IJIRSET
International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY

e-ISSN: 2319-8753 | p-ISSN: 2347-6710

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 11, Issue 1, January 2022

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.569

9940 572 462 | 6381 907 438 | ijirset@gmail.com | www.ijirset.com

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

Using Neural Nets to Classify Bipolar NPN Transistors into Equivalent Classes

TOLOJANAHARY Dina Andriamiaina [1], RABOANARY Roland [2] ,

HASINA TAHIRIDIMBISOA Nirina Maurice[3]

PhD Student, Department of High Energy Physics, University of Antananarivo, MADAGASCAR[1]

Full Professor, Department High Energy Physics, University of Antananarivo, MADAGASCAR[2]

Doctor Lecturer, Department of High Energy Physics, University of Antananarivo, MADAGASCAR[3]

ABSTRACT: In this paper, we consider supervised neural networks (neural nets) to classify NPN transistors into equivalent classes. Based on our algorithm, an equivalent class is defined by collection of transistors that share at least certain characteristics from their respective data-sheets. In different words, a transistor that belongs to an equivalent class can be substituted by another different transistor in the same class when implemented in certain electrical circuit. We collect many data-sheets from different transistor manufacturers that are available and free online. These data are fed into our neural networks as a feed-forward propagation. Our results are compelling and suggest that we are in the right track. So far our proposed neural nets architecture reaches 97.5% of accuracy during training and only 70% of accuracy from testing samples. The second part of this article is focused on a very broad review and reproduction of characteristic curves of a transistor. To achieve this, we realize a simple circuit with a bipolar NPN transistor. Numerical and analytic computation are considered for this last part. The results are summarized as a linear progression problem. We coded everything using python and our algorithm are presented in this manuscript. We claim that even if our study is only focused on the type of NPN bipolar transistor, one can extend easily our concept to different types of transistor such as PNP, JFET , MOSFET and many other well-known types.

KEYWORDS: Neural Networks, Neural nets, Feed-Forward, bipolar transistor.

I. INTRODUCTION

In the high level of electronic domain, nowadays, the AI(Artificial Intelligent) is the one of the most indispensable weapon, for the development of technology. In this scientific review, we are going to see at which point the ANN is efficient for contribute the electronic task on transistor. In this instance, we want to apply the Feed-forward ANN algorithm on the bipolar transistor type NPN. For that, in the first task we consider a classic problem that may help amateur electrician or electrical engineer. We consider a classification problem. To be precise we classify transistors that are widely available to use and also their data-sheets are accessible to everyone. Our ultimate motivation is to use machine learning to explore the set of similar electronic components from different manufacturers. In this way, our goal is to have an automated and intelligent machine that decide and classify components into in equivalent classes depending on their characteristic and properties to use when implemented in a circuit. To achieve our goal and based on the availability of data as we mentioned earlier, we only consider transistors of type NPN. However, our study here can be generalized to other electronic components than transistors. Moreover, one extra motivation why we choose transistor is explained as follow. Transistors are the fundamental building blocks of most integrated electronic circuits (ICs). A quick example of these are CPU, RAM and many other ICs. In the second part, we will construct a data from a transistor functioning on simple circuit in common emitter. Our goal is to learn a neural networks for predicting the output voltage Vout with the variable value of the base resistor.

II. RELATED WORK

II-1 Neural networks review

During the last couple of decades, application and development of an artificial intelligent or AI has never cease to increase. Nowadays, artificial neural nets has specifically become, among many others, one indispensable tool to achieve outstanding results from wide range of area. For more details, we recommend reader to visit [1]-[6] and their list of references. A natural explanation to this is based on the observation that well trained machine can nowadays outperform human performance capability for some specific task. At least, this is indeed the case when we consider the

