Traffic Sign Recognition
by Jisun Lee (U37416487)

ASSIGNMENT 1: PROJECT PROPOSAL DRAFT

Please use this template in creating your response. Retain the gray text and MS Word headings, and supply your responses where indicated. Your materials, in black 12-point Times New Roman, should not exceed 5 pages excluding this gray text, the references and your figures. You may add appendices, which should be referred to in the body of the paper, and which will be read on an as-needed basis. Note the evaluation criteria below, and leave plenty of time for editing so that your paper responds to them and you obtain the most favorable grade. You may alter your project plans as the term progresses: that is to be expected. Your changes will fit with this growing document. Keep notes of what you read so that you can provide references. Responses considered "good" should go beyond the minimum of what's requested.

1.1 SUMMARY DESCRIPTION

One- or (if necessary) two-paragraph overall purpose of a proposed term project. You will be free to change this in future but we want you to think through your most promising project idea as early as feasible because implementation can be time-consuming. The *clarity* of your summary is especially relevant, so avoid details in this summary. An example: *RealEval* will estimate the value of a house in the ABC neighborhood of Tucson based on recent sales of comparable properties.

We know about self-driving automobiles, in which the passenger may completely rely on the vehicle for transportation. However, in order to attain level 5 autonomy, cars must comprehend and observe all traffic rules. Many researchers and large companies, including as Tesla, Google, Apple, Mercedes-Benz, Ford, Audi, and others, are working on autonomous vehicles and self-driving automobiles in the realm of Artificial Intelligence and technological innovation. To achieve accuracy with this technology, cars must be able to comprehend traffic signs and make appropriate judgments.

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.

1.2 I/O EXAMPLES

At least two concrete examples of projected output for designated input. You will not be held to this output exactly—it is just illustrative at this stage. The *specificity* of your examples is especially relevant: avoid generality. An example: Input 123 Main Street, 4 bedrooms, 2 bathrooms, …. Output $650,000 with confidence 80%.

Input the image of the traffic sign, the program will classify what the image is based on shape and the color of the image.

1.3 REQUIREMENTS

Provide high-level requirements. Separate your requirements into three approximately even categories using triage (place requirements the two extreme categories first—*definite* and *nice-to-do*—and then place the remaining requirements in the middle category). Be conservative with your "definite" requirements: make them the bare necessity to have an actual project but no more. State requirements in declarative language such as "NUMRECO shall recognize numbers 0-9 from a 12 by 35 array of black-or-white pixels" (not "First I will build a neural net" because this is a procedure and a design element rather than a requirement). The *clarity* and properly declarative form of your requirements are especially relevant.

1.3.1 Definite Requirements

Using pandas, I need to extract the picture path and labels. Then, in order to forecast the model, I need to scale the photographs to 30x30 pixels and create a numpy array holding all of the image data. I need to import the accuracy score from sklearn.metrics and watch how the model predicted the real labels.

1.3.2 Requirements Not Classified Yet

Labels for learning and evaluation data are stored as ClassId columns in the csv file, so they are retrieved and stored as arrays. When learning deep learning, validation data is separated from learning data to prevent overfitting. And save all data as numpy array.

1.3.3 Nice-to-do Requirements

Create the multiple layers to make clear classification and to have high accuracy. Then, the SoftMax function is utilized in the Output Layer to determine the class of an input picture.

1.4 HOW SUCCESS WILL BE ASSESSED

Explain, as specifically as possible how success of the project will be assessed. Quantification is ideal. We realize that you can't know at this stage what realistic goals are, nor will we evaluate you project on this a lot but we want you to think this through. An example is "90% successful recognition of a cat in 1000 random images containing animals taken from the Web." The *clarity*, especially this specificity, of your assessment is especially relevant.

My goal for this project is that if the learning is done well, make sure that the evaluation data performs well with at least more than 70% of test accuracy.

1.5 TECHNOLOGY EXPLANATION

Explain what two machine learning technologies you are seriously considering--and why you feel they apply. One technology may be emphasized as the implementation and the other as an alternative for discussion. The *technical correctness* in this part is especially relevant, including your explanation of "why."

