Imperial College London

COMP70050

Introduction to Machine Learning

October 2023

Imperial College London

Department of Computing

—Coursework 2—

Artificial Neural Networks

Team members:

Kyoya Higashino [redacted]

[redacted]

Table of Contents

Section 1 – Model Description and Justification

Figure 1 below details the chosen neural network architecture and hyperparameters.

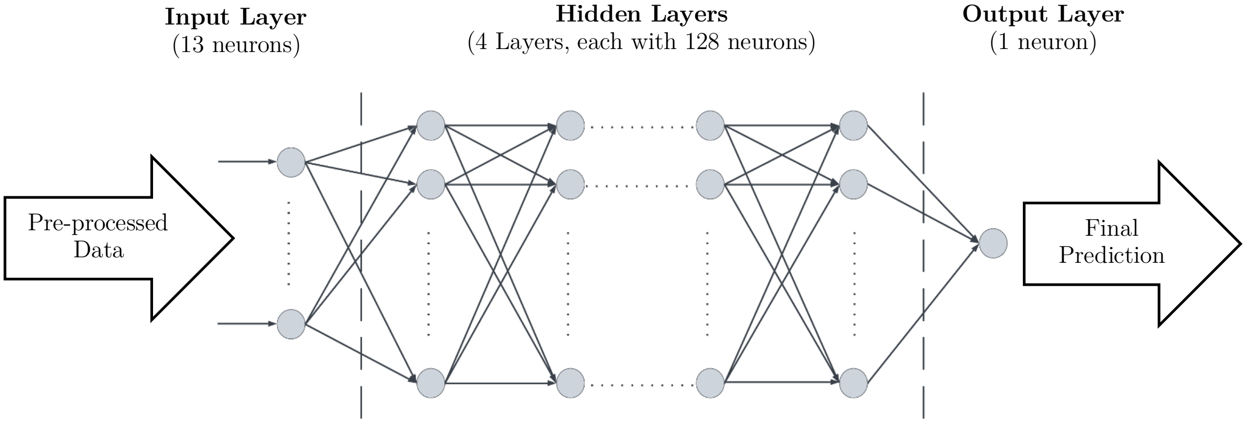

Figure 1: Chosen neural network architecture

Pre-processing

Several pre-processing operations were performed, including **standardisation**, **substitution of null values**, and **one-hot encoding** categorical values.

Standardisation ensures that all numerical features are on a consistent scale, aiding in gradient descent's efficient navigation of the loss landscape, and preventing features with larger magnitudes from disproportionately influencing the model training. Unlike min-max normalisation, standardisation is less sensitive to outliers, thus preserving the original distribution of features without distortion.

The substitution of null values served as a crucial step in handling missing features, ensuring that impacted data points remain viable for both training and testing purposes, retaining as much data as possible. As replacement values, the feature average—referred to as mean imputation—was used, acting as the most reasonable estimate considering the overall feature distribution. Other methods, such as median imputation and k-nearest neighbours, were considered; however, the mentioned mean imputation was selected due to its simplicity and efficiency in preserving the distribution of data. As large sample sizes are crucial in building robust models, this strategy tries to maximise the availability of the dataset while minimising introduced bias.

As the 'ocean proximity' feature is categorical, **one-hot encoding** was applied to transform it into a numerical format suitable for the neural network. One-hot encoding replaces the single categorical column into multiple (here, five) binary columns, each representing a distinct proximity category and preventing any implicit ordinal relationships that might interfere with the model's learning process. The resulting one-hot encoded representation facilitates a clearer understanding of the categorical distinctions, enhancing the model's ability to discern and learn patterns related to different proximity categories.

Neural Network Model

After one-hot encoding, the data included 13 independent features, resulting in an **input layer of 13 neurons**. After an extensive hyperparameter search, the final neural network utilises an architecture of **4 hidden layers of 128 neurons** each along a ReLU activation function at each hidden layer. Conversely, as a regression problem, the **output layer consists of a single neuron** with a linear activation function to produce a continuous output.

The architecture's complexity, with multiple layers and neurons, was chosen to capture complex patterns within the data, while avoiding overfitting—a common risk with high-capacity networks— which was a key consideration in the model's design.

Neural Network Training

Several training parameters were also considered, either theoretically derived or empirically chosen. This included the learning rate, optimiser, loss function, and train-validation-test split.

