The AI Application Report

CI512 - Intelligent Systems 1

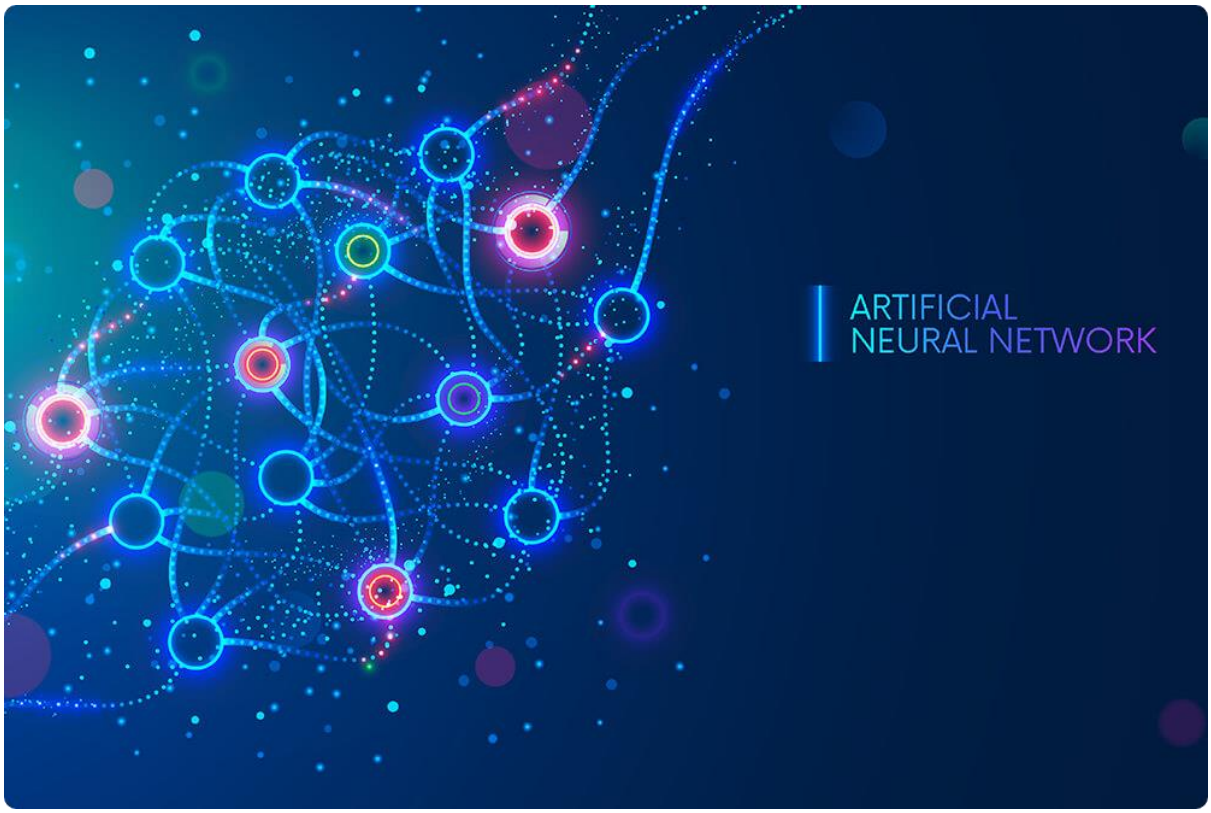

Figure 1 - https://bernardmarr.com/img/What%20is%20an%20Artificial%20Neural%20Networks.jpg

Table of Contents

Introduction

My project proposal is to make an AI that can predict how bad lung cancer is in patients, this is crucial because you can prioritise the people that need help the most. In the UK 34,800 people died from lung cancer in 2019 ***[Ref1]***. If the cancer was treated earlier, I believe that number would be much lower.

