Multi-Layer Perceptron Model for Multi-Country Crop Yield Predictions

Matthew A. Ford

Department of Informatics, University of Sussex, BN1 9QH, UK

In this report, a multilayer perceptron model (MLP) is created to predict the yields of crops given a countries agricultural and environmental data. This model is a prediction model and therefore considers the data from previous years to predict the yield from the following year. This form of MLP is known as an autoregressive model.

The structure of this report is split into four main sections. First, the performance of the system is analysed. Following this is the model architecture, including the techniques used for optimisation. Then, the features and labels passed to the model are discussed before finally, the preprocessing techniques are outlined and justified.

Together, these sections aim to provide a clear understanding on how the MLP was constructed, optimised, trained and evaluated for the task of multi-country yield prediction.

1. Performance

This section discusses how the data was split for training and validation before evaluating the metrics for the model's performance.

1.1 | Splitting the Data

To create the datasets, a sliding window approach was used. This method segments the year data into overlapping sequences, known as *"windows"*. Each widow has the same fixed length and captures the input features (discussed in section 3) for a given number of years (window size).

This data is then paired with the yield data (y_t) for the following year. This setup allows the model to predict the next year's crop yields by identifying the patterns in the window data from a given number of past years.

$$Data_{w=3} = [[X_{t-3}, X_{t-2}, X_{t-1}], y_t]$$

This makes a model that can capture temporal dependencies which is key to a time-series style model. As a result, this makes the model generalize better for predicting yields in unseen years.

To split the data into a training and validation set, all years that have a target year of 2020 or later, are used for validation, while the previous sequences are for training. As the data provided ranged from 2010 to 2022, this meant that there was approximately a 75:25 split.

Each input had the size:

$$1 + (45 * window_{size}) + 102$$

Where there are 45 features per window, one country ID and 102 crop yields. This will be further discussed in *section 3*.

Once the model had been trained for several epochs using the windows of data for each country, the metrics for evaluation could be calculated.

1.2 | Evaluation Metrics

The main metrics used to evaluate the model are the mean-squared error (MSE), R-squared value (R^2), the mean-absolute error (MAE), and the average distance correlation. All of which are shown in *Table 1*.

Metric	Value
Mean Squared Error	6,531,575
Mean Absolute Error	656.9
R-Squared	0.875
Avg. Distance Correlation	0.947

Table 1: Evaluation Metric Values for the Final Model.

Before any metrics could be calculated, it was important to un-scale the predictions and targets. This made sure that the metrics were calculated with the original units.

As the yield values could never be negative, it was made sure that all predictions were clipped at 0 if they happened to be negative.

MSE is calculated by averaging the differences between the predicted and actual values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

The model achieved a MSE of ~***6.5 million*** on the validation dataset. While at first this value seems large, when observed relative to the magnitude of the yield values predicted, it is not surprising to see.

The crop yields in the dataset can range from *0* to the tens of thousands, this means that a high MSE is generally expected, especially in cases where the predictions deviate greatly from high-yield crops. Even a few deviations can cause the value to increase significantly as MSE tends to be sensitive to outliers. This made it a good metric to use as the loss measure for optimisation, as a lower score means the model is generally closer to the true values. However, it is not the most informative metric for analysis.

Instead of MSE, other metrics such as R^2 value, MAE and distance correlation can provide a better understanding of the model's performance.

The MAE is the mean of the magnitude of errors.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

This shows that the model predictions were off by ***656.9*** units on average. MAE is less sensitive to deviations and therefore is a better way of assessing a model where the output is so high. Hence this is a better way of building a picture of the model's performance in this instance.

The R^2 value gives an insight into the proportion of the variance in the target yield that can be explained by the model. The closer this value is to 1.0, the more accurate the model is. It is calculated using the sum of the squared errors, alongside the variance in the actual values and their means.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

A higher R^2 value implies the model has a better predictive accuracy. An R^2 value of ***0.875*** suggests that the model can explain *87.5%* of the variance in the given crop yield data. This means that the model has learnt patterns within the input features and is able to account for most fluctuations in the yields.

While a value of *1.0* is ideal, *0.875* is still considerably strong given the complexity of real-world data as this is generally prone to containing a large amount of noise.

The average distance correlation measure captures linear and non-linear associations between datasets and helps to show how strongly the predictions are related to the actual yields of each crop.