In hyperparameter tuning, a grid search identified a **learning rate of 0.01** with an **Adam optimiser** to be the most effective (more details in *Section 3 – Hyperparameter Search and Results*). Note that the Adam optimiser is a common choice, preferred for its adaptive learning rate and as it combines the advantages of both Adagrad and RMSProp. While the learning rate as an independent parameter seemed to improve performance at higher values, experiments showed that is not consistent across all neural network configurations, resulting in a small learning rate for the optimal architecture.

The chosen **loss function was a mean square error** (MSE) to penalise large errors more heavily during the training process. Simultaneously, the **train-test split chosen was 80/20** to achieve a good balance between training and test data size, aligning with common practices in machine learning.

Section 2 – Evaluation Set-up

As mentioned previously, the initial dataset was separated into an 80/20 train/test split, leaving 20% of the data only to be used for testing. Subsequently, the training data is separated again into a 90/10 pure-train/validation split, leaving 10% of the training data for validation. As a result, the initial dataset is split into **training, validation, and testing** in the ratio of **72%, 8%,** and **20%,** respectively.

During training, only the pure training dataset (72%) is used to train the neural network model, while the validation dataset (8%) is used to evaluate the training process. The training-validation split is implemented to evaluate model performance more accurately on 'unseen data' and prevent overfitting. To do so, an **early stopping** condition was employed to stop training once overfitting started to occur with the overfit criterion known as the 'patience'. The 'patience' of the early stopping condition is set as ten, defined as observing no improvement in the last ten training epochs. Ten was identified as the optimal balance between avoiding premature and late stoppages, which can lead to underfitting and overfitting, respectively.

Early stopping is our chosen method to prevent overfitting. However, L2 regularisation (lambda weight decay) and dropout are also effective. L2 penalises the square values of the weights to simplify the model, while dropout adds robustness as it prevents neuron co-adaptation by random deactivation during training. These methods were not applied in this instance to preserve the natural dynamics of the loss function and model interpretability and simplicity.

Once trained, the model is tested on the held-out test dataset (20%) for final evaluation, acting as the best evaluation of the model's ability to generalise on unseen data and predict accurately. As the loss function used is MSE, the Root Mean Square Error (RMSE) is used as the final evaluation metric.

Section 3 – Hyperparameter Search and Results

To conduct hyperparameter tuning, a grid search method was used to train models with different parameter combinations on the pure-train dataset (72% of the initial dataset). The performance criterion used for evaluation is the RMSE, comparing predicted model outputs versus validation data (8% of the initial dataset).

The hyperparameters for the neural network are the **number of hidden layers**, the **number of neurons per hidden layer**, the **activation function** used for hidden layers, the **optimiser**, and the **learning rate**. Table 1 shows the range of the search space for all the hyperparameters to be tuned.

Table 1: Hyperparameter search space for five hyperparameters

Hyperparameter Search Space	
Number of Hidden Layers	1, 2, 3, 4
Number of Neurons per Hidden Layer	16, 32, 64, 128
Activation Function used for Hidden Layers	ReLU, Sigmoid, Tanh
Optimiser	Adam, Adagrad, RMSprop
Learning Rate	0.001, 0.005, 0.01, 0.05, 0.1

In hyperparameter tuning, all independent variable model parameters were investigated. These parameters are defined/set before training and are manually changed through multiple experiments. Conversely, the early stopping mechanism during training makes the number of epochs a dependent variable and not an independent parameter. Indeed, while the epoch limit is set at 1000, training may terminate before the 1000 epoch limit is reached.

From the grid search, 720 different models were tested. Of the 720, the 10 best-performing models are shown in Table 2 below. Note that the best-performing model is also the final neural network described in *Section 1 – Model Description and Justification*.