For this project, I will use deep learning which is a type of machine learning, that uses supervised learning and a brain-like structure to learn and predict. The type of deep learning I am going to use is artificial neural networks (ANN), as they are effective at processing data in table format. The reason I am going with deep learning is that compared to other machine learning techniques, deep learning can achieve higher accuracy, and can find patterns with more complex datasets. Deep learning, in my opinion, is the most exciting type of machine learning and has the most potential to achieve Artificial general intelligence. ***[Ref2]***

There are several challenges I might face while making this AI application. One of the challenges is the quality of the data might be bad, by that I mean there might be the same, incomplete or incorrect data. The data might also be biased by having more data about one outcome than another. To solve this, I need to choose a good dataset.

Another issue I might come across is overfitting, this is when you train the models, and it starts to memorise the training data rather than learning and finding patterns. To solve this, I can break the data set down into two parts the first part to train the model and the second part to test the model, this is called cross-validation.

Section 1 - Dataset

The data set I am going to use is one I found on Kaggle, this dataset has 1000 training data. All the patients have cancer, and the results are broken down into three parts low medium and high. The dataset has no missing data and is fairly distributed to avoid bias.

There are 26 features in the dataset including age, fatigue and wheezing, however, there are two columns I don't need, index and patient ID, as they have nothing to do with predicting the type of cancer.

Column name	Description
Age	The age of the patient. (Numeric)
Gender	The gender of the patient. (Categorical)
Air Pollution	The level of air pollution exposure of the patient. (Categorical)
Alcohol use	The level of alcohol use of the patient. (Categorical)
Dust Allergy	The level of dust allergy of the patient. (Categorical)
OccuPational Hazards	The level of occupational hazards of the patient. (Categorical)
Genetic Risk	The level of genetic risk of the patient. (Categorical)
chronic Lung Disease	The level of chronic lung disease of the patient. (Categorical)
Balanced Diet	The level of balanced diet of the patient. (Categorical)
Obesity	The level of obesity of the patient. (Categorical)
Smoking	The level of smoking of the patient. (Categorical)
Passive Smoker	The level of passive smoker of the patient. (Categorical)
Chest Pain	The level of chest pain of the patient. (Categorical)
Coughing of Blood	The level of coughing of blood of the patient. (Categorical)
Fatigue	The level of fatigue of the patient. (Categorical)
Weight Loss	The level of weight loss of the patient. (Categorical)
Shortness of Breath	The level of shortness of breath of the patient. (Categorical)
Wheezing	The level of wheezing of the patient. (Categorical)
Swallowing Difficulty	The level of swallowing difficulty of the patient. (Categorical)
Clubbing of Finger Nails	The level of clubbing of finger nails of the patient. (Categorical)

Figure 2 - Image of the dataset. Src: https://www.kaggle.com/datasets/thedevastator/cancer-patients-and-air-pollution-a-new-link

Section 2 - Implementation

2.1 Data manipulation

I got the data from the CVS file to python, using pandas. The actual results were Low, Medium and High which needed to be converted to numbers, so the model can work with them. I also used sklearn to split the data into training and testing, 25% of the data was used for testing and the rest was used for training.

2.2 Making an ANN

ANN works by having a set of parameters and each parameter has a weight that is randomly set at the beginning, as the model trains the weights are adjusted to match the output result.

A layer is made of multiple units which receives inputs from other units and processes them to return an output. Each unit computes the dot product between x (input) and w (weight). The notation for each unit is $x^T w$. The T stands for transposing which is when you convert columns into rows and vice versa, this is used so we can multiply the matrix of inputs and weights. When you multiply matrixes the row of the first matrix must match the columns of the second matrix and the size of the output matrix is going to be the column of the first matrix and the row of the second matrix.

