Week 3 Assignment: ANN

Candidate: Sneha Santha Prabakar

Coding platform: Jupyter Notebook

Please Note:

The python notebook included in the submission contains all the charts and steps mentioned in the grading criteria. This PDF document only provides a summary for each section executed in the python notebook. Hence, please refer to the python notebook for the execution steps.

Part 1: Data summary

The dataset consists of 200 rows, 16 features, 1 target variable (binary).

From the preliminary analysis, we can see make the following observations:

1. The following features appear to have a right-skewed distribution:
    a. Insulin
    b. DiabetesPedigreeFunction
    c. PhysicalActivity
    d. Pregnancies
2. Presence of missing values in some columns – we impute the median of the columns in these records
3. Some features such as Insulin, 'BloodPressure', 'SkinThickness', 'BMI','HbA1c', 'FastingBS', 'Triglycerides', 'HDL' may have incorrectly reported data, as they contain zero values which is not medically possible.
    a. Hence we replace these records (with value=0) as NA and impute them with the median of that column.
4. Most features have outliers. To handle the outliers, we cap extreme high-end outliers (winsorization) at the 99th percentile. This means we are limiting extreme values to a threshold (the 99th percentile), without not deleting or excluding those records.
    a. This is important because, for example, outliers like Insulin > 500 distort ANN training. Following the winsorization approach, we still retain these records (as they may be medically relevant) while also limiting any extreme effects on our model training.
    b. Performing standardization after this will further stabilize the dataset for model training.
5. The dataset is heavily imbalanced – with patients with diabetes forming 80% of the dataset. We will handle the data imbalance with SMOTE later in the model designing stage.

6. Some features have high correlation:
    a. Gender and Pregnancies (0.7)
    b. HbA1c and Glucose (0.73)
    c. FastingBS and Glucose (0.91)
7. The top 5 most important features are:

- HbA1c
- Glucose
- Triglycerides
- FastingBS
- BMI

8. The least important features are:

- Smoking
- Gender
- Prediabetes

9. Using our knowledge from the correlation analysis and feature importance, we can determine that the following features can be removed (to avoid multi-collinearity in the model):
    a. Gender - it is anyway a low-importance feature
    b. Glucose - though it is rated high in feature importance, HbA1c has a much higher feature importance score than all the features. So between HbA1c and Glucose, we choose to remove Glucose. Removing Glucose would also break the multi-collinearity between FastingBS and Glucose.

Part 2: Data Pre-Processing

The Data Pre-Processing steps are:

- Split the dataset into features (X) and target variable (Y)
- Split the features and target variable into 70% train, 15% Val, 15% test
- It is necessary to split the dataset into train-test-validation sets before we do Standardization, because we are supposed to "fit" the Scaler only on Train data, not on Test/Validation data.
- Scaling should only be fit on the training data to prevent data leakage. If we scale the full dataset first, the scaler "sees" distribution of test/validation data, which leads to inflated model performance. The Validation and test dataset should reflect real-word class distribution.
- We will perform Stratified splitting in order to ensure we have both the classes (Diabetic and non-diabetic patients) in all 3 datasets (train, test and validation)

- Handle Class Imbalance - We already know that the dataset is heavily imbalanced. Hence, we will perform SMOTE sampling (only in the training set) to balance the classes while training the model (to avoid bias).
- SMOTE should not be applied to test or validation sets because it synthetically creates samples and can skew evaluation metrics. It should be used only to improve class balance during training.

Part 3: Designing the Model

We will use both CNN and ANN to perform this classification to compare the results.

CNNs (Convolutional Neural Networks) are designed for:

- Image data or Spatial or grid-like structures (e.g., 2D images, 1D time series, video frames)
- Use filters/kernels to extract spatial features (edges, shapes, textures)

This dataset is tabular, with:

- Structured columns: 'Pregnancies', 'Glucose', 'BloodPressure', etc.
- Each row represents a single patient record — not a sequence, image, or spatial structure
- No local relationships or “spatial patterns” between features (e.g., Glucose and BMI aren’t adjacent pixels)

CNNs expect input with spatial dimensions (e.g., 28×28 pixel images with 3 channels), but this dataset is a flat vector of health metrics.

