Predicting and Estimating Crop Yields with Deep Learning

Kao Chiam Saelee
DTSA-5511

INTRODUCTION

The application of deep learning techniques to train models for forecasting future outcomes are powerful and practical tools. Agriculture, a pivotal component of the global economy, necessitates a comprehensive understanding of worldwide crop yield as the human population continues to grow. This knowledge is crucial for tackling food security issues and mitigating the effects of climate change.

The prediction of crop yield stands out as a significant challenge in agriculture. Weather conditions (such as rainfall, temperature, etc.), pesticide usage, and precise historical data on crop yield play critical roles in making informed decisions related to agricultural risk management and future predictions.

INTRODUCTION

For this project, we'll be using the Crop Yield Prediction Dataset to predict crop yields of the 10 most consumed crops in the world.

We'll build out, tune and train different deep learning architectures such as the Convolutional Neural Network (CNN), Deep Neural Network (DNN) and Recurrent Neural Network (RNN) to predict yield, utilizing data such as pesticide use, rainfall and temperature and features in our prediction.

We will progress with the hyperparameter tuning a deep learning model architecture and ultimately compare all of those model performances to machine learning models such as linear regression and random forest.

OVERVIEW

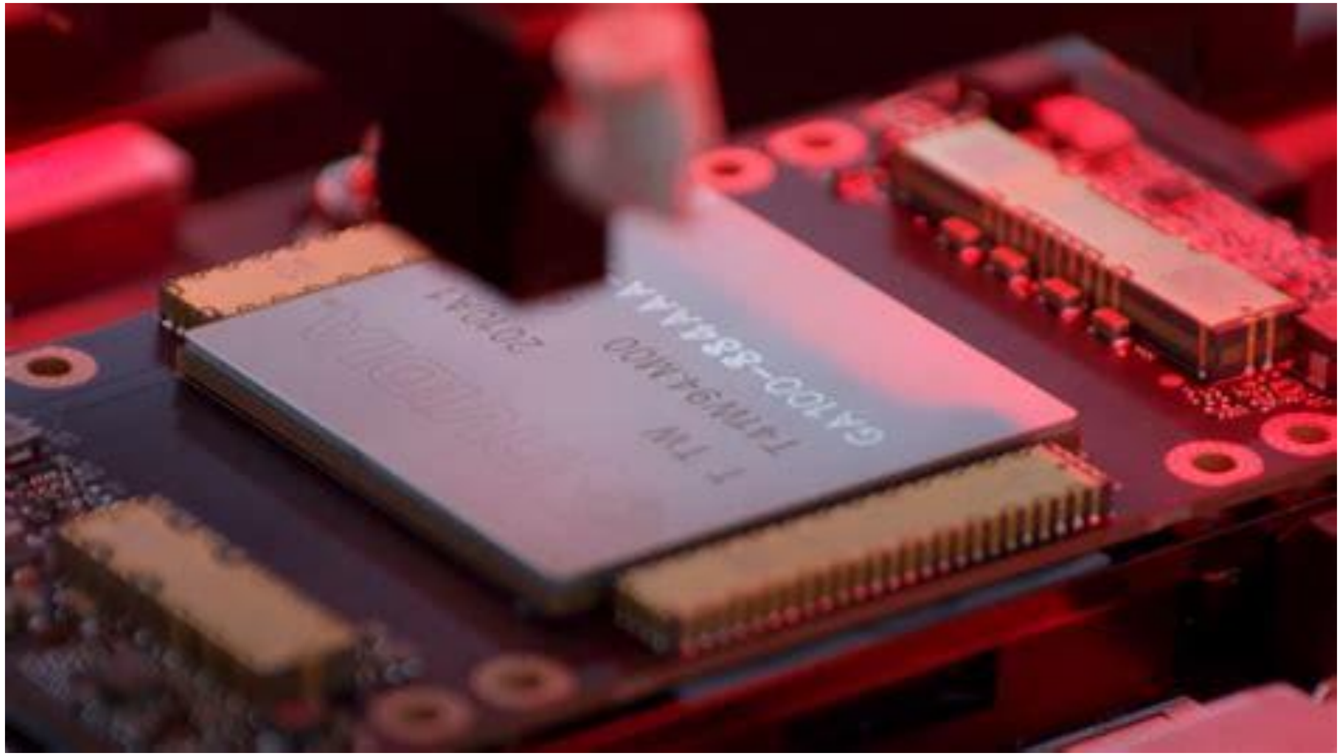
1. Setting Up Environment
2. Exploratory Data Analysis (EDA)
3. Data Preprocessing
4. Convolutional Neural Network
5. Deep Neural Network
6. Recurrent Neural Network
7. Model performance comparison
8. Hyperparameter tuning
9. Comparison to other Machine Learning models
10. Discussion/ Summary
11. Final Remarks

SETTING UP THE ENVIRONMENT

IMPORT PACKAGES

- Numpy and Pandas
- Scikit-learn
- Tensorflow and Keras

MOUNT SYSTEM’S GPU



EXPLORATORY ANALYSIS AND VISUALIZATION (EDA)

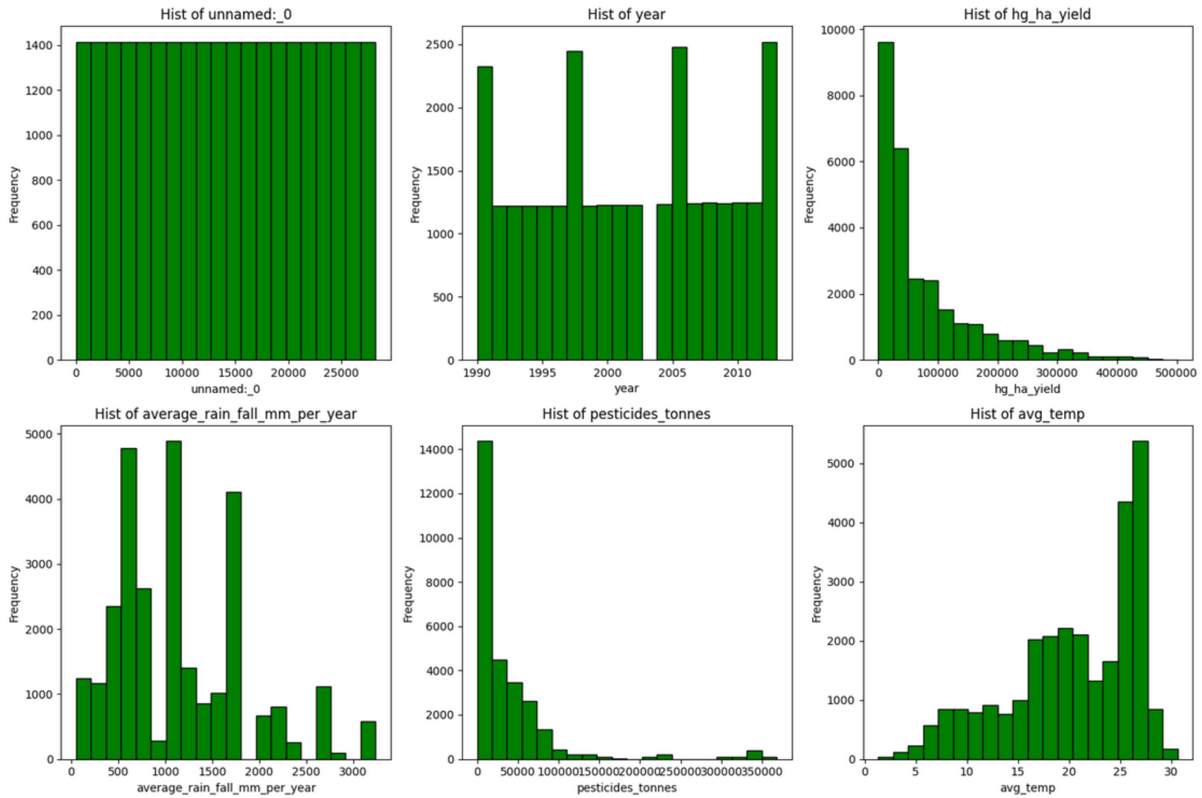
1. View our loaded dataset
2. Clean up the column names
3. Check the shape and data types
4. Assess for missing or duplicate data
5. Graph and view the numeric columns
6. Visualize the categorical columns
7. Final cleaning or column headers
8. Correlation analysis of the features