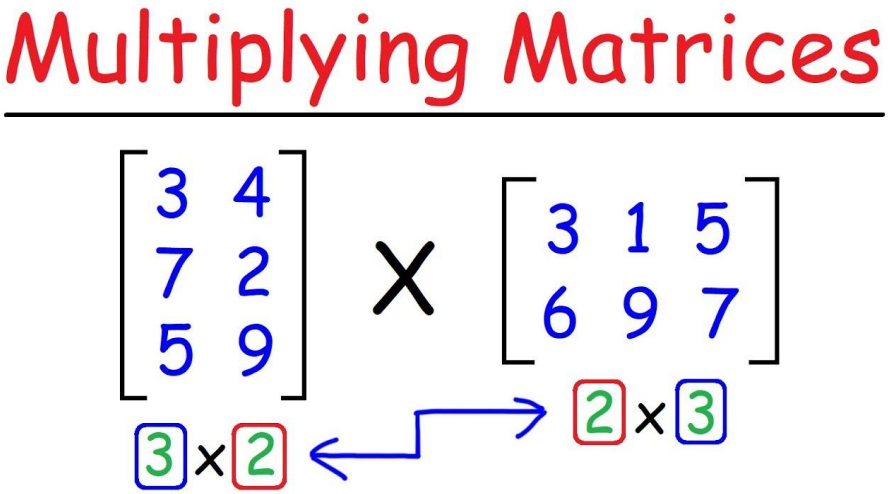

Figure 3 - Image of multiply matric. Src: https://i.ytimg.com/vi/2spTnAiQg4M/maxresdefault.jpg

The activation function converts the linear output to a non-linear output as that is needed for the model to have multiple layers.

There are two main activation functions:

Sigmoid - which is used for the final output of the model. The output is always between 0 and 1, so it is used for binary classification. ***[Ref3]***

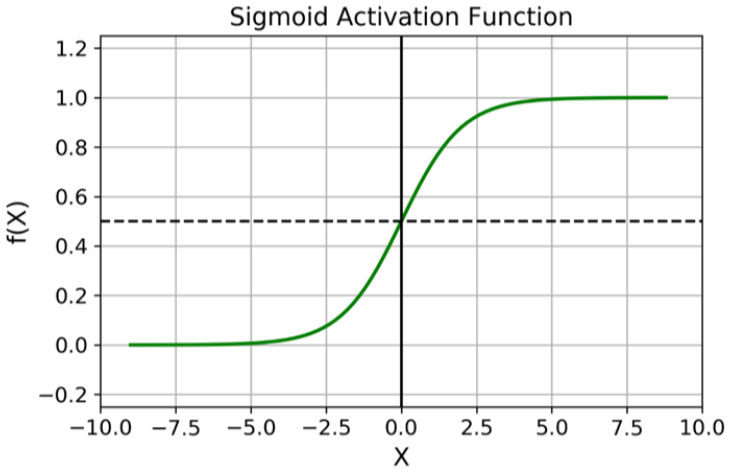

Figure 4 - Graph of sigmoid function. Src: https://ambrapaliaidata.blob.core.windows.net/ai-storage/articles/Untitled_design_13.png

ReLU (Rectified Linear Unit) - this is used for the nodes in the hidden layers. This is because ReLU is constant, making it simpler to work out the derivative. ReLU helps reduce the vanishing gradient problem. ***[Ref4]***

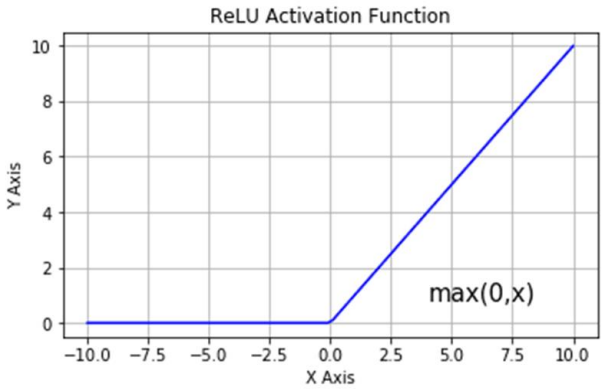

Figure 5 - Graph of the relu function. Src: https://www.nomidl.com/wp-content/uploads/2022/04/image-10.png

The data set has 23 columns (not including the target and the columns I dropped) so the model takes 23 inputs. The model also has 1 hidden layer, which takes 29 units as the input and returns 37 units. I got the number of units by running the model multiple times and adjusting the units to see what returns the highest accuracy. The output of the previous layer must have the same number of units as the input of the next layer.