Table 2: Top 10 Best Models from Hyperparameter Tuning

Rank	**Number of Hidden Layers**	**Number of Neurons per Hidden Layer**	**Activation Function**	**Optimiser**	**Learning Rate**	**Score (RMSE) [in $]**
1	4	128	ReLU	Adam	0.010	**56,097.510**
2	4	128	ReLU	Adam	0.005	**57,094.180**
3	4	32	ReLU	Adam	0.005	**59,007.150**
4	4	64	ReLU	Adam	0.005	**59,108.445**
5	4	128	ReLU	Adagrad	0.050	**59,447.617**
6	4	64	ReLU	Adam	0.050	**61,020.676**
7	4	16	ReLU	Adagrad	0.100	**62,009.074**
8	4	64	ReLU	Adam	0.100	**62,105.156**
9	2	128	ReLU	Adam	0.100	**62,208.305**
10	4	64	ReLU	Adam	0.010	**62,231.125**

In Table 2, some patterns can be observed such as the frequent appearance of certain parameter values. To better visualise the grid search results and the effects of individual parameters on model performance, the average RMSE is plotted against models of varying hyperparameter values. The average RMSE must be taken as different models with the same value of a certain hyperparameter will still have differences in their other hyperparameters.

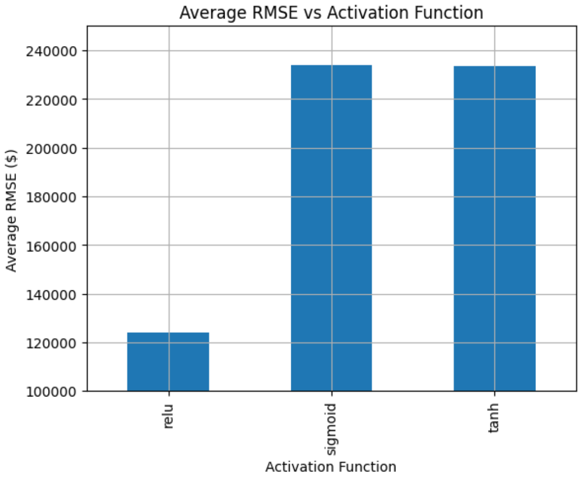

Figure 2: Average RMSE vs. Activation Functions.

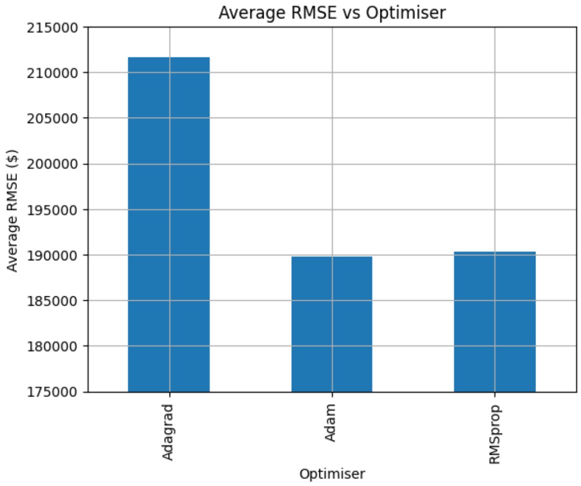

Figure 3: Average RMSE vs. Learning Rate.

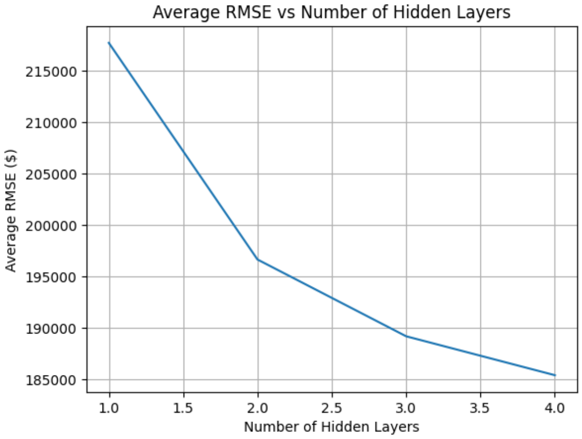

Figure 4: Average RMSE vs. Number of Hidden Layers.

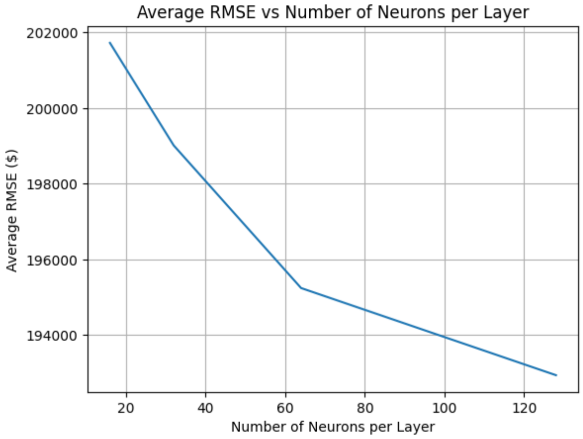

Figure 5: Average RMSE vs. Number of Neurons per Layer.

