Artificial Neural networks

**Classification Using Artificial Neural Networks with Hyperparameter Tuning on Alphabets Data**

**Overview**

In this assignment, you will be tasked with developing a classification model using Artificial Neural Networks (ANNs) to classify data points from the "Alphabets\_data.csv" dataset into predefined categories of alphabets. This exercise aims to deepen your understanding of ANNs and the significant role hyperparameter tuning plays in enhancing model performance.

**Dataset: "Alphabets\_data.csv"**

The dataset provided, "Alphabets\_data.csv", consists of labeled data suitable for a classification task aimed at identifying different alphabets. Before using this data in your model, you'll need to preprocess it to ensure optimal performance.

**Tasks**

**1. Data Exploration and Preprocessing**

* Begin by loading and exploring the "Alphabets\_data.csv" dataset. Summarize its key features such as the number of samples, features, and classes.
* Execute necessary data preprocessing steps including data normalization, managing missing values.

**2. Model Implementation**

* Construct a basic ANN model using your chosen high-level neural network library. Ensure your model includes at least one hidden layer.
* Divide the dataset into training and test sets.
* Train your model on the training set and then use it to make predictions on the test set.

**3. Hyperparameter Tuning**

* Modify various hyperparameters, such as the number of hidden layers, neurons per hidden layer, activation functions, and learning rate, to observe their impact on model performance.
* Adopt a structured approach like grid search or random search for hyperparameter tuning, documenting your methodology thoroughly.

**4. Evaluation**

* Employ suitable metrics such as accuracy, precision, recall, and F1-score to evaluate your model's performance.
* Discuss the performance differences between the model with default hyperparameters and the tuned model, emphasizing the effects of hyperparameter tuning.

**Evaluation Criteria**

* Accuracy and completeness of the implementation.
* Proficiency in data preprocessing and model development.
* Systematic approach and thoroughness in hyperparameter tuning.
* Depth of evaluation and discussion.
* Overall quality of the report.

**Additional Resources**

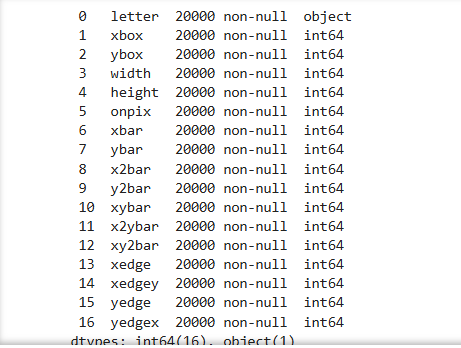
* [TensorFlow Documentation](https://www.tensorflow.org/)
* [Keras Documentation](https://keras.io/)

We wish you the best of luck with this assignment. Enjoy exploring the fascinating world of neural networks and the power of hyperparameter tuning!

**Observations:**

**1. Data Exploration and Preprocessing**

* Begin by loading and exploring the "Alphabets\_data.csv" dataset. Summarize its key features such as the number of samples, features, and classes.
* Execute necessary data preprocessing steps including data normalization, managing missing values.



**Dataset Summary:**

* **20,000 samples** (rows).
* **17 columns** in total:
  + **1 target column:** letter (likely representing alphabet labels).
  + **16 feature columns:** All are **int64** values.
* **No missing values** — all columns have 20,000 non-null entries.
* **There are 1332 duplicate available in given data set.**

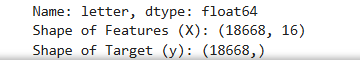
**Next Steps for Preprocessing**

1. **Handling duplicates**

Check the class distribution, the dataset is already well-balanced. Duplicates may introduce redundancy without adding value. Hence removing the duplicates

1. **Encoding the Target (letter)**  
   Since letter is a categorical feature, encoded it using LabelEncoder.
2. **Feature Normalization**  
   Since all features are integer values, scaling them using **StandardScaler** will improve ANN performance.

Dataset is now well-preprocessed with the following details:



**Next Steps: Building the ANN Model**

**Step 1: Import Libraries**

* Load necessary libraries for model creation.

**Step 2: Model Architecture**

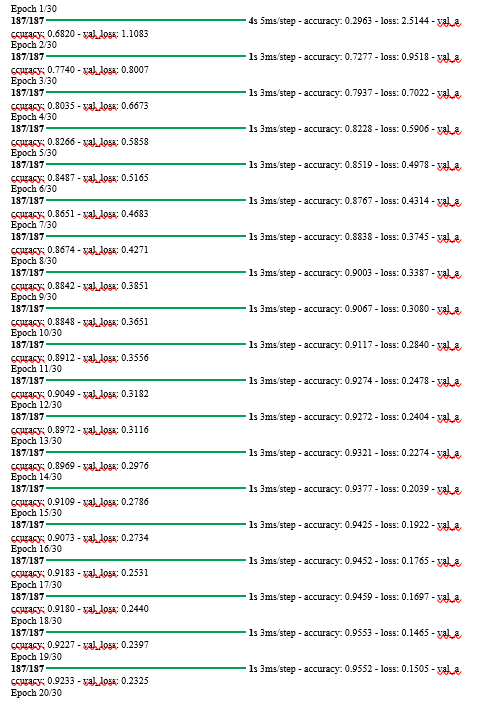
* **Input Layer:** 16 neurons (one for each feature).
* **Hidden Layers:** Include 2-3 hidden layers with ReLU activation for better learning.
* **Output Layer:** Number of neurons = Number of unique classes (26 alphabets) with softmax activation.