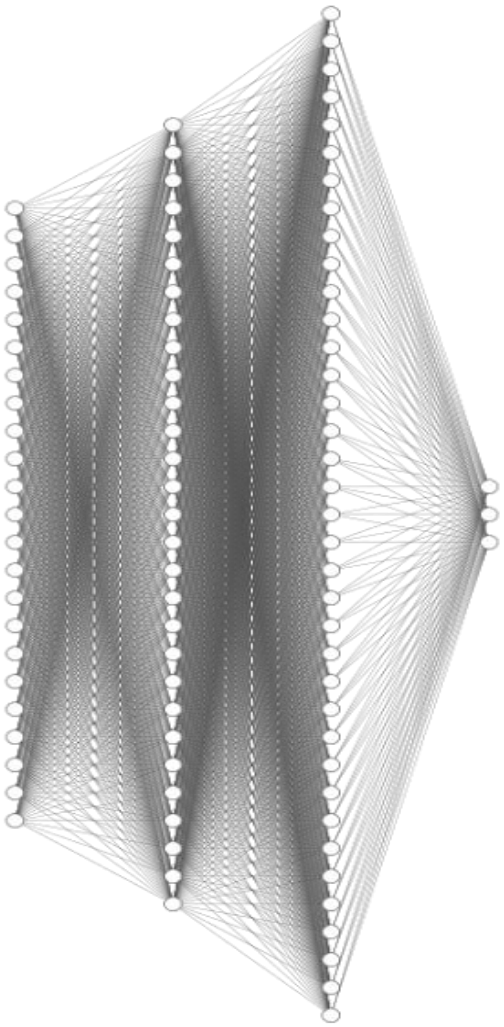

Figure 6 - Image of what my ann looks like. Src: https://alexlenail.me/NN-SVG/

The model output returns 3 values, which are the probabilities of each category. The activation function of the final output can't be a sigmoid because I need multi-class classification. So, I used a SoftMax function, which is an extension of sigmoid to return the probabilities. ***[Ref5]***

The formula for the SoftMax function is $\frac{e^z i}{\sum e^z}$

- z being the outputs
- e is the exponent (this is used so the output never goes under 0 as you can't get a negative probability)

Using the argsmax function I found the index of the highest value, meaning the highest probability, which is the output.

2.3 Training the ANN

3 Steps to train the model.

Forward propagation - starts from the input goes through all the layers and then the output, this runs the model.
Getting the loss - The output of the forward propagation ($\hat{y}$) is compared to the target (y). The difference is called error, the loss function takes that error to generate losses. ***[Ref6]***

There are two types of errors

- Binarized error – Is the output right or wrong? Easier to understand but provides less information. This is used to evaluate the model's performance for the final output.
- Continuous error – How much the output was off by? This is harder to understand but provides more information. Making it useful to train the model.

There are two main loss functions

- Mean squared error (MSE) - for continuous data.
- Cross-entropy - for categorical data. This is what I used in the model. Cross entropy is the difference between two probabilities. The equation for this is $-\sum_{i=1}^{n} p \ log2\,(q)$.

Backpropagation - starts from the output and works backwards to adjust the weights, this trains the model and adjusts the weights using the errors. Backpropagation works by implementing gradient descent.

The gradient descent algorithm:

- Sets a random number of minimums
- Loops over training iterations
- Calculates the derivate at the number
- Updates the value using this equation: number = number - learning rate * derivative

The goal gets represented as an error function, so the solution is the point with the smallest error. The derivative is used to know how to adjust the weights to find the best solution. This is done by finding the minima.

