Report On

Thyroid Disease Detection Using Multiclass Multilayer Neural Network

Submitted

For

Soft Computing Project

By

PraveenKumar Suthar (151070018)
Prafull Parmar (151070025)

DEPARTMENT OF COMPUTER ENGINEERING & INFORMATION TECHNOLOGY
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE
MUMBAI-400019

2018-2019

Statement Of Candidates

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources.

I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in my submission.

I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Mr. PraveenKumar Suthar	**Mr. Prafull Parmar**
(151070018)	**(151070025)**

Date:

Statement of Candidates

We state that work embodied in this Project titled **"Thyroid Detection Using Multiclass Multilayer Neural Network"** forms our own contribution of work. The report reflects the work done during the period of candidature but may include related preliminary material provided that it has not contributed to an award of previous degree. No part of this work has been used by us for the requirement of another degree except where explicitly stated in the body of the text and the attached statement**.**

Mr. PraveenKumar Suthar	**Mr. Prafull Parmar**
(151070018)	**(151070025)**

Date:

Approval Sheet

This is to certify that Mr. PraveenKumar Suthar, Mr. Prafull Parmar, students of B.Tech Computer Engineering, has completed the report entitled, “**Thyroid Detection Using Multiclass Multilayer Neural Network**” to our satisfaction.

Dr. S.G.Bhirud	**Name and Signature**
Professor, Computer Dept.	**(Lab-In-Charge)**

Place: Veermata Jijabai Technological Institute, Mumbai.

Date: / /2019

Contents

ABSTRACT

Thyroid disease is one of the widespread disease in the world. It has different levels of danger to the patient. By using neural networks for classifying the particular symptoms we can help in predicting the particular category of thyroid disease thus providing a valuable addition to the detection and treatment of patients.

Chapter 1

Introduction

The aim of this project is to use multiclass multilayer neural network to detect if a patient has hypothyroid disease given the record and symptoms of the patient.

1.1 Why detecting Thyroid is important?

Hypothyroidism, also called underactive thyroid or low thyroid, is a disorder of the endocrine system in which the thyroid gland does not produce enough thyroid hormone. It can cause a number of symptoms, such as

1. Fatigue
2. Increased sensitivity to cold
3. Constipation
4. Dry skin
5. Weight gain
6. Puffy face
7. Hoarseness
8. Muscle weakness
9. Elevated blood cholesterol level
10. Muscle aches, tenderness and stiffness
11. Pain, stiffness or swelling in your joints
12. Heavier than normal or irregular menstrual periods
13. Thinning hair
14. Slowed heart rate
15. Depression
16. Impaired memory
17. Enlarged thyroid gland (goiter)

Occasionally there may be swelling of the front part of the neck due to goiter. Untreated hypothyroidism during pregnancy can lead to delays in growth and intellectual development in the baby or congenital iodine deficiency syndrome.

Worldwide, **too little iodine in the diet** is the most common cause of hypothyroidism. In countries with enough iodine in the diet, the most common cause of hypothyroidism is the autoimmune condition Hashimoto's thyroiditis. Less common causes include: previous treatment with radioactive iodine, injury to the hypothalamus or the anterior pituitary gland, certain medications, a lack of a functioning thyroid at birth, or previous thyroid surgery. The diagnosis of hypothyroidism, when suspected, can be confirmed with blood tests measuring thyroid-stimulating hormone (TSH) and thyroxine levels.

1.2 How many people are suffering from Thyroid?

Worldwide about one billion people are estimated to be iodine deficient; however, it is unknown how often this results in hypothyroidism.In the United States, hypothyroidism occurs in 0.3–0.4% of people.

Subclinical hypothyroidism, a milder form of hypothyroidism characterized by normal thyroxine levels and an elevated TSH level, is thought to occur in 4.3–8.5% of people in the United States.

