Predicting GSNR using ResNet for Tabular Data

Hira Sardar

***Abstract*—This report presents an approach to predict Generalized Signal-to-Noise Ratio (GSNR) using a modified ResNet model for tabular data. The model was trained and validated on a European dataset, and we report the results and challenges encountered, particularly those related to computational limitations during the fine-tuning phase.**

I. Introduction

Predicting the Generalized Signal-to-Noise Ratio (GSNR) is a crucial task in various telecommunications applications. Traditional methods often struggle with the complexity and volume of data. In this paper, we leverage a modified ResNet architecture to handle tabular data for this prediction task.

II. Dataset

The dataset used for training and evaluation was obtained from a European dataset consisting of various features such as Power, NLI, ASE, Frequency, Number of Spans, and Total Distance. The target variable is GSNR. The dataset preprocessing involved filling missing values with the mean, normalizing features, and splitting into training, validation, and test sets.

III. Methodology

A. Data Preprocessing

- Filled missing values with column means.
- Normalized numerical features to have zero mean and unit variance.

B. Model Architecture

The base model is a ResNet-50, modified to accept tabular data. The final layers were adjusted to fit the regression task of predicting GSNR.

C. Training

The training involved splitting the data into 70% training, 15% validation, and 15% test sets. The model was trained using the MSE loss function and Adam optimizer for 50 epochs.

Listing 1. Training Loop

```
def train_model(model, train_loader, val_loader, criterion, optimizer, epochs=50):
    model.train()
    for epoch in range(epochs):
        train_loss = 0
        for features, target in train_loader:
            optimizer.zero_grad()
            outputs = model(features)
            loss = criterion(outputs, target.unsqueeze(1))
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
    train_loss /= len(train_loader)

    val_loss = 0
    model.eval()
    with torch.no_grad():
        for features, target in val_loader:
            outputs = model(features)
            loss = criterion(outputs, target.u
            val_loss += loss.item()
    val_loss /= len(val_loader)
```

D. Evaluation

The model's performance was evaluated using Mean Squared Error (MSE), R-squared (R^2), and Mean Absolute Error (MAE).

TABLE I
Model Performance on Validation Set

Metric	Value
MSE	1.4174
R^2	-0.3533
MAE	0.9590

IV. Challenges and Future Work

A. Computational Limitations

Over the last two weeks, we faced significant challenges related to computational power while attempting to perform freezing and fine-tuning of the ResNet model. These steps are crucial for improving the model's performance, but due to the limited resources, the process could not be completed successfully.

V. Conclusion

This study demonstrates the potential of using a modified ResNet model for predicting GSNR from tabular data. Despite computational challenges, the initial results are promising, and future work will focus on optimizing the model and leveraging higher computational resources to complete the fine-tuning process.