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

"Alpha Game Go". In fact, the authors of [7] manage to master this game using well trained machine and beat the world champion player. Motivated by these long list of success, we prefer to use the AI as algorithm which performs our goal. The loss function implemented in our neural nets follows our consideration that we have categorical inputs data. Accordingly, we penalize the neural nets with the cross-entropy loss function,

$$E(W)= - \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^{(n)} \log(\hat{y}_k (x^{(n)}, w)). \quad (1)$$

Our convention for the training data set is D=$\{x^{(n)}, t^{(n)}\}$. The pair {x , t} respectively the input target pair. This way, an intuitive interpretation of the objective function (1) is to measure the relative distance between the output $\hat{y}$ (x,w) and the target t. The training process consist of minimizing the overall distance between the input/target pairs. This is accomplished by repetition evaluation of the gradient.

$$\frac{\partial E}{\partial W}$$

We use the backpropagation [8] algorithm which is a traditional method by (Rumel-hart et al., 1986) to evaluate this gradient efficiently. For simplicity, we consider fully connected networks architecture in this study. Since our problem is a multi-class classification networks type, our output nodes are,

$$\hat{y}_i = \frac{e^{a_i(w.x)}}{\sum_k e^{a_k(w.x)}} . \quad (2)$$

Indeed this is the appropriate expression to model ŷ. A straightforward to explain this choice is to reduce the problem to binary classification. In this way, equation (2) reduces to the well-known logistic function,

$$y = \frac{1}{1+e^{-a(w.x)}}. \quad (3)$$

The expression (2) is conventionally called softmax in the literature of neural nets AI.
For some completeness shake of our discussion we are now ready to give a brief review on a transistor.

II.2 Overview of NPN transistors

For concreteness, we choose a type of NPN transistor for our review. Briefly, transistor was original invented by Willian Shokle [9]. This was during the electronic revolution around the fifties. More elaborated and detailed review is found in [10]. It is more often to use transistors as linear current amplification. But they can also be operated as electronic gates, depending on how we implement them in a circuit. We focus only on this linear operation and identify parameters that will become as input data to our neural networks. Naturally, NPN transitors have three electrodes attached to them disregarding their sizes or and their shapes. **Figure 1** is an illustration of the inside of a NPN transistor exhibiting the name of the three different electrodes.

Figure 1: To the left, the three different electrodes are specified in the first two.The last figure to right is commonly used as a symbol of a NPN transistor incircuit.

In general, there are four functioning modes of a bipolar transistor [11]. These modes are blocked mode, active mode, saturate mode and reverse-active mode. To exhibit these modes, one plots the curve obtain from the equation where I_C measures the current flow going through the collector and V_{BE} is the differential potential between the emitter and base electrodes. Concretely, a typical curve of I_C vs V_{BE} is illustrate in the following figure (**Figure 2**).

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

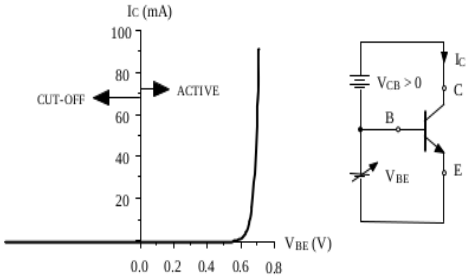

Figure 2: this picture is extracted from \Aero2 Signals and systems (Part2),Notes on BJT and transistors circuit

- Active mode: $I_C \backsimeq 0A$ for $V_{BE}<0,5V$. Increasing V_{BE} slowly in the region $V_{BE}>0,5V$, the current I_C start to raise and take asymptotically exponential value of V_{BE} . $V_{BE}= 0,7V$is the common value at operating time of a transistor when used as linear amplifier.

- Blocked mode: this is the region where $I_C= I_B= I_E \backsimeq 0$.

For completeness of our discussion, a typical curve from $I_C = f (V_{CE})$ also is needed to give a full curve characteristic of a transistor. In fact, curves obtain from I_C vs V_{CE} is usually plotted alongside I_B as a parameter. The following figure, represented in **Figure 3**, is again captured from "Aero2 Signals and systems (Part2), Notes on BJT and transistors circuit".