Chapter 2

2.1 Neural Networks

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurones) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

Why use Neural Networks ?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyse. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

A Simple Neuron

An artificial neuron is a device with many inputs and one output. The neuron has two modes of operation; the training mode and the using mode. In the training mode, the neuron can be trained to fire (or not), for particular input patterns. In the using mode, when a taught input pattern is detected at the input, its associated output becomes the current output. If the input pattern does not belong in the taught list of input patterns, the firing rule is used to determine whether to fire or not.

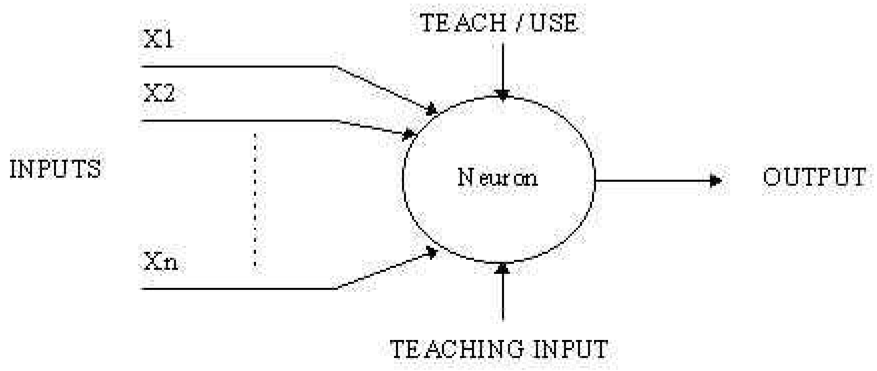


Fig:1 A simple neuron

Firing Rules

The firing rule is an important concept in neural networks and accounts for their high flexibility. A firing rule determines how one calculates whether a neuron should fire for any input pattern. It relates to all the input patterns, not only the ones on which the node was trained.

2.2 Architecture of Neural Network

2.2.1 Feed-forward networks

Feed-forward ANNs (figure 2) allow signals to travel one way only; from input to output. There is no feedback (loops) i.e. the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. They are extensively used in pattern recognition. This type of organisation is also referred to as bottom-up or top-down.

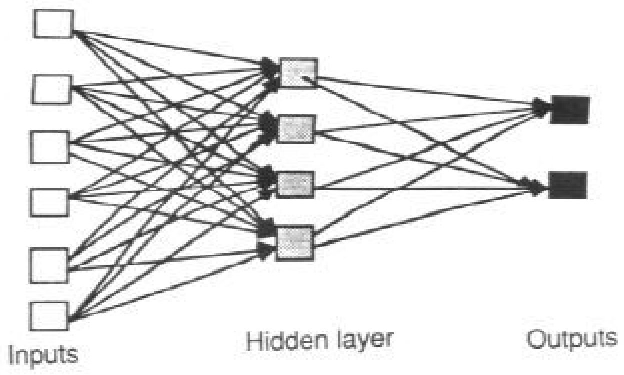


Figure 2 : An example of a simple feedforward network

2.2.2 Feedback networks

Feedback networks (figure 2) can have signals travelling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic; their 'state' is changing continuously until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organisations.

2.2.3 Network Layers

The commonest type of artificial neural network consists of three groups, or layers, of units: a layer of "input" units is connected to a layer of "hidden" units, which is connected to a layer of "output" units. (see Figure 2)

1. The activity of the input units represents the raw information that is fed into the network.

2. The activity of each hidden unit is determined by the activities of the input units and the weights on the connections between the input and the hidden units.

3. The behaviour of the output units depends on the activity of the hidden units and the weights between the hidden and output units.

This simple type of network is interesting because the hidden units are free to construct their own representations of the input. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents.
We also distinguish single-layer and multi-layer architectures. The single-layer organisation, in which all units are connected to one another, constitutes the most general case and is of more potential computational power than hierarchically structured multi-layer organisations. In multi-layer networks, units are often numbered by layer, instead of following a global numbering.

2.3 Hyperparameter tuning for machine learning models

When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to as **hyperparameters** and thus this process of searching for the ideal model architecture is referred to as *hyperparameter tuning*.