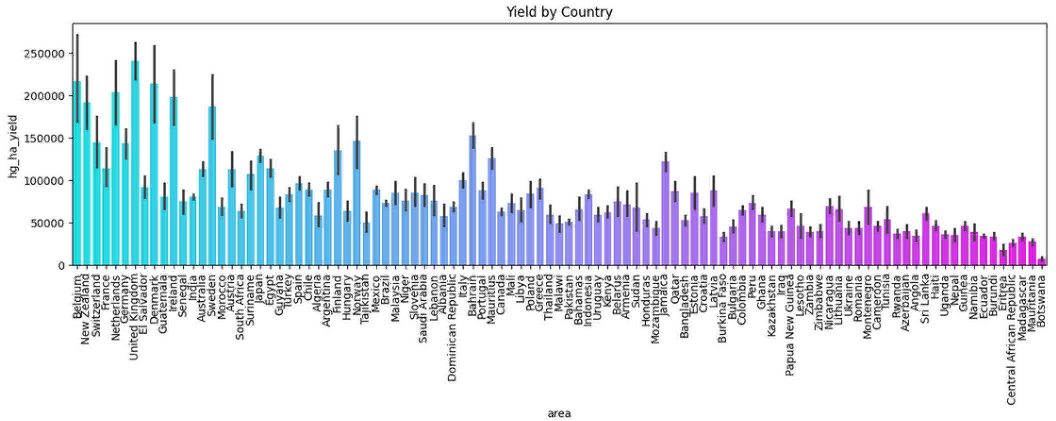
	Unnamed: 0	Year	hg/ha_yield	average_rain_fall_mm_per_year	pesticides_tonnes	avg_temp
count	28242.000000	28242.000000	28242.000000	28242.00000	28242.000000	28242.000000
mean	14120.500000	2001.544296	77053.332094	1149.05598	37076.909344	20.542627
std	8152.907488	7.051905	84956.612897	709.81215	59958.784665	6.312051
min	0.000000	1990.000000	50.000000	51.00000	0.040000	1.300000
25%	7060.250000	1995.000000	19919.250000	593.00000	1702.000000	16.702500
50%	14120.500000	2001.000000	38295.000000	1083.00000	17529.440000	21.510000
75%	21180.750000	2008.000000	104676.750000	1668.00000	48687.880000	26.000000
max	28241.000000	2013.000000	501412.000000	3240.00000	367778.000000	30.650000

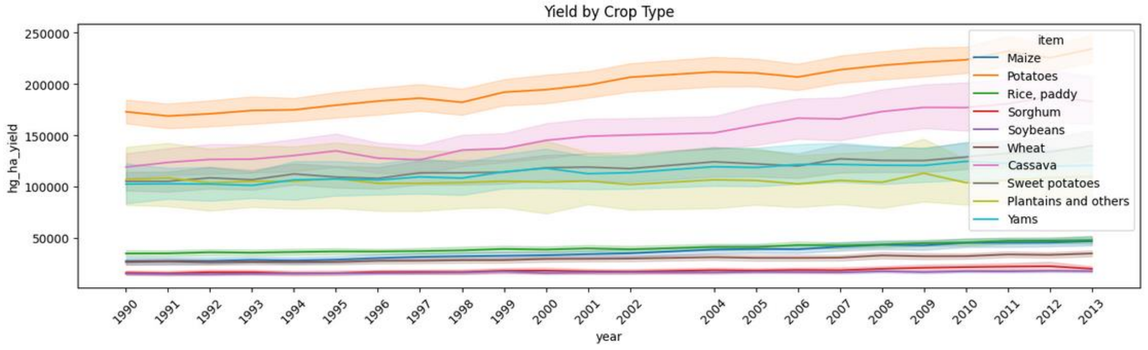
```
Missing Data:
 unnamed:_0                       0
area                             0
item                             0
year                             0
hg_ha_yield                      0
average_rain_fall_mm_per_year    0
pesticides_tonnes                0
avg_temp                         0
dtype: int64

Duplicate Rows:
 Empty DataFrame
Columns: [unnamed:_0, area, item, year, h
Index: []
```

EXPLORATORY ANALYSIS AND VISUALIZATION (EDA)

1. View our loaded dataset
2. Clean up the column names
3. Check the shape and data types
4. Assess for missing or duplicate data
5. Graph and view the numeric columns
6. Visualize the categorical columns
7. Final cleaning or column headers
8. Correlation analysis of the features

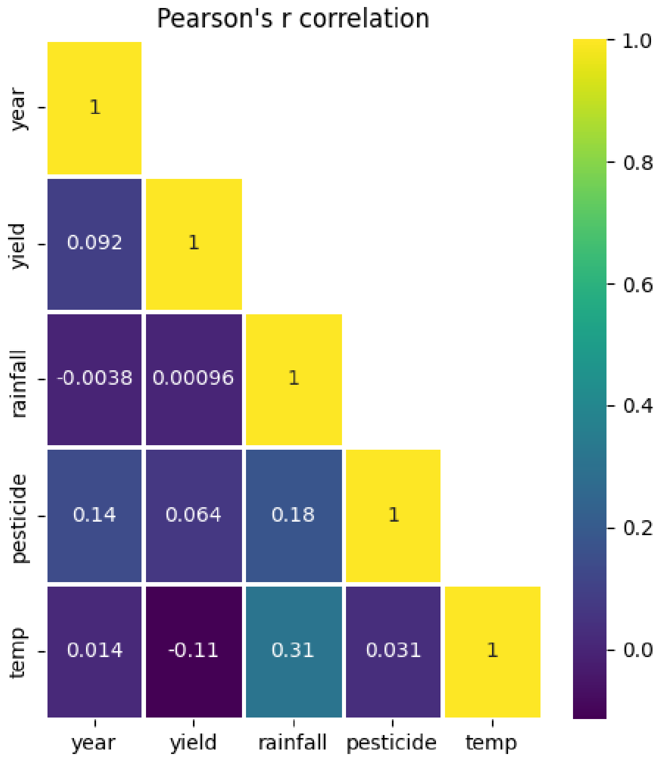


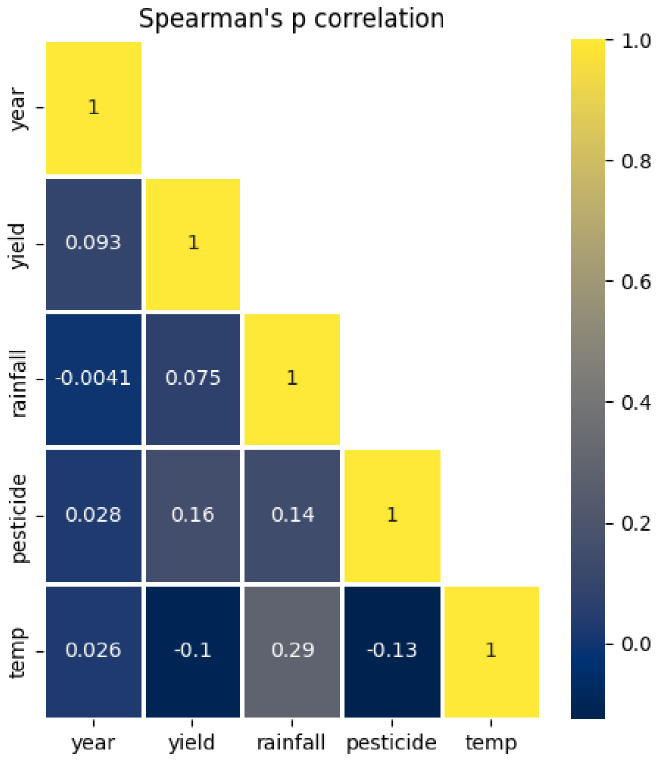




EXPLORATORY ANALYSIS AND VISUALIZATION (EDA)

