***MIDTERM EVALUATION***

**NAME: -Taneshwar**

**ROLL NO: -221124**

INTRODUCTION

In this assignment, we aimed to explore the impact of different optimizers and loss functions on the performance of a neural network model for a crystal structure classification task. The dataset used contains information about various crystal structures, and the objective is to predict the lowest distortion of the crystal structure

EXPLANATION

1. **Data Preprocessing:**

* Imported the dataset and unnecessary columns ('Compound', 'In literature') were dropped.
* Missing values marked with '-' were replaced with NaN value.
* Categorical data was encoded using label encoding.
* Invalid labels were identified and replaced with valid ones to ensure consistency in the target variable.
* The dataset was split into features (X) and labels (y).

1. **Model Building:**

* A feedforward neural network model was constructed using TensorFlow's Keras API.
* The model architecture consisted of three dense layers, with ReLU activation functions and dropout layers to prevent overfitting.
* The output layer utilized a softmax activation function for multi-class classification (4 in this case).

1. **Training Procedure:**

* The training data was further split into training and validation sets to monitor model performance during training.
* Features were standardized using sklearn's StandardScaler to ensure uniformity in feature scales.

1. **Exploration of Optimizers:**

* I used four different optimizers: Adam, SGD , RMSprop, and Adadelta.
* Each optimizer was evaluated in terms of its accuracy.

1. **Evaluation Metrics:**

* The primary evaluation metric used was accuracy, which measures the percentage of correctly classified instances.

**Results and Discussion:**

**Adam Optimizer:**

* + Adam optimizer showed the highest accuracy.
  + Adam optimizer shows us robust convergence behavior.
  + It reaches High training and accuracy performance.
  + It is relatively quick as compared to other Optimizers.

**SGD (Stochastic Gradient Descent):**

* + SGD showed slower convergence compared to Adam and other optimizers.

**RMSprop:**

* + It was similar to adam optimizer but showed less performance to Adam i.e in accuracy
  + It exhibited stable convergence behavior and reached high accuracy on both training and validation sets.

**Adadelta:**

* + Adadelta optimizer demonstrated slower convergence and it hardly reaches high accuracỵ
  + There was much difference in validation accuracy and Training accuracy.

.

RESULT & ANALYSIS

1. **Data Preprocessing:**

* Loaded the dataset and dropped unnecessary columns.
* Handled missing values marked with '-' and replaced them with NaN.
* Label-encoded the target column 'Lowest distortion'.
* Corrected invalid labels in the target column.

1. **Model Building and Training:**

* Built a neural network model with two hidden layers and dropout.
* Used four different optimizers: 'adam', 'sgd', 'rmsprop', 'adadelta'.
* Trained the model for 200 epochs and monitored training/validation accuracy.

1. **Visualization:**

* Plotted training and validation accuracy for each optimizer across epochs.

CONCLUSION

In conclusion, Adam and RMSprop are best in this case, but the choice of optimizer may depend on different parameters like (parameters, model complexity). Adam and RMSprop optimizers generally showed superior performance in terms of convergence speed and accuracy compared to SGD and Adadelta.

**Data Preprocessing:**

Appropriate handling of missing values and label encoding is crucial for preparing the data for deep learning models.

**Model Performance:**

The model achieved varying degrees of accuracy for different optimizers.

The performance of the model was evaluated using training and validation accuracy.

**Optimizers Comparison:**

'adam' optimizer generally provided the best training and validation accuracy.

'sgd' (Stochastic Gradient Descent) and 'rmsprop' also showed good performance but may require further tuning.

**Epochs Consideration:**

Training for 200 epochs seemed reasonable for this task, as it allowed the model to converge without overfitting.

Scalability:

The model's scalability and generalization to unseen data can be further assessed using a larger test set or cross-validation.

**Hyperparameter Tuning:**

The model architecture and hyperparameters may need further tuning to optimize performance.

**Further Analysis:**

Additional analysis, such as confusion matrices, precision-recall curves, or F1 scores, could provide a more detailed understanding of the model's performance.