2.3.1 Grid Search

Grid search is arguably the most basic hyperparameter tuning method. With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model, and selecting the architecture which produces the best results.

Consider the standard classification framework - you have a sample which you divide into training sample (*Strain*) and validation sample (*Svalid*). You are solving an optimization problem *P* (which would usually be something like minimize training error plus a regularization term), which is a function of the model parameters, say *w*, the training sample *Strain* and some hyperparameters, say *α* and *β*. Solving the optimization problem for a *fixed* set of values of α and β gives you a value of w. Since the optimal value of w (call it $w*$) is a function of α and β , we can write it as follows:

$$w*(\alpha,\beta)=\text{argmin}wP(w,\alpha,\beta,Strain)$$

Now you use this $w*$ to predict on the validation sample to get validation error. You can view this scenario in terms of a "validation error function": the function takes as inputs the hyperparameters α and β , and returns the validation error corresponding to $w*(\alpha,\beta)$.
So the goal of hyperparameter optimization is to find the set of values of α and β, that minimize this validation error function.

Note that this validation error function is very expensive to evaluate -- for each value of α and β , to find the value of this function, you need to solve the optimization problem P. Further, this function might be non-convex, non-smooth, etc. so that it is impractical to find the global minimum of this function in a principled way. Therefore, we resort to grid search : pick a bunch of values of α -- (α1,α2,…) , pick a bunch of values of β -- (β1,β2,…) and for each pair of values, evaluate the validation error function. Then pick the pair that gives the minimum value of the validation error function. The pairs (α1,β1),(α1,β2),…,(α2,β1),(α2,β2) when plotted in space look like a grid, hence the name.

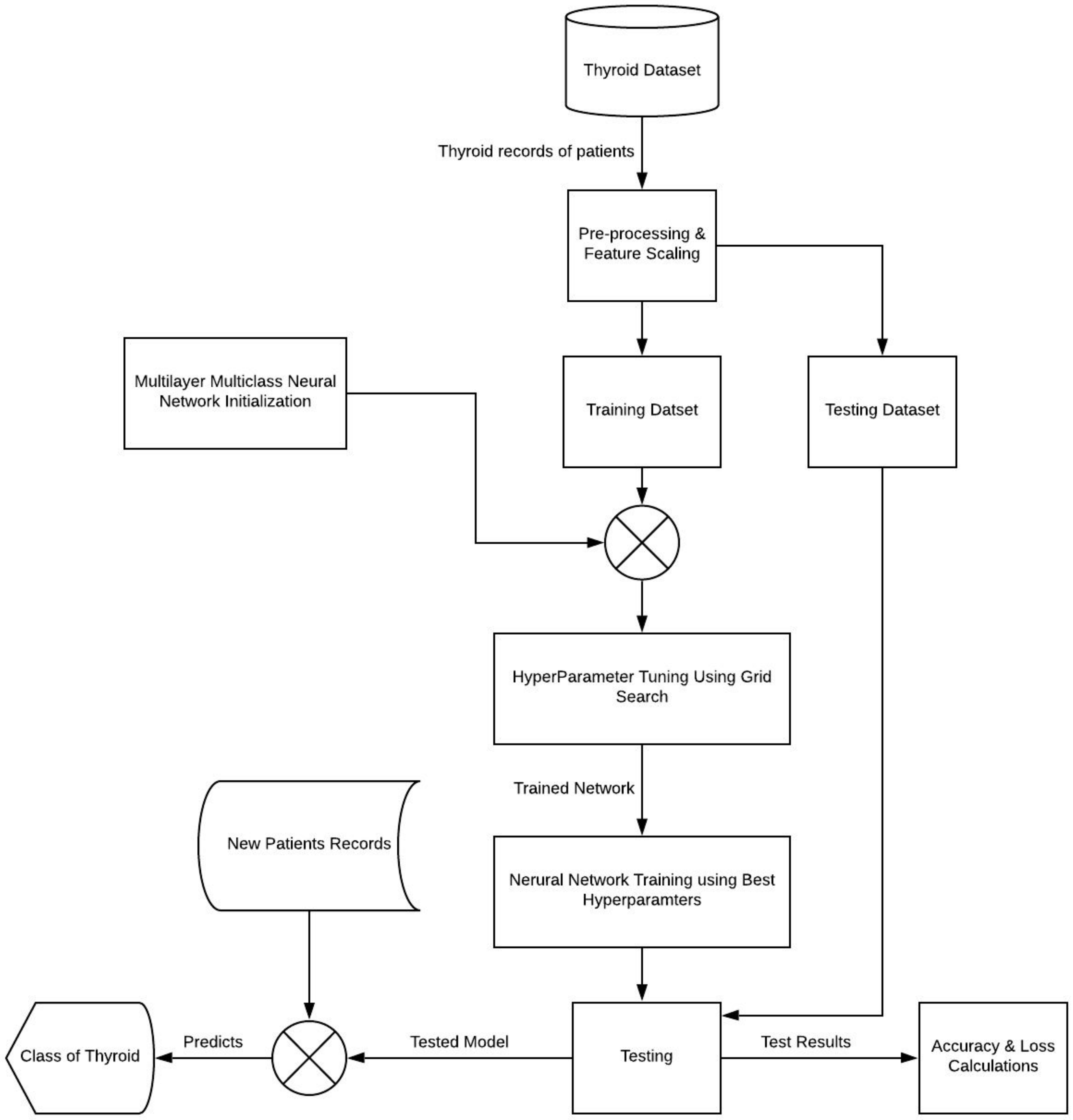
2.4 Proposed System for Thyroid Disease Detection

