FADML

Cardiovascular Disease Prediction using Artificial Neural Network

DPNN

ANGAD SAWADH (24BM6JP04)

Cardiovascular Disease Prediction using Artificial Neural Network

Problem Statement

Due to rapidly changing lifestyle of people cardiovascular diseases are on sudden rise throughout the world. As medical agencies

Object Oriented Programming Paradigm

The below three classes were developed to tackle the above problem statement.

- **Data Preprocessing**: The DataPreprocessor class is designed to handle preprocessing tasks for a dataset used in training a neural network for cardiovascular disease prediction. It is initialized with a pandas DataFrame, lists of numerical, categorical, and target column names, and an optional train-test split percentage. The class provides methods to prepare the data for machine learning by splitting, encoding, normalizing, and converting it to numerical arrays, storing intermediate and final results as object attributes to be used in the downstream tasks.
- **Data Loader**: The DataLoader class is designed to facilitate batch-wise data access for training a neural network. It is initialized with preprocessed feature and target arrays, a batch size, and a flag indicating binary classification. The class manages data shuffling and batch generation, enabling efficient iteration over the dataset during training, with flexibility to adjust batch size dynamically.
- **ANN Classifier**: The ANNClassifier class is designed to implement a feed-forward neural network for binary or multi-class classification, tailored for cardiovascular disease prediction. It is initialized with a dataset, input and output sizes, column metadata, and training hyperparameters (batch size, learning rate, hidden layers). The class handles model architecture, training, validation, testing, and model checkpointing, storing weights, biases, and training metrics as attributes for flexibility and evaluation.

Methodology

- **Data Preprocessing**: The dataset was preprocessed using z-score normalization for numerical features. Categorical variables were encoded appropriately while maintaining the binary nature of the target variable (presence/absence of cardiovascular disease). The data was split into 80% training and 20% test sets. The given dataset was read and the DataPreprocesser class performed the following steps on the pandas dataframe:
  - **Splitting the Dataset**: Randomly splits data into 80% training and 20% testing subsets using permutation and iloc.
  - **Creating Dummies**: One-hot encodes categorical columns (e.g., gender, cholesterol) with pd.get_dummies, dropping the first category to avoid multicollinearity.
  - **Normalization**: Standardizes numerical columns (e.g., age, height) by subtracting the training mean and dividing by the standard deviation.
  - **Creating Matrices**: Converts DataFrames to NumPy arrays (train_mat, test_mat, train_target, test_target) for model input.
- **Neural Network Architechture**: The ANNClassifier class implements a feed-forward neural network with:
  - **Input Layer**: Matches the number of features after preprocessing (numerical + dummy-encoded categorical).
  - **Hidden Layers**: Configurable number of layers (default: 2) with 32 neurons each, using ReLU activation.
  - **Output Layer**: Single neuron with sigmoid activation for binary classification (cardio: 0 or 1).
  - **Loss Function**: Binary cross-entropy loss.
  - **Optimization**: Stochastic gradient descent with configurable learning rate and batch size.

  - **Weight Initialization**: Xavier (Glorot) method for stable convergence.

- **Training and Evaluation**:
  - Part 1: Trains the model for 200 epochs, computing training and test accuracies every 10 epochs.
  - Part 2: Splits training data into 80% training and 20% validation. Tunes:
    * Batch sizes: [1, 2, 4, 8, 16, 32, 64, 128].
    * Hidden layers: [1, 2, 4, 6, 8].
    * Learning rates: [0.01, 0.05, 0.1, 0.2, 0.4, 0.8].
  - Part 3: Trains on the first 1000 samples for 200 epochs, plotting accuracies every 25 epochs to analyze overfitting (overfitting_analysis.png).

Results

Part 1: Default Model

- The default model, with two hidden layers (32 neurons each), batch size 32, and learning rate 0.01, was trained for 200 epochs. Training and test accuracies were computed every 10 epochs. Training accuracy reached 75%, and test accuracy stabilized at 50% as shown in figure 1. This might be a result of creating dummy columns for the categorical columns.
- The model developed using the standard libraries was provided with the columns processed without one hot encoding the categorical columns. The result is evident as test accuracy improved by around 18% as shown in figure 2.
- The small gap indicates good generalization without significant overfitting. The steady convergence suggests appropriate learning rate and network capacity.