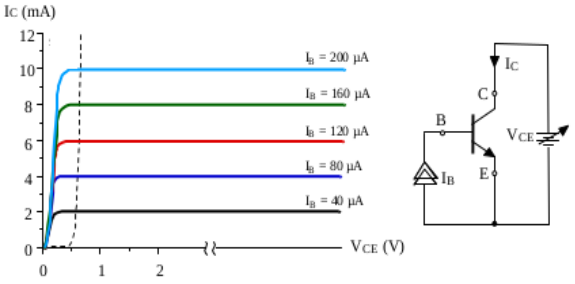

Figure 3: The two remaining modes active and saturate are easily read fromthe above curves.

- Active region: $V_{CE} > V_{BE}$, $I_C= \beta I_B$. It does not depend on V_{CE}.

- Saturate region: $V_{CE}< V_{BE}$, $I_C \longrightarrow 0$ if $V_{CE} \longrightarrow 0$. $V_{CEsat} = 0,2V$

III. METHODOLOGY

The following figure (depicted in the **Figure 4**) is an illustration of a fully connected networks that contains n + 1 nodes in the input layer, L +1 nodes in the hidden layer and K nodes for the K classes in the output layer.

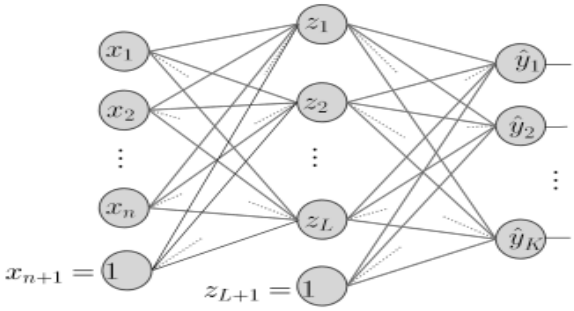

Figure 4: Shows a fully connected networks

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

The above figure illustrate the type of architecture we use to achieve our results. More details about this will be discussed in the next section.

III-1 Neural Nets Architecture

In this paper, our task consist of a feed-forward neural networks, [12] is well written review on this topic. But here, we just use it directly alongside a backpropagation optimization as our learning algorithm. To achieve our results, we use fully connected layers with only one single hidden layer. In total we have three layers if we include the input and output layers. We fix the number of nodes in the input layer to be equal to 7. This includes the bias node which always corresponds to an input $x_7 = 1$. Then, we tested three different scenario for the number of nodes in the hidden layer. We respectively set the number of nodes to be equal to 6, 8 and finally 12. However, the nodes in the last (output) layer is again fixed. We set it equal to 20 nodes. Naturally, this number of nodes in the last stage set the number of disjoint class we consider. In this way our label for each input $\vec{X}$ is a vector of dimension equal to 20, for instance $\vec{t} = [0, \cdots, 0, 1, 0, \cdots, 0]$. The components of $\vec{t}$ is populated with 0 except at the class where it belongs to. In this perspective, the components of $\vec{t}$ are probabilities in one to one corresponding to each of the 20 different classes. We will see the results and conclusions in the remaining section the summary of the above experiments.

IV. EXPERIMENTAL RESULTS

IV-1 Classification Results

In first experiment we train our neural nets, we prepare collection of data into a CSV file. Following [13], the available raw data we collected from the manufactures are summarized into a table. A sample of such table is presented in **Table 1**.

Table 1: Shows the example of data from manufactures.

Mark	P_C(W)	I_C(A)	V_{CEmax}(V)	F_{max}{Mhz}	HFE(min/max)
2N5929	100	30	80	200	20/100
2N6585	125	10	350	25	7/35
BC183L	0.300	0.200	30	150	120/...
BC184L	0.300	0.200	30	150	240/....

Our dataset consist of data collected from 50 transistors. Accordingly, we process the transformation of these raw dataset into a CSV file in order to use them as train/test data for our neural networks. We parameterize our python codes presented as **Algorithm 1**, see **Appendix A**, in terms of the following steps. First, when launching, it asks the number of hidden layers according to the wishes of any user. Then, it ask the number of units in each hidden layer following the desired number of the hidden layers in the previous step.