In this project, I will create a deep neural network model capable of categorizing traffic signs in an image. I can read and comprehend traffic signs using the model, which is a critical duty for all autonomous cars. The dataset contains almost 50,000 images of various traffic signals. It is further subdivided into 43 distinct classes. The dataset is highly varied; some classes contain many photos, while others have few images. The dataset is around 300 MB in size. The dataset has a train folder with photos for each class and a test folder with images for evaluating our model.

Input Layer
Conv2d
Conv2d
Max Pool
Conv2d
Max Pool
Conv2d
Max Pool
Flatten Layer
Fully connected Layer
Output Layer

Hidden Layers

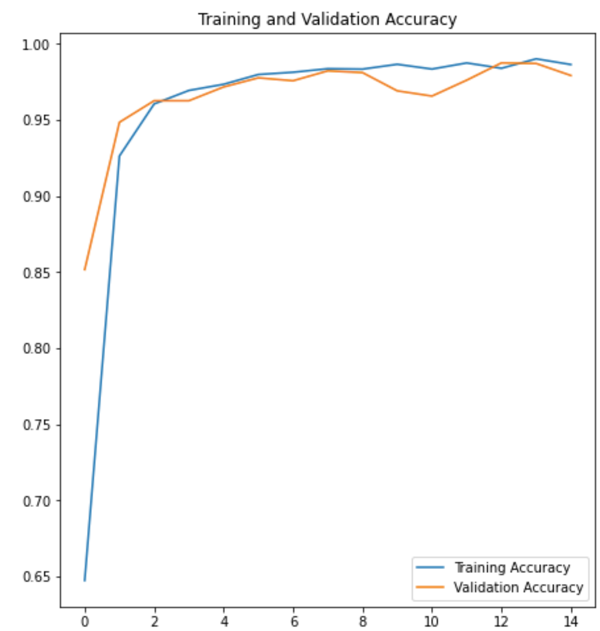
In short, the Conv2d layer is used to filter the picture and identify similarity, the Max Pool is used to extract the important data, and the Flatten Layer is used to convert a 2-dimensional vector into a 1-dimensional vector as a preparation for the linked layer that follows. Finally, the Output Layer uses the SoftMax function, which may offer the class of an input picture.

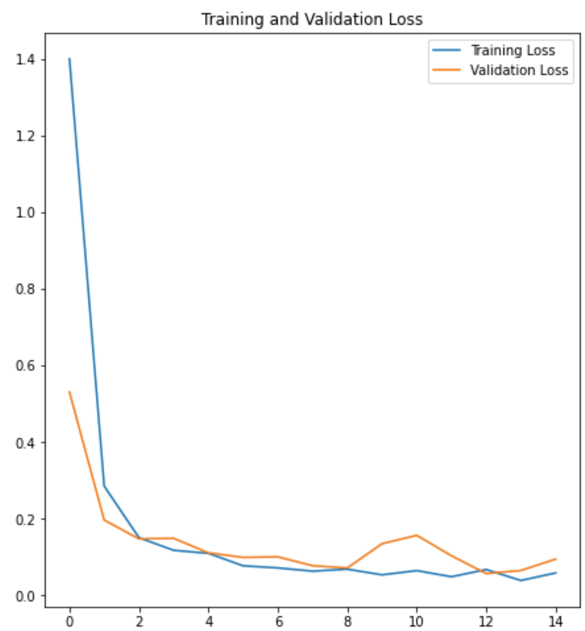
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 31, 31, 32)        896

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 14, 14, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 12, 12, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_3 (Conv2D)           (None, 4, 4, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 64)         0
 2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 256)               65792

 dense_1 (Dense)             (None, 43)                11051

=================================================================
Total params: 142,411
Trainable params: 142,411
Non-trainable params: 0
_________________________________________________________________
```

```
epoch = 15
history = model.fit(x_train, y_train, batch_size = 64, epochs=epoch, validation_data = (x_val, y_val))
```