2.4.1 Dataset

The Dataset is taken from the UCI dataset repository.
Data set summary:

1. Number of attributes: 21 (15 attributes are binary,6 attributes are continuous)
2. Number of classes: 3
    a. Class 1: HypoThyroid
    b. Class 2: Subclinical hypothyroid
    c. Class 3: Normal
3. Number of learning examples: 3772
4. Number of testing examples: 3428

2.4.2 FlowChart


Thyroid Dataset
Thyroid records of patients
Pre-processing & Feature Scaling
Multilayer Multiclass Neural Network Initialization
Training Datset
Testing Dataset
HyperParameter Tuning Using Grid Search
Trained Network
New Patients Records
Nerural Network Training using Best Hyperparamters
Class of Thyroid
Predicts
Tested Model
Testing
Test Results
Accuracy & Loss Calculations

2.4.2 Method

The Neural network will be a multiclass neural network. Number of Hidden layers will be selected from a range of 1 to 3 hidden layers. Hyperparameters of the model will be tuned for increasing the accuracy. The neural network model will be Evaluated using k-cross validation technique.

The architecture of the neural network for the classification will have 3 hidden layers with 10 hidden neurons in first hidden layer and 6 neurons units in each second and third hidden layer with ReLu as the activation function and the output layer will consist of 3 neuron units so as to classify the input into one of the 3 categories of thyroid. The activation function for the output layer neuron is chosen as softmax function.

This network uses the cross_entropy loss function to find the errors in the network output and uses Adam optimizer to find the optimal values of weights that minimizes the loss function and hence increase the accuracy of our classifier.

2.5 Code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import confusion_matrix

# Importing the dataset
dataset = pd.read_csv('Ann_train.csv')
X = dataset.iloc[:, 0:21].values
y = dataset.iloc[:, 21].values
y = pd.get_dummies(y)

# Separate training and testing files
X_train, y_train = X, y
dataset = pd.read_csv('Ann_test.csv')
X_test = dataset.iloc[:, 0:21].values
```

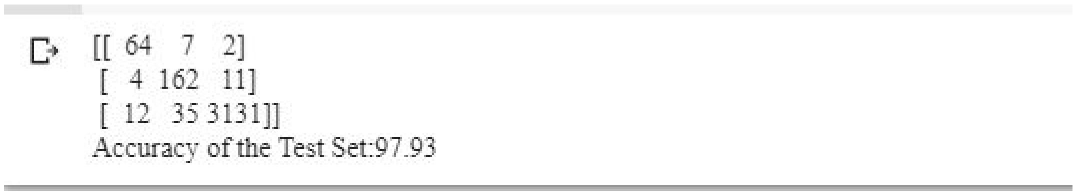
```
y_test = dataset.iloc[:, 21].values

