# Assignment-I Report

# DNNGP for Genomic Prediction

## Problem Statement:

### Problem:

Genomic selection is crucial for plant breeding, helping predict desirable traits from genetic data. Traditional methods like GBLUP (Genomic Best Linear Unbiased Prediction) struggle with complex genotype-phenotype relationships due to their linear assumptions. While machine learning (ML) methods like SVR (Support Vector Regression) improve accuracy, they still face challenges in handling high-dimensional genomic data efficiently.

### Limitations of Existing Methods:

- Linear models (e.g., GBLUP) fail to capture non-linear genetic interactions.  
- Machine learning models (e.g., SVR, LightGBM) require extensive hyperparameter tuning and may struggle with high-dimensional datasets.  
- Deep Learning models need large datasets and optimization to prevent overfitting.

### Proposed Method:

We implement DNNGP (Deep Neural Network Genomic Prediction), a deep learning-based model, to enhance genomic selection accuracy by:  
1. Using deep neural networks to model complex non-linear genetic interactions.  
2. Employing PCA-transformed genomic features (PC1–PC251) from the Wheat599 dataset.  
3. Comparing DNNGP’s performance against SVR to evaluate prediction improvements.

## 2. Requirements Gathering

### Abstract :

Genomic prediction accelerates crop breeding by estimating breeding values from genetic data. Traditional methods like GBLUP have limited accuracy, while machine learning methods (e.g., SVR) improve predictions but struggle with high-dimensional data. This study implements DNNGP (Deep Neural Network Genomic Prediction) on the Wheat599 dataset, applying deep learning with PCA features to improve phenotype predictions. We compare DNNGP with SVR (Support Vector Regression) using Mean Squared Error (MSE) and R² score. Our results highlight DNNGP’s predictive power, demonstrating its potential for large-scale genomic selection applications.

### Dataset Description

- Dataset Used: Wheat599  
- Features: 251 Principal Components (PC1-PC251) derived from genomic markers.  
- Target Label: env1 (environmental phenotype response for genomic prediction).  
- Sample Size: 599 entries (M1–M599).

### Target Label Analysis

**env1** is the dependent variable representing an environmental factor affecting plant phenotype. Values range between -3 to +3, requiring normalization.

### Type of Model or Network

- DNNGP (Deep Neural Network with Fully Connected Layers).  
- SVR (Support Vector Regression) for comparison.

### Metrics Used

- Mean Squared Error (MSE): Measures prediction error.  
- R² Score: Evaluates model accuracy.

## 3. Implementation

Chosen methodology includes data preprocessing, model selection, and comparison between deep learning and machine learning models.

**Code:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error, r2\_score

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization

# File paths

wheat1\_path = "/content/wheat1.tsv"

wheat599\_pc95\_tsv\_path = "/content/wheat599\_pc95.tsv"

# Load datasets

wheat1\_df = pd.read\_csv(wheat1\_path, sep="\t")

wheat599\_pc95\_tsv\_df = pd.read\_csv(wheat599\_pc95\_tsv\_path, sep="\t")

# Merge datasets on 'ID'

df = pd.merge(wheat599\_pc95\_tsv\_df, wheat1\_df, on="ID")

# Preprocessing

def preprocess\_data(df):

X = df.drop(columns=["env1", "ID"]) # Features

y = df["env1"] # Target

# Normalize Features

scaler = StandardScaler()

X\_scaled = scaler.fit\_transform(X)

# Split Data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)

return X\_train, X\_test, y\_train, y\_test, scaler

X\_train, X\_test, y\_train, y\_test, scaler = preprocess\_data(df)

# Build DNNGP Model

def build\_dnngp(input\_shape):

model = Sequential([

Dense(128, activation='relu', input\_shape=(input\_shape,)),

BatchNormalization(),

Dropout(0.3),

Dense(64, activation='relu'),

BatchNormalization(),

Dropout(0.3),

Dense(32, activation='relu'),

Dense(1, activation='linear') # Regression output

])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])

return model

# Train DNNGP

model = build\_dnngp(X\_train.shape[1])

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

# Predictions & Evaluation

y\_pred\_dnn = model.predict(X\_test).flatten()

mse\_dnn = mean\_squared\_error(y\_test, y\_pred\_dnn)

r2\_dnn = r2\_score(y\_test, y\_pred\_dnn)

# Train SVR for Comparison

svr = SVR()

svr.fit(X\_train, y\_train)

y\_pred\_svr = svr.predict(X\_test)

mse\_svr = mean\_squared\_error(y\_test, y\_pred\_svr)

r2\_svr = r2\_score(y\_test, y\_pred\_svr)

# Results

print("DNNGP Performance: ", "MSE:", mse\_dnn, "R²:", r2\_dnn)

print("SVR Performance: ", "MSE:", mse\_svr, "R²:", r2\_svr)

# Plot Loss Curve

plt.figure(figsize=(10, 5))

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.title('DNNGP Training Loss Curve')

plt.legend()

plt.show()

# Scatter Plot of Actual vs. Predicted

plt.figure(figsize=(8, 6))

sns.scatterplot(x=y\_test, y=y\_pred\_dnn, label='DNNGP', color='blue')

sns.scatterplot(x=y\_test, y=y\_pred\_svr, label='SVR', color='red')

plt.plot(y\_test, y\_test, color='black', linestyle='--') # Perfect predictions line

tlplt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title('Actual vs. Predicted Values')

plt.legend()

plt.show()

# Residual Plot

plt.figure(figsize=(8, 6))

sns.histplot(y\_test - y\_pred\_dnn, bins=30, kde=True, color='blue', label='DNNGP Residuals')

sns.histplot(y\_test - y\_pred\_svr, bins=30, kde=True, color='red', label='SVR Residuals')

plt.xlabel('Residuals')

plt.title('Residual Distribution')

plt.legend()

plt.show()

## 4. Execution

**Steps performed**:  
🔹 Load & Merge Data: Combine wheat599\_pc95.tsv (features) with wheat1.tsv (target values).  
🔹 Feature Scaling: Standardize PC1-PC251 values using StandardScaler().  
🔹 Train-Test Split: 80% training, 20% testing to validate performance.  
🔹 DNNGP Training: 50 epochs, batch size 32, Adam optimizer for training.  
🔹 SVR Training: Fit Support Vector Regression on the same dataset.  
🔹 Performance Evaluation: Compare MSE and R² scores between DNNGP vs. SVR.

## 5. Results & Comparison

|  |  |  |
| --- | --- | --- |
| Model | Mean Squared Error (MSE) | R² Score (Higher = Better) |
| DNNGP (Deep Learning) | 0.695 | 0.269 |
| SVR (Support Vector) | 0.641 | 0.326 |

### Observations:

* SVR outperformed DNNGP, showing lower error (MSE) and higher accuracy (R² score).
* Deep Learning models may require hyperparameter tuning to achieve better performance.
* Feature selection (reducing dimensions or selecting relevant PCs) may improve results.