In [2]:

```
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 [3]:

```
images = []
classNo = []
class_sayısı = 10
path = "myData"

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)   #toplam veri sayısı
```

Out[3]:

```
10160
```

In [4]:

```
# her bir rakamdan kaç tane resim verisi var?
print(classNo.count(0))
print(classNo.count(1))
print(classNo.count(2))
```

```
1016
1016
1016
```

In [5]:

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

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)
```

```
(10160, 32, 32, 3)
(10160, 32, 32)
```

In [6]:

```
#veriyi validasyonlu bir şekilde ayırma
```

```
x_train,x_test,y_train,y_test = train_test_split(new_images,labels,test_size=0.4)
x_train,x_validation,y_train,y_validation = train_test_split(x_train,y_train,test_size=0.2)

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

```
(10160, 32, 32, 3)
(4876, 32, 32)
(4064, 32, 32)
(1220, 32, 32)
```

In [7]:

```
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)
```

```
(4876, 32, 32, 1)
(4064, 32, 32, 1)
(1220, 32, 32, 1)
```

In [8]:

```
#evrişimsel sinir ağı resim içindeki nesne konumuna duyarlıdır
#x_train verisini zoom, rotasyon gibi değişiklik yaparak çeşitlendir
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 [9]:

```
#farklı bir şekilde OneHotEncoder işlemi
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 [10]:

```
model = models.Sequential()

girdi = (32,32,1)

# "same" padding --> 1 sıra padding
model.add(layers.Conv2D(8,kernel_size=(5,5),input_shape=girdi,padding="same",activation="relu")
model.add(layers.MaxPooling2D(pool_size=(2,2)))

model.add(layers.Conv2D(16,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())

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

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

In [11]:

```
batch_size = 100
generator = dataGenerator.flow(x_train,y_train,batch_size=batch_size)
steps = x_train.shape[0] // batch_size
# 4876 // 100 --> steps = 48    (kalansız bölüm)
# shuffle --> veriyi karıştırır

history = model.fit_generator(generator, epochs = 20,
                              validation_data = (x_validation,y_validation),
                              steps_per_epoch = steps, shuffle = 1)
```

```
WARNING:tensorflow:From <ipython-input-11-69b235c4dc93>:7: Model.fit_generator (from tensorflo
w.python.keras.engine.training) is deprecated and will be removed in a future version.
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/20
48/48 [==============================] - 3s 60ms/step - loss: 1.8538 - accuracy: 0.3725 - val_l
oss: 0.6826 - val_accuracy: 0.8459
Epoch 2/20
48/48 [==============================] - 3s 56ms/step - loss: 0.8154 - accuracy: 0.7431 - val_l
oss: 0.2420 - val_accuracy: 0.9279
Epoch 3/20
48/48 [==============================] - 3s 55ms/step - loss: 0.5152 - accuracy: 0.8405 - val_l
oss: 0.1629 - val_accuracy: 0.9426
Epoch 4/20
48/48 [==============================] - 3s 57ms/step - loss: 0.3815 - accuracy: 0.8882 - val_l
oss: 0.1203 - val_accuracy: 0.9639
Epoch 5/20
48/48 [==============================] - 3s 57ms/step - loss: 0.3213 - accuracy: 0.8999 - val_l
oss: 0.1133 - val_accuracy: 0.9664
Epoch 6/20
48/48 [==============================] - 3s 57ms/step - loss: 0.2655 - accuracy: 0.9175 - val_l
oss: 0.0871 - val_accuracy: 0.9697
Epoch 7/20
48/48 [==============================] - 3s 59ms/step - loss: 0.2165 - accuracy: 0.9343 - val_l
oss: 0.0783 - val_accuracy: 0.9754
Epoch 8/20
48/48 [==============================] - 3s 55ms/step - loss: 0.1986 - accuracy: 0.9355 - val_l
oss: 0.0651 - val_accuracy: 0.9770
Epoch 9/20
48/48 [==============================] - 3s 56ms/step - loss: 0.1800 - accuracy: 0.9470 - val_l
oss: 0.0634 - val_accuracy: 0.9746
Epoch 10/20
48/48 [==============================] - 3s 59ms/step - loss: 0.1730 - accuracy: 0.9462 - val_l
oss: 0.0597 - val_accuracy: 0.9746
Epoch 11/20
48/48 [==============================] - 3s 57ms/step - loss: 0.1513 - accuracy: 0.9518 - val_l
oss: 0.0546 - val_accuracy: 0.9803
Epoch 12/20
48/48 [==============================] - 3s 55ms/step - loss: 0.1490 - accuracy: 0.9560 - val_l
oss: 0.0534 - val_accuracy: 0.9811
Epoch 13/20
48/48 [==============================] - 3s 57ms/step - loss: 0.1352 - accuracy: 0.9588 - val_l
oss: 0.0505 - val_accuracy: 0.9803
Epoch 14/20
48/48 [==============================] - 3s 57ms/step - loss: 0.1224 - accuracy: 0.9611 - val_l
oss: 0.0426 - val_accuracy: 0.9861
Epoch 15/20
48/48 [==============================] - 3s 61ms/step - loss: 0.1115 - accuracy: 0.9629 - val_l
oss: 0.0427 - val_accuracy: 0.9861
Epoch 16/20
48/48 [==============================] - 3s 58ms/step - loss: 0.1112 - accuracy: 0.9650 - val_l
oss: 0.0409 - val_accuracy: 0.9852
Epoch 17/20
48/48 [==============================] - 3s 56ms/step - loss: 0.1065 - accuracy: 0.9675 - val_l
oss: 0.0403 - val_accuracy: 0.9869
Epoch 18/20
48/48 [==============================] - 3s 57ms/step - loss: 0.1192 - accuracy: 0.9644 - val_l
oss: 0.0394 - val_accuracy: 0.9877
Epoch 19/20
48/48 [==============================] - 3s 56ms/step - loss: 0.1032 - accuracy: 0.9650 - val_l
oss: 0.0379 - val_accuracy: 0.9885
```

```
Epoch 20/20
48/48 [==============================] - 3s 56ms/step - loss: 0.0876 - accuracy: 0.9742 - val_l
oss: 0.0338 - val_accuracy: 0.9893
```