```
Epoch 1/15
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`,
  return dispatch_target(*args, **kwargs)
491/491 [==============================] - 118s 238ms/step - loss: 1.4012 - accuracy: 0.6474 - val_loss: 0.5312 - val_accuracy: 0.8517
Epoch 2/15
491/491 [==============================] - 99s 202ms/step - loss: 0.2855 - accuracy: 0.9263 - val_loss: 0.1968 - val_accuracy: 0.9485
Epoch 3/15
491/491 [==============================] - 100s 203ms/step - loss: 0.1508 - accuracy: 0.9605 - val_loss: 0.1477 - val_accuracy: 0.9626
Epoch 4/15
491/491 [==============================] - 99s 201ms/step - loss: 0.1178 - accuracy: 0.9694 - val_loss: 0.1491 - val_accuracy: 0.9626
Epoch 5/15
491/491 [==============================] - 99s 201ms/step - loss: 0.1099 - accuracy: 0.9734 - val_loss: 0.1111 - val_accuracy: 0.9717
Epoch 6/15
491/491 [==============================] - 99s 202ms/step - loss: 0.0774 - accuracy: 0.9799 - val_loss: 0.0992 - val_accuracy: 0.9777
Epoch 7/15
491/491 [==============================] - 99s 202ms/step - loss: 0.0718 - accuracy: 0.9813 - val_loss: 0.1007 - val_accuracy: 0.9758
Epoch 8/15
491/491 [==============================] - 100s 203ms/step - loss: 0.0633 - accuracy: 0.9837 - val_loss: 0.0776 - val_accuracy: 0.9823
Epoch 9/15
491/491 [==============================] - 100s 203ms/step - loss: 0.0685 - accuracy: 0.9834 - val_loss: 0.0717 - val_accuracy: 0.9811
Epoch 10/15
491/491 [==============================] - 99s 201ms/step - loss: 0.0537 - accuracy: 0.9866 - val_loss: 0.1350 - val_accuracy: 0.9691
Epoch 11/15
491/491 [==============================] - 99s 201ms/step - loss: 0.0647 - accuracy: 0.9835 - val_loss: 0.1570 - val_accuracy: 0.9657
Epoch 12/15
491/491 [==============================] - 99s 202ms/step - loss: 0.0487 - accuracy: 0.9875 - val_loss: 0.1034 - val_accuracy: 0.9762
Epoch 13/15
491/491 [==============================] - 98s 200ms/step - loss: 0.0675 - accuracy: 0.9840 - val_loss: 0.0571 - val_accuracy: 0.9875
Epoch 14/15
491/491 [==============================] - 98s 200ms/step - loss: 0.0390 - accuracy: 0.9902 - val_loss: 0.0651 - val_accuracy: 0.9871
Epoch 15/15
491/491 [==============================] - 99s 201ms/step - loss: 0.0587 - accuracy: 0.9864 - val_loss: 0.0947 - val_accuracy: 0.9792
```



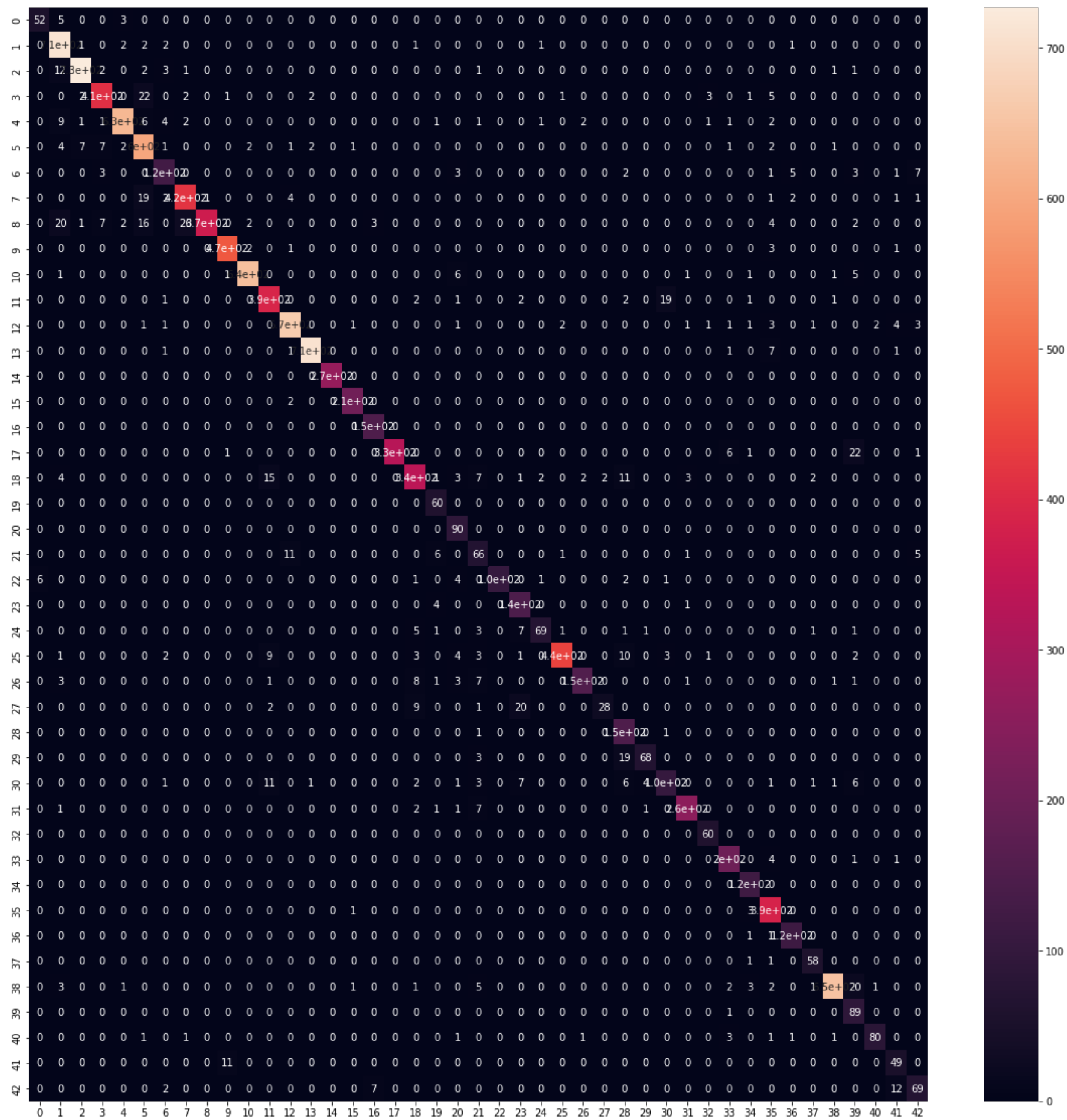



