In [24]:

```
import os
import cv2
import numpy as np
import seaborn as sbn
import matplotlib.pyplot as plt
from tensorflow.keras import models
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
```

In [25]:

```
images = []
classNo = []
class_sayısı = 7
path = "Traffic_Data"

for i in range(class_sayısı):
    img_folders = os.listdir(path + "\\" + str(i))
    for j in img_folders:
        img = cv2.imread(path + "\\" + str(i) + "\\" + j)
        img = cv2.resize(img, (32,32))
        images.append(img)
        classNo.append(i)

len(images)
```

Out[25]:

```
10502
```

In [26]:

```
images = np.array(images)
labels = np.array(classNo)
new_images = []

def scaling_process(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img = cv2.equalizeHist(img)
    img = img / 255
    return img

for img in images:
    img = scaling_process(img)
    new_images.append(img)

new_images = np.array(new_images)
print(new_images.shape)
```

```
(10502, 32, 32)
```

In [27]:

```
x_train,x_test,y_train,y_test = train_test_split(new_images,labels,test_size=0.1)
x_train,x_validation,y_train,y_validation = train_test_split(x_train,y_train,test_size=0.2)

x_train = x_train.reshape(-1,32,32,1)
x_test = x_test.reshape(-1,32,32,1)
x_validation = x_validation.reshape(-1,32,32,1)

print(x_train.shape)
print(x_test.shape)
print(x_validation.shape)
```

```
(7560, 32, 32, 1)
(1051, 32, 32, 1)
(1891, 32, 32, 1)
```

In [28]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
dataGenerator = ImageDataGenerator(width_shift_range=0.1,
                                   height_shift_range=0.1,
                                   zoom_range=0.1,
                                   rotation_range=10)
dataGenerator.fit(x_train)
```

In [29]:

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train,class_sayısı)
y_test = to_categorical(y_test,class_sayısı)
y_validation = to_categorical(y_validation,class_sayısı)
```

In [30]:

```
model = models.Sequential()
girdi = (32,32,1)

model.add(layers.Conv2D(32,kernel_size=(5,5),input_shape=girdi,padding="same",activation="relu"
model.add(layers.Conv2D(64,kernel_size=(3,3),padding="same",activation="relu"))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Conv2D(32,kernel_size=(5,5),padding="same",activation="relu"))
model.add(layers.Conv2D(32,kernel_size=(3,3),padding="same",activation="relu"))
model.add(layers.MaxPooling2D(pool_size=(2,2)))
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())

model.add(layers.Dense(256, activation="relu"))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(class_sayısı, activation="softmax"))

model.compile(optimizer="adam",loss="categorical_crossentropy",metrics=["accuracy"])
```

In [31]:

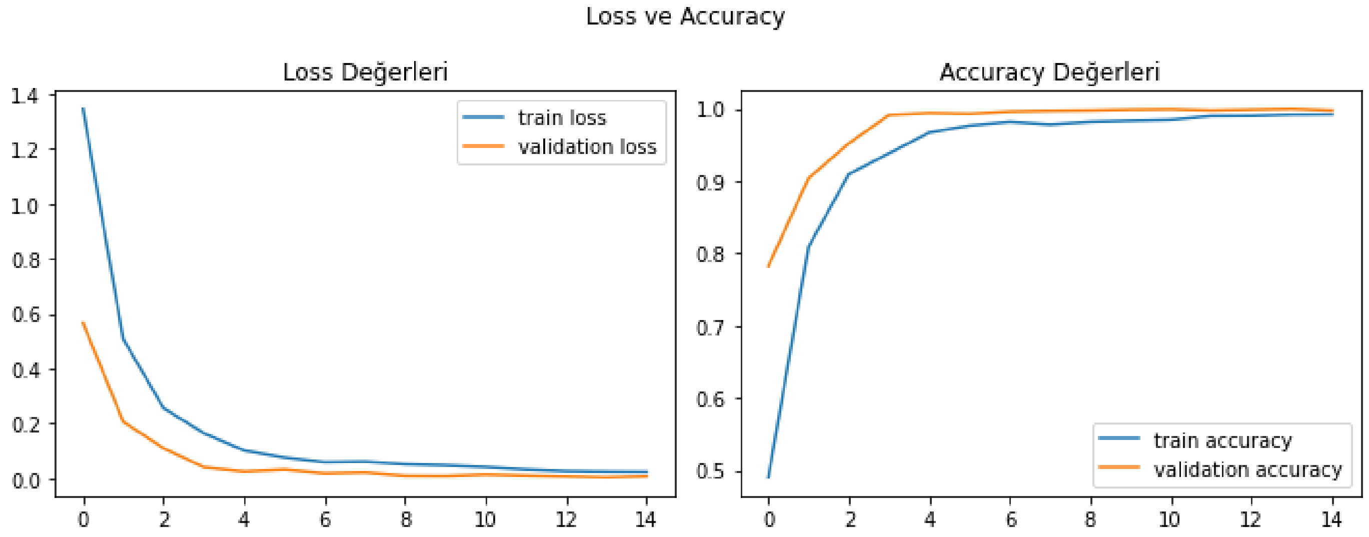
```
batch = 40
generator = dataGenerator.flow(x_train,y_train,batch_size=batch)
steps = 80

history = model.fit_generator(generator, epochs=15,
                              validation_data = (x_validation,y_validation),
                              steps_per_epoch = steps, shuffle=1)
```

```
Epoch 1/15
80/80 [==============================] - 15s 187ms/step - loss: 1.3446 - accuracy: 0.4903 - val
_loss: 0.5650 - val_accuracy: 0.7821
Epoch 2/15
80/80 [==============================] - 15s 183ms/step - loss: 0.5068 - accuracy: 0.8091 - val
_loss: 0.2057 - val_accuracy: 0.9038
Epoch 3/15
80/80 [==============================] - 15s 183ms/step - loss: 0.2554 - accuracy: 0.9094 - val
_loss: 0.1093 - val_accuracy: 0.9519
Epoch 4/15
80/80 [==============================] - 15s 185ms/step - loss: 0.1640 - accuracy: 0.9381 - val
_loss: 0.0405 - val_accuracy: 0.9910
Epoch 5/15
80/80 [==============================] - 15s 183ms/step - loss: 0.1013 - accuracy: 0.9669 - val
_loss: 0.0247 - val_accuracy: 0.9937
Epoch 6/15
80/80 [==============================] - 15s 184ms/step - loss: 0.0757 - accuracy: 0.9762 - val
_loss: 0.0320 - val_accuracy: 0.9926
Epoch 7/15
80/80 [==============================] - 15s 186ms/step - loss: 0.0586 - accuracy: 0.9816 - val
_loss: 0.0181 - val_accuracy: 0.9958
Epoch 8/15
80/80 [==============================] - 15s 187ms/step - loss: 0.0605 - accuracy: 0.9778 - val
_loss: 0.0216 - val_accuracy: 0.9968
Epoch 9/15
80/80 [==============================] - 15s 188ms/step - loss: 0.0513 - accuracy: 0.9816 - val
_loss: 0.0093 - val_accuracy: 0.9974
Epoch 10/15
```

```
80/80 [==============================] - 15s 187ms/step - loss: 0.0472 - accuracy: 0.9831 - val
_loss: 0.0084 - val_accuracy: 0.9984
Epoch 11/15
80/80 [==============================] - 15s 187ms/step - loss: 0.0409 - accuracy: 0.9844 - val
_loss: 0.0139 - val_accuracy: 0.9989
Epoch 12/15
80/80 [==============================] - 15s 188ms/step - loss: 0.0321 - accuracy: 0.9897 - val
_loss: 0.0101 - val_accuracy: 0.9974
Epoch 13/15
80/80 [==============================] - 15s 188ms/step - loss: 0.0259 - accuracy: 0.9900 - val
_loss: 0.0070 - val_accuracy: 0.9984
Epoch 14/15
80/80 [==============================] - 15s 188ms/step - loss: 0.0240 - accuracy: 0.9916 - val
_loss: 0.0035 - val_accuracy: 0.9995
Epoch 15/15
80/80 [==============================] - 15s 188ms/step - loss: 0.0232 - accuracy: 0.9919 - val
_loss: 0.0075 - val_accuracy: 0.9974
```

In [32]:

```
fig,axes = plt.subplots(1,2, figsize=(10,4))
fig.suptitle("Loss ve Accuracy")
axes[0].plot(history.history["loss"], label="train loss")
axes[0].plot(history.history["val_loss"], label="validation loss")
axes[0].set_title("Loss Değerleri")
axes[0].legend()
axes[1].plot(history.history["accuracy"], label="train accuracy")
axes[1].plot(history.history["val_accuracy"], label="validation accuracy")
axes[1].set_title("Accuracy Değerleri")
axes[1].legend()
plt.tight_layout()
plt.show()
```