1. View our loaded dataset
2. Clean up the column names
3. Check the shape and data types
4. Assess for missing or duplicate data
5. Graph and view the numeric columns
6. Visualize the categorical columns
7. Final cleaning or column headers
8. Correlation analysis of the features





area	crop	year	yield	rainfall	pesticide	temp
Albania	Maize	1990	36613	1485.0	121.0	16.37
Albania	Potatoes	1990	66667	1485.0	121.0	16.37
Albania	Rice, paddy	1990	23333	1485.0	121.0	16.37
Albania	Sorghum	1990	12500	1485.0	121.0	16.37
Albania	Soybeans	1990	7000	1485.0	121.0	16.37

DATA PREPROCESSING

```
## Split the data into features (X) and target variable (y)
X_numeric = df2.drop(['area', 'crop', "year",'yield'], axis=1)
X_categorical = pd.get_dummies(df2[['area', 'crop', 'year']])
X = pd.concat([X_numeric, X_categorical], axis=1)
y = df2['yield'].values

##random seed
rs= 1234

##convert booleans to numeric of false= 0, and true= 1
X= X.astype(int)

##Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=rs)

## Standardize the numeric features using StandardScaler
# scaler = StandardScaler()
scaler= MinMaxScaler()
# Xtrain_sc = scaler.fit_transform(X_train.iloc[:, :X_numeric.shape[1]])
# Xtest_sc = scaler.transform(X_test.iloc[:, :X_numeric.shape[1]])
Xtrain_sc = scaler.fit_transform(X_train)
Xtest_sc = scaler.transform(X_test)
```

```
array([[0.06240201, 0.03680209, 0.17241379, ..., 0.        , 0.        ,
        0.        ],
       [0.18344309, 0.07573047, 0.55172414, ..., 0.        , 0.        ,
        0.        ],
       [0.69708373, 0.00354018, 0.72413793, ..., 0.        , 0.        ,
        0.        ],
```

CONVOLUTIONAL NEURAL NETWORK (CNN)

CNNs are good for tasks where spatial information is important such as image data.

If the dataset consists of a sequential structure or related neighboring datapoints, CNNs can capture those patterns effectively.

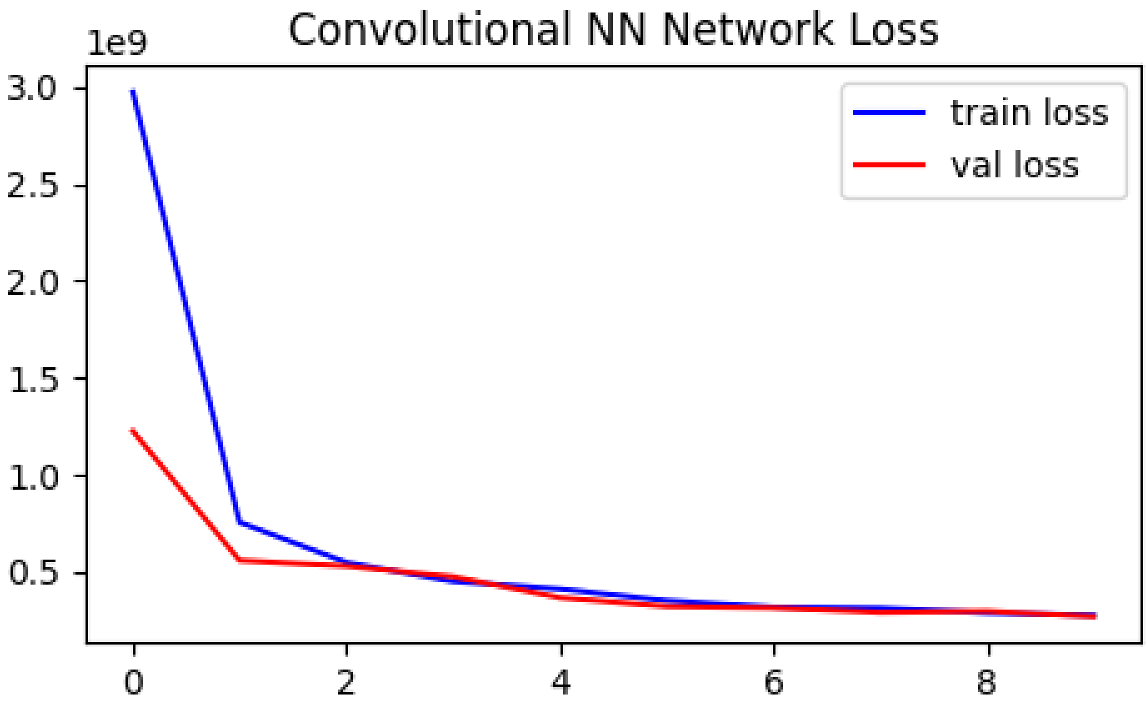
1. A Conv1D layer with 32 filters and a size of 3 with the data's input shape of (115,1). The activation = RELU
2. maxpooling layer after the conv layer to down-sample the spatial dims of the input volume
3. A flatten layer to flatten data into one dimenisonal array to prepare for fully connect layers
4. A fully connected layer with 64 naurons and RELU activation function
5. The last output layer with a single neuron for regression