To work out the minima, calculate the derivative, set it to 0 and solve for x. This will return the values where the derivative is 0, which are the points that are changing from positive to negative and vice versa. To find the minima, the point has to go from a negative derivative on the left to a positive derivative on the right. ***[Ref7]***

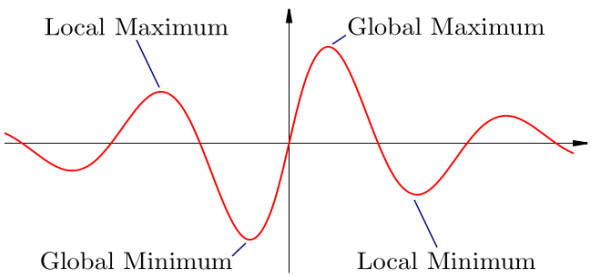

Figure 7 - Graph of the minima. Src: https://images.squarespace-cdn.com/content/v1/58757ed7f5e231cc32494a1b/1509464620786-ODRGQHAZZYMZURUVS0WG/main-qimg-5f0742ae54865cfebc9dae6a6d7fb66b%5B1%5D.png?format=1500w

The learning rate is a small number that is used to take small steps (opposite to the gradient) so that the minima is not missed. If the learning rate is too large the local minima might get missed (Exploding gradient). If the learning rate is too small the model might not reach the local minima because it is moving very slowly (Vanishing gradient). For my model, I chose 0.01 because it is usually the default value, and it seems to work. ***[Ref8]***

The epoch is the number of times the model gets trained, this is done by looping through the model, according to Gretel ***[Ref9]*** a good starting point is “3 times the number of columns in your data”. For me this is 3000, I looked at the loss and, if the loss plateaus, I reduced the number of epochs. This means the model finished learning.