In [33]:

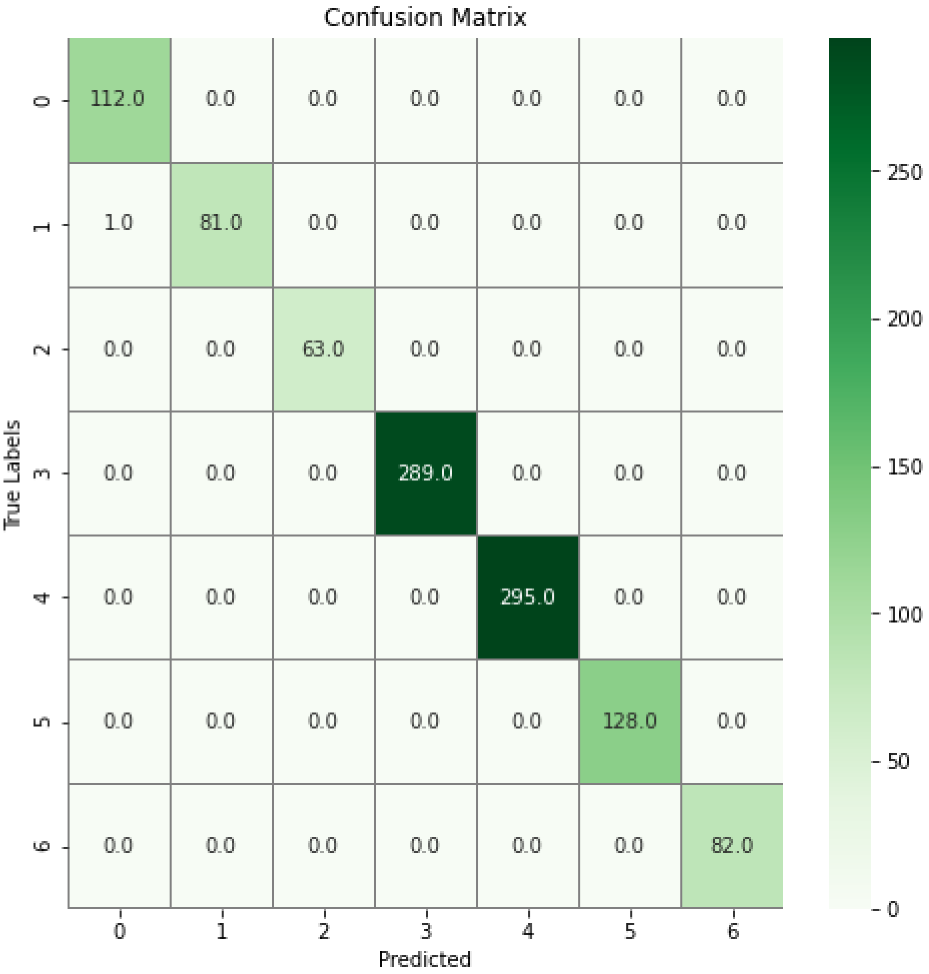
```
score_train = model.evaluate(x_train,y_train)
print("Eğitim Doğruluğu: %",score_train[1]*100)
score_test = model.evaluate(x_test,y_test)
print("Test Doğruluğu: %",score_test[1]*100)
```

```
237/237 [==============================] - 6s 26ms/step - loss: 0.0028 - accuracy: 0.9989
Eğitim Doğruluğu: % 99.89417791366577
33/33 [==============================] - 1s 25ms/step - loss: 0.0034 - accuracy: 0.9990
Test Doğruluğu: % 99.90485310554504
```

In [34]:

```
from sklearn.metrics import confusion_matrix

y_predict = model.predict(x_test)
y_predict_class = np.argmax(y_predict, axis = 1)
Y_true = np.argmax(y_test, axis = 1)

cm = confusion_matrix(Y_true, y_predict_class)
```

```
fig, axes = plt.subplots(figsize=(8,8))
sbn.heatmap(cm, annot = True, linewidths = 0.01, cmap = "Greens",
linecolor = "gray", fmt = ".1f", ax=axes)
plt.xlabel("Predicted")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.show()
```



In [36]:

```
import pickle

pickle_out = open("model_trained.p","wb")
pickle.dump = (model, pickle_out)
pickle_out.close()
```

In [ ]: