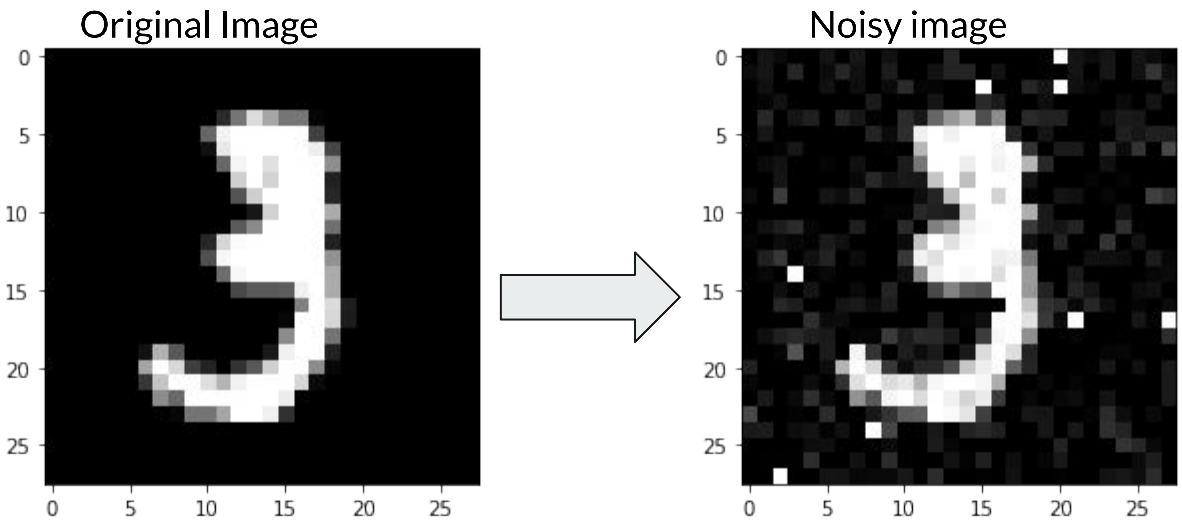
Autoencoder for Image Denoising

Shubhankar Poundrik
Nakshatra Yalagach
Anushruti H

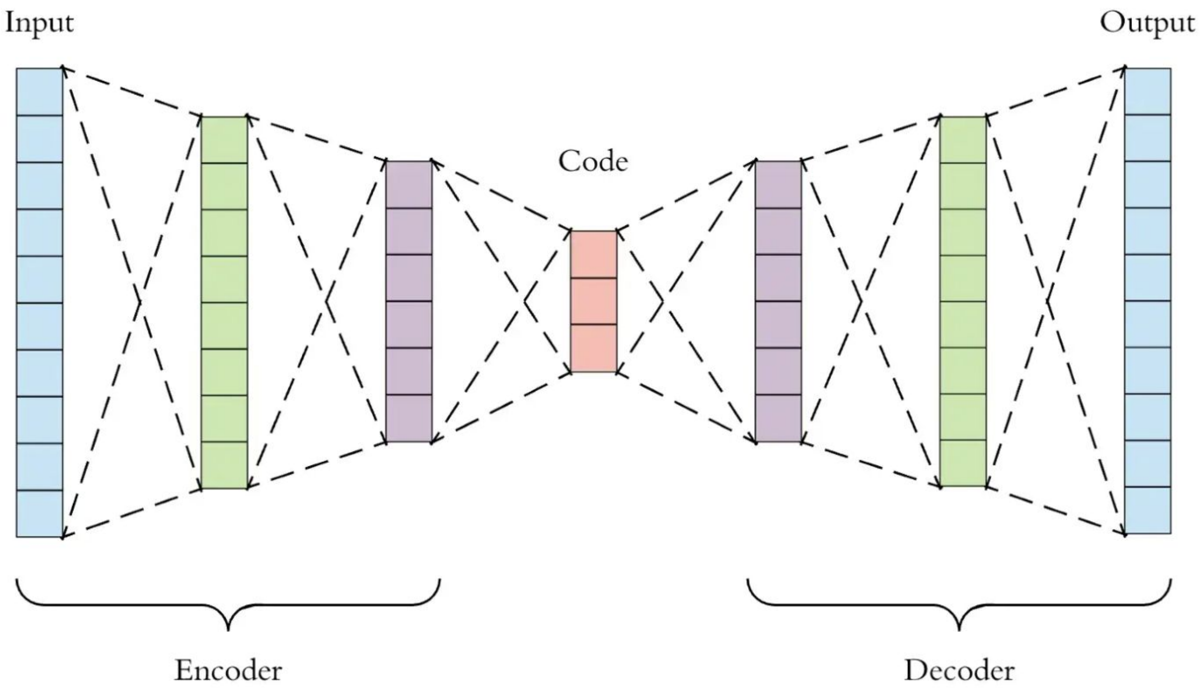
Noise in images



Images can suffer from various types of noise

We add Gaussian and salt-and-pepper noise to original image

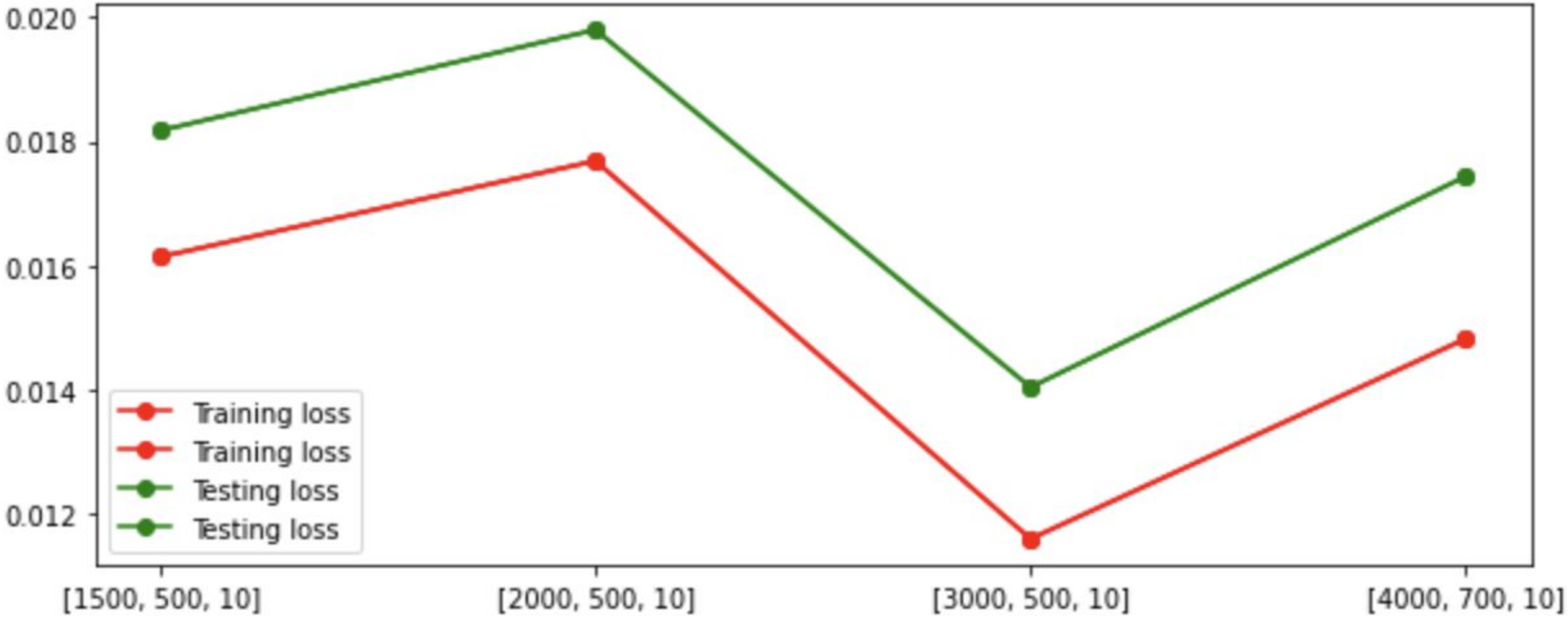
Fully Connected Autoencoder



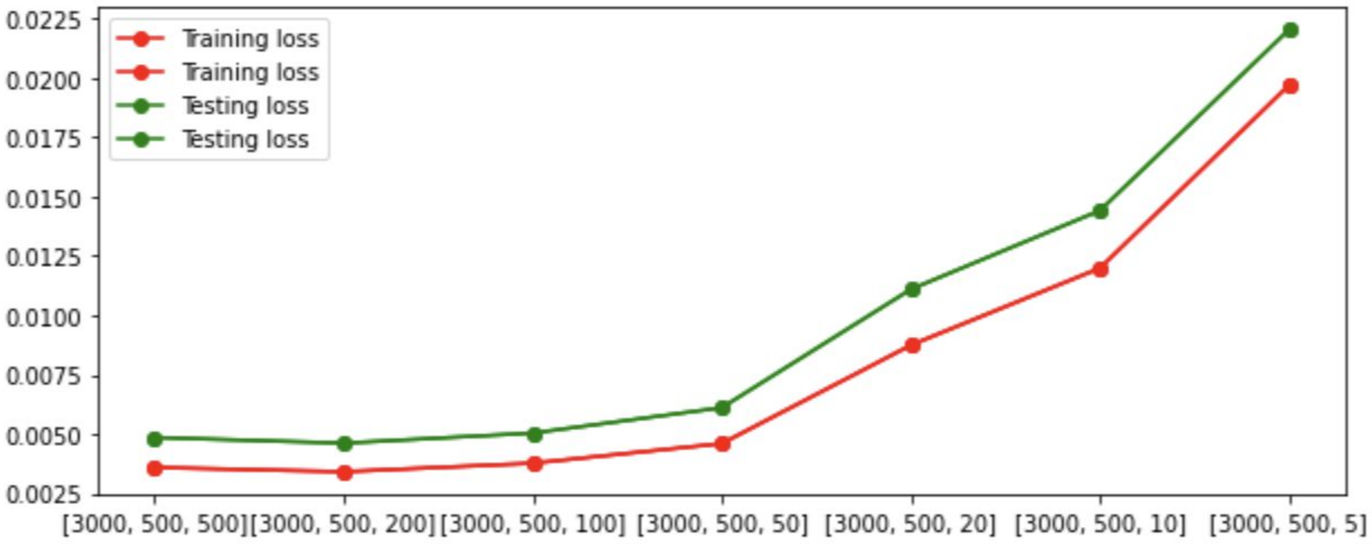
Architecture

- Both the encoder and decoder are fully-connected feedforward neural networks.
- Data is “compressed” at the layer of smallest width.
- Goal: To obtain an denoised image as output.

Results - Fully Connected

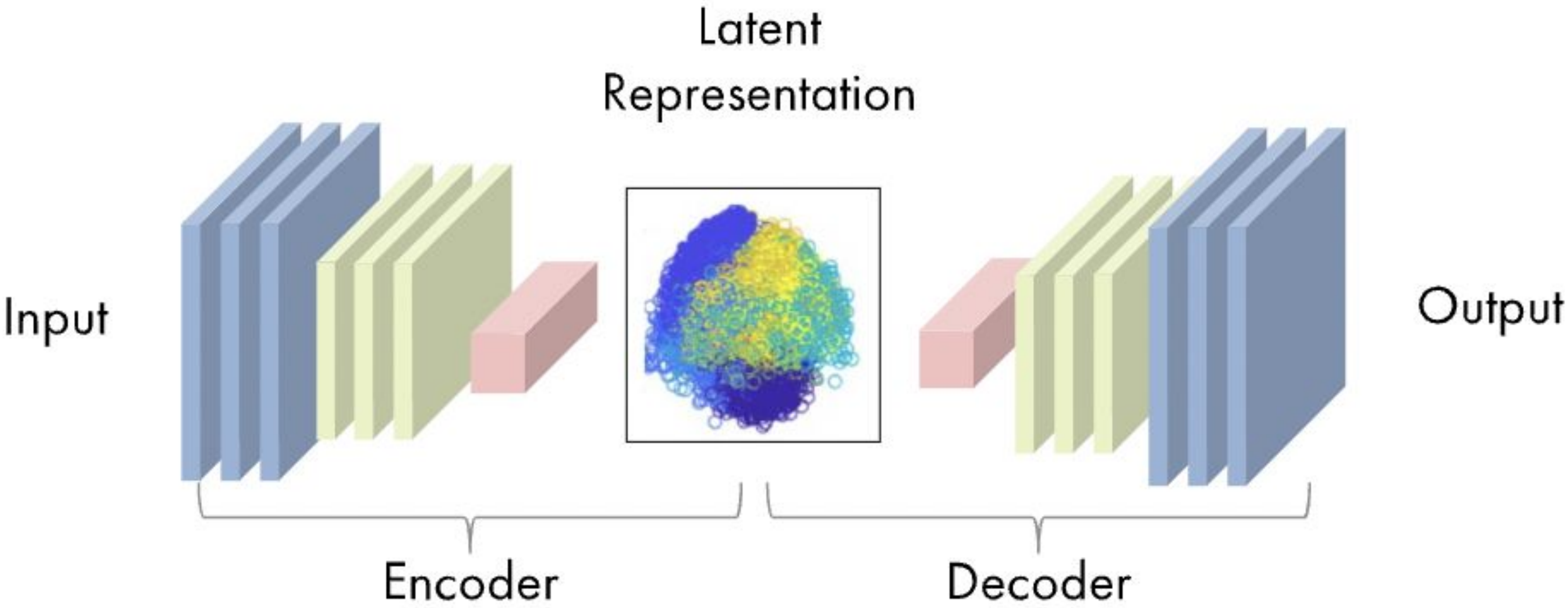


Results - Fully Connected



- If the feature map (middle) layer is too small, the reconstruction is not good due to the loss of data.
- The loss converges to a value for large width of narrowest (middle/feature map) layer.