Most yield predictions have a high distance correlation, with the average being ***0.947***. This average suggests that the model is understanding the underlying structure of the data significantly well and therefore is showing the model's ability to predict with the same complexity as the true yields.

As both the R^2 value and distance correlation are high, this means the model is accurate in both a linear and non-linear sense.

Figure 1 shows the mean predicted and actual yields of all crops plotted with respect to each other. The closer the points are to the linear line, the more accurate the average prediction was.

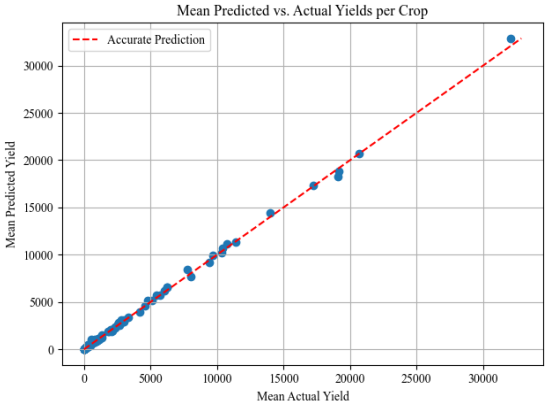

Figure 1: Mean Predicted vs Actual Yields per Crop.

The graph shows that the average predictions are generally very close to the actual yield values. Therefore, reinforcing the conclusions given by the R^2 and distance correlation values.

1.3 | Performance Conclusion

To conclude, the metrics used to evaluate the model show that it has learnt the underlying patterns in the data very well. Having both the R^2 value and average distance correlation being high, it shows that the model has captured linear and non-linear trends which shows its ability to decipher the complex patterns that inherently come from real-world data. This therefore results in accurate and stable yield predictions.

2. Model

This section discusses the model architecture as well as the parameters chosen to optimise, and the process used to do so. Also, the problem of overfitting is discussed alongside the choices made to prevent it.

The hyperparameter values used to achieve the results in *section 1.2* are shown in *Table 2*.

Hyperparameter	Optimisation Result
Optimiser	AdamW
Activation Function	ReLU
LR Scheduler	ReduceLROnPlateau
Window Size	7
Batch Size	70
Dropout	0.005
Learning Rate	0.001878
Weight Decay	6.9e-7

Table 2: Gene values from the best performing genome at the end of the genetic algorithm.

The optimisation process to get these values is later discussed in *Section 2.2*.

2.1 | Model Architecture

The decision made for the model was to allow it to train on all countries with viable data. Then, to keep the output consistent, each prediction will output 102 numbers - one for each crop.

To create a model that was appropriate for this task, there had to be some way to outline region-specific characteristics. To do this, the model had a layer for country embeddings which transformed each country ID to a vector of a specified size.

This allowed for the country feature to be learnable instead of using a simple one-hot encoding. This country vector was then concatenated with the input data to create the model input.

$$X_{in} = [Data_{w=i}, country_{vec}]$$

If the model was training, the input data would also include the yields for the target year.

$$X_{in} = [Data_{w=i}, Yields_{target}, country_{vec}]$$

Then the model follows a simple feed-forward neural network consisting of an input layer, two hidden layers and the output layer.

The hidden layers are of sizes ***1024*** and ***512***. Through trial and error, these were found to be the most appropriate as the dataset is large and complex.

After the input layer and each of the hidden layers, the ***ReLU*** activation function was applied to the data. This was essential to the model as it introduces non-linearity for learning complex patterns. *ReLU* makes sure there are no negative inputs, making all negative values 0, as well as helping to avoid the 'vanishing gradient' problem which in turn helps these deep-learning models to train more effectively.

This was one of the optimised hyperparameters and so the model with the lowest validation loss was found using this function.

Proceeding each activation function, there was then a dropout layer which again had a value that was optimised to ***0.005***.

Both the dropout layer and country embeddings help to encourage generalisation, which is key to prediction models as it must account for unseen data that may come from newly added year or country data. This also helps to capture the general trends that are not as specific to the training data.

To train the model, the training data was fed through for a given number of epochs. Each time, the loss was calculated using the ***MSELoss*** function. This the allowed for the network weights to be adjusted through backpropagation.

The process of backpropagation is essential for a model to learn as it allows for the model to recognise the adjustments needed to lower the loss. The weights can then be adjusted to match using an optimiser.