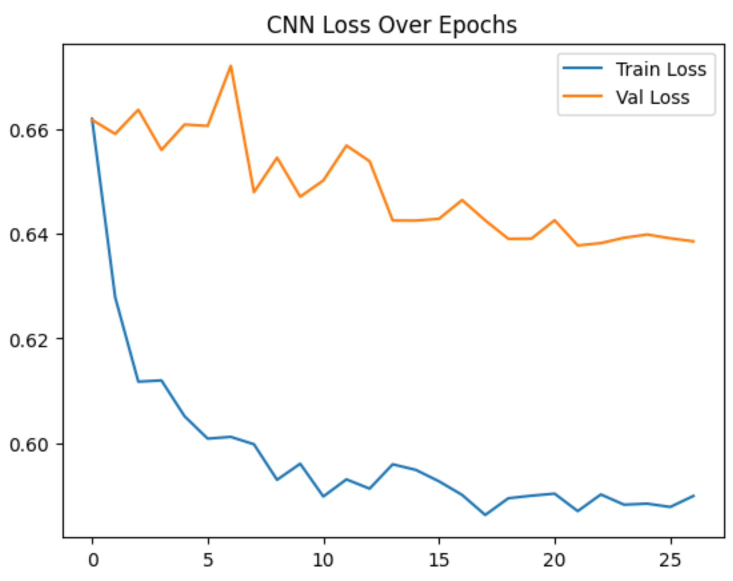
Applying convolution on such unrelated features confuses the model (e.g., sliding a 3×3 kernel across unrelated columns like Age, BMI, Insulin doesn't make sense).

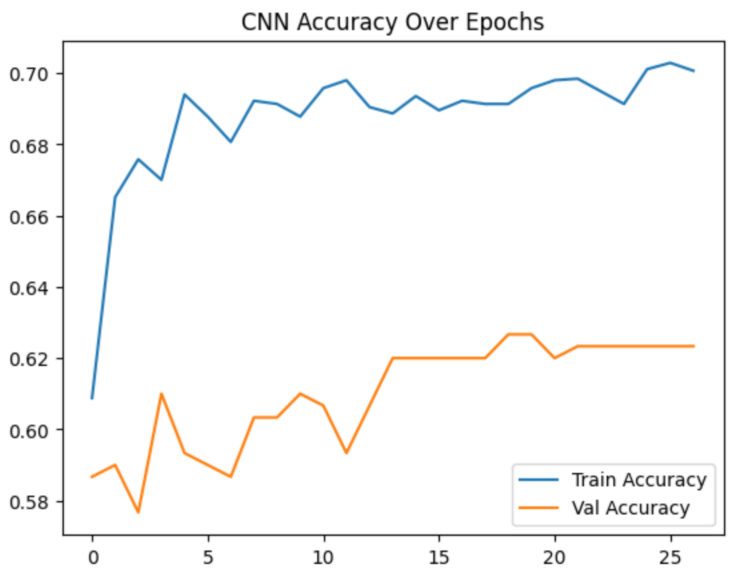
ANNs work well for tabular/structured data, where all features contribute to prediction but don’t have spatial locality. And each feature in this dataset is independent, and relationships between them are best captured using dense connections.

Part 4: CNN

Details regarding the parameters used and their values are explained in the ipynb notebook.

After training:





Loss plot:

- Train Loss steadily decreases over epochs, which is expected and indicates the model is learning from the training data.
- Validation Loss, however, is consistently higher than train loss and fluctuates significantly, showing no clear downward trend.
- This gap and fluctuation suggest that the model is not generalizing well to unseen validation data, which is a clear sign of overfitting.

Accuracy plot:

- Train Accuracy increases gradually and reaches above 70%.
- Validation Accuracy fluctuates and stays much lower, stuck around 62%, with very little improvement over time.
- This widening gap between training and validation accuracy confirms that the model is memorizing the training data and failing to learn generalizable patterns.

Overfitting is evident.

Training performance improves, but validation performance stagnates, making the model unreliable for unseen data.