As seen in Figure 2, the ReLU activation function greatly outperforms the sigmoid and tanh functions, with ReLU models having RMSE values of nearly half that of sigmoid and tanh. In Figure 3, the Adam optimiser slightly outperforms RMSprop for the lowest average RMSE, also being used in 8 of the 10 best hyperparameter configurations. Simultaneously, multiple trials through grid search tuning showed that learning rates are very sensitive to the choice of optimiser and other hyperparameters. As other hyperparameters change the model complexity, the learning rate also needed to be uniquely tuned to match changing gradient weights and vector lengths for gradient descent. Eventually, the best learning rate was empirically found to be 0.01.

In Figure 4 and Figure 5, line graphs are used as the number of hidden layers and number of neurons can take a wide range of values. Both figures show a convex downward curve, suggesting that higher values positively affect model performance but will eventually converge to a minimum average RMSE. As a result, the highest values for the number of hidden layers and neurons per layer (4 and 128, respectively) were used for the final hyperparameters.

Section 4 – Final Model Evaluation

After training on the pure-train dataset (72%) and employing a validation dataset (8%) for hyperparameter tuning and early stopping, the model underwent final testing on the held-out test set (20%). Common practice suggests retraining the model on the combined training and validation datasets once hyperparameters have been finalised; however, in this instance, the early stopping mechanism necessitates maintaining a separate validation set. This approach ensures that early stopping continues to act as a form of regularisation against overfitting during the final training phase. Consequently, the validation set was not merged with the pure-train set to preserve its role in the model's development. The learning curve of the model is presented in Figure 6 below.

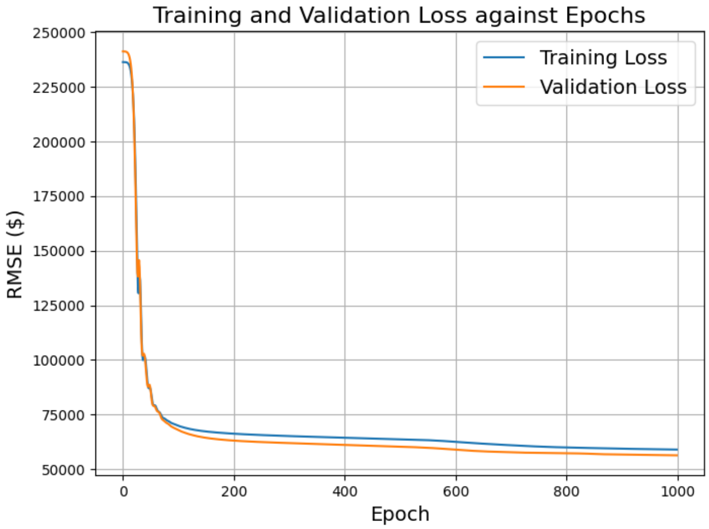

Figure 6: Learning curve for the final neural network model

Figure 6 illustrates the close alignment between the training and validation loss lines, indicating a **well-fitted model with minimal susceptibility to overfitting**. However, it is worth noting that a gradual divergence between the loss lines occurs during the later stages of training, suggesting a potential inclination of the model towards fitting the training data more than unseen data.

The **final RMSE score** calculated on the held-out test dataset is **$56,814**, indicative of the model's robust performance, notably when contrasted with the standard deviation of the 'median house values' which stands at approximately $116,000. This lower RMSE suggests that the model's predictions are significantly more precise than a model that would naively predict the mean value for all observations. Additionally, this performance exceeds the LabTS's benchmark RMSE of $90,000 by a margin of 37%. Such a comparison underscores the model's capability to provide accurate and stable predictions, rather than merely reflecting the central tendency of the data.

All in all, the RMSE score provides a clear measure of the model's predictive accuracy, particularly relevant for house price predictions where outliers can disproportionately affect performance. To augment the robustness of future evaluations, incorporating a spectrum of metrics, such as Mean Absolute Error for its direct interpretability and the coefficient of determination (R^2) for its measure of explained variability, could yield a multifaceted view of the model's performance, enabling a more granular assessment of its predictive capabilities and generalisability.