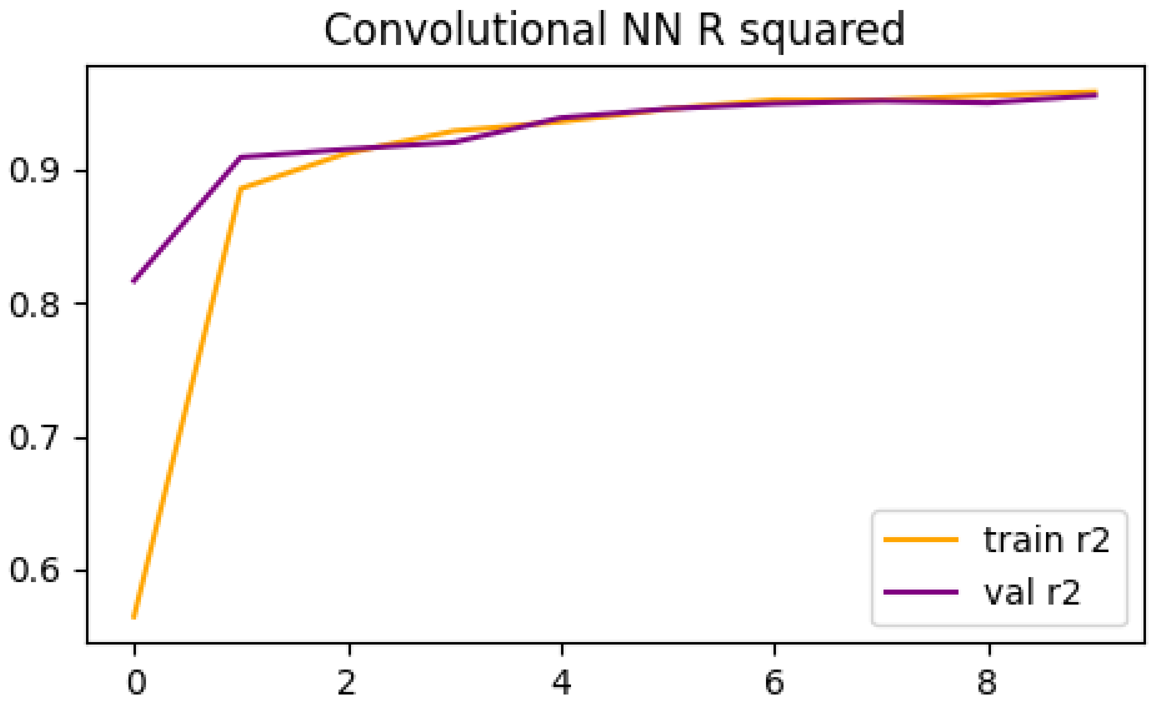
Optimizer= Adam, Learning rate= 0.1, Loss= MSE, R-squared

MODEL ARCHITECTURE

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 113, 32)           128

 max_pooling1d (MaxPooling1D  (None, 56, 32)           0
 )

 flatten (Flatten)           (None, 1792)              0

 dense (Dense)               (None, 64)                114752

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 114,945
Trainable params: 114,945
Non-trainable params: 0
```

CONVOLUTIONAL NEURAL NETWORK (CNN)





Our base CNN did a really good job in predicting the yield values without over or underfitting. Being a deep learning model, it is not surprising that it performs well. The model can be built upon in many different ways to achieve a better outcome.

DEEP NEURAL NETWORK (DNN)

DNNs are versatile and suited for a wide range of data types including tabular data where spatial or sequential data may be lacking.

DNNs are great for data that are very complex because it's innate deep architecture, hence the name.

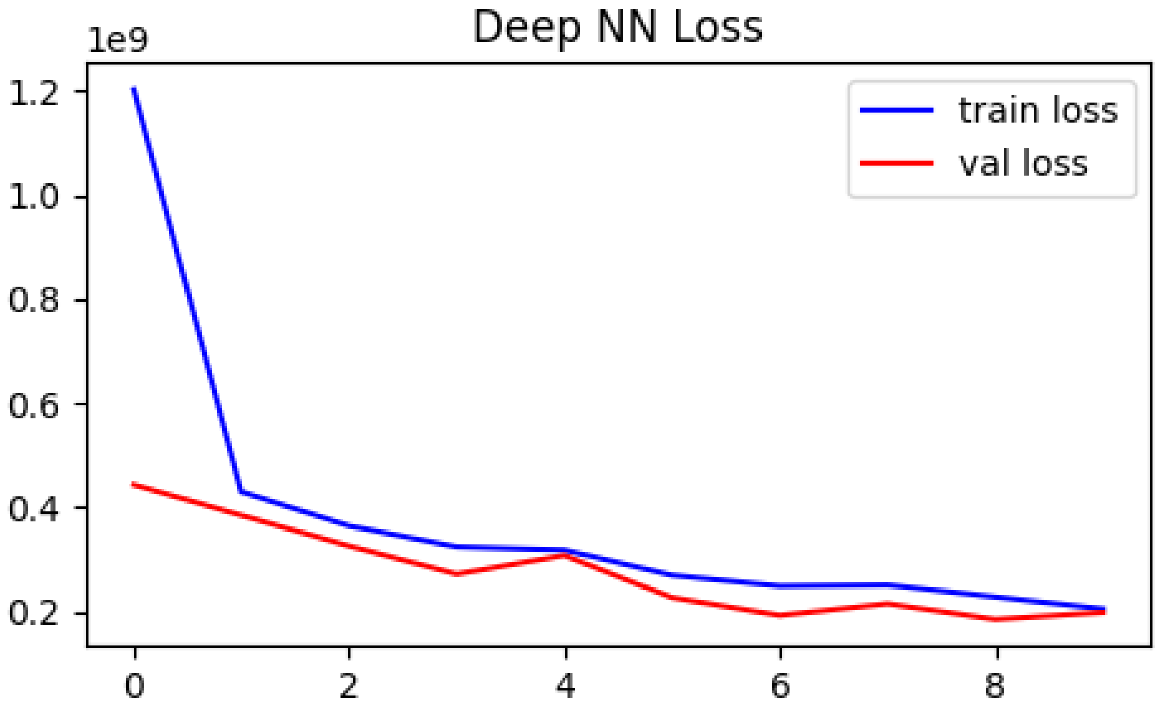
1. A Dense layer with 128 filters and a size of 3 with the data's input shape of (115,). The activation is RELU is applied to introduce non-linearity.
2. Another dense layer with 64 filters and same activation
3. A third dense layer with 32 filters and same activation
4. The last output layer with a single neuron for regression

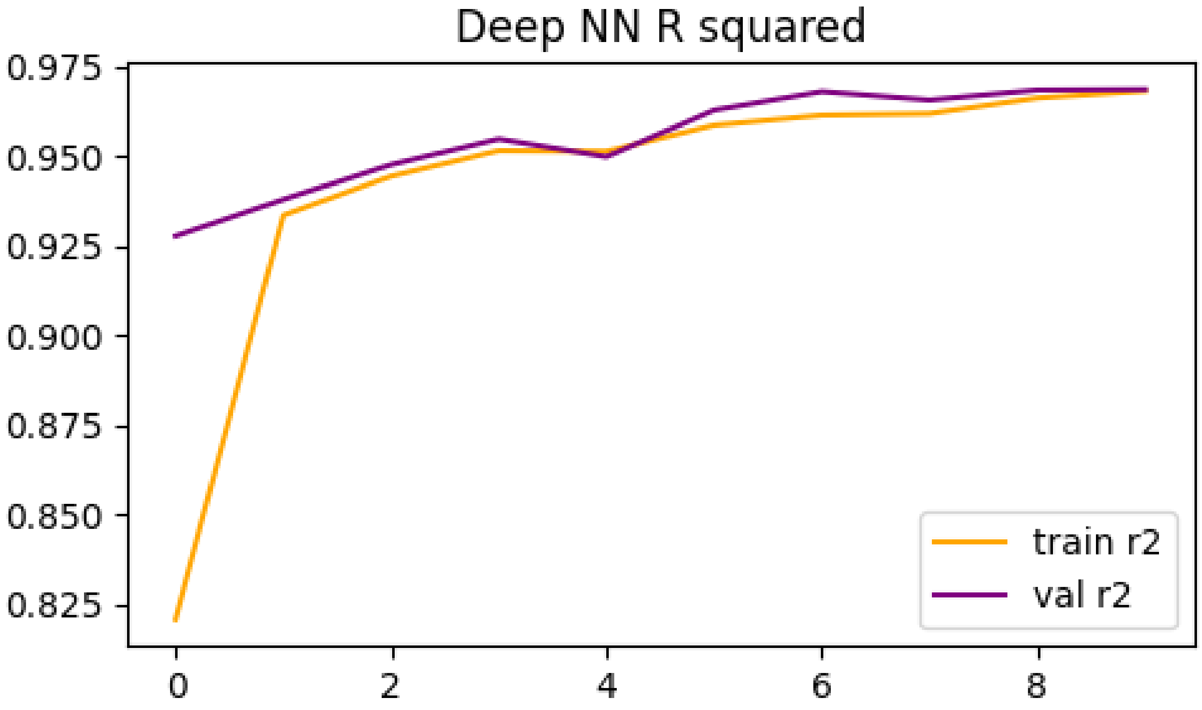
Optimizer= Adam, Learning rate= 0.1, Loss= MSE, R-squared

MODEL ARCHITECTURE

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 128)               14848

 dense_3 (Dense)             (None, 64)                8256

 dense_4 (Dense)             (None, 32)                2080

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 25,217
Trainable params: 25,217
Non-trainable params: 0
```

DEEP NEURAL NETWORK (DNN)





The base DNN is starting to converge to a global minimum yet still yield low loss and high r-squared. It'd be interesting to see how a tuned model would look especially because the DNN is describe to be the type model architecture that would work well for our prediction task

RECURRENT NEURAL NETWORK (DNN)

RNNs by design, are very useful for sequential data of data where the order input matters.