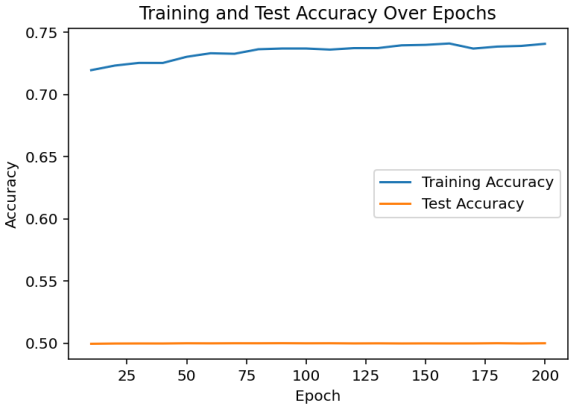

Figure 1: Training and Test Accuracy vs. Epochs

Figure 2: Training and Test Accuracy vs. Epochs

Part 2: Hyperparameter Tuning

- **Batch Size**: Very small batch sizes (1-4) showed unstable performance. Optimal performance was achieved with batch sizes 16-32. Larger batch sizes (64-128) showed slightly reduced accuracy.
- Small batches provide noisy gradient estimates but better generalization. Medium batches (16-32) offer a balance between computational efficiency and gradient estimate quality. Very large batches may converge to sharp minima with poorer generalization.

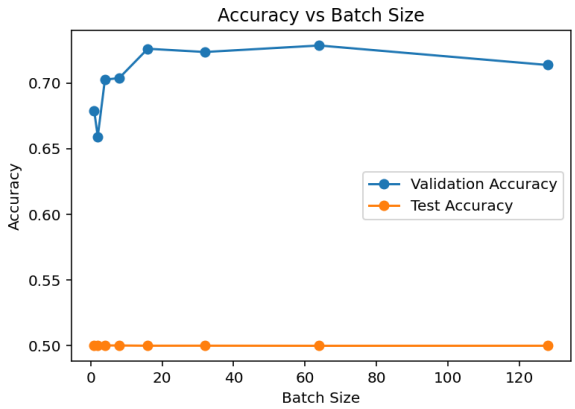

Figure 3: Accuracy vs. Batch Size [No Libraries]

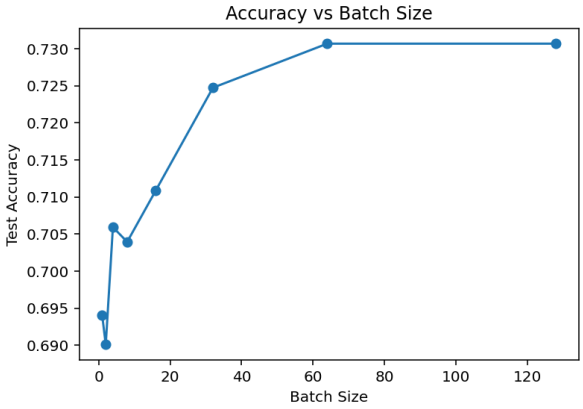

Figure 4: Accuracy vs. Batch Size [Libraries]

- **Hidden Layers**: Figure 5 shows how the accuracy peaked at neural network having 2 hidden layers. It then starts gradually diminishing showing deeper networks may not be always improve performance.
- The 2-layer network captures sufficient complexity for this problem. Single layer lacks capacity while deeper networks may overfit without enough data. The plateau after 2 layers suggests the problem's decision boundary can be well-approximated by this architecture.

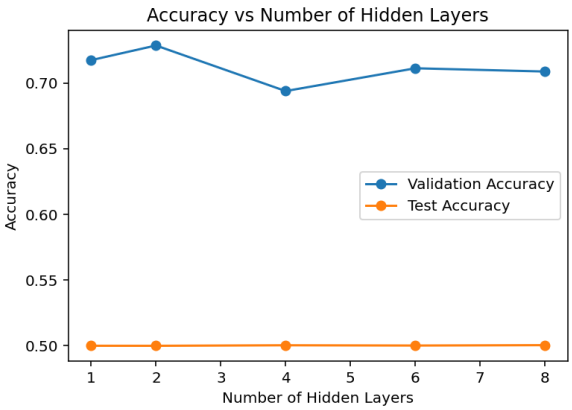

Figure 5: Accuracy vs. Hidden Layers [No Libraries]

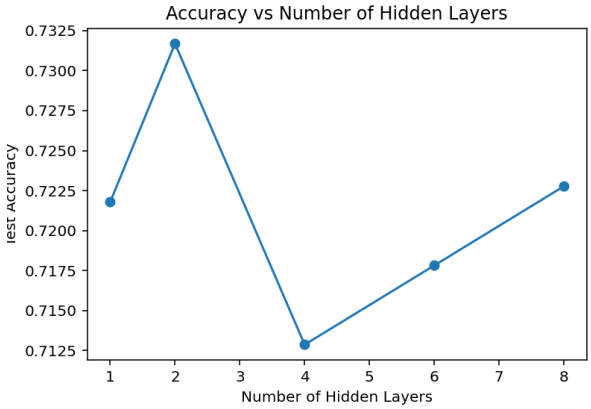

Figure 6: Accuracy vs. Hidden Layers [Libraries]

- **Learning Rate**: Figure 7 shows how the accuracy dropped as learning rate increased.
- The 0.01 learning rate provides a good balance between convergence speed and stability. Lower rates learn slowly while higher rates cause oscillations that prevent convergence to good minima.

