REPORT ON TEXT AND SEQUENCE DATA

Initial Insights:

1. **Transformers or RNNs on sequence and text data:**
   Tokenization, padding, and embedding layers are used for preparation for text data.Textual data is transformed into a dense vector representation and transmitted through a fully connected network to create a basic embedding-based model using Keras.
2. **Increasing efficiency with less data:**
   The previewed cells don't specifically address data augmentation or transfer learning methods, which are popular approaches for enhancing performance on sparse data. It focuses on dense layers and embedding.
3. **Methods for improving predictions:**
   It makes use of a dense classifier, embedding layers, and the IMDB dataset. The cells under examination do not yet display methods like hyperparameter tuning, sophisticated models (like RNNs or Transformers), or pre-trained embeddings.

1. **How to apply RNNs or Transformers to text and sequence data?**
   Text data is transformed into dense vector representations using embedding layers.
   To improve the processing of text data, models incorporate pre-trained embeddings (such as GloVe) to establish weights.

2. **How to improve performance of the network, especially when dealing with limited data.?**
   Pre-trained Embedding:
   GloVe embeddings, which don't require training on extensive datasets, are utilized to improve the model's comprehension of word relationships.
   To maintain the previously learned information, the embedding layer is made to be non-trainable.
   Validation Data:
   To keep an eye on overfitting and adjust model hyperparameters, a validation set is utilized.

3. **Determine which approaches are more suitable for prediction improvement.?**

RNNs:

Use LSTMs or GRUs for capturing sequential patterns in text, such as contextual dependencies.

Transformers:

Leverage attention mechanisms or pre-trained Transformer models (e.g., BERT, GPT) to achieve state-of-the-art results in text processing tasks.

Transfer Learning:

Use fine-tuning on large pre-trained models, such as BERT or RoBERTa, for even better predictions.

Results:

Embedding Technique	Training Sample Size	Training Accuracy (%)	Test Loss
Custom-trained Embedding Layer	100	100	0.69
Custom-trained Embedding layer	5000	97.75	0.38
Custom-trained Embedding layer	1000	98.75	0.67
Custom-trained Embedding Layer	10000	97.44	0.35
Pretrained word Embedding (GloVe)	100	100	0.82
Pretrained word Embedding (GloVe)	5000	100	0.89
Pretrained word Embedding (GloVe)	1000	98.59	1.27

Pretrained word Embedding(GloVe)	10000	97.96	1.19

Conclusion:

The comparison between custom-trained embedding layers and pre-trained GloVe embeddings reveals distinct strengths for each approach. When compared to pre-trained embeddings, custom-trained embeddings exhibit lower test loss, indicating superior generalization, particularly with smaller datasets. This implies that when training data is scarce, task-specific embeddings are more flexible. Conversely, pre-trained GloVe embeddings show increased test loss, maybe as a result of overfitting or a mismatch with the dataset, even though they achieve nearly perfect training accuracy across all sample sizes. Both approaches perform better as the sample size grows, while in terms of test loss, custom-trained embeddings perform somewhat better than GloVe embeddings. Both methods work well with bigger datasets, although custom-trained embeddings are generally better suited for problems with little data.