This includes but not limited to, data for time series prediction and Natural language Processing. We will see how it compares to CNN and DNN.

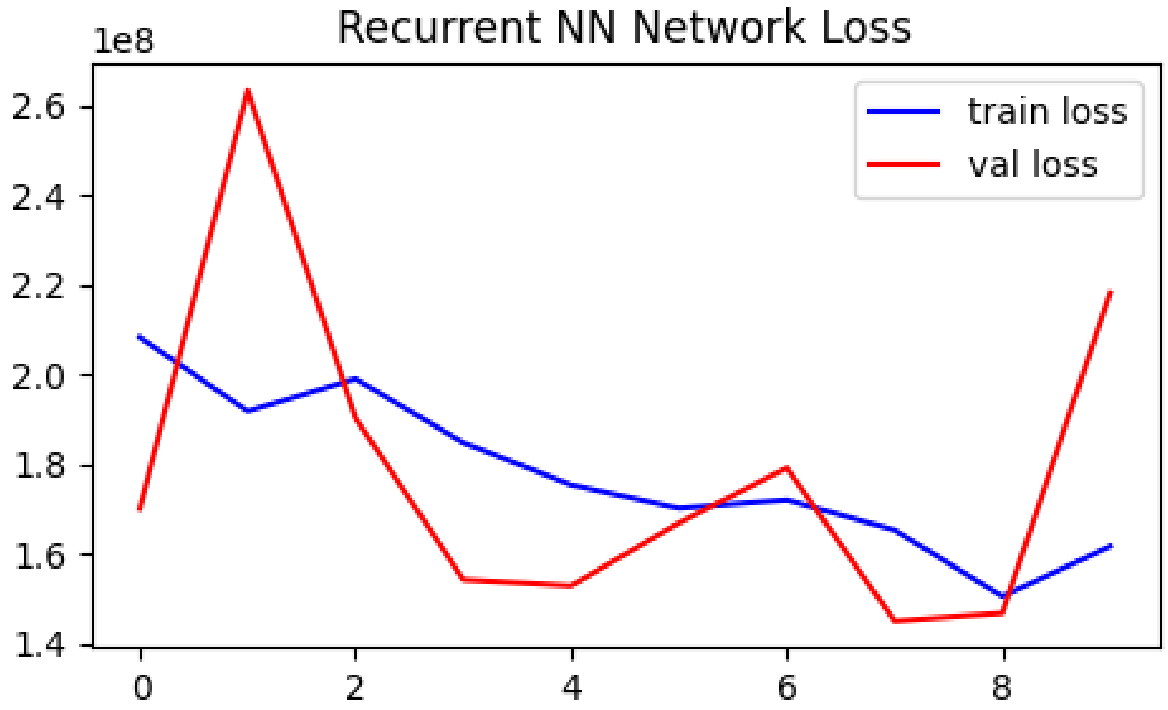
1. The first layer is a Simple RNN layer with 32 neurons. It processes sequences by iterating through the input sequence elements and maintaining a state. The activation is RELU to introduce non-linearity.
2. Another dense layer with 64 filters and same activation for more abstraction and complexity to learn from.
3. The last output layer with a single neuron for predicting continuous output. Using default linear activation.

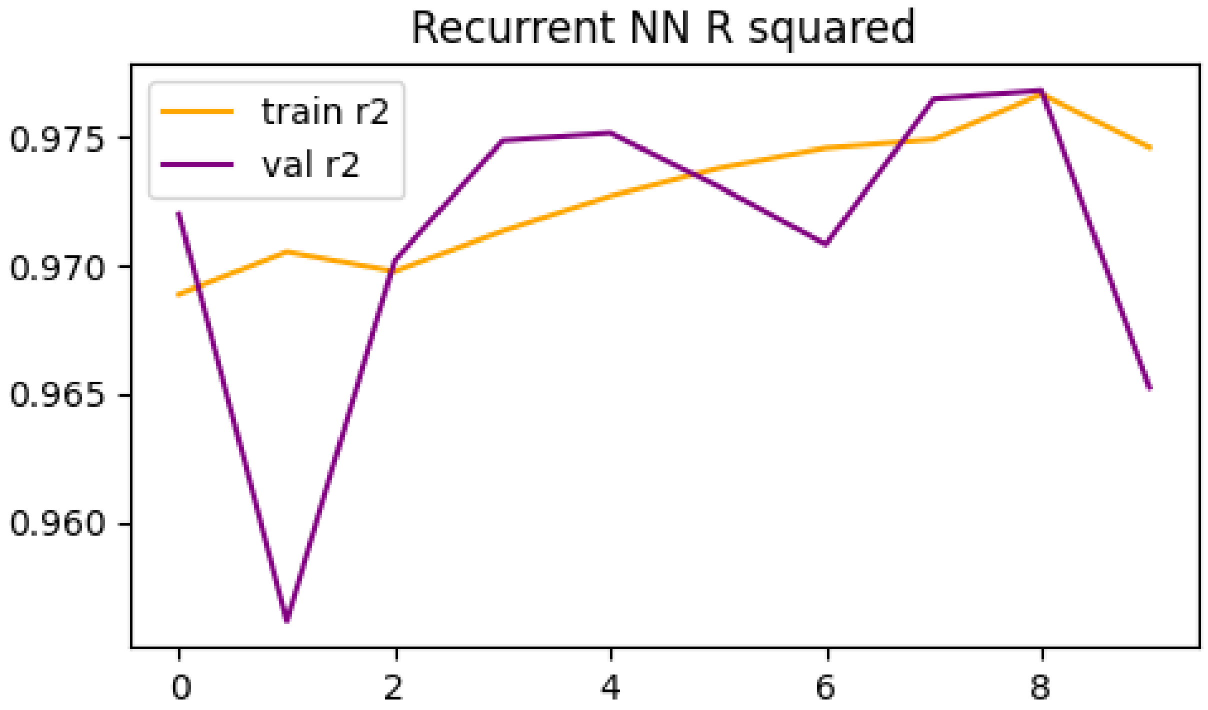
Optimizer= Adam, Learning rate= 0.1, Loss= MSE, R-squared