Finally, just before the training step, it ask the number of epochs according to the wishes of the user. In all of this, we fixed the learning rate to be equal to 0.05. But different user of our codes can easily change this to any appropriate value depending on their use. Following many other authors, the splitting ratio to separate our dataset into train/test data is 0.80.

IV-1-1 Results:

We now summarize in terms of the following table (**Table 2**) the results of our learning algorithm.

Table 2: Show the accuracy rate after the algorithm learning with number neuron of hidden layer is 6,8,12.

Neurons of Hidden Layer	Epochs	l-rate	Train(error/accuracy)	Test(error/accuracy)
6	10000	0.05	4,723/97,5%	18,607/70%
8	10000	0.05	2,221/97,5%	20,849/70%
12	10000	0.05	0,867/97,5%	23,77 /70%

Based on the above results, one finds that the outcomes accuracy of our learning algorithm are independent of the size of the single hidden layer. In fact, we only observe a fast decrease in error values during both the training and test process when we increase the numbers of units in the hidden layer.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

IV-2 Regression simulation results

In this second experiment, we consider the circuit presented in **Figure 4**. We take V_{BB} = 10V, V_{CC} = 10V ,V_{BE} = 0,7V , HFE = 10, and the base resistor R_B = 50kΩ. Data is collected by varying the linear potentiometer at the collector which takes values between 0.3Ω to 3000Ω. In this way, we collect the corresponding voltage V_{OUT}. Our goal is to explore the linearity functioning of a transistor and predict that this is indeed the case when using linear regression prediction algorithm. To achieve this, we refer to our **Algorithm 2** presented in the **Appendix B** of this manuscript.

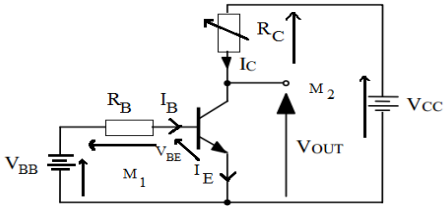

Figure 4: A simple circuit to study the linear mode functioning of a NPN transistor.

s The linear regression model we are interested is,

$$V_{OUT} = b_0 + b_1 . R_C \quad (4)$$

The coefficients b_0 and b_1 are unknown and yet they are to be predicted. This is the main goal in this second experimental study. Now, run a simulation of the circuit presented in **Figure 4**. During this simulation, we collect discrete values of V_{OUT}. In total, the length of the collected data is 50. But we again split these data into train/test dataset with the split ratio equal to 70%. Approach of **Algorithm 2** for prediction uses least squares methods to fit our linear model in equation (4). The final, results of this second experiments is summarized as follow. After training and test simulations, we reach RMSE≃0.0, where RMSE:= Root Mean Squared Error, b_0= 20.0, b_1= −1.86.
A recapitulating table of the above results is presented in **Table 3**.

Table 3: Shows the result of simulation.

R_C (Kilo Ohm)	V_{OUT}(measuread)(Volt)	V_{OUT}(predicted)(Volt)
2	16,281	16,285
1,54	17,212	17,214
1,2	17,768	17,767
2,3	15,722	15,725
2,75	14,885	14,887
0,7	18,698	16,699
1,82	16,615	18,829
0,63	18,828	18,829
0,91	18,307	18,308
2,65	15,075	15,074

In summary, **Table 3** shows that up to two point decimal our linear model predicts very accurately the data measured from the circuit. This is in perfect agreement that we are indeed working at the linear mode functioning of the transistor as described in the circuit **Figure 4**.

