THESIS SUPERVISOR'S REPORT

I. IDENTIFICATION DATA

Thesis title:	**Applying Computer Vision, and MachineLearning Techniques for modelling andsimulation of autonomous Car**
Author's name:	**Omar Alif Abelhakim Allam**
Type of thesis :	master
Faculty/Institute:	Faculty of Mechanical Engineering (FME)
Department:	Department of Instrumentation and Control Engineering
Thesis reviewer:	Ing. Cyril Oswald, Ph.D.
Reviewer's department:	Department of Instrumentation and Control Engineering

II. EVALUATION OF INDIVIDUAL CRITERIA

Assignment **challenging**

How demanding was the assigned project?

The goal of the thesis is to design basic algorithms for autonomous driving of a car using machine learning methods and to test them on a suitable simulator.

Fulfilment of assignment **fulfilled with major objections**

How well does the thesis fulfil the assigned task? Have the primary goals been achieved? Which assigned tasks have been incompletely covered, and which parts of the thesis are overextended? Justify your answer.

The main objection is that although the assignment is fulfilled, the practical implementation is almost a pure copy of the on-line tutorial "Self Driving Simulation using NVIDIA's Model" available from https://www.computervision.zone/courses/self-driving-simulation-using-nvidias-model/ without any added value, differing only in the naming of some functions and variables.

Activity and independence when creating final thesis **F - failed.**

Assess whether the student had a positive approach, whether the time limits were met, whether the conception was regularly consulted and whether the student was well prepared for the consultations. Assess the student's ability to work independently.

The student could consult the work both with me and with another nominated specialist. Neither option was used practically the whole time. Thus, although he created the work without our help, he only copied the solution to the assigned practical problem from an online source.

Technical level

Is the thesis technically sound? How well did the student employ expertise in his/her field of study? Does the student explain clearly what he/she has done?

The technical level is adequate, but I do not rate it for the plagiarism mentioned.

Formal level and language level, scope of thesis

Are formalisms and notations used properly? Is the thesis organized in a logical way? Is the thesis sufficiently extensive? Is the thesis well-presented? Is the language clear and understandable? Is the English satisfactory?

The formal and linguistic level of the thesis is good, but I do not rate it for the reasons mentioned above.

Selection of sources, citation correctness **F - failed.**

Does the thesis make adequate reference to earlier work on the topic? Was the selection of sources adequate? Is the student's original work clearly distinguished from earlier work in the field? Do the bibliographic citations meet the standards?

The thesis does not contain any original work by the student.

III. OVERALL EVALUATION, QUESTIONS FOR THE PRESENTATION AND DEFENSE OF THE THESIS, SUGGESTED GRADE

Summarize your opinion on the thesis and explain your final grading.

The thesis does not contain any original work by the student. All code given as a solution to the given problem is a plagiarized online tutorial from https://www.computervision.zone/courses/self-driving-simulation-using-nvidias-model . Even the data used for training is taken without any change from https://github.com/rslim087a/track. Both source codes, i.e. the code submitted by the student and the code from the online tutorial, are attached to this review.

The grade that I award for the thesis is **F - failed.**

Date: **30.8.2023** Signature:

Appendix A - student source code vs on-line tutorial code

USED TRAINING DATA

```
!git clone https://github.com/oallam40/track # pure fork from https://github.com/rslim087a/track
```

MAIN PARTS OF MODEL TRAINING FROM SUBMITTED CNNModel.py
parts of the code used to create the graphs are omitted, although they also match the original

```
datadir = 'track'
columns = ['center', 'left', 'right', 'steering', 'throttle', 'reverse', 'speed']
data = pd.read_csv(os.path.join(datadir, 'driving_log.csv'), names = columns)

def path_leaf(path):
  head, tail = ntpath.split(path)
  return tail

data['center'] = data['center'].apply(path_leaf)
data['right'] = data['right'].apply(path_leaf)
data['left'] = data['left'].apply(path_leaf)

def load_img_steering(datadir, df):
  image_path = []
  steering = []
  for i in range(len(data)):
    indexed_data = data.iloc[i]
    center, left, right = indexed_data[0], indexed_data[1], indexed_data[2]
    image_path.append(os.path.join(datadir, center.strip()))
    steering.append(float(indexed_data[3]))
  image_paths = np.array(image_path)
  steerings = np.array(steering)
  return image_paths, steerings
image_paths, steerings = load_img_steering(datadir +'/IMG', data)

X_train, X_valid, y_train, y_valid = train_test_split(image_paths, steerings, test_size=0.2,
random_state=6)

def nvidia_model():
  model = Sequential()
  model.add(Convolution2D(24, kernel_size=(5,5), strides=(2,2), input_shape=(66,200,3),
activation='elu'))
  model.add(Convolution2D(36, kernel_size=(5,5), strides=(2,2), activation='elu'))
  model.add(Convolution2D(48, kernel_size=(5,5), strides=(2,2), activation='elu'))
  model.add(Convolution2D(64, kernel_size=(3,3), activation='elu'))
  model.add(Convolution2D(64, kernel_size=(3,3), activation='elu'))
  model.add(Dropout(0.5))
  model.add(Flatten())
  model.add(Dense(100, activation='elu'))
  model.add(Dropout(0.5))
  model.add(Dense(50, activation='elu'))
  model.add(Dropout(0.5))
  model.add(Dense(10, activation ='elu'))
  model.add(Dropout(0.5))
  model.add(Dense(1))
  optimizer = Adam(lr=1e-3)
  model.compile(loss='mse', optimizer=optimizer)
  return model

history = model.fit(X_train, y_train, epochs=30, validation_data = (X_valid, y_valid), batch_size=100,
verbose=1, shuffle=1)
```

MATCHING PARTS OF MODEL TRAINING FROM ORIGINAL SOURCE CODE

```
def getName(filePath):
    return filePath.split('\')[-1]

def importDataInfo(path):
    columns = ['Center', 'Left', 'Right', 'Steering', 'Throttle', 'Brake', 'Speed']
    data = pd.read_csv(os.path.join(path, 'driving_log.csv'), names = columns)
    data['Center']=data['Center'].apply(getName)
    print('Total Images Imported',data.shape[0])
    return data
path = 'myData'
data = importDataInfo(path)

def loadData(path, data):
  imagesPath = []
  steering = []
  for i in range(len(data)):
    indexed_data = data.iloc[i]
    imagesPath.append(f'{path}/IMG/{indexed_data[0]}')
    steering.append(float(indexed_data[3]))
  imagesPath = np.asarray(imagesPath)
  steering = np.asarray(steering)
  return imagesPath, steering
imagesPath, steerings = loadData(path,data)

xTrain, xVal, yTrain, yVal = train_test_split(imagesPath, steerings,
test_size=0.2,random_state=10)

def createModel():
  model = Sequential()

  model.add(Convolution2D(24, (5, 5), (2, 2), input_shape=(66, 200, 3),
activation='elu'))
  model.add(Convolution2D(36, (5, 5), (2, 2), activation='elu'))
  model.add(Convolution2D(48, (5, 5), (2, 2), activation='elu'))
  model.add(Convolution2D(64, (3, 3), activation='elu'))
  model.add(Convolution2D(64, (3, 3), activation='elu'))

  model.add(Flatten())
  model.add(Dense(100, activation = 'elu'))
  model.add(Dense(50, activation = 'elu'))
  model.add(Dense(10, activation = 'elu'))
  model.add(Dense(1))

  model.compile(Adam(lr=0.0001),loss='mse')
  return model

history = model.fit(dataGen(xTrain, yTrain, 100, 1),
                                  steps_per_epoch=300,
                                  epochs=10,
                                  validation_data=dataGen(xVal, yVal, 100, 0),
                                  validation_steps=200)
```

Second submitted file Flask_Socket_IO.py is the pure copy from the original source.