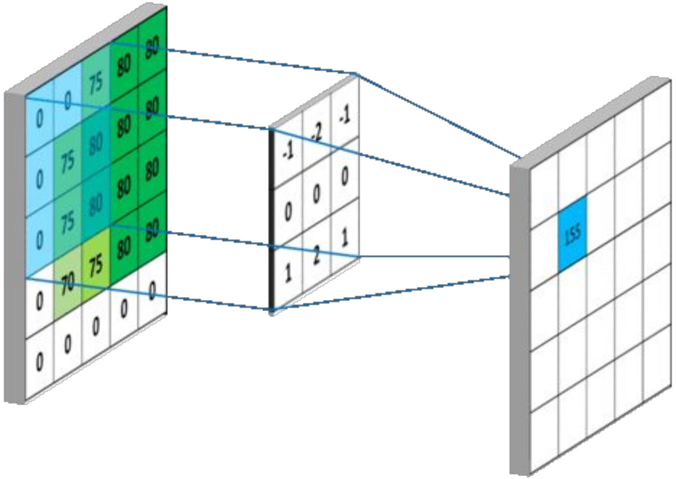
Convolutional Autoencoders



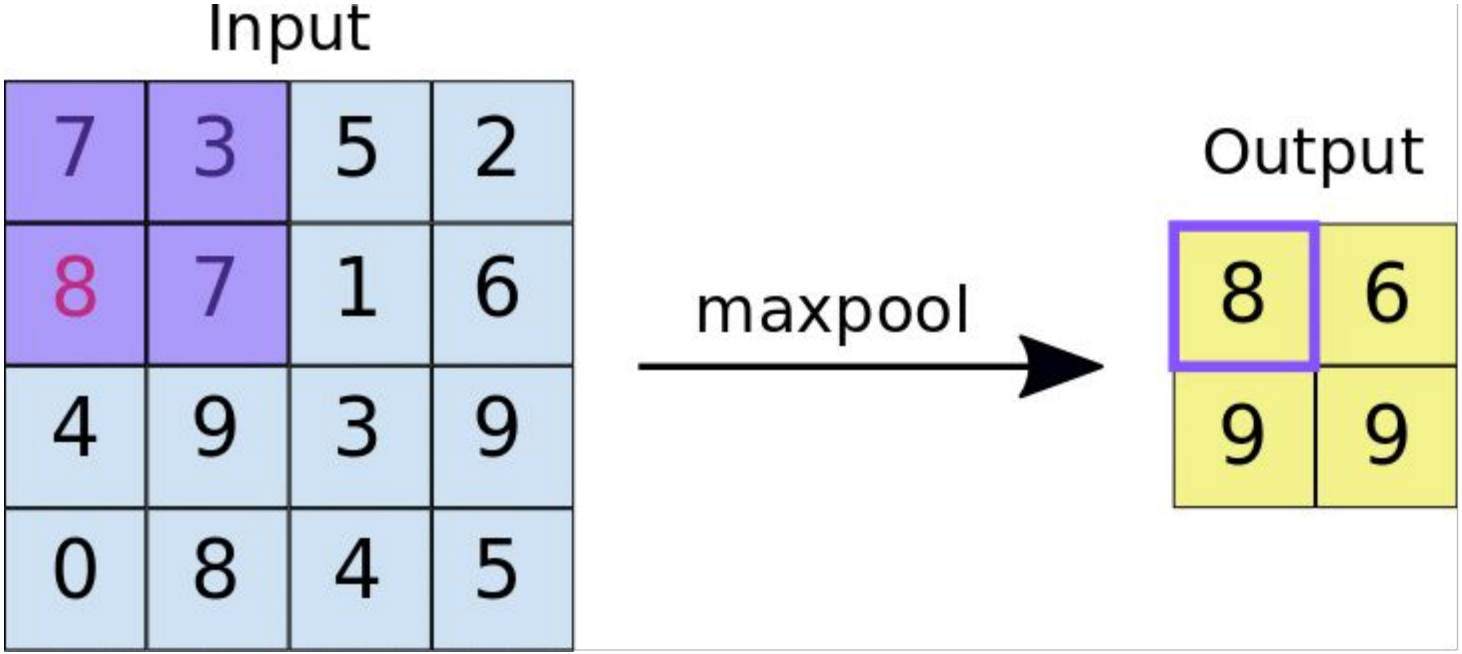
How Do the Convolutional Autoencoders Work?

1. Convolution Layer
    a. Padding
    b. Strides
2. ReLU
3. Max Pooling Layer

Convolutional Layer



Maxpool Layer



Model design

```
Input(shape=(28, 28, 1)),
Conv2D(16, (3, 3), activation='relu', padding='same', strides=2),
Conv2D(8, (3, 3), activation='relu', padding='same', strides=2),
Conv2DTranspose(8, kernel_size=3, strides=2, activation='relu', padding='same'),
Conv2DTranspose(16, kernel_size=3, strides=2, activation='relu', padding='same'),
Conv2D(1, kernel_size=(3, 3), activation='sigmoid', padding='same')
```