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Function to create model, required for KerasClassifier
def create_model():
  # Initialising the ANN
  classifier = Sequential()

  # Adding the input layer and the first hidden layer
  classifier.add(Dense(units = 10, activation = 'relu', input_dim =21 ))

  #Adding the second hidden layer
  classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))

  #Adding the third hidden layer
  classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))

  # Adding the output layer
  classifier.add(Dense(units = 3, activation = 'softmax'))

  # Compiling the ANN
  classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
  return classifier

#Doing GridSearch for tuning Hyperparameters
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
# create model
model = KerasClassifier(build_fn=create_model, verbose=0)
# define the grid search parameters
batch_size = [10, 20, 40, 60, 80, 100]
epochs = [10, 50, 100,200]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1)
```

```
grid_result = grid.fit(X_train, y_train)
# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))

# Fitting the ANN with the best hyperparameters  to the Training set
dic = grid.best_params_
best_batch_size= dic['batch_size']
best_epochs = dic['epochs']
classifier = create_model()
classifier.fit(X_train, y_train, batch_size= best_batch_size, epochs = best_epochs)

# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred =1* (y_pred > 0.5)

# Creating the pred_class array
pred_class = []
for row in y_pred:
  row = list(row)
  pred_class.append(row.index(max(row))+1)
pred_class = np.array(pred_class)

# Making the Confusion Matrix
cm = confusion_matrix(y_test, pred_class)

#Printing the Confusion matrix and accuracy
print(cm)
from sklearn.metrics import accuracy_score
print("Accuracy of the Test Set:%.2f" % (accuracy_score(pred_class,y_test)*100))
```

Chapter 3

Results and Discussions

Grid search was used for finding the best set of hyperparameters for tuning the Neural Network. Best result was found for **batch size=20** and **number of epochs= 200**. Best result were obtained when three hidden layers were used in the neural network. The number of neurons in each hidden layer were also a crucial factor in determining the overall accuracy of the classifier. Increasing the number of neurons in the first hidden layer had now effect on the accuracy of the model. However changing the number of neurons in the second and third hidden layer had a significant effect on the accuracy of the classifier. A classifier with three hidden layers was selected as the final model for the prediction of the thyroid disease. The first hidden layer contains 10 neurons with ReLU (Rectified Linear Unit) as activation function was selected. Both Second and Third hidden layers included 6 neurons each with ReLU as their activation functions.
An Accuracy of **97.81%** was successfully achieved.



```
[[  64    7    2]
 [   4  162   11]
 [  12   35 3131]]
Accuracy of the Test Set:97.93
```

Fig 4: Output

Chapter 4

Summary and Conclusions

A Multiclass Multilayer Neural Network was developed and was successfully trained and tested on the Thyroid disease datasets, which yielded an accuracy of 97.81% . Neural Network was able to perform the multiclass classification of the three classes of the Thyroid disease.As the dataset contained around 92% records with class 3(normal subjects) it was very important to for the accuracy of the neural network to be greater than 92% and as the accuracy achieved is 97.81%, thus it would be safe to conclude that the neural network has achieved a respectable accuracy in predicting the Thyroid disease.

Chapter 5

Future Work

Different Neural Network architectures could be tried and tested for increasing the accuracy and reliability of neural networks in classifying thyroid diseases. The approach could be expanded for diagnosis of other diseases as well.

Chapter 6

References

1. **UCI Machine Learning Repository for Thyroid Dataset.**
2. **https://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html**
3. **https://www.jeremyjordan.me/hyperparameter-tuning/**
4. **https://en.wikipedia.org/wiki/Neural_network**