**Predicting Mosfet Drain current Through**

**Neural Networks.**

**Youssef Khaled Jomaa**

**Table of Contents**

[Introduction 3](#_Toc149614016)

[1.1. Problem Definition 4](#_Toc149614017)

[1.2. Problem Overview 4](#_Toc149614018)

[1.3. Project Approach 4](#_Toc149614019)

[1.5. Project Development Methodology 6](#_Toc149614020)

[1.6. The Used Tools in The Project (SW and HW) 6](#_Toc149614021)

[1.7. Report Organization 7](#_Toc149614022)

[Chapter 2: System Analysis 9](#_Toc149614023)

[2.1 Project specification 9](#_Toc149614024)

[2.1.1 Functional requirements 9](#_Toc149614025)

[2.1.2 Non-Functional requirements 9](#_Toc149614026)

[Chapter 3: Implementation and Testing 10](#_Toc149614027)

[3.1. Overview 10](#_Toc149614028)

[3.1.1. Dataset 10](#_Toc149614029)

[3.1.2 Models Overview 10](#_Toc149614030)

[3.2. Models Implementation and Testing 10](#_Toc149614031)

[3.2.1. Random Forest Model 10](#_Toc149614032)

[● Trial 1 11](#_Toc149614033)

[5.2.1.2. Traditional NN Model 12](#_Toc149614034)

[● Trial 12](#_Toc149614035)

[5.2.2.2. Deep Neural Network with Cross-Validation 13](#_Toc149614036)

[● Trial 1 13](#_Toc149614037)

[Chapter 4: Conclusion and Future Work 14](#_Toc149614038)

[4.1. Conclusion 14](#_Toc149614039)

[4.2. Future work 14](#_Toc149614040)

[4.3. Technical Obstacles 14](#_Toc149614041)

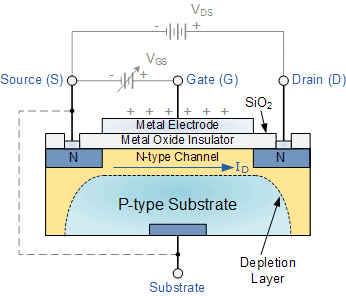
[4.3.1. Missing Values 14](#_Toc149614042)

[References 14](#_Toc149614043)

## Introduction

Transistor generally was designed when scientists aimed to amplify signals that were being downsized due to the long-travelled distance….. Where later on, enhancements took place leading to great inventions like switches.

Speaking which, the EFT type of transistors ex. MOSFET. Mosfet is a type of transistor

. that can work as a switch according to the voltage intake.

Mosfet consists of 4 parts [1. drain 2. source 3. P-type

Semiconductor 4.gate] Where a flow of current is generated

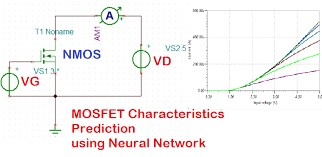
“Drain Current” from drain to source when the effective

amount of voltage is provided. And voltage deplete when

the voltage provides negatively

Our aim is to implement machine learning model to predict the drain current

i.e. The current flow from drain to source, Upon several input features like

* Gate voltage
* Drain Source Voltage
* Mobility
* Width
* Length
* …

**Figure SEQ Figure \\* ARABIC 1**

## 1.1. Problem Definition

-We are trying to implement machine learning model on MOSFET type to predict the drain current. Where MOSFET doesn’t depend on applying current like basic type transistor, but depends on applying voltage gate on semiconductor p-type material, and applying voltage between drain and source, which stimulates current flow between drain and source, so our aim is to predict this current flow with respect to different parameters.

## 1.2. Problem Overview

There are two types of MOSFET, N-channel and P-channel, and there are two modes for MOSFET, the enhancement mode and the depletion mode, talking of enhancement mode in N- channel type, it takes a positive voltage connected to the gate to attract the electrons, creating a stimulated flow of electrons between source and drain, with the effective voltage provided which has to be greater than threshold voltage, and with another battery connected between drain and source, Current flow is provided ,till when vds is equal to or greater than VGS, current is conducted through the transistor and become depend and only depend on VGS to strength the current flow.

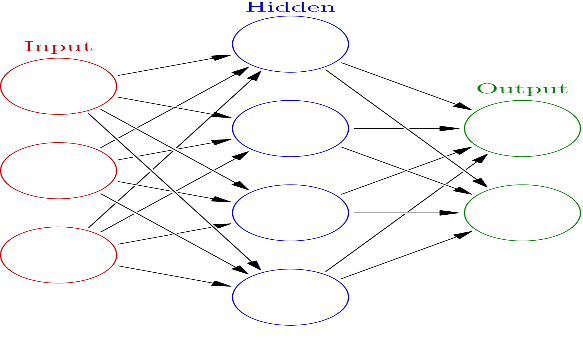
## 1.3. Project Approach

There are three approaches to developing a model of any semiconductor device.

1. ***Physics-based model:*** They are the most accurate, they are derived by applying device physics, solving equations, etc. As a result, they are difficult to develop, time taking and sometimes they are impossible to develop especially for new devices/materials where the device physics is unknown.
2. ***Empirical model:*** They are based upon curve fitting using a polynomial. Due to their fitting nature, they are less accurate and might not capture the variation of any parameter properly. But they are much easier to develop. Most of the MOSFET models available so far are semi-empirical in nature.
3. ***Look-up Table approach:*** It is another approach for model development where the data points are stored in form of a table. The simulator will pick up the input variable and its corresponding output without solving any equations. But a large number of data points are required. Accuracy is less compared to physics-based models.

MOS transistor characteristics can’t be fully described by conventional device physics and some properties are almost impossible to model or a huge computational resource is required. in this situation, the data -drive machine learning-based models are gaining the attention of researchers. In this approach in-depth knowledge of devices, and physics is not required. The training data is collected from real device measurement results or may be from a mixture of the real device and commercial device simulator (TCAD). The dataset contains feature variables such as gate and drain voltage, width, length, and other parameters. The target/output variable is drain current. After proper scaling, training is performed and then the trained model is tested with a new set of input variables. This approach reduces the gap between physics-based model development and development time with accuracy.

**Optimum Approach:** We build a NN model to train data to predict current drain.



**Figure SEQ Figure \\* ARABIC 3**

**Figure SEQ Figure \\* ARABIC 4**

## 1.5. Project Development Methodology

-We used the Hybrid approach, as the name implies, is a combination of the Waterfall and Agile methodologies. It takes the best parts of both Waterfall and Agile and combines them in a flexible yet structured approach that can be used across different projects. It is appropriate for projects that require the team to meet continuously changing requirements in a short amount of time.

-In our project we use Waterfall method where the steps are sequential where we cannot return to a previous step.

-The first step was EDA. Next step features preprocessing. After that we built NN model.

-We use Agile method where the steps are incremental where we can return to a previous step.

-In NN model weights and architecture are updated regularly and the model is rebuilt.

**Figure SEQ Figure \\* ARABIC 6**

## 1.6. The Used Tools in The Project (SW and HW)

**Software:**

* Deep Learning Algorithms:
* Neural Network (NN):

In deep learning, a neural network (NN) is a class of deep neural networks, most commonly predict outcome.

* Google Collaboratory.
* Python Programming Language.
* Python Libraries:
* Numpy v1.19.5
* Pandas v1.1.5
* Matplotlib v3.2.2
* Sklearn v0.24.2
* Keras
* TensorFlow

**Hardware:**

Our intention is to build a model to run on embedded chips, this means that our main focus is software, and we do not use any other hardware rather than a machine that can execute the program.

## 1.7. Report Organization

**Chapter 2**

We determine the suitable functional requirements and non-functional requirements to our project.

**Chapter 3**

We will present the project by designing System Component Diagram, System Class Diagrams, Sequence Diagrams and System GUI Design

**Chapter 4**

This chapter covers future purposes and potential work to be done to improve application for the sake of user experience and to overcome all possible issues.

# Chapter 2: System Analysis

## 2.1 Project specification

### 2.1.1 Functional requirements

* + - * GetUserInput:

Several parameters affecting the MOSFET like VDS, VGS, Mobility,

Length, Width,

* + - * PredictOutput:

Predict the current according to the input features.

### 2.1.2 Non-Functional requirements

* Performance: Response time is as quick as possible.
* Scalability: The capability of the app to handle growing number of features

**Figure SEQ Figure \\* ARABIC 8**

# Chapter 3: Implementation and Testing

## 3.1. Overview

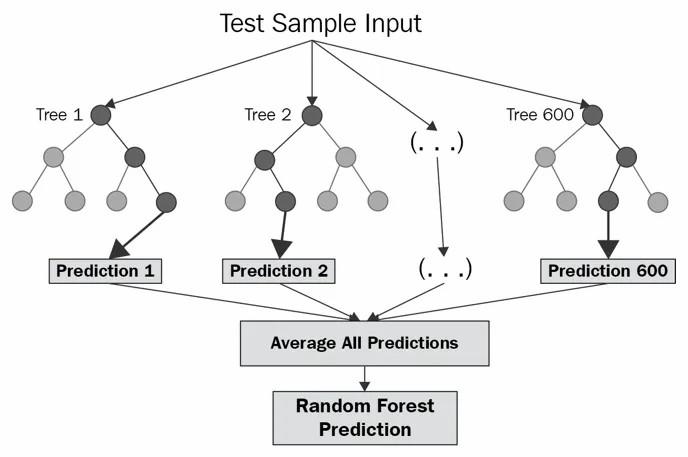
### 3.1.1. Dataset

**a- Dataset Description:**

* One Large Excel sheet (xlsx format) that contains data all for every feature as a row.

### 3.1.2 Models Overview

I chose 2 different models, Random Forest model, And neural network model in two different approaches.



## 3.2. Models Implementation and Testing

### 3.2.1. Random Forest Model

**Figure SEQ Figure \\* ARABIC 20**

A random forest is a supervised machine learning algorithm that is constructed from decision tree algorithms. That’s used to solve regression and classification problems. It establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome. Also, it reduces the overfitting of datasets and increases precision.

**How random forest algorithm works**

Bagging involves using different samples of data (training data) rather than just one sample. A training dataset comprises observations and features that are used for making predictions. The decision trees produce different outputs, depending on the training data fed to the random forest algorithm. These outputs will be ranked, and the highest will be selected as the final output.

##### Trial 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Trial 1** | | **RandomForestClassifier (n\_estimators=40,**  **random state=66)** | | |  |  | | --- | --- | | **Train Data Size** | **60%** | | **Test Data Size** | **40%** |   Trial Details |
| **Trial Description:**  The CF-based classification of patients into two classes was implemented in a Random Forest classifier with the number of trees equal to 40. In contrast to CNNs and DNNs, Random Forest did not have convolutional and pooling layers.  **Note:** This model is good at training and testing and takes less time because n\_estimators are small. |  |

#### 3.2.2. Traditional NN Model

##### Trial

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Trial 1 Architecture** | | **Flatten ()**  **Dense (64, activation='relu')**  **Dropout (0.5)** | | **Dense (32, activation='relu')**  **Dropout (0.5)** | | **Dense (1, activation=sigmoid)** | | |  |  | | --- | --- | | **Number of**  **Epochs** | **10** | | **Train Data Size** | **70%** | | **Test Data Size** | **30%** |   Trial Details |
| **Trial Description:**  We classified individual CT images into two types, (i) positive CT (pCT) images where imaging features associated with COVID-19 pneumonia could be unambiguously discerned, and (ii) negative CT (nCT) images where imaging features in both lungs were irrelevant to COVID-19 pneumonia. To enable image-based prediction, we adopted a deep learning framework of 6-layer convolutional neural networks (CNNs), Containing one input layer with input\_shape (200\*200\*5), 3 sets of dual convolutional, 3 fully connected (dense) layers, Containing stride (1,1), Batch Normalization, kernel\_initializer(uniform), padding (same), activation function Relu/Sigmoid and one output layer. |  |

#### 3.2.3. Deep Neural Network with Cross-Validation

##### Trial 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **Model 4** | | **Dense (38, input\_shape=38, activation='relu')**  **Dropout (0.5)** | | **Dense (64, activation='relu')**  **Dropout (0.2)** | | **Dense (64, activation='relu')**  **Dropout (0.2)** | | **Dense (48, activation='relu')**  **Dropout (0.2)** | | **Dense (16, activation='relu')**  **Dropout (0.5)** | | **Dense (2, activation='softmax')** | | |  |  | | --- | --- | | **K** | **5** | | **Number of Epochs** | **2000** | | **Train Data Size** | **80%** | | **Test Data Size** | **20%** | | **Training**  **Accuracy** | **40.02%, 43.12%,**  **41.86%,43.74% ,42.87%** | | **Testing**  **Accuracy** | **43.02%, 46.12%, 41.86%, 40.46%, 43.96%** | | **Average**  **Accuracy** | **43.08%** | | **Trial Evaluation** | **44.96%** | |
| **Trial Description:**  The CF-based classification of patients into two classes was implemented in a DNN Model. We adopted a deep learning framework of 6-layer Deep Neural Networks (DNNs), Containing one input layer with input\_shape (38),6 fully connected (dense) layers, with shuffling data, Activation function Relu/SoftMax and one output layer.  **Note:** This model is bad at training and testing |  |

# Chapter 4: Conclusion and Future Work

## 4.1. Conclusion

Critically, the prediction of drain current is not applicable and can’t be accurately predicted but requires training the model each time with different parameters.

## 4.2. Future work

We want to improve the model by building a NN model and implement different strategies to improve the accuracy like standardization and normalization.

## 4.3. Technical Obstacles

### 4.3.1. Missing Values

As we said before some features have missing values so features reduction was a mandatory step but after that many features have many NaN value and this is a reason to less accuracy and cannot predict well.

to handle this problem, we Fill these missing values with:

1. Column means.
2. Column median.
3. Column normal range.
4. Data Imputation.
5. Zero.

# References

[1[] https://rb.gy/kppzr](%5d%20https:/rb.gy/kppzr)