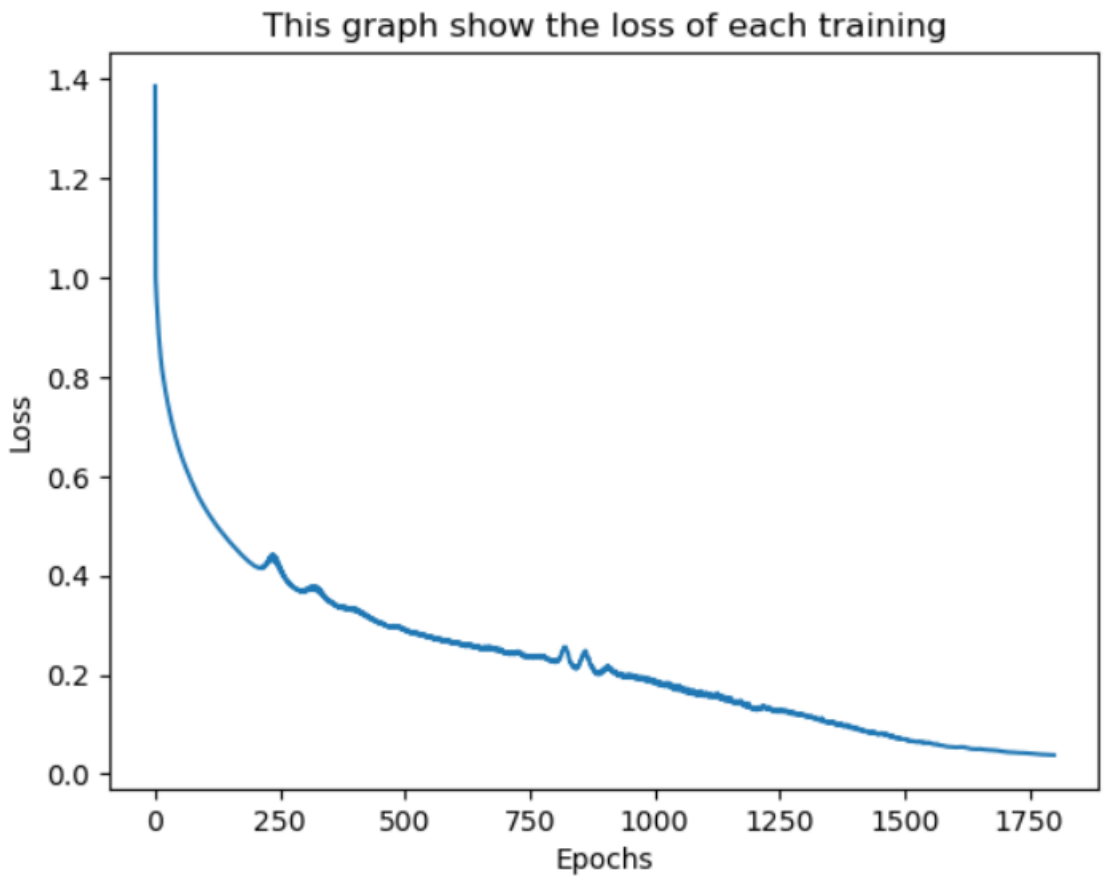

Figure 8 - Graph to show loss function using pyplot

Section 3 - Evaluation

3.1 How I measured the performance of my Model?

1. **Accuracy** - To get the accuracy I compared y to $\hat{y}$ and if it is correct, it gets added to an array. The length of that array is divided by the total predictions and multiplied by 100 to give a percentage. I ran this 100 times and got the average.
2. **Loss** - A loss close to 0 means the error rate is really low, I used the loss to adjust the epoch to get closest to 0.

3.2 Results

When I ran my model 100 times my total accuracy was 98%

```
totalAcc = 0

for i in range(100):
  ann , loss_function ,optimizer = makeANN(29, 37)
  train_target , loss_arr =  train(ann , loss_function ,optimizer ,train_target)
  acc  = result_function(ann , loss_function ,optimizer)
  totalAcc += acc



totalAcc = totalAcc/100
print(f"The total average accuracy of this ANN is {round(totalAcc * 100)}%")
```

```
The total average accuracy of this ANN is 98%
```

Figure 9 - Screenshot of my total accuracy

References

[Ref1] - Cancer Research UK. Lung cancer statistics. Available at: https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/lung-cancer

[Ref2] - Deep Learning and Artificial General Intelligence: Still a Long Way to Go MaciejŚwiechowski. (n.d.). Available at: https://arxiv.org/pdf/2203.14963.pdf.

[Ref3] - Wood, T. Sigmoid Function. DeepAI. Available at: https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function.

[Ref4] - Vivek Praharsha. ReLU (Rectified Linear Unit) Activation Function. OpenGenus IQ: Computing Expertise & Legacy. Available at: https://iq.opengenus.org/relu-activation/.

[Ref5] - Furnieles, G. Sigmoid and SoftMax Functions in 5 minutes - Towards Data Science. Available at: https://towardsdatascience.com/sigmoid-and-softmax-functions-in-5-minutes-f516c80ea1f9.

[Ref6] - Wambui, R. Cross-Entropy Loss and Its Applications in Deep Learning - neptune.ai. Available at: https://neptune.ai/blog/cross-entropy-loss-and-its-applications-in-deep-learning.

[Ref7] - Mathsisfun.com. (2017). Maxima and Minima of Functions. [online] Available at: https://www.mathsisfun.com/algebra/functions-maxima-minima.html [Accessed 5 Jan. 2023].

[Ref8] - Setting the learning rate of your neural network. Jeremy Jordan. Available at: https://www.jeremyjordan.me/nn-learning-rate/.

[Ref9] - Gretel.ai. How many epochs should I train my model with? Available at: https://gretel.ai/gretel-synthetics-faqs/how-many-epochs-should-i-train-my-model-with#:~:text=The%20right%20number%20of%20epochs,again%20with%20a%20higher%20value.

[Ref10] **–** I took a course in Udemy to understand deep learning. Available at: https://www.udemy.com/share/104Ylw3@miWWni4CFwYcdnT8X7x0XvtKKP4NI5XuXbKnV2da6z7ReKfV5yFiKMjP2ZmreN4e9w==/.

Table of Figures