MODEL ARCHITECTURE

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 32)                1088

 dense_6 (Dense)             (None, 64)                2112

 dense_7 (Dense)             (None, 1)                 65

=================================================================
Total params: 3,265
Trainable params: 3,265
Non-trainable params: 0
```

RECURRENT NEURAL NETWORK (DNN)





The RNN's loss is less at 1e8 compared to the other two model's 1e9. Also, a validation accuracy of 0.970 in was reached just after two epochs. However the gradient seems to be much less stable for RNN than CNN or DNN

MODEL PERFORMANCE COMPARISON

177/177 [==============================] - 0s 2ms/step

Mean Squared Error (CNN): 257149287.11623612

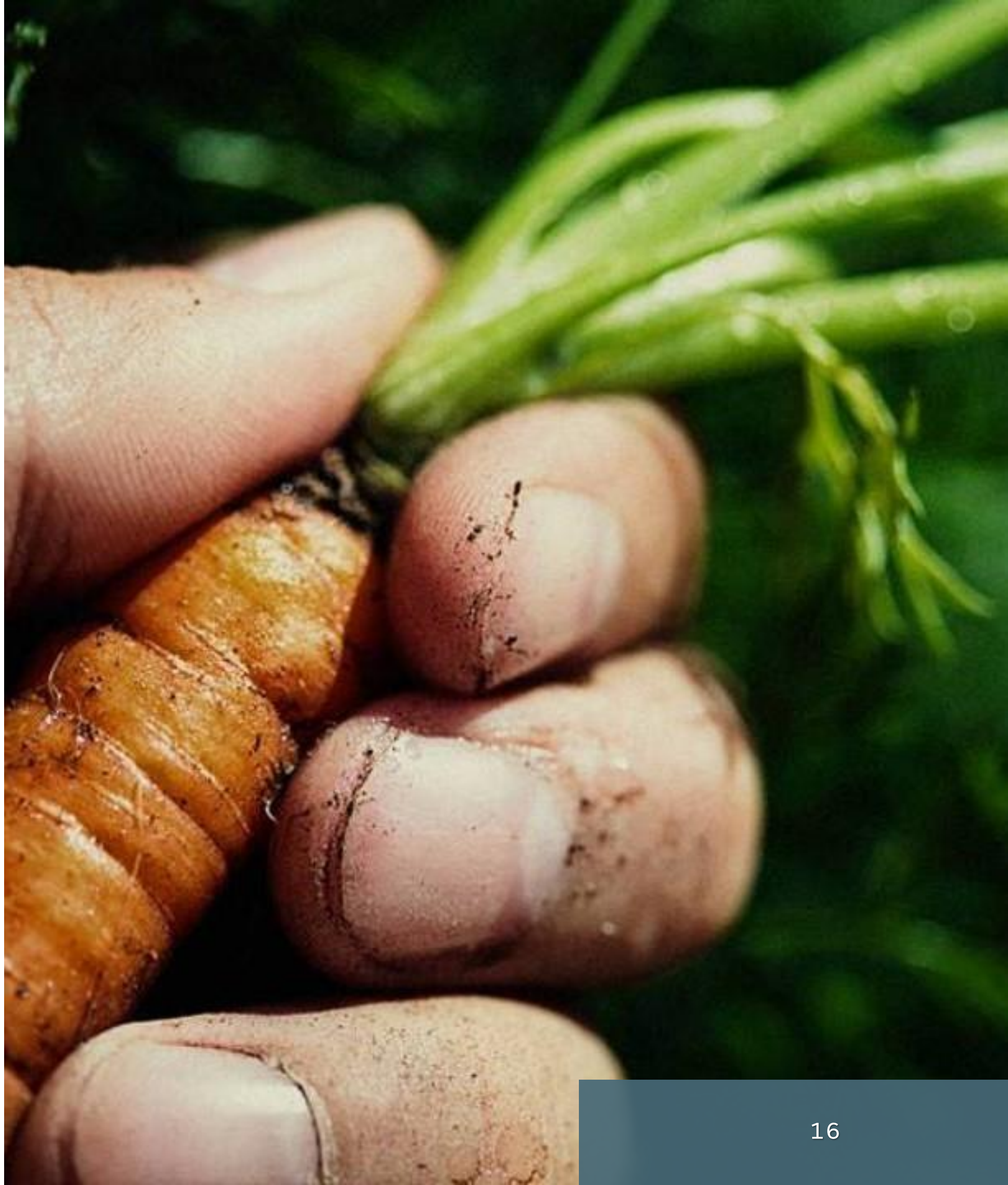
Mean Squared Error (DNN): 195975768.1491566

Mean Squared Error (RNN): 155900308.53059205

R_squared (CNN): 0.9637

R_squared (DNN): 0.9724

R_squared (RNN): 0.978



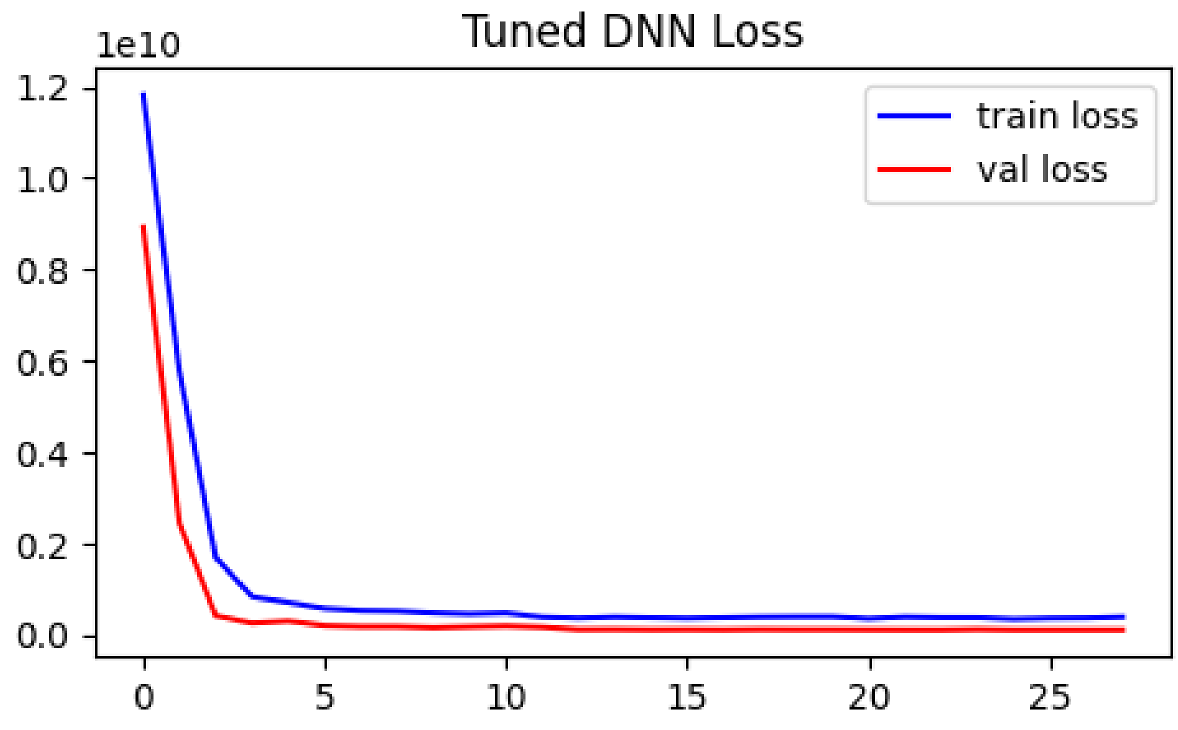
HYPERPARAMETER TUNING

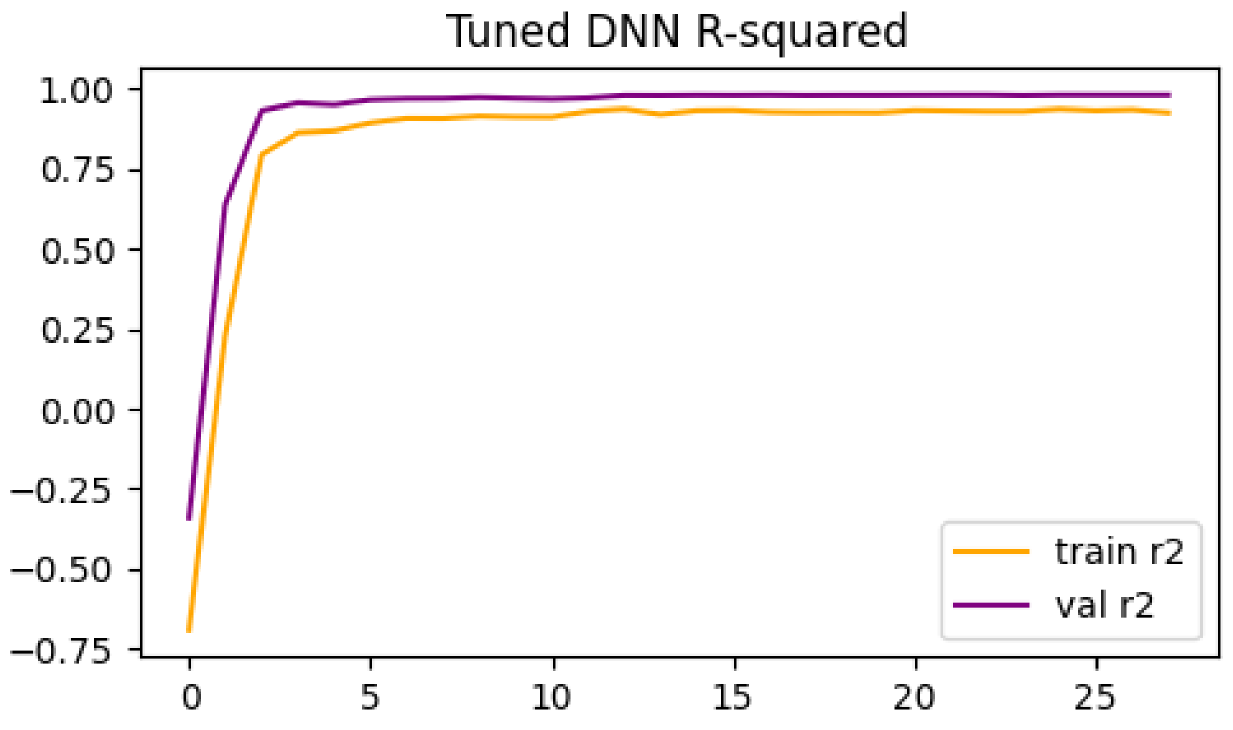
```
Trial 10 Complete [00h 01m 01s]
val_loss: 9613351936.0