V. CONCLUSIONS

In this paper, we claim that we achieve our goal in classifying bipolar NPN transistors into different inequivalent classes. There were 20 classes to be predicted from our expectations when we prepare our dataset. We reach respectively accuracy of 97.5% for train and 70% for the test process. However, our proposal if far from being perfect. Indeed, there are still some improvement to be made in order to increase our accuracy. But on average, as a first experiment concerning this topic, we are already satisfied on the progress we made so far. After this first attempt of study, many project comes to ones mind for future projects if one is interested to continue and extrapolate this work.

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

Appendix:

Appendix A

==========

ALGORITHME I: TRANSISTOR'S CLASSIFICATION

==========

```
1.import random
2.import numpy as np
3.from  random import seed
4.from  random import random
5.from  random import randrange
6.from csv import reader
7.def load_csv(filename):
8.    dataset = list()
9.    with open(filename, 'r' ) as file:
10.        csv_reader = reader(file)
11.        for row in csv_reader:
12.            if not row:
13.               continue
14.            dataset.append(row)
15.   return dataset
16.def str_column_to_float(dataset, column):
17.    for row in dataset:
18.        row[column] = float(row[column].strip())
19.def Pmax_min_and_Pmax_max(power):
20.       tol=(power*30)/100
21.       Pmin=power-tol
22.       Pmax=power+tol
23.       returnPmin,Pmax
24.def Imax_min_and_Imax_max(intensity):
25.    tol=(intensity*35)/100
26.    Imin=intensity-tol
27.    Imax=intensity+tol
28.    returnImin,Imax
29.def Fmin_and_Fmax(frequency):
30.    tol=(frequency*75)/100
31.    Fmin=frequency-tol
32.    Fmax=frequency+tol
33.    returnFmin,Fmax
34.def HFEmin_and_HFEmax(hfe):
35.    tol=(hfe*75)/100
36.    HFEmin=hfe-tol
37.    HFEmax=hfe+tol
38.    returnHFEmin,HFEmax
39.def train_test_split(dataset, split):
40.    train = list()
41.    train_size = split * len(dataset)
42.    dataset_copy = list(dataset)
43.    while len(train) <train_size:
44.          index = randrange(len(dataset_copy))
45.          train.append(dataset_copy.pop(index))
46.    return train, dataset_copy
47.def activate(weights, inputs):
48.    activation = weights[-1]
49.    for i in range(len(weights)-1):
```

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

```
50.         activation += weights[i]*inputs[i]
51.     return activation
52.def softmax(activation,total):
53.     soft= np.exp(activation)/total
54.     return soft
56.def initialize_network(n_inputs,n,x, n_outputs):
57.     network = list()
58.     hidden_layer1= [{ ' weights ':[random() for i in
        range(n_inputs)]}  for i in range(x)]
59.     network.append(hidden_layer1)
60.     for i in range(n):
61.         hidden_layer2= list()
62.         hidden_layer2= [{ ' weights ':[random() for i in
            range(x+1)]} for i in range(x)]
63.         network.append(hidden_layer2)
64.         output_layer=  [{ ' weights ':[random() for i in
            range(x+1)]} for i in range(n_outputs)]
65.     network.append(output_layer)
66.     return network
67.def transfer(activation):
68.     return 1.0 / (1.0 + np.exp(-activation))
69.def transfer_derivative(output):
70.     return output *(1.0 - output)
71.def forward_propagate(network,row):
72.     inputs = row[:-2]
73.     for i in range(len(network)):
74.         layer=network[i]
75.         if i!=len(network)-1:
76.            new_inputs = []
77.            for neuron in layer:
78.                neuron['act']= activate(neuron[ ' weights ' ],
                   inputs)
79.                neuron[ ' output ' ]=transfer(neuron['act'])
80.                new_inputs.append(neuron[ ' output ' ])
81.            inputs = new_inputs
82.         elif i==len(network)-1 :
83.              new_inputs = []
84.              total=0.0
85.              for neuron in layer:
86.                  neuron['act'] =activate(neuron[' weights '],
87.                  inputs)
88.                  total+=np.exp(neuron['act'])
89.              for neuron in layer:
90.                  neuron[' output ']=softmax(neuron['act'],total)
91.                   new_inputs.append(neuron[' output '])
92.      inputs =new_inputs
93.      return inputs
94.def backward_propagate_error(network,expected):
95.     for i in reversed(range(len(network))):
96.         layer = network[i]
97.         errors = list()
98.         ifi!= len(network)-1:
99.            for j in range(len(layer)):
100.               error = 0.0
101.               for neuron in network[i + 1]:
102.                   error+=(neuron['weights'][j]*neuron['delta'])
103.               errors.append(error)
```