In [12]:

```
history.history.keys()
```

Out[12]:

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

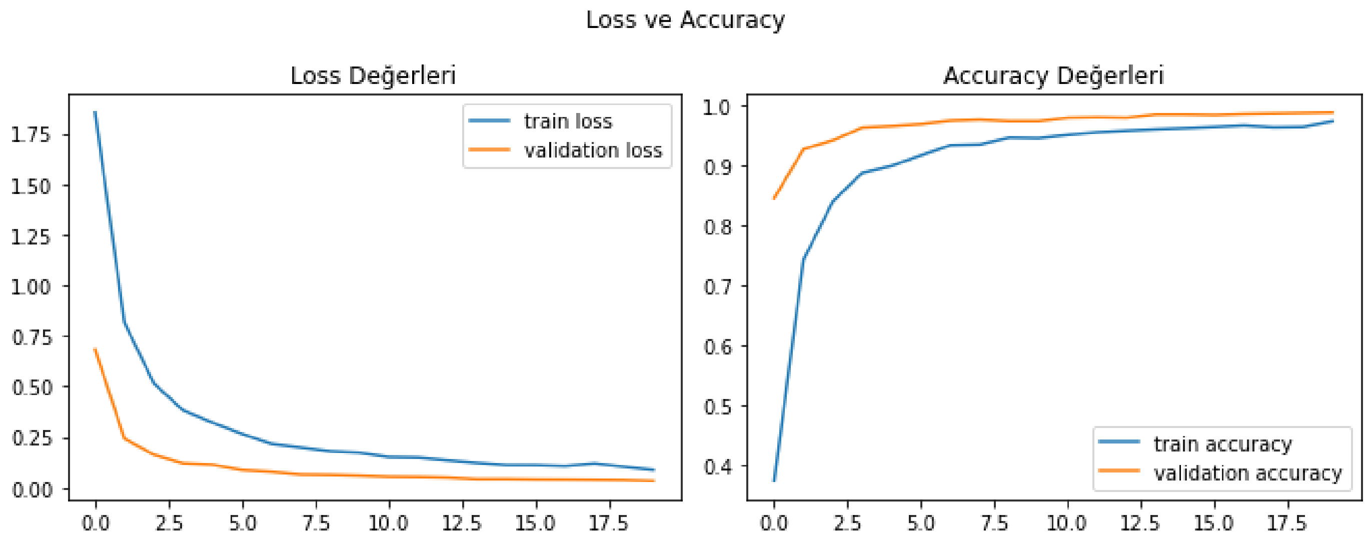
In [13]:

```
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 [14]:

```
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)
```

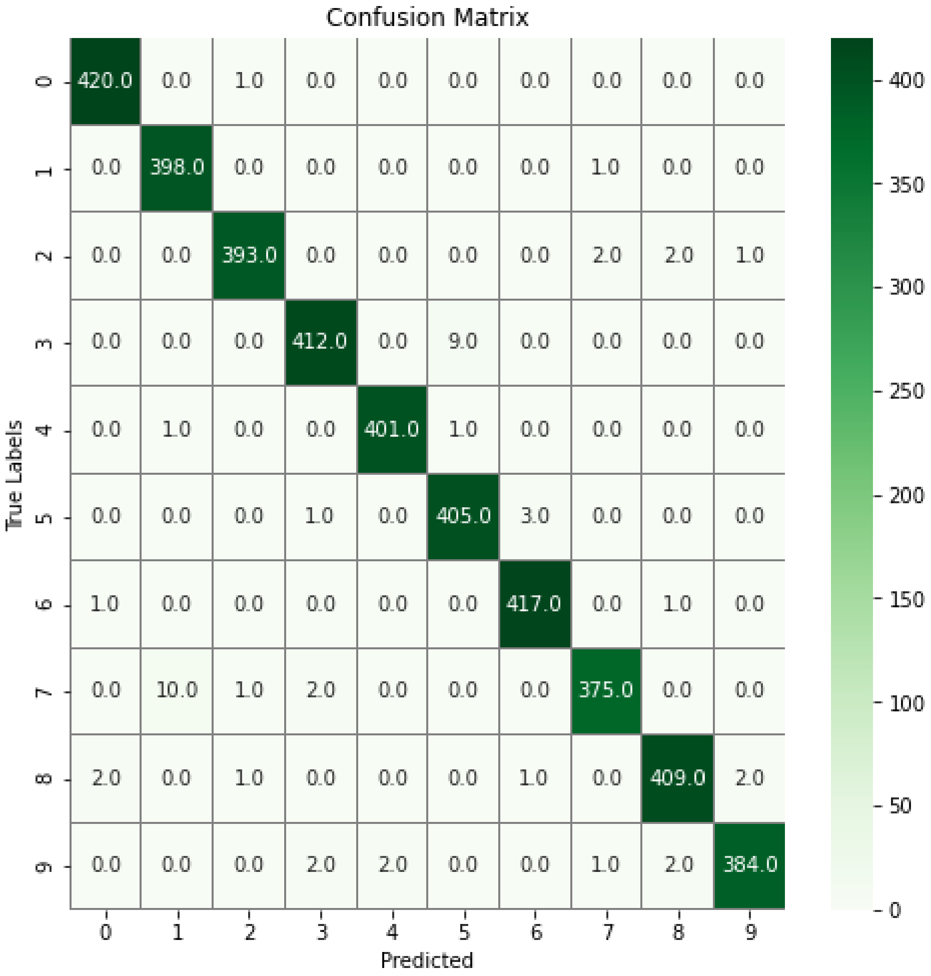
```
153/153 [==============================] - 1s 6ms/step - loss: 0.0248 - accuracy: 0.9928
Eğitim Doğruluğu: % 99.2821991443634
127/127 [==============================] - 1s 6ms/step - loss: 0.0352 - accuracy: 0.9877
Test Doğruluğu: % 98.76968264579773
```

In [15]:

```
# Resim Tahminleri Doğruluk Skalası
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()
```



	0	1	2	3	4	5	6	7	8	9
0	420.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	398.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0
2	0.0	0.0	393.0	0.0	0.0	0.0	0.0	2.0	2.0	1.0
3	0.0	0.0	0.0	412.0	0.0	9.0	0.0	0.0	0.0	0.0
4	0.0	1.0	0.0	0.0	401.0	1.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	1.0	0.0	405.0	3.0	0.0	0.0	0.0
6	1.0	0.0	0.0	0.0	0.0	0.0	417.0	0.0	1.0	0.0
7	0.0	10.0	1.0	2.0	0.0	0.0	0.0	375.0	0.0	0.0
8	2.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	409.0	2.0
9	0.0	0.0	0.0	2.0	2.0	0.0	0.0	1.0	2.0	384.0

In [16]:

```
open("Rakamlar_model.json","w").write(model.to_json())
model.save("Rakamlar_model.h5")
model.save_weights("Rakamlar_weights.h5")
```

In [ ]:

```

```