Best val_loss So Far: 127869912.0
Total elapsed time: 00h 10m 11s
```

```
{'input_units': 32,
 'num_layers': 2,
 'layer_0_units': 160,
 'dropout': 0.0,
 'learning_rate': 0.015407870950216256,
 'layer_1_units': 96,
 'layer_2_units': 160}
```

```
Epoch 18/50
565/565 [==============================] - 3s 5ms/step - val_loss: 127508592.0000 - val_mse: 127508
592.0000 - val_rsq: 0.9796 - lr: 0.0015
Epoch 19/50
565/565 [==============================] - 3s 5ms/step - val_loss: 124860376.0000 - val_mse: 124860
376.0000 - val_rsq: 0.9800 - lr: 0.0015
Epoch 20/50
565/565 [==============================] - 2s 4ms/step - val_loss: 124745104.0000 - val_mse: 124745
104.0000 - val_rsq: 0.9800 - lr: 0.0015
Epoch 21/50
565/565 [==============================] - 3s 5ms/step - val_loss: 122171488.0000 - val_mse: 122171
488.0000 - val_rsq: 0.9804 - lr: 1.5408e-04
Epoch 22/50
565/565 [==============================] - 3s 5ms/step - val_loss: 119788480.0000 - val_mse: 119788
480.0000 - val_rsq: 0.9807 - lr: 1.5408e-04
Epoch 23/50
565/565 [==============================] - 3s 5ms/step - val_loss: 119095168.0000 - val_mse: 119095
168.0000 - val_rsq: 0.9808 - lr: 1.5408e-04
Epoch 24/50
565/565 [==============================] - 3s 5ms/step - val_loss: 128898272.0000 - val_mse: 128898
272.0000 - val_rsq: 0.9794 - lr: 1.5408e-04
Epoch 25/50
565/565 [==============================] - 2s 4ms/step - val_loss: 119493384.0000 - val_mse: 119493
384.0000 - val_rsq: 0.9808 - lr: 1.5408e-04
Epoch 26/50
565/565 [==============================] - 2s 4ms/step - val_loss: 119270016.0000 - val_mse: 119270
016.0000 - val_rsq: 0.9808 - lr: 1.5408e-04
Epoch 27/50
565/565 [==============================] - 3s 5ms/step - val_loss: 119311632.0000 - val_mse: 119311
632.0000 - val_rsq: 0.9808 - lr: 1.0000e-04
Epoch 28/50
565/565 [==============================] - 3s 5ms/step - val_loss: 119515528.0000 - val_mse: 119515
528.0000 - val_rsq: 0.9808 - lr: 1.0000e-04
```

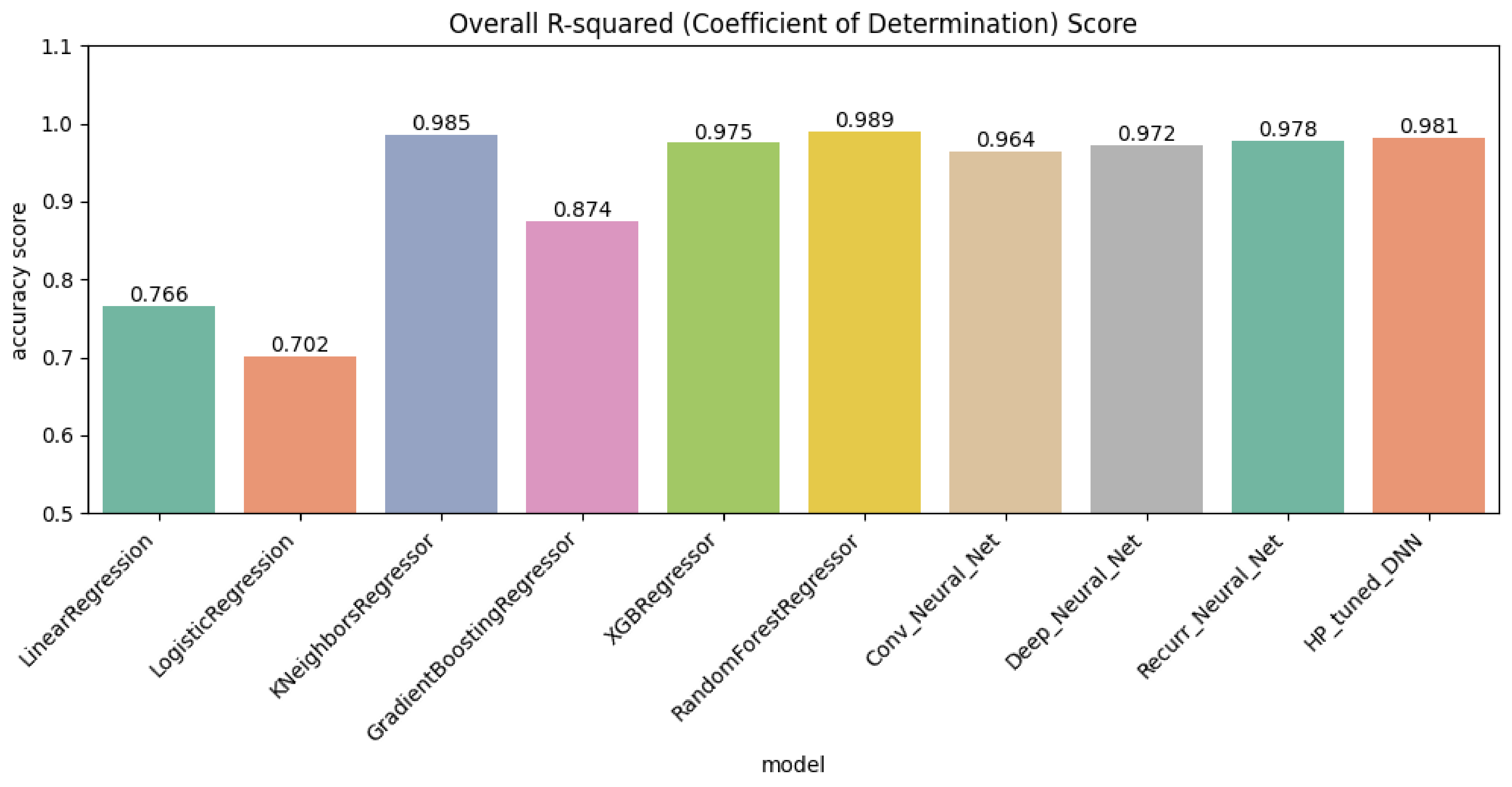
HYPERPARAMETER TUNING



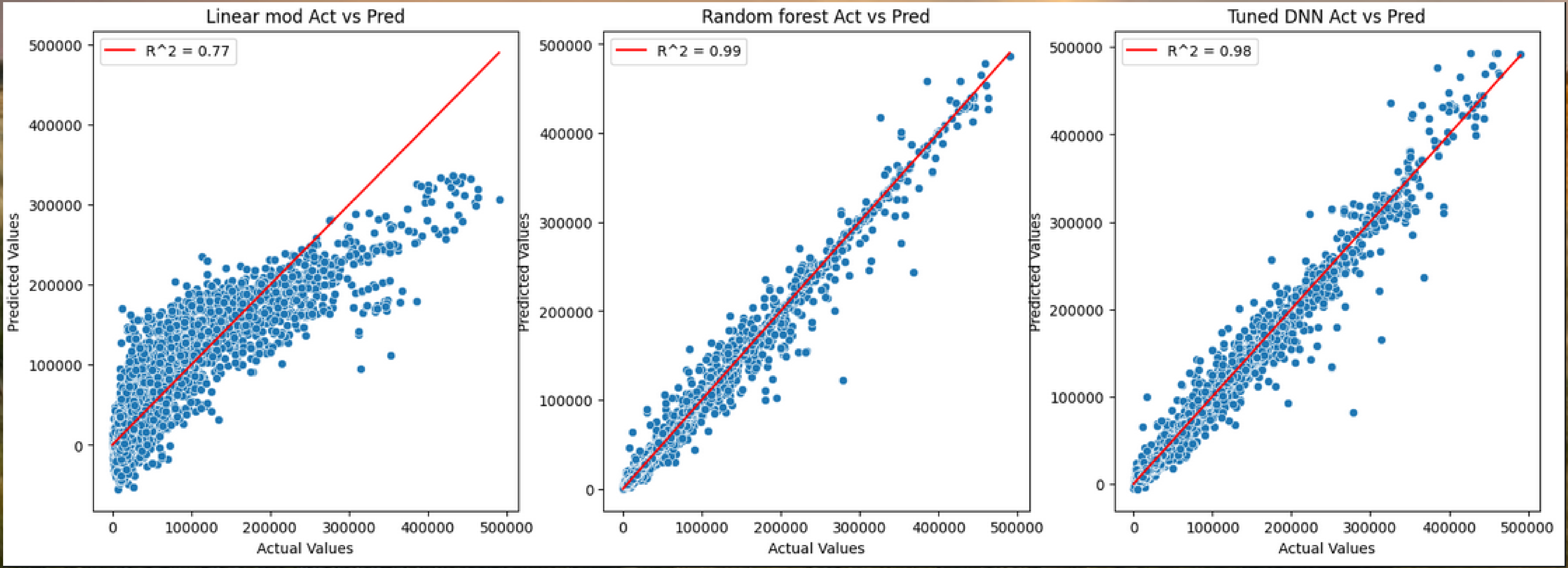


The model fits well, perhaps just a tiny bit of underfitting but the nonetheless the model generalizes well to the test set and validations sets.

Comparison to Other Machine Learning Models



Comparison to Other Machine Learning Models



DISCUSSION/ SUMMARY

In this project, we used Deep learning to predict yield based on various features from our dataset including, country location, crop type, rainfall and temperature.

The process started off with loading up python package essentials such as numpy, sckitlearn and keras and setting up our GPU environment. We then cleaned our dataset, checking the its shape and datatypes and visualizing our features in relation to our dependent variable, crop yield. We preprocessed the dataset by one-hot encoding out object (categorical) variables, split our dataset into training and testing sets and normalized/scaled the features so they are in the ready for deep learning.

