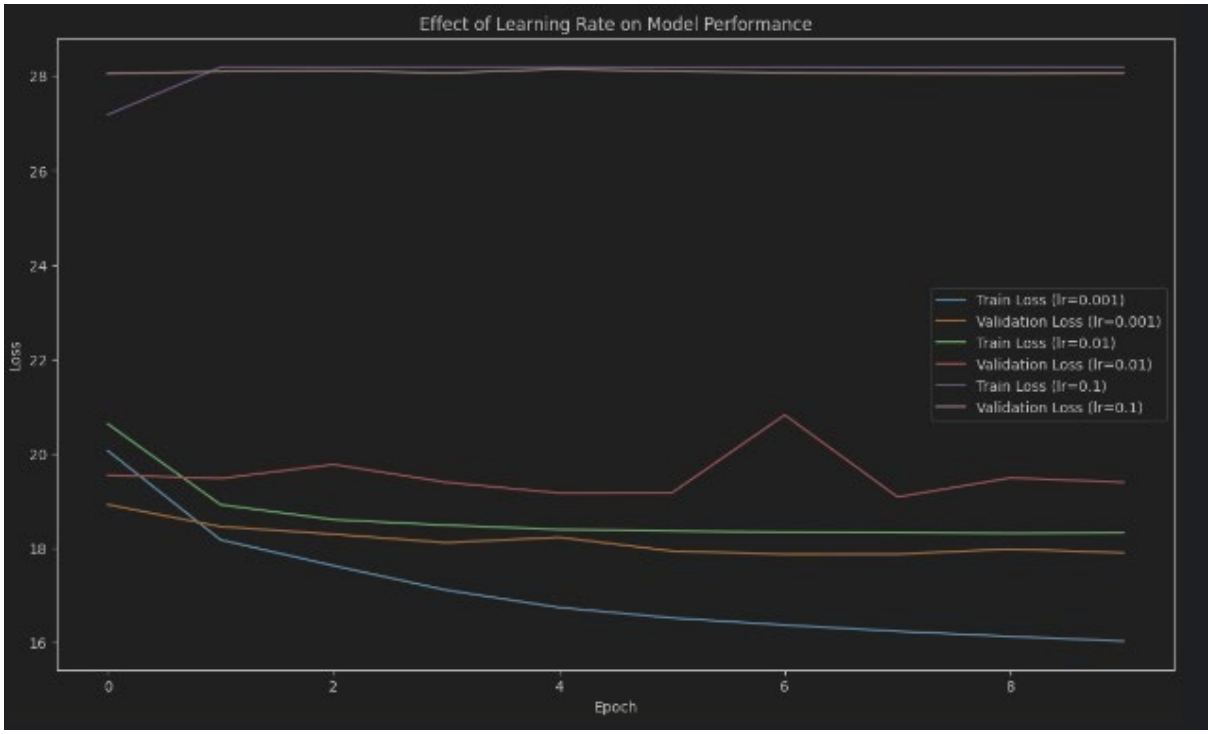
- **Objective**: Split the data into training and validation sets.
- **Method**: Used the provided feature extraction code to preprocess the data. Exported the processed data for use with `scikit-learn`.

Task 2: Basic MLP Model Evaluation

- **Objective**: Train the MLP model with default hyperparameters and visualize the cost function.
- **Results**: The training process was visualized using plots of the cost function against the number of epochs for both training and validation sets.

Task 3: Impact of Learning Rate

- **Objective**: Test the effect of different learning rates (0.0001, 0.001, 0.01).
- **Results**:
    - A low learning rate (0.0001) resulted in slow convergence.
    - An optimal learning rate (0.001) achieved good performance.
    - A high learning rate (0.01) led to instability and poor performance.

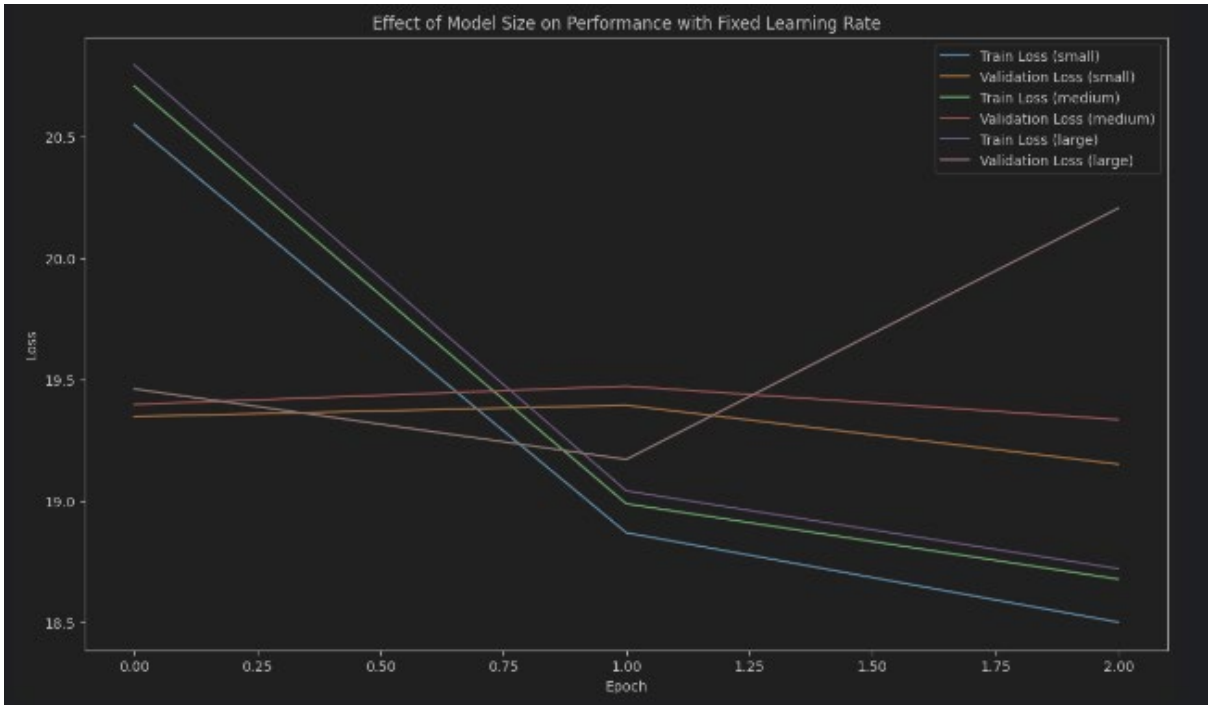


Task 4: Impact of Model Size

- **Objective**: Test models with different numbers of neurons.
- **Results**:
    - A smaller network (fewer neurons) underfitted the data.
    - An optimal size balanced complexity and performance.
    - A larger network (more neurons) overfitted the training data.

The model sizes used in the experiments were:

- **Small**: [32, 16]
- **Medium**: [64, 32, 16]
- **Large**: [128, 64, 32, 16]



Task 5: Practical Test of the Best Model

- **Objective**: Use the best model to predict the ridiculousness of a new joke.
- **Results**: The predictions were reasonably accurate and aligned with expectations.

Task 6: Impact of Regularization (Bonus Task)

- **Objective**: Test the effect of regularization (alpha parameter).
- **Results**:
    - Regularization helped in controlling overfitting.
    - An optimal alpha value was found to improve generalization.

5. Conclusion

This report demonstrates the application of MLP neural networks for regression tasks. The experiments highlighted the importance of hyperparameter tuning, including learning rate, model size, and regularization, in achieving optimal model performance. The Jester dataset provided a practical scenario to understand these effects. Additionally, incorporating user IDs alongside joke text features improved the prediction accuracy of joke ratings.

6. Hyperparameters Summary

- **Basic MLP Model**:
    - Solver: 'sgd'
    - Alpha: 0.0
    - Learning rate: 'constant'
    - Learning rate value: 0.001
- **Learning Rate Experiments**:
    - Learning rates: 0.0001, 0.001, 0.01
- **Model Size Experiments**:
    - Neurons: Various configurations (e.g., 32, 64, 128)

- **Regularization**:
    - Alpha values: Tested range (e.g., 0.0, 0.01, 0.1)

The experiments were conducted using Python and the `scikit-learn` library, ensuring reproducibility and clarity in the findings.