SAKARYA
ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Derin Öğrenme ve Evrişimli Sinir Ağları Dersi

Proje Ödevi

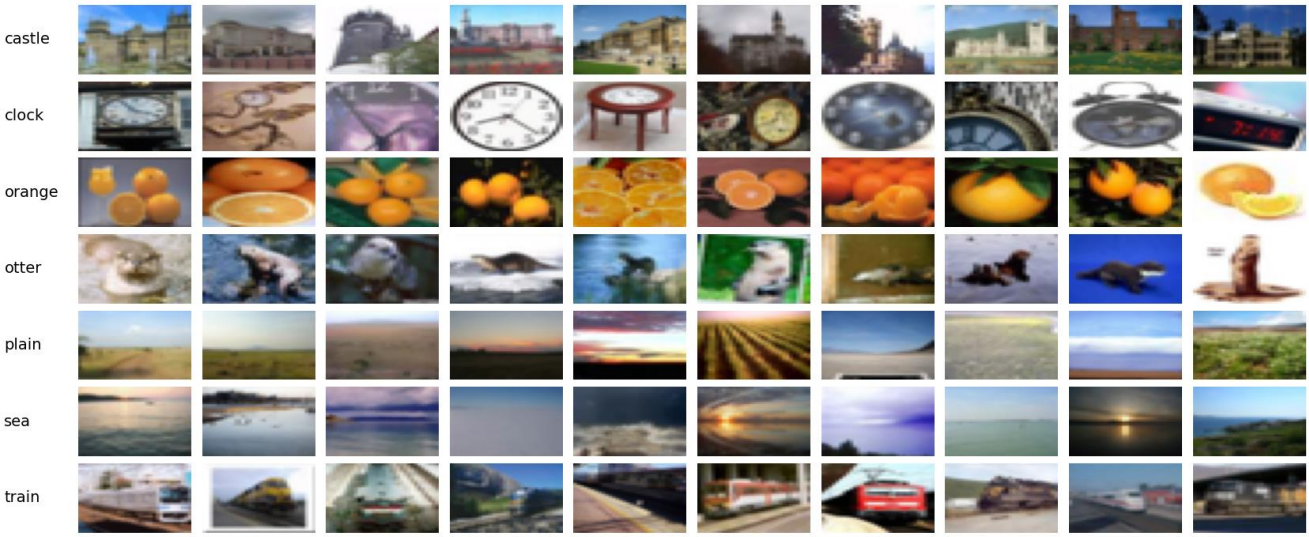
B201210024 – Ahmet Furkan SÖĞÜTCÜ

1/A Grubu

Dr. Öğr. Üyesi Ulaş YURTSEVER

1.Adım

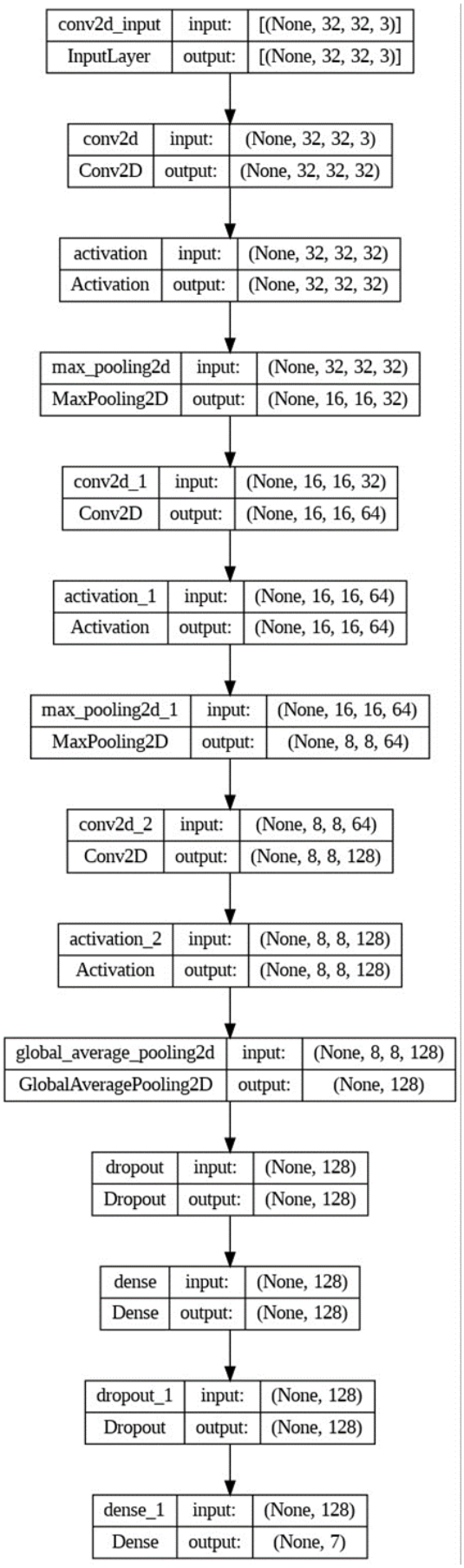
Bu ilk aşamada, bize verilen sınıfların içeriğindeki görüntülerin bir kısmının gösterilmesi istendi. Bu işlem için de en son kısımda verilecek olan kodlar içinde 1.Adım kodunda yazılanlar ile örnek görüntüler listelendi. Aşağıda örneklerin görüntüsü bulunmakta.



Resim1. – Veri Setinden Örnekler

2.Adım

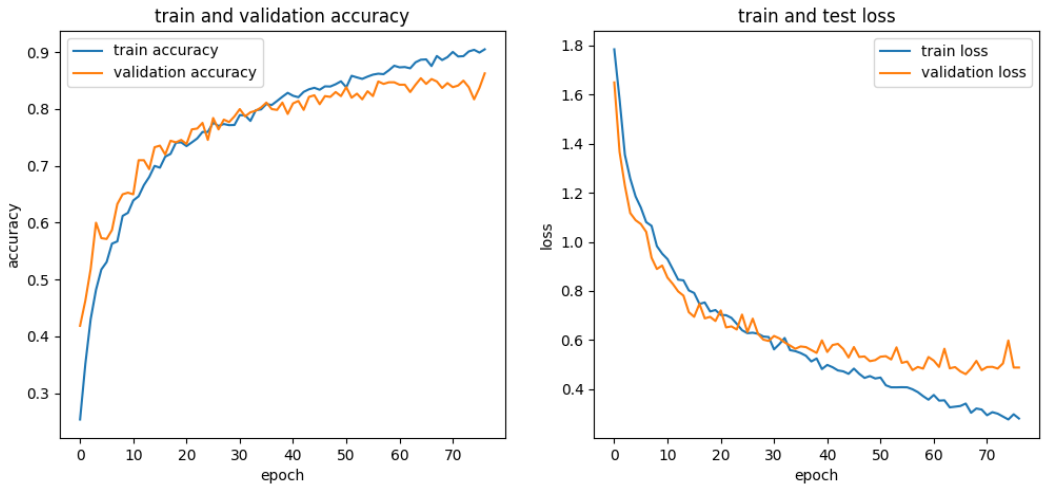
2. aşamada ise model tasarımının yapılması istenilmişti. Öncelikle model katmanlarında bulunacaklar belirlendi ve ardından bu modelin blok şeması çizildi. Blok şeması bir sonraki sayfada görüldüğü gibidir.



Resim2.- Blok Şeması

3.Adım

Bu adımda ise tasarlanan modelin eğitim aşaması bulunuyor. Eğitilecek modeli uygun epoch sayısı ile eğitilip istenilen grafikler programda çizdirilecekti. Bu grafikler doğruluk ve kayıp grafiklerinin, eğitim ve geçerleme verilerini içerecek şekilde olacaktı. Model eğitiminde bir de validation için test veri seti kullanılacaktı. Aşağıda çizdirilen grafikler mevcut.