The optimiser chosen through optimisation was ***AdamW***. This variant of the Adam optimiser which has weight decay. Instead of applying L2 regularisation by adding to the gradients, AdamW applies the decay to the weights directly after the gradients have been updated. The benefit being that the model can have better generalisation, and the training process is more stable. Here the weight decay was optimised to ***6.9e-7***.

Learning rate (LR) is important for a model to train either efficiently or effectively. This value controls the amount a model can update its weights to account for the loss. With a static learning rate, the model has the risk of overshooting minima if the value is too high; or having a training cycle that is extremely slow if the value is too low. Therefore dynamic / adaptable learning rates are a better choice.

To enhance the training efficiency, a learning rate scheduler was chosen as an optimisable hyperparameter which resulted in a **'ReduceLROnPlateau'** scheduler being employed. This scheduler starts off with a larger learning rate of ***0.001878*** before reducing when there is no improvement in the validation loss after 3 epochs. This therefore helps the model to converge more effectively and allows for local or global minima to be reached.

In terms of model input, the window size is an important factor. This determined the amount of past information the model would train on. If this was too high the model would have a chance of overfitting, and if it was too low the model may not have enough detail to predict with an acceptable accuracy as historical patterns may not be recognised. The optimisation process found that the best window size was ***7***.

The batch size was the final hyperparameter. This determined the number of samples that were processed before the weights within the model were updated. Having larger batches allows for smoother gradients, whereas smaller batches can help with generalisation. Therefore, a suitable balance had to be found.

After the optimisation process was completed, a batch size of ***70*** was found to give the best [performance. Allowing the model to train with stability whilst upholding the generalisation factor.

2.2 | Parameter Optimisation

The hyperparameters discussed had set ranges and options for optimisation, which are shown in *Table 3*.

Parameter	Value Options
Optimiser	Adam | AdamW
Activation Function	ReLU | LeakyReLU | Tanh | Sigmoid
LR Scheduler	CosineAnnealingLR | StepLR | ReduceLROnPlateau
Window Size	1 – 9
Batch Size	16 – 256
Dropout	0.0 – 0.4
Learning Rate (LR)	0.00001 – 0.01
Weight Decay	0.0 – 0.00001

Table 3: Parameter Optimisation Options and Ranges.

Due to the large number of options, a parameter sweep was seen as unsuitable, as the time taken to cover every combination would be extremely long. Instead, a genetic algorithm was created to traverse the fitness landscape.

2.2.1 | Genetic Algorithm for Optimisation

Genetic algorithms (GAs) take inspiration from natural selection and are applied here to optimise the hyperparameters of the model. Each solution is represented as a genome, where each gene corresponds to a choice for one hyperparameter, as shown in *Table 3*.

$$Genome_i = [gene_1, gene_2, \dots gene_n]$$

To assess the suitability of the genomes, a fitness function is used. This function trains a model using the hyperparameters encoded in the genome and evaluates its performance on the validation set. For this instance, the fitness is inversely proportional to the average validation loss.

$$fitness = \frac{1}{ValidationLoss_{Avg} + 1}$$

The approach for the algorithm was to follow a form of elitism. This means that genomes with higher finesses are preferred and thus are preserved through generations while also being used for gene crossovers.

This algorithm has four main steps:

1. **Initialisation:** A population of 10 genomes is generated with random hyperparameter values.
2. **Evaluation:** The population is passed through a fitness function, and each genome is evaluated.
3. **Selection:** The best overall genome, along with the top two genomes of the generation are preserved.
4. **Crossover and Mutation:** The rest of the population is generated through a gene crossover between two of the top three genomes. This has an 80% chance of happening, otherwise, a random genome from the top three is chosen to be directly copied.

   To retain diversity through the population, there is a 20% chance of mutation per genome per generation. Here, a single gene is randomly changed to a new value from its available choices in *Table 3*.

Each generation is one cycle of this process. The algorithm was run for 100 generations, with an increasingly larger number of training epochs to allow for deeper exploration.

Figure 2 shows the best-found average validation loss along with the best validation loss per generation.