Results - CNN

Training loss:
0.0026

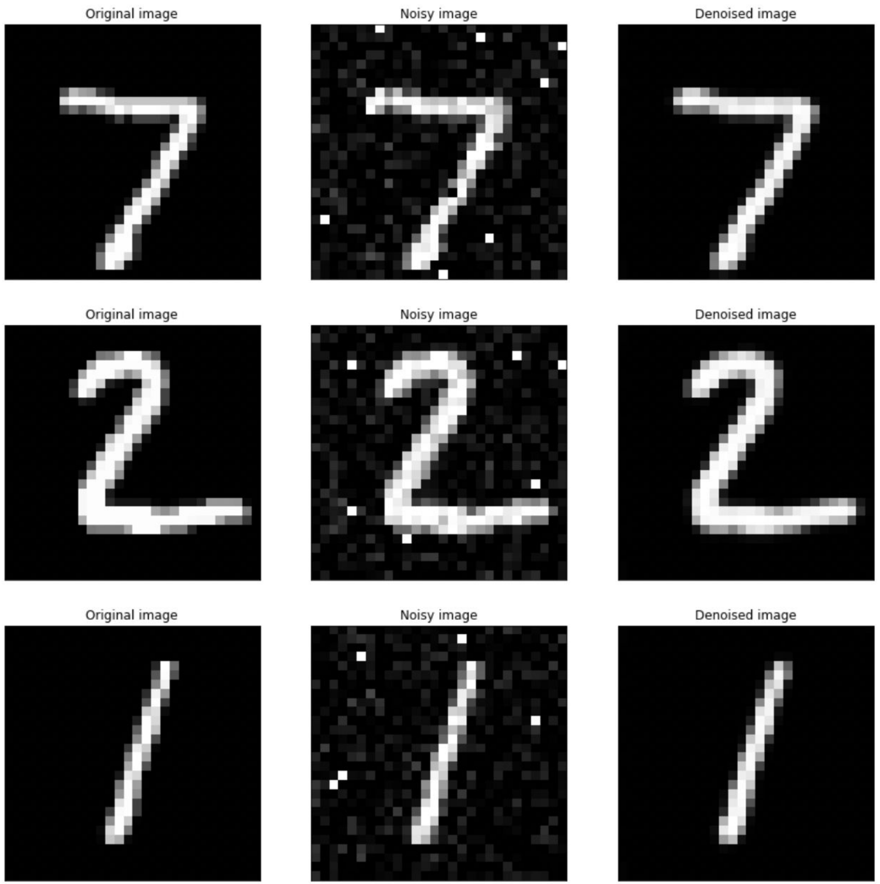
Testing loss:
0.0026

Total params: 3,217

VS

Best fully connected AE
Testing loss: 0.0049

Params: 7,721,794



KNN

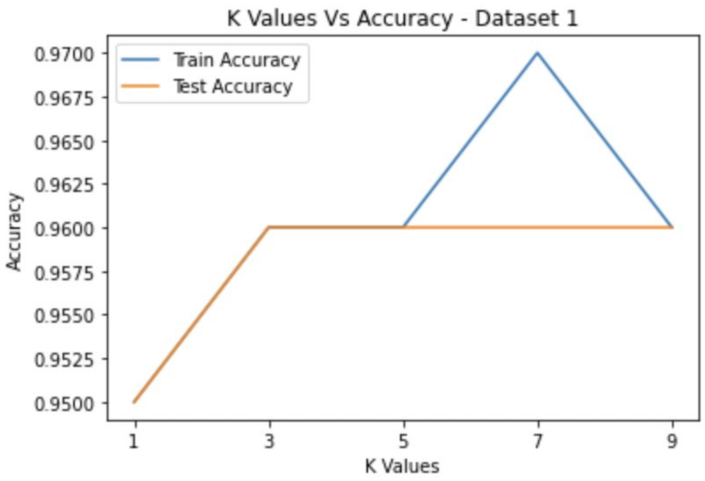
- Instance-based Learning or Lazy Learning: The function is only approximated locally and all computation is deferred until classification.
- Algorithm is used for both classification and regression.
- Classification: The KNN algorithm works by finding the closest training examples in the feature space and using their class labels to predict the class of the new example.
- Regression: The KNN algorithm predicts the value of the new example by taking the average of the values of its nearest neighbors.
- KNN is called a non-parametric method because it does not make any assumptions about the functional form of the underlying distribution of the data.

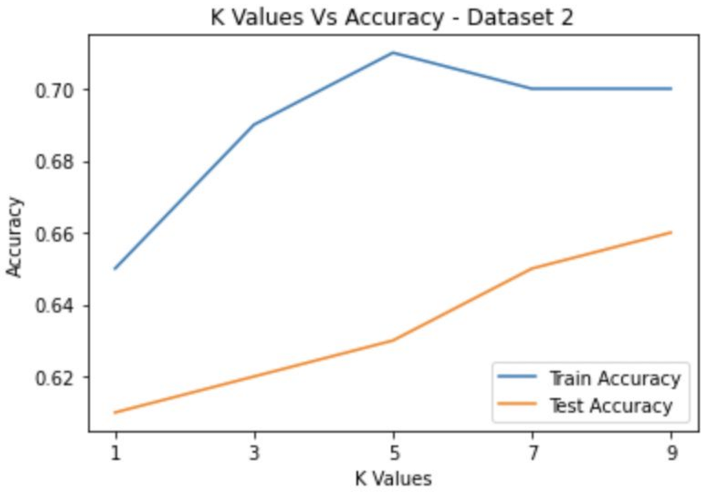
Hyperparameter Testing was performed on the following:

1. K value: 1,3,5,7,9
2. Distance: euclidean, manhattan, minkowski, cosine, jaccard, hamming

KNN - Hyperparameter Testing

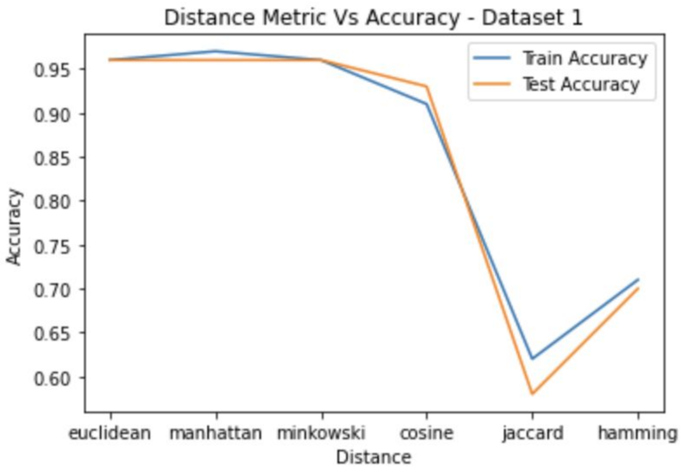
Determining Optimal K value

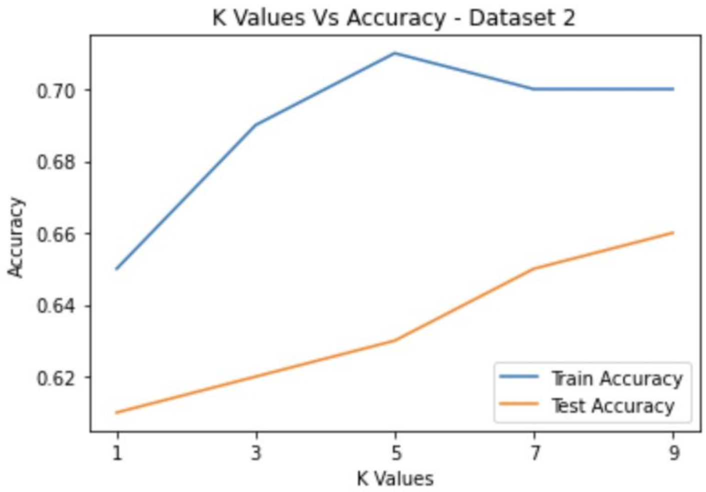




KNN - Hyperparameter Testing

Determining Optimal Distance Metrics





KNN - Hyperparameter Testing

Best Parameters for Dataset -1

- **K = 3, 5, 7 and 9** with test accuracy of 0.96
- Distance metric = **Euclidean, Manhattan. Minkowski** with test accuracy of 0.96

Best Parameters for Dataset - 2

- **K = 9** with test accuracy of 0.66
- **Distance metric = Hamming** with test accuracy of 0.72

Decision Tree

- Comes under the concept of supervised learning.
- Next step is decided based on the response of the previous step.
- We have used the gini index to evaluate the splits in the dataset.

- The accuracy obtained on the dataset 1 is : 91.2%
- The accuracy obtained on the dataset 2 is : 64.7%

Decision Tree - Hyperparameter Tuning

- Used gridsearchCV for hyperparameter tuning.
- Parameter grid :
  - 'criterion' : ['gini','entropy'],
  - 'max_depth': [2, 3, 5],
  - 'min_samples_split': [2, 3, 5],
  - 'min_samples_leaf': [1,5,8]
- For dataset 1 : The maximum accuracy obtained from using these best parameters is : 96%, Precision : 0.96, Recall : 0.97, f1-score : 0.97, AUC : 0.95
- For dataset 2 : The maximum accuracy obtained from using these best parameters is : 70%, Precision : 0.71, Recall : 0.91, f1-score : 0.80, AUC : 0.5874

Decision Tree - Best Parameters

- Dataset 1 :
    - 'criterion': entropy
    - 'max_depth': 3
    - 'min_samples_leaf': 1
    - 'min_samples_split': 2
- Dataset 2: '
    - criterion': 'gini'
    - 'max_depth': 2
    - 'min_samples_leaf': 5
    - min_samples_split': 2

Random Forest

- Group of decision trees with very low correlation between them.
- Class of supervised machine learning algorithm.

- The accuracy we obtained for the dataset 1 is : 95.9%
- The accuracy obtained for the dataset 2 is : 64.74%

Random Forest - Hyperparameter Tuning

- Parameter grid :
  - 'max_depth': 10
  - 'max_features': 'auto'
  - 'min_samples_leaf': 1
  - 'min_samples_split': 2
  - 'n_estimators': 200
- For dataset 1 : The maximum accuracy obtained from using these best parameters is : 96%, Precision : 0.97, Recall : 0.96, f1-score : 0.97, AUC : 0.9577
- For dataset 2 : The maximum accuracy obtained from using these best parameters is : 70%, Precision : 0.78, Recall : 0.85, f1-score : 0.81, AUC : 0.6856

Random Forest - Best Parameters

- Dataset 1 :
  - 'max_depth': 10
  - 'max_features': 'auto'
  - 'min_samples_leaf': 1
  - 'min_samples_split': 2
  - 'n_estimators': 200
- Dataset 2 :
  - 'max_depth': 20
  - 'max_features': 'auto'
  - 'min_samples_leaf': 1
  - 'min_samples_split': 5
  - 'n_estimators': 300

Support Vector Machine

- Employed to find the hyperplane that would best divide the dataset into 2 classes.
- SVM algorithm chooses a hyperplane as the best when the distance between the hyperplane and the support vectors are to be maximum.

