Prediction of Current gain in BJT
Common-Emitter configuration Amplifier

Submitted By- Abhishek Sharma (15MI435)

Arun Sagar(15MI433)

OBJECTIVE

To design a Deep Learning Algorithm that can calculate the current gain of the bi-polar junction transistor-based amplifier in common emitter-based configuration.

METHODOLOGY (Data generation & Collection)

The following steps were taken to reach the objective:

1. The data set for the different values of the resistor, capacitor to predict the current gain is prepared using **OrCAD** Software and other simulation tools by varying the values of the components (resistor and capacitor) and collecting the corresponding current gain for the amplifier.

2. Secondly the data is stored in the **.CSV** file as shown. The values of the resistor are in Ω, capacitor in μF, nF and current gain comes out as shown.

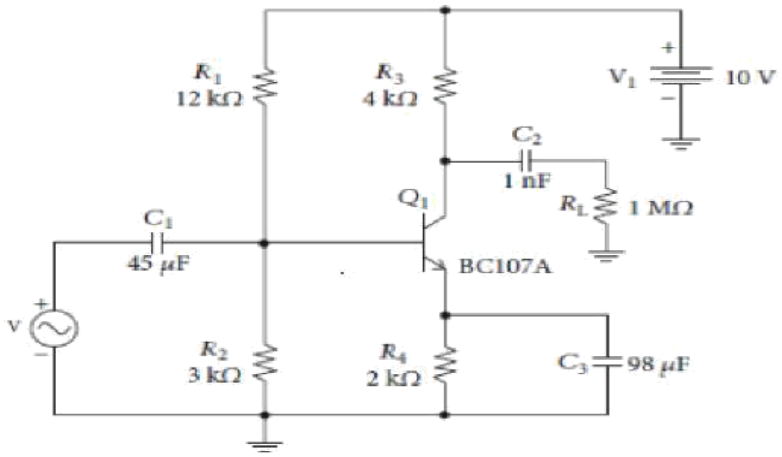


FIGURE 1: Nominal values of the amplifier circuit.

R1	R2	R3	R4	Ic/ib
15	3	4	2.5	149.41
15	2.25	5	1.5	150.66
9	2.25	4	1.5	162.94
12	3.8	4	1.5	166.54
15	3	3	2.5	149.84
12	2.25	5	1.5	156.84
9	2.25	5	1.5	162.25
9	3.8	3	2.5	163.44
9	3.8	5	2	164.41
15	3.8	5	2	157.53
15	3.8	3	2.5	154.99
9	3.8	3	1.5	171.68
15	3.8	4	2	158.3
9	3.8	3	2	167.19
9	3	3	2	163.91
15	3.8	4	1.5	163.1
12	3.8	4	2.5	158.32
15	3.8	5	1.5	162.04

Fig.2 data set for circuit in.csv file

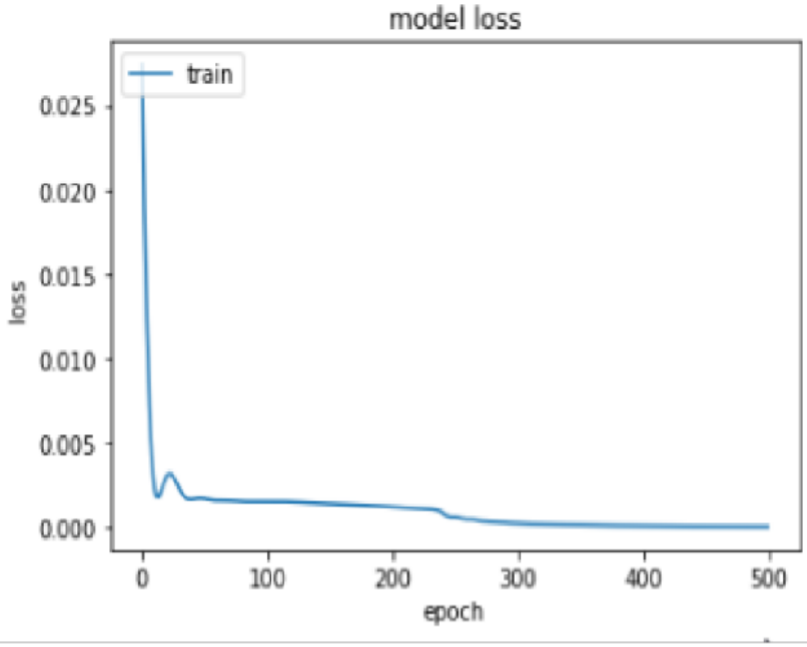
Nominal values of amplifier are shown taken in reference to Research Article**- Regression and ANN Models for Electronic Circuit Design**

3. To train the model first we import necessary pre-processing libraries like Panda and Numpy. For mathematical tasks we imported matplot.lib, then we imported sequential model from keras to create dense layers. With Pandas we imported data set we created in above steps and necessary normalization task is performed.
4. We split the data set in X_train, X_test, y_train and y_test with 80:20 ratio. Then we load sequential model and added the three dense layers with the following parameters

```
[ ] X=X/10
    y=y/1000
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)
    model = Sequential()
    model.add(Dense(20, activation='relu',input_dim = 4))
    model.add(Dense(15, activation='relu'))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')
    print(model.summary())
```

5. And fit the data set in the model file and train it on batch size of 50 and run it over 500 epochs.

Output Results



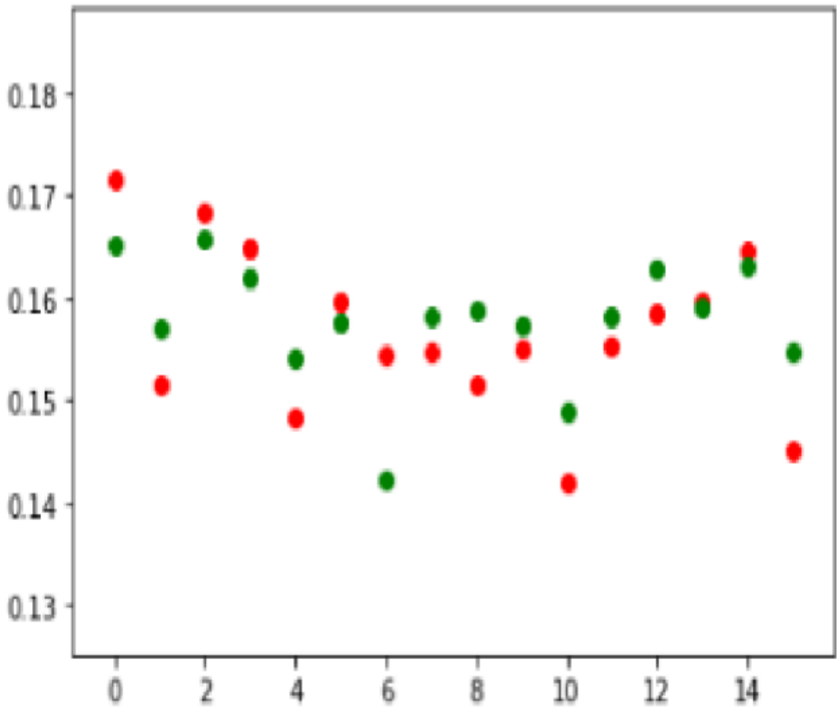


Fig.3 predicted result in green and actual in red