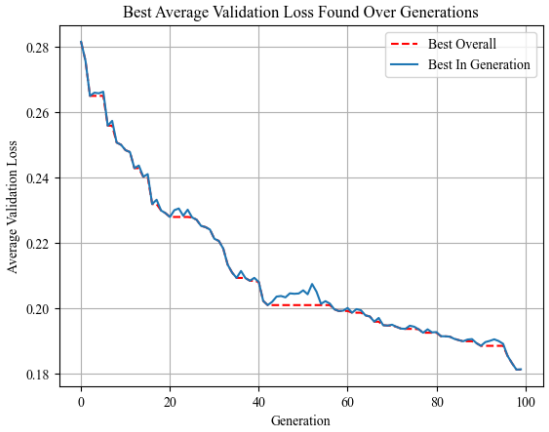

Figure 2: Best Found Genome for each Generation of the Genetic algorithm (GA).

As seen in the graph, the average validation loss goes down quickly in the earlier generations before the gradient decreases and the exploration slows down. There is a lot of fluctuation with the best per population results, but the general trend remains the same.

In the later generations, the model seemingly starts to converge, although it is shown that there is a jump near the end. This is possibly due to a jump between minima because of a random mutation for one of the genes. This solidifies the importance of maintaining population diversity through mutation as without this, the population would not be able to jump from local minima or "traps".

At the end of the optimisation process, the results in *Table 2* were produced. These values achieved an average validation loss of **0.1812** and was used to train the final model evaluated in *Section 1*.

2.3 | Overfitting

To prevent overfitting, several techniques were used in the model design. As discussed earlier, a dropout layer was included to randomly disable a fraction of the neurons whilst training. This prevents the model from becoming too dependent on any single pattern in the training data.

Another technique used was L2 regularisation. This is controlled through the weight decay hyperparameter inputted into the optimiser. This helps discourage larger weights and therefore penalises the model if it becomes overly complex.

As mentioned earlier, data windows also helped to reduce overfitting as it forced the model to focus on a specific time frame instead of having all the available data for previous years.

3. Features & Labels

This section outlines the features and labels used to train and validate the model as well as discussing the process of extracting these from the provided dataset, giving reason to any choices made along the way.

As mentioned in *Section 1*, a sliding window approach was employed restructure the data into a suitable format for an auto-regressive time-series model. The main goal of this approach was to allow the model to learn meaningful temporal patterns across multiple years, rather than treating each year's data individually.

As discussed in *Section 1.1*, each window consists of the feature data for the previous x years, where x is the size of the window. This sequence is then paired with the crop yield data for the subsequent year. This trains the model to make predictions based on historical trends, instead of using current-year features to predict current-year yields.

Each input is generated per country to help retain the regional and temporal context, which is critical to this model as data varies across different regions and countries.

3.1 | Features & Labels

The input data was made up of several agricultural and environmental values recorded monthly for each country over several years. These features were compiled from multiple datasets after being processed and represented the conditions that may influence crop production. Applying a sliding window over 7 consecutive years allowed the model to capture the relevant temporal patterns.

The output data was the crop yield values for each country for the year following the input data. This is a vector of size 102, where each element corresponds to a predicted yield of a specific crop.

Before training, the features and yield values were scaled. This ensured that the data was within the same range and as a result ensured numerical stability between crop types. This

was necessary due to the variation in yield amounts as some countries produced crops that others didn't, and there could be thousands of crops difference between yields. This in turn helped the model learn more efficiently.

Each training example was therefore made from a country's feature window over 7 years, and the yields for the following year as the label. This allowed the model to learn to predict future yields.

3.2 | Data Extraction

The aim of the data extraction was to get meaningful data with a low number of features to then be used as inputs and outputs for training and validating the model.

3.2.1 | Country Mapping

Each feature's dataset originally recorded values at specific latitude and longitudes. To train a country-specific model, it was vital to assign each point to a country. This was achieved using a provided lookup table which contained:

- Country Centroids (Latitude & Longitude)
- Approximate Country Radii
- Country Area
- Country Name

To keep the mapping process simple, each country was approximated as a circle centred around its centroid. This was deemed to be sufficient for providing some geographical accuracy alongside keeping the computation efficient.

For each data point, the following process was run:

- Calculated the Euclidean distance from the point to each country's centroid.
- Identified the nearest centroid using these distances.
- Compared the distance to the nearest centroid against an adjusted radius which was calculated by scaling the base radius by a factor derived from the area. This helped better represent larger countries or those with irregular shapes.
- If the point was within the radius, the data point was then assigned the respective country.
- If there was no centroid to pass the condition, the data was marked as *"OUT OF AREA"* and removed from the dataset.