- The accuracy on the test set for dataset 1 was found to be : 95.6%
- The accuracy on the test set for dataset 2 was found to be : 79.56%

SVM - Hyperparameter Tuning

- Parameter grid :
  - 'C': [0.1, 1, 10]
  - 'gamma': [1, 0.1, 0.01]
  - 'kernel': ['linear']
- For dataset 1 : The maximum accuracy obtained from using these best parameters is : 97%, Precision : 0.96, Recall : 0.91, f1-score : 0.93, AUC : 0.9152
- For dataset 2 : The maximum accuracy obtained from using these best parameters is : 77%, Precision : 0.79, Recall : 0.90, f1-score : 0.84, AUC : 0.7097

SVM - Best Parameters

- Dataset 1 :
  - 'C': 1
  - 'gamma': 1
  - 'kernel': 'linear'
- Dataset 2 :
  - 'C': 10
  - 'gamma': 1
  - 'kernel': 'linear'

Boosting

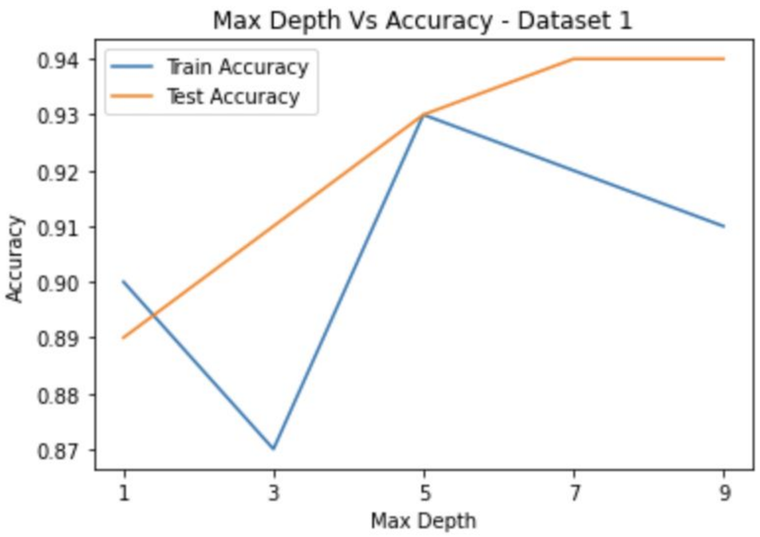
- Ensemble learning algorithm.
- Goal: To combine the weak models into a single strong model that is able to make accurate predictions on new data.
- Involves training a sequence of weak models in a way that each model tries to correct the mistakes of the previous model.
- The model implemented uses decision trees as the weak models.

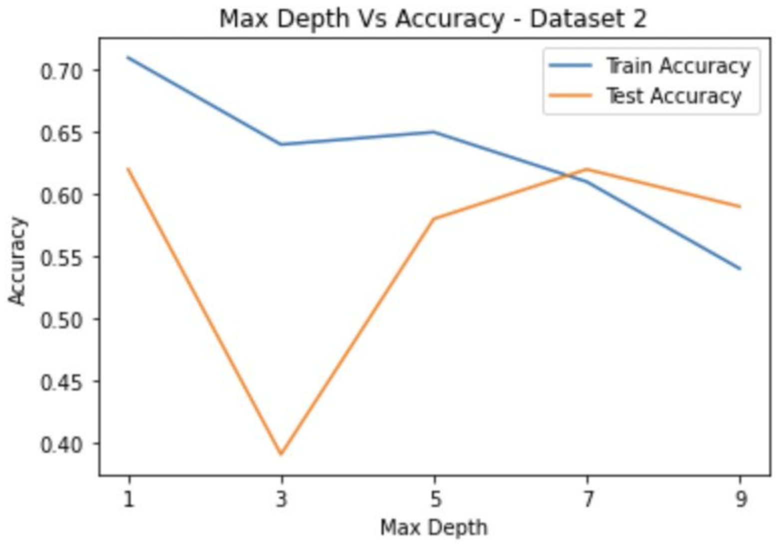
Hyperparameter Testing was performed on the following:

1. Max_depth

Boosting - Hyperparameter Testing

Determining Optimal Max Depth value





Boosting - Hyperparameter Testing

Best Parameters for Dataset -1

- **Max Depth = 7 and 9** with test accuracy of 0.94

Best Parameters for Dataset - 2

- **Max Depth = 1 and 7** with test accuracy of 0.62

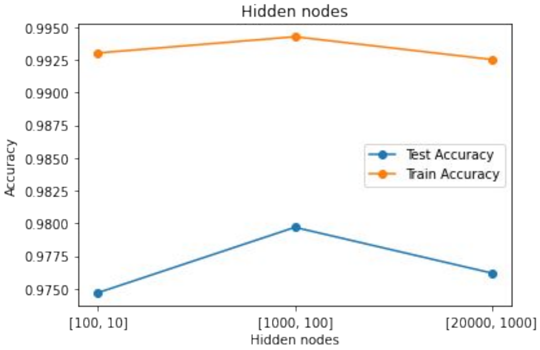
Neural Network

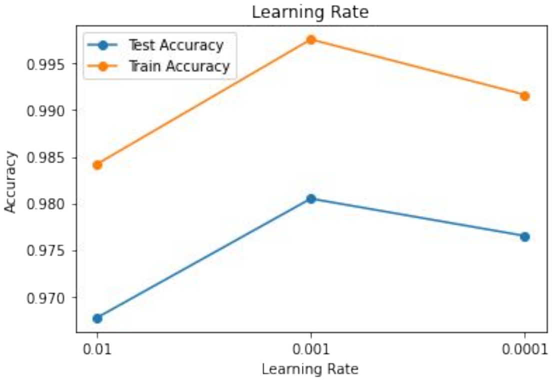
- Neural networks are a type of machine learning algorithm that is inspired by the structure and function of the human brain.
- They are composed of many interconnected units called neurons, which process and transmit information.
- The weights of the connection b/w neurons are adjusted to fit the model to the training data.
- The algorithm uses a process called gradient descent, which involves calculating the gradient of the error with respect to the weights of the connections between neurons.
- This gradient is then used to update the weights in a way that reduces the error.
- This process is repeated for many iterations, allowing the network to gradually learn and improve its performance.

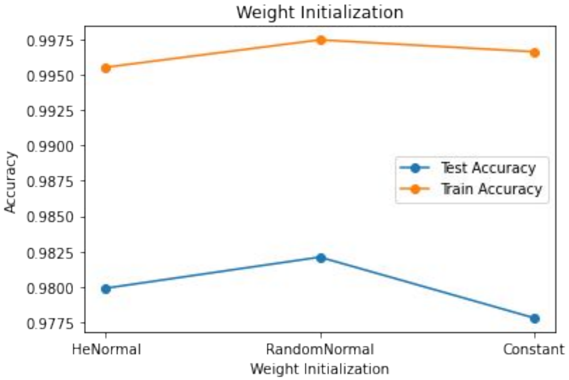
Neural Network - Hyperparameter tuning

- Hidden nodes tested:[100, 10], [1000, 100], [20_000, 1000]
- Learning rate tested: 0.01, 0.001, 0.0001
- Weight initialization tested: HeNormal(), RandomNormal(mean=0.0, stddev=0.05, seed=42), Constant(0.1)
- Best hyperparameters: Hidden units = [1000, 100], learning rate = 0.001, Weight initialization = <keras.initializers.initializers_v2.RandomNormal object at 0x7fc29e4a2e80>

Neural Network - Accuracies for different hyperparameters







Logistic Regression

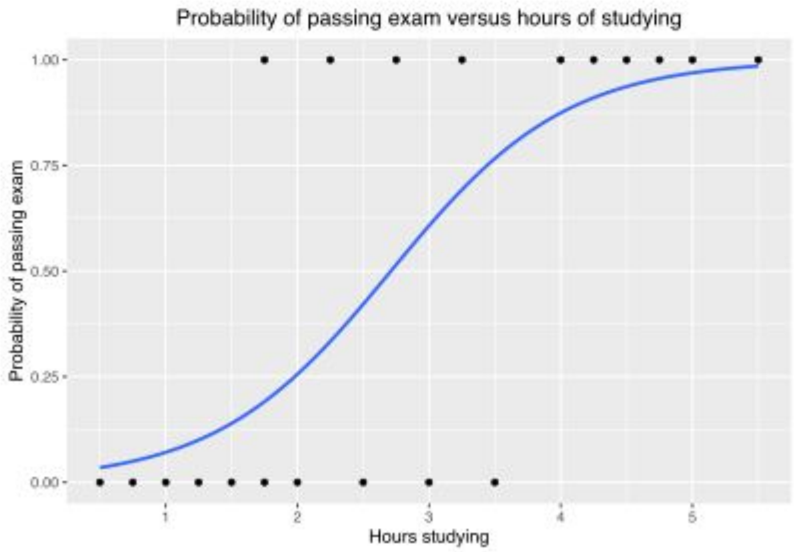
- The algorithm fits the given data to a sigmoid / logistic function using gradient descent.
- The equation for logistic regression can be written as,

  $$p(x) = 1/(1 + e^{-\beta^T x'}), \quad where\ x' = [1 \quad x^T]^T$$

  Here the parameter to estimate is beta.

- For Logistic Regression, no hyperparameter tuning is required.

Logistic Regression



Dataset 1

Average across folds:
Accuracy = 0.965, precision = 1.000, recall = 0.903, f1 = 0.948, AUC = 0.952
Test set results:
Accuracy = 0.956, precision = 0.956, recall = 0.935, f1 = 0.945, AUC = 0.953

Dataset 2

Average across folds:
Accuracy = 0.729, precision = 0.652, recall = 0.520, f1 = 0.575, AUC = 0.683
Test set results:
Accuracy = 0.785, precision = 0.615, recall = 0.615, f1 = 0.615, AUC = 0.733

Thank You!

Q & A