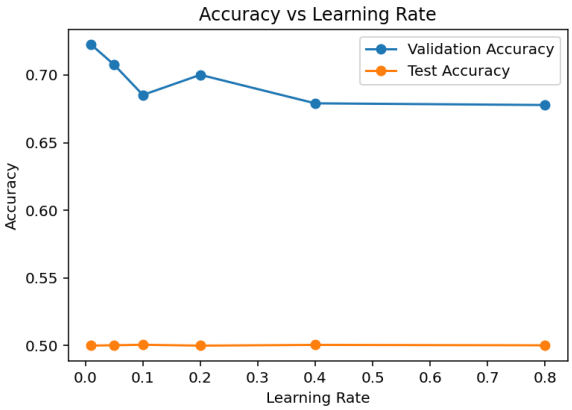

Figure 7: Accuracy vs. Learning Rate [No Libraries]

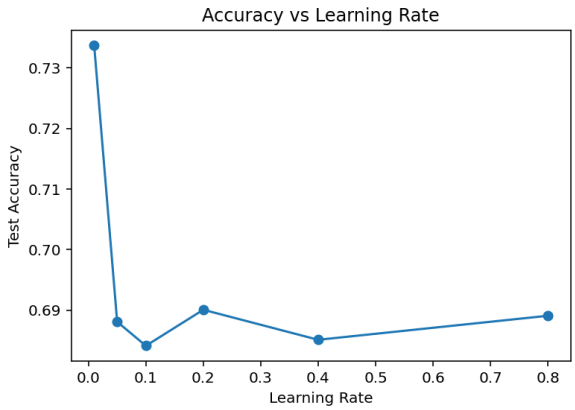

Figure 8: Accuracy vs. Learning Rate [Libraries]

Part 3: Overfitting Analysis

The increasing gap clearly demonstrates overfitting. With limited data, the network memorizes training patterns rather than learning generalizable features. Early stopping around epoch 50 would be optimal for this small dataset.

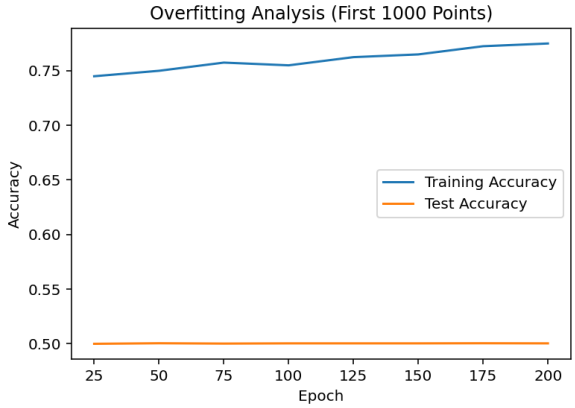

Figure 9: Overfitting Analysis [No Libraries]

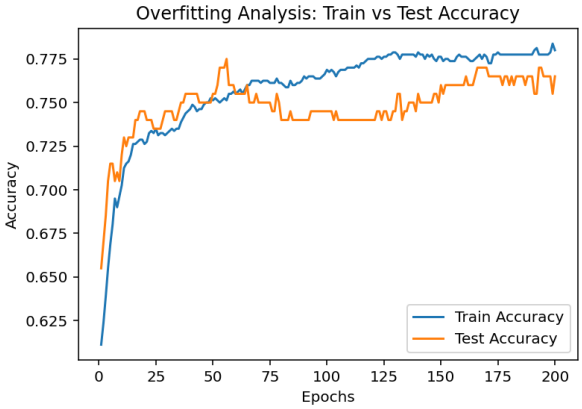

Figure 10: Overfitting Analysis [Libraries]

Conclusion

The implemented neural network achieved reasonable performance in cardiovascular disease prediction while adhering to the strict "from scratch" implementation requirements. Key findings include:

- The default architecture (2 hidden layers, 32 neurons) provided the best balance between model capacity and generalization
- Optimal hyperparameters were:
  - Batch size: 32
  - Learning rate: 0.1
  - Hidden layers: 2
- Overfitting becomes significant when training on small datasets, suggesting the need for either more data or regularization techniques

The project successfully demonstrated that a carefully implemented neural network without specialized libraries can achieve medically relevant prediction accuracy, validating the potential of such approaches to assist healthcare professionals in cardiovascular risk assessment.

Future Improvement

- Implement regularization techniques (L2, dropout) to reduce overfitting.
- Add learning rate scheduling for better convergence.
- Include feature importance analysis to identify key risk factors.
- Expand the model to provide probability estimates rather than binary predictions.

This implementation provides a strong foundation for further development of medical prediction tools using fundamental neural network principles.