To add, all countries that did not have any matching yield data were removed as it was necessary to at least have some yield data to train the model accurately.

This process allowed for definitive and consistent country aggregation and filtered out any data points that were ambiguous or irrelevant. This helped maintain the integrity and clarity of the features for training.

3.2.2 | Data Aggregation

To reduce the dimensionality of the data into meaningful statistics, the remaining raw data was aggregated.

Firstly, as there were multiple data points per country per year, the data was grouped and averaged to get a single row of data for each country, consisting of the averages of every data row. This was commonly on a per moth per year basis.

This was also done to the land coverage, where the land cover percentage was averaged per country with any data points that were found.

The data being given at a monthly resolution. This meant that there were a significant number of features causing the dimensionality to be high.

To reduce the complexity, and to make the data more suited to predicting annual yields, the monthly data was aggregated to a yearly scale by using both the mean and standard deviation of each feature (column) for every country per year provided.

The average represented the annual level of each feature, and the standard deviation allowed for variability within the year to be captured alongside strengthening the temporal dynamics.

By performing these aggregations on the feature data, the result was a compact but informative summary. This helped to reduce the noise in the data as well as making it easier for the model to detect per-year patterns as the temporal scale was now aligned between the features and labels.

3.2.3 | Missing Output Data Handling

Due to the nature of real-world data collection, it is very probable that some data may be missing. This would be challenging for the sliding window approach as it relies on continuous sequences data. If any point was missing, the training sample would be invalid.

With the given dataset, there were no missing years of feature data, but there were some years missing for the label data. To address this, the missing values were estimated using a linear extrapolation algorithm. This was a simple method that effectively estimates steady trends over time. From analysing the yield values, this was seemingly common.

By filling in these missing values, the continuity of the data was upheld, no relevant data was wasted, and the sliding window approach could then be applied.

3.2.4 | Resulting Data

For each country-year pair, the finalised features included the mean and standard deviations of the environmental and agricultural data as well as a country ID that is passed through the embedding layer discussed in *section 2.1*.

3.3 | Reasoning

The steps taken to prepare the feature data were made strategically. They aimed to reduce the complexity and improve the generalisation.

As the monthly data had a high dimensionality, applying a yearly aggregation to get the average and standard deviation of each feature allowed for the dataset to become easily managed and helped reduce the noise that is inherent to monthly data due to fluctuations. This pulled the focus of the model onto long-term trends which is ideal for predictive models.

The process of linear extrapolation provided missing data with values estimated from adjacent data. This helped to fill any gaps in a logical way and ensured the temporal trend was upheld.

Overall, by summarising the data to a yearly resolution, the model was encouraged to learn patterns from long-term trends instead of having the risk of reacting to short-term patterns. This again was a way of improving the generalisation of the model whilst ensuring the feature resolution matches that of the labels.

4. Preprocessing

This section discusses any other forms of preprocessing performed on the data, any problems that had to be rectified, and the reasoning behind any choices made.

Whilst cleaning the land coverage dataset, there were some large inconsistencies that were discovered. The file itself contained data points across the entire world, each with a latitude, longitude, and 17 land classes, with percentage values where the total is 100, showing how the land at that point is split.

To visualise this data, the choice was made to plot the points over a world map. This is where the issues became apparent. Firstly, when removing points that logically cannot have any crops growing (such as ocean, and permanent snow / ice), the points would be removed at seemingly incorrect places.

This led to the conclusion that the 17 land coverage percentage columns had been shifted by 1 to the right. Therefore, this had to be adjusted accordingly. However, this made the second issue apparent.

It appeared that the data points were flipped in the y-axis (latitude). And therefore, all the values in that column had to be inversed. This made the data consistent with the rest and allowed for the remaining processing to go ahead.

In terms of file column and row naming, any columns that were addressed by name were manually checked for spelling errors. Other than this, calling columns by name was generally avoided unless the dataset was one that was custom as spelling errors can be common for large datasets and it is not always possible to check manually.

It was also noticed that the yield data had uppercase naming, therefore when working with these, it was made sure that all names were lowered. To add, the yield data was mixed with production data per crop, per country, per year. These production data rows were removed as they were deemed irrelevant to the model due to it only predicting yields.