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

```
104.          for s in range(len(layer)):
105.              neuron = layer[s]
106.              neuron['delta']=errors[s]*transfer_derivative(
                  neuron['output'])
107.          else:
108.              for j in range(len(layer)):
109.                  neuron = layer[j]
110.                  errors.append(expected[j]-neuron['output'])
111.              for s in range(len(layer)):
112.                  neuron = layer[s]
113.                  neuron[ ' delta ' ] = errors[s]
114.def update_weights(network, row, l_rate):
115.    for i in range(len(network)):
116.        inputs = row[:-2]
117.        if i != 0:
118.           inputs = [neuron['output'] for neuron in network[i- 1]]
119.        for neuron in network[i]:
120.            for j in range(len(inputs)):
121.                neuron['weights'][j]+=l_rate*neuron['delta']
                    *inputs[j]
122.            neuron['weights'][-1]+=l_rate*neuron[ ' delta ' ]
123.def train_network(network, train, l_rate, n_epoch, n_outputs):
124.    for epoch in range(n_epoch):
125.        sum_error = 0.0
126.        for row in train:
127.            outputs = forward_propagate(network, row)
128.            expected = [0 for i in range(n_outputs)]
129.            expected[row[-1]] = 1
130.            sum_error += -sum([expected[i]*np.log10(outputs[i])
131.            for i in range(len(expected))])
132.            backward_propagate_error(network, expected)
133.            update_weights(network, row, l_rate)
134.        print('>epoch=%d,lrate=%.3f,error=%.3f'%(epoch, l_rate,
            sum_error))
135.def test_network(network,test,n_outputs):
136.    sum_error =0.0
137.    for row in test:
138.        outputs = forward_propagate(network, row)
139.        expected = [0 for i in range(n_outputs)]
140.        expected[row[-1]] = 1
141.        sum_error += -sum([expected[i]*np.log10(outputs[i])
142.        for i in range(len(expected))])
143.        print("Transistor is ",row)
144.        print("The result of test is:",outputs)
145.    print("The error of test is:",sum_error)
146.    filename='TransistorData2.csv'
147.dataset=load_csv(filename)
148.for i in range(len(dataset[0])-1):
149.    str_column_to_float(dataset, i)
150.    dataset_copy=dataset
151.pool=list()
152.while len(dataset_copy)!=1:
153.      A=list()
154.      index=randrange(len(dataset_copy))
155.      A.append(dataset_copy.pop(index))
156.      for row in A:
157.          Pmax_min,Pmax_max=Pmax_min_and_Pmax_max(row[0])
```