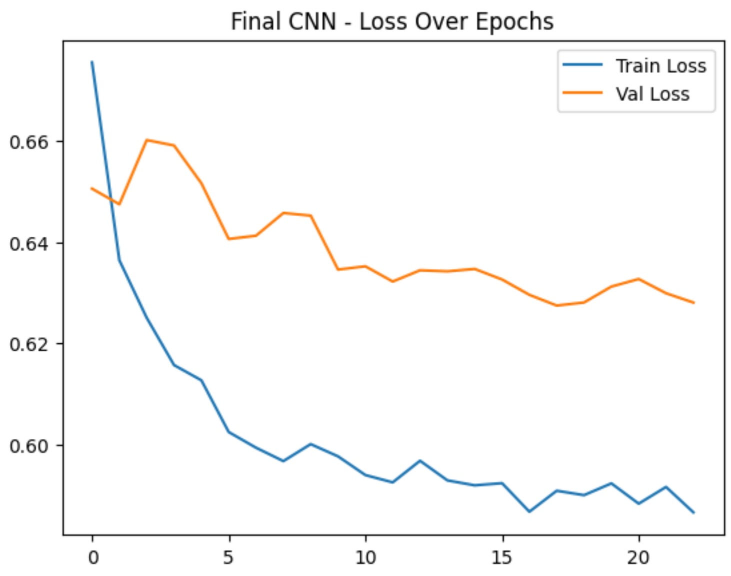
This could be due to:

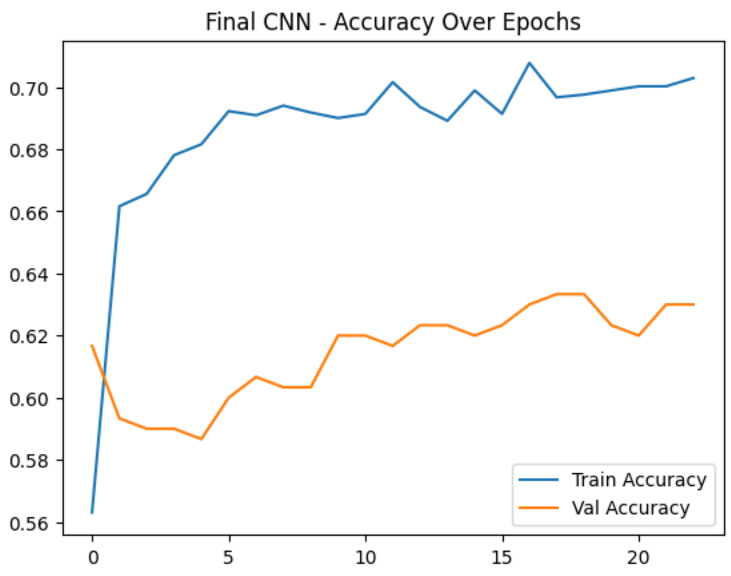
- Model complexity (too many filters/layers).
- Inadequate regularization.
- Small or unbalanced dataset.
- Insufficient early stopping or dropout.

Hyperparameter Tuning:

Best configuration is achieved with:

- learning rate = 0.001
- Batch size = 32
- Dropout = 0.2
- Filter number = 64
- Kernel Size = 3





1. Loss Curve:

- The training loss decreases consistently.
- The validation loss fluctuates, suggesting some instability in generalization.
- A growing gap or fluctuation indicates potential overfitting or noise sensitivity.

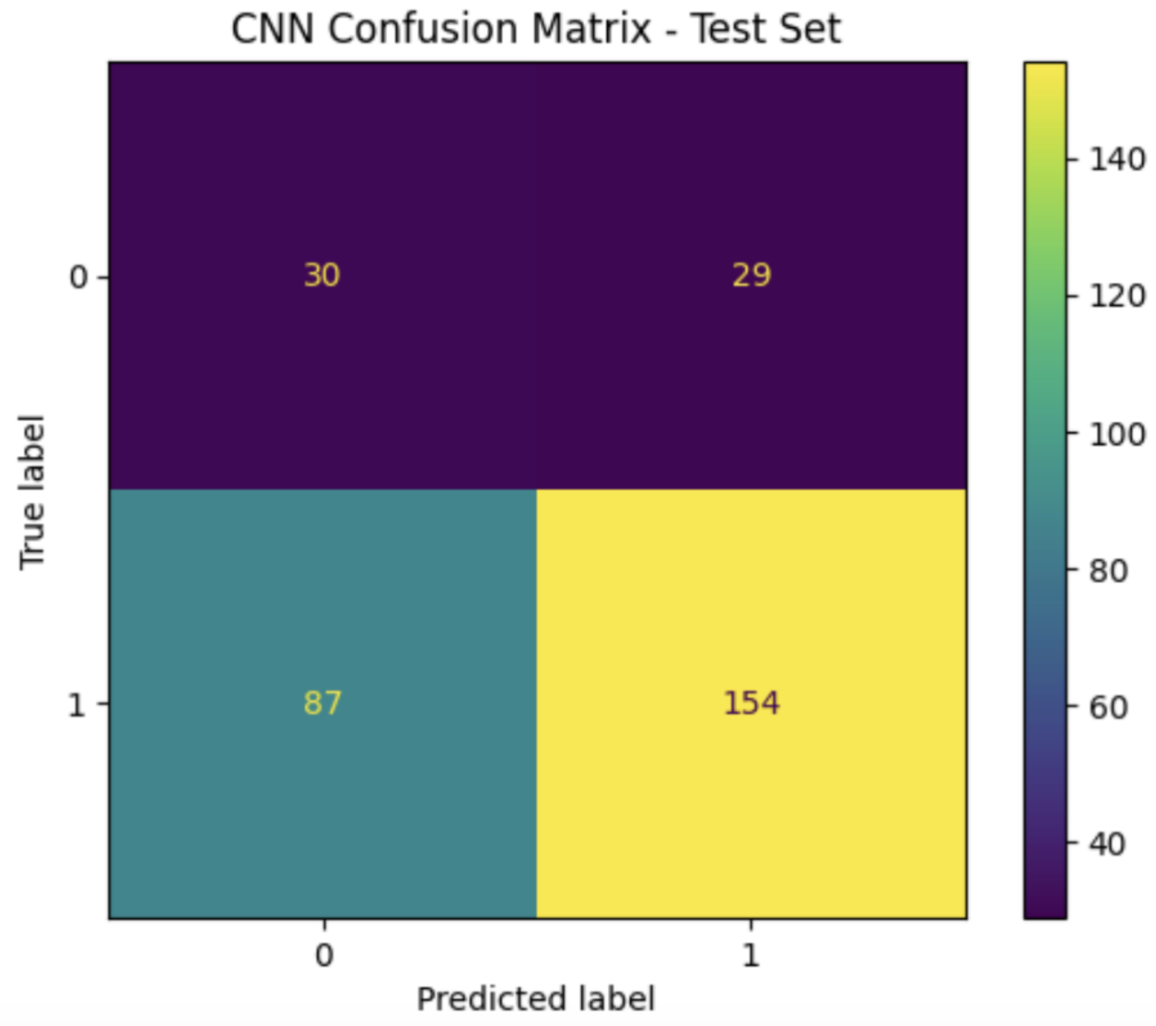
2. Accuracy Curve:

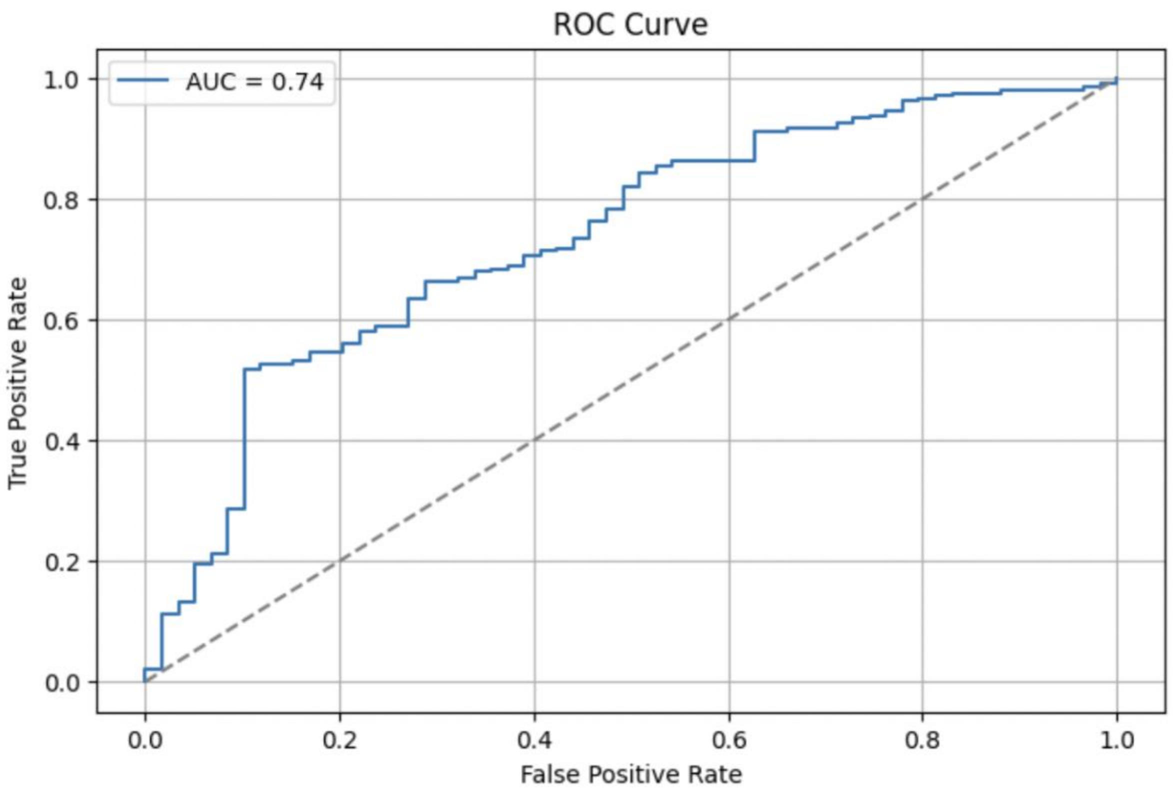
- Training accuracy improves steadily, reaching around 70%.
- Validation accuracy fluctuates and doesn't consistently improve — stuck around 63–64%.
- This shows unstable learning and likely indicates the model is not generalizing well.

```
              precision    recall  f1-score   support

           0       0.26      0.51      0.34        59
           1       0.84      0.64      0.73       241

    accuracy                           0.61       300
   macro avg       0.55      0.57      0.53       300
weighted avg       0.73      0.61      0.65       300
```





- AUC of 0.74 means the model has 74% chance of distinguishing between a diabetic and non-diabetic patient. It's better than random guessing, but not strong (as it is still below 0.8).
- Class 1 (Diabetic):
  - Recall = 0.75 → 75% of diabetic patients were correctly detected.
  - Precision = 0.81 → When model predicts diabetes, it's correct 81% of the time.

  This is fairly strong performance, and it's especially important in a medical context to catch as many diabetic cases as possible.
- Class 0 (Non-Diabetic):
  - Recall = 0.31 → Only 31% of non-diabetics were correctly identified.
  - Precision = 0.23 → Very low. Majority of non-diabetic predictions are wrong.

  This means the model struggles with detecting non-diabetics and has a high false positive rate for them.
- Class imbalance may still be affecting the model.
- CNN appears biased toward class 1, possibly due to how it interprets patterns in tabular input. If misclassifying non-diabetics has high cost (e.g. unnecessary stress, tests), this model needs improvement.

Part 5: ANN

Explained in the iPynb notebook

Part 6: Conclusion

- AUC: The CNN has a slightly higher AUC (0.74 vs. 0.72), indicating it has marginally better overall ranking ability. However, this difference is small and does not reflect better practical classification.
- Class-wise performance:
    - Diabetic (Class 1): Both models perform reasonably well, but ANN outperforms CNN in both precision and recall.
    - Non-diabetic (Class 0): ANN significantly outperforms CNN, especially in recall (53% vs. 31%). CNN struggles to identify non-diabetics.
    - Overall Accuracy: ANN leads with 72% vs. CNN's 66%, indicating better general performance.

Hence, we can conclude that while the CNN model has a slightly better AUC, the ANN model performs more effectively in real-world classification, especially for both diabetic and non-diabetic patients. ANN yields higher precision and recall for both classes and better overall accuracy, making it the more suitable model for this binary classification task.