Our CNN, DNN and RNN we're equally great at predicting with r-squared values for test set prediction results of 0.9637, 0.9724, and 0.978 respectively.

We chose to tune our DNN model and after discovering the best hyperparameters form our search, we built a model that was even better at predicting the unseen test set than the simpler base models (r-squared= 0.981).

Even comparing our tuned model performance to machine learning models, we've learned that it was on par with the versatile and equally powerful randomforest and much better than the simpler regression model.

DISCUSSION/ SUMMARY

Although we were able to build a prediction model using Deep Learning that had high prediction accuracy, we can further improve our work.

Here are a few things worth exploring that may help improve model:

1. Remove outliers from our dataset
2. Select and use less features for our model
3. Use other types of encoders such as label or ordinal encoding
4. Using other normalization or scaling techniques such as standard scaler
5. kfolds cross validation (although may not be needed unless using machine learning)
6. Conduct a deeper hyperparameter search

DISCUSSION/ SUMMARY

Additionally, running models for classifications would also be very useful. For example, specific yield values might align with particular crop types at specific locations. This insight could prove invaluable for growers in optimizing farm management practices. Growers can assess whether certain pesticide levels result in diminishing returns or determine which crops thrive in specific locations. This information is crucial for identifying ideal locations, allowing growers to explore factors such as better soils, optimal seed varieties, and more effective farm management strategies.

FINAL REMARKS

Forecasting crop yields based on measurable attributes is crucial for several reasons. Conducting field trials is a lengthy and costly process, requiring a substantial number of trials to gain insights into potential yields for a given year. Moreover, the geographical location plays a significant role, as different areas experience varying weather and soil conditions.

Deep learning and other advanced predictive modeling techniques offer a solution, enabling the accurate simulation of crop yield values. By analyzing data such as temperature, rainfall, and fertilizer inputs, these models can achieve high precision. Additionally, deep learning models can be further refined to assist businesses and organizations addressing complex challenges, such as engineering microorganisms through in-vitro and growth chamber experiments and determining how that translates to large field testing. Predictive modeling also contributes prescriptive insights for enhancing management practices. With the global population on the rise, the demand for food is increasing, making the optimization of digital farming through predictive modeling an essential strategy to address this global crisis and maximize food production worldwide.


THANK YOU!