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

```
158.           Icmax_min,Icmax_max=Imax_min_and_Imax_max(row[1])
159.           Vmax=row[2]
160.           Fmin,Fmax=Fmin_and_Fmax(row[3])
161.           HFEmin,HFEmax=HFEmin_and_HFEmax(row[4])
162.       for row1 in dataset_copy:
163.           if row1[0]>=Pmax_min and row1[0]<=Pmax_max and
164.              row1[1]>=Icmax_min and row1[1]<=Icmax_max and
165.              row1[3]>=Fmin and row1[3]<=Fmaxand row1[4]>=HFEmin
166.              and row1[4]<=HFEmax and row1[2]==Vmax:
167.              A.append(dataset_copy.pop(dataset_copy.index(row1)))
168.       pool.append(A)
169.DATA=list()
170.i=0
171.for x in pool:
172.    B=list()
173.    for row2 in x:
174.        row2.append(i)
175.        B.append(row2)
176.    DATA.append(B)
177.    i=i+1
178.FINALDATA=list()
179.for line in pool:
180.    for line1 in line:
181.        FINALDATA.append(line1)
182.print(FINALDATA)
183.seed(1)
184.Dataset=FINALDATA
185.train,test=train_test_split(Dataset,0.80)
186.n_inputs = len(Dataset[0]) - 2
187.n_outputs = len(set([row[-1] for row in Dataset]))
188.hidden_number=int(input("please enter the number of hidden layer:"))
189.neuron_number=int(input("please enter the number of neuron for each hidden
layer:"))
190.epoch_number=int(input("please enter the number of epoch:"))
191.network = initialize_network(n_inputs,hidden_number,neuron_number,n_outputs)
192.train_network(network,train,test,0.05,epoch_number, n_outputs)
```

==========

Appendix B

==========

ALGORITHM II:PREDICTION OF OUTPUT VOLTAGES

==========

```
1.import random
2.from  random import randrange
3.from math import sqrt
4.def mean(values):
5.    cov=sum(values) / float(len(values))
6.    returncov
7.def covariance(x, mean_x, y, mean_y):
8.    covar = 0.0
9.    for i in range(len(x)):
10.       covar+= (x[i] - mean_x) * (y[i] - mean_y)
11.   returncovar
12.def variance(values, mean):
```

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

```
13.     return sum([(x-mean)**2 for x in values])
14.def coefficients(dataset):
15.     x = [row[0] for row in dataset]
16.     y = [row[1] for row in dataset]
17.     x_mean, y_mean = mean(x), mean(y)
18.     b1 = covariance(x, x_mean, y, y_mean) / variance(x, x_mean)
19.     b0 = y_mean - b1 * x_mean
20.     print("The coefficient  b0=%f  and b1=%f"%(b0,b1))
21.     return [b0, b1]
22.def train_test_split(dataset, split):
23.     train = list()
24.     train_size = split * len(dataset)
25.     dataset_copy = list(dataset)
26.     whilelen(train) <train_size:
27.           index = randrange(len(dataset_copy))
28.           train.append(dataset_copy.pop(index))
29.     return train, dataset_copy
30.def simple_linear_regression(train,test):
31.     predictions=list()
32.     b0,b1=coefficients(train)
33.     for row in test:
34.         yhat=b0+b1*row[0]
35.         predictions.append(yhat)
36.     return predictions
37.def current_base_in_mA(vbb,resistor):
38.     res=(vbb-0.7)/resistor
39.     return res
40.def V_out_in_V(tension,hfe,I_B,R):
41.     vout=tension-hfe*I_B*R
42.     returnvout
43.def rmse_metric(actual, predicted):
44.     sum_error = 0.0
45.     for t in range(len(actual)):
46.         prediction_error = predicted[t] - actual[t]
47.         sum_error += (prediction_error ** 2)
48.         mean_error = sum_error / float(len(actual))
49.     returnsqrt(mean_error)
50.def evaluate_algorithm(dataset, algorithm, split, *args):
51.     train, test = train_test_split(dataset, split)
52.     test_set = list()
53.     for row in test:
54.         row_copy = list(row)
55.         row_copy[-1] = None
56.         test_set.append(row_copy)
57.     predicted = algorithm(train, test_set, *args)
58.     actual = [row[-1] for row in test]
59.     rmse = rmse_metric(actual, predicted)
60.     returnrmse
61.HFE=float(input("Please enter the current gain of the transistor:"))
62.while HFE<=0:
63.       HFE=float(input("Please enter the current gain of the
transistor:"))
64.VBB=float(input("Please enter the voltage in VBB(V):"))
65.VCC=float(input("Please enter the voltage in VCC(V):"))
66.R_B=float(input("Please enter the base resistor in Kilo_ohm:"))
67.R_C_limitmax=float(input("Please enter the collector resistor max
limit in Kilo_ohm:"))
```

| e-ISSN: 2319-8753, p-ISSN: 2347-6710| www.ijirset.com | Impact Factor: 7.569|

|| Volume 11, Issue 1, January 2022 ||

| DOI:10.15680/IJIRSET.2022.1101001 |

```
68.Dataset=list()
69.j=0
70.epochs=int(input("please enter the number of epochs :"))
71.Error=[]
72.while True:
73.       j+=1
74.       RMSE=0
75.       for i in range(150):
76.           R_C=random.uniform(0.3,R_C_limitmax)
77.           I_B=current_base_in_mA(VBB,R_B)
78.           Vout=V_out_in_V(VCC,HFE,I_B,R_C)
79.           Dataset.append([R_C,Vout])
80.       split=0.7
81.       RMSE = evaluate_algorithm(Dataset, simple_linear_regression
          ,split)
82.       Error.append(RMSE)
83.       print('The number of epoch=%d The Root Mean Square Error RMSE:
                %.3f ' % (j,RMSE))
84.       if RMSE==0:
85.          break
86.       if j>=epochs:
87.          break
88.index_min=Error.index(min(Error))+1
89.print(index_min)
```

REFERENCES

[1]Andrea Belgrano,BjornA.Malmgren and Odd Lindhal;Application of artificial neural networks (ANN) to primary production time series data;Journal of Plankton research/volume 23/number 6/pages 651-658 /2001.
[2]Igor N.Aizenberg,NaumN.Aizenberg,Pattern recognition Using Neural Network Based on Multi- valued neurons, LectureNotes in computer science , International Work-Conference on Artificial and Natural Neural Networks,IWANN'99 Alicante, Spain ,June 2-4,1999.Proceedings,volume II. p383-392.
[3]Juan Manuel Alonso –Wiber, M.PazSesmero, German Gutierrez, AgapitoLedezma ,And Araceli Sanchis.Handwritten Digit Recognition with Pattern Transformations and Neural Network Averaging. V.Mladenov et al (Eds),ICANN 2013, LNCS 8131, pp.335-342 20123.
[4]J.J Mhelo,V.Rivas,G.Romero,P.Castillo, A.Pascual, J.M Carazo;Improved Automatic Classification of Biological Particles from Electron –Microscopy Images Using Genetic Neural Nets,International Work-Conference on Artificial and Natural Neural Networks,IWANN'99 Alicante, Spain ,June 2-4,1999.Proceedings,volume II.p373-382.
[5]Francis Jeason and Tony White,Dynamic Memory For Robot Control Using Delay-Based Coincidence Detection Neurons.V.Mladenov et al (Eds).ICANN 2013, LNCS 8131,pp. 162-169, 2013.
[6]Francisco Morales, Paloma Ballesterros,Neural Networks in Automatic Diagnosis Malignant Brain Tumours.G.Goos,J. International Work-Conference on Artificial and Natural Neural Networks,IWANN'99 Alicante, Spain ,June 2-4,1999.Proceedings,volume II.p778-787.
[7]Silver ,David ; Huang, Aga; Maddison Chris ;Guez,Arthur; Sifre Laurent ;Driessche,George Van den ;Schrittwiser, Julian;Antonoglou,Ioannis;PanneerShelvam Veda (January 2016)."Mastering the game of Go with deep neural networks and tree search"Nature 529 ..4845.
[8] Daniel Groupe(2017);Principles of artificial network ;Avanced series on circuits and Systems-vol.6.
[9] Principles of transistor circuits ,Stan Amos and Make James, 9^{th}$ edition,2000.
[10] R.M MARTSON ,Diodes,Transistors and FET circuits Manual,1991.
[11] Aero2 Signals and systems (Part2),Notes on BJT and transistors circuit
[12] Murat H.Sazly A Brief Review of Feed forward Neural Networks.
[13] Tableau de transistor équivalence/transistor divers germanium.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor:
7.569

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

Scan to save the contact details