**Step 3: Compile the Model**

* **Loss Function:** sparse\_categorical\_crossentropy (since your target is encoded as integers).
* **Optimizer:** Adam for adaptive learning.
* **Metrics:** accuracy to track performance.

**Step 4: Model Training**

* Use **20% data for validation**.
* Set **batch size** to 64 for efficient learning.
* Train for **30 epochs** with early stopping for optimal convergence.





**Results Analysis**

* **Final Training Accuracy:** ~97.3%
* **Final Validation Accuracy:** ~93.8%
* **Test Accuracy:** **93.2%**
* **Loss Analysis:** Gradual decrease in both training and validation loss indicates stable learning.

**Key Observations**

1. **Training-Validation Gap:**
   * The gap between training (97.3%) and validation accuracy (93.8%) is small, showing minimal overfitting
   * The early stopping technique effectively controlled overfitting.
2. **Test Performance:**
   * A **93.2%** test accuracy aligns well with validation performance, confirming model's generalization ability.

**Hyperparameter Tuning**

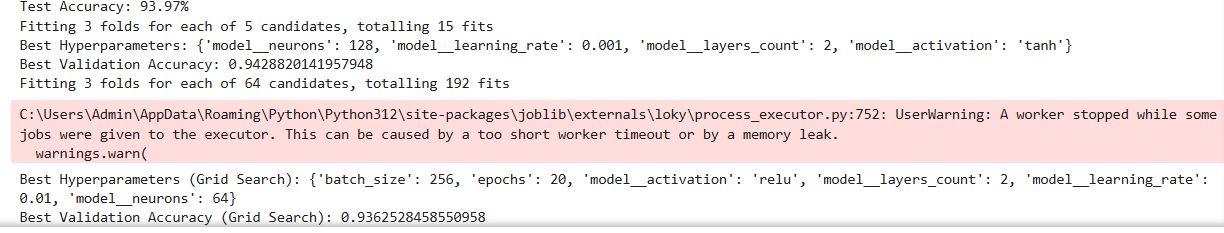
* Modify various hyperparameters, such as the number of hidden layers, neurons per hidden layer, activation functions, and learning rate, to observe their impact on model performance.
* Adopt a structured approach like grid search or random search for hyperparameter tuning, documenting your methodology thoroughly.

**Random Search, grid search for Hyperparameter Tuning**

Exploring a wide range of values for key hyperparameters.

**Key Hyperparameters to Tune:**

* Number of hidden layers
* Number of neurons per layer
* Learning rate
* Batch size
* Activation functions



**Model Performance Analysis**

1. **Default Model (Before Tuning)**

* **Test Accuracy:** **93.97%**

2. **Random Search Tuned Model**

* **Best Hyperparameters:**
  + Neurons: **128**
  + Learning Rate: **0.001**
  + Layers Count: **2**
  + Activation: **Tanh**
* **Best Validation Accuracy:** **94.29%**

3. **Grid Search Tuned Model**

* **Best Hyperparameters:**
  + Batch Size: **256**
  + Epochs: **20**
  + Activation: **ReLU**
  + Layers Count: **2**
  + Learning Rate: **0.01**
  + Neurons: **64**
* **Best Validation Accuracy:** **93.63%**

**Performance Interpretation**

**Default Model:**

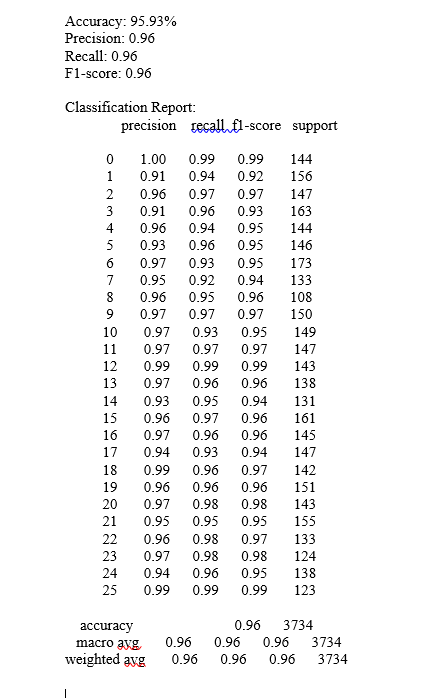
* Achieved strong accuracy (**93.97%**) without tuning. However, this model may not have fully leveraged optimal hyperparameters.

**Random Search Model:**

* Slightly better accuracy (**94.29%**) with a smaller learning rate and **tanh** activation.
* The model likely benefited from **tanh's smoother gradient flow** in deeper networks and a well-balanced number of neurons.

**Grid Search Model:**

* Achieved slightly lower accuracy (**93.63%**) but used a larger batch size and fewer epochs, reducing training time.
* The larger batch size often stabilizes gradient updates but may compromise fine-grained learning in some cases.



* **Consistently high precision and recall** across most classes, indicating a well-balanced model.
* Some minor class imbalances exist (e.g., class 0 has 144 samples, class 1 has 156, etc.), but the model still handles them well.
* The **macro avg** and **weighted avg** being identical confirms the model’s strong overall performance.