```
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)

print('test set accuracy: ', test_accuracy)
```

```
test set accuracy:  0.9370546340942383
```

The network finally provides a 98% accuracy on validation data, and 94% accuracy on testing data. And, the accuracy rises slowly after 6 epochs, which means the model has almost reach the local minimum, which could determine the parameter of the model.

Although having high percentage of outcome, there are still some error when I compare the predicted results by entering the test data.



1.6 DATA SOURCES

Explain whether or not your project requires data. If so, describe specifically were you will obtain it. If you intend to gather the data yourself, please explain how this will be practical. The *clarity*, especially this specificity, of your account is especially relevant.

I got my data source from Kaggle.com (https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign)

1.8 REFERENCES FOR PROPOSAL PHASE

Fill in the following, and cite each reference at least once (e.g., "[2]") within the text. References can include specific places in the notes and textbook. Note that "Use of Resources" is an entire evaluation criterion equal in weight to the others. This section is not hard to do, so it's easy to gain points (and the use of references always improves a paper in any case). On the other hand, if you don't complete this section, the grade could suffer up to 20%. There should be a minimum of three meaningful references.

https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign/code

I take a look what other people did on their project that have same topic with me. Through skim other people's work, I build a tentative plan about how I make my code.

Codes:

```
# -*- coding: utf-8 -*-
"""Project.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1pVpIsk3R0PXb8YejFyaN0s0f9lh36Zui
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
from PIL import ImageDraw
from tqdm import tqdm

import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
from sklearn.metrics import accuracy_score

from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array,
array_to_img, load_img

os.chdir('/content/drive/MyDrive/CS_767/Project/archive')

"""Data"""

file_list = os.listdir('/content/drive/MyDrive/CS_767/Project/archive')
file_list

df_Meta = pd.read_csv('/content/drive/MyDrive/CS_767/Project/archive/Meta.csv')
```

```
df_Meta

Meta_images = []
Meta_labels = []

plt.figure(figsize=(16,16))
for i in range(len(df_Meta)):
    img = load_img("/content/drive/MyDrive/CS_767/Project/archive/" + df_Meta['Path'][i])
    plt.subplot(1, 3, 1)
    plt.imshow(img)
    Meta_images.append(img)
    Meta_labels.append(df_Meta['ClassId'][i])

df_Train = pd.read_csv('/content/drive/MyDrive/CS_767/Project/archive/Train.csv')
df_Train

import seaborn as sns

plt.figure(figsize=(20,10))
ax = sns.countplot(x="Width", data=df_Train)

df_cutWidth = pd.cut(df_Train['Width'], np.arange(0,200,10)).value_counts(sort=False)

fig, ax = plt.subplots(figsize=(20,10))
ax.bar(range(len(df_cutWidth)),df_cutWidth.values)
ax.set_xticks(range(len(df_cutWidth)))
ax.set_xticklabels(df_cutWidth.index)
fig.show()

image_height = 33
image_width = 33
image_channel = 3

img_sample =
Image.open("/content/drive/MyDrive/CS_767/Project/archive/"+df_Train['Path'][0])

draw = ImageDraw.Draw(img_sample)
draw.rectangle([df_Train['Roi.X1'][0], df_Train['Roi.Y1'][0], df_Train['Roi.X2'][0],
df_Train['Roi.Y2'][0]], outline="red")
img_sample_resized = img_sample.resize((300,300))
img_sample_resized

img_sample_crop = img_sample.crop((df_Train['Roi.X1'][0], df_Train['Roi.Y1'][0],
df_Train['Roi.X2'][0], df_Train['Roi.Y2'][0]))

# Shows the image in image viewer
```

```
img_sample_crop_resized = img_sample_crop.resize((300,300))
img_sample_crop_resized

df_Test = pd.read_csv('./Test.csv')
df_Test

image_height = 33
image_width = 33
image_channel = 3

Train_images = []
Train_labels = []

for i in tqdm(range(len(df_Train))):
   img = load_img("/content/drive/MyDrive/CS_767/Project/archive/"+df_Train['Path'][i],
target_size = (image_height, image_width))
   img = img_to_array(img)
   Train_images.append(img)

image_height = 33
image_width = 33
image_channel = 3

Test_images = []
Test_labels = []

for i in tqdm(range(len(df_Test))):
   img = load_img("/content/drive/MyDrive/CS_767/Project/archive/"+df_Test['Path'][i],
target_size = (image_height, image_width))
   img = img_to_array(img)
   Test_images.append(img)

Train_labels = df_Train['ClassId'].values
Train_labels

Test_labels = df_Test['ClassId'].values
Test_labels

x_train, x_val, y_train, y_val = train_test_split(np.array(Train_images), np.array(Train_labels),
test_size=0.2)

x_test = np.array(Test_images)
y_test = np.array(Test_labels)

"""CNN Model"""
```

```
model = Sequential([
    Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(image_height,
image_width, image_channel)),
    Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2, 2)),
    Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2, 2)),
    Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    MaxPool2D(pool_size=(2, 2)),

    Flatten(),
    Dense(256, activation='relu'),
    Dense(43, activation='softmax')

])

model.summary()

model.compile(optimizer='adam',loss =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])

epoch = 15
history = model.fit(x_train, y_train, batch_size = 64, epochs=epoch, validation_data = (x_val,
y_val))

accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epoch)

plt.figure(figsize=(16, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, accuracy, label='Training Accuracy')
plt.plot(epochs_range, val_accuracy, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)

print('test set accuracy: ', test_accuracy)

test_prediction = np.argmax(model.predict(x_test), axis=-1)

plt.figure(figsize = (13, 13))

start_index = 0
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = test_prediction[start_index + i]
    actual = y_test[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(actual, prediction), color = col)
    plt.imshow(array_to_img(x_test[start_index + i]))
plt.show()

import seaborn as sns
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, test_prediction)
plt.figure(figsize = (20, 20))
sns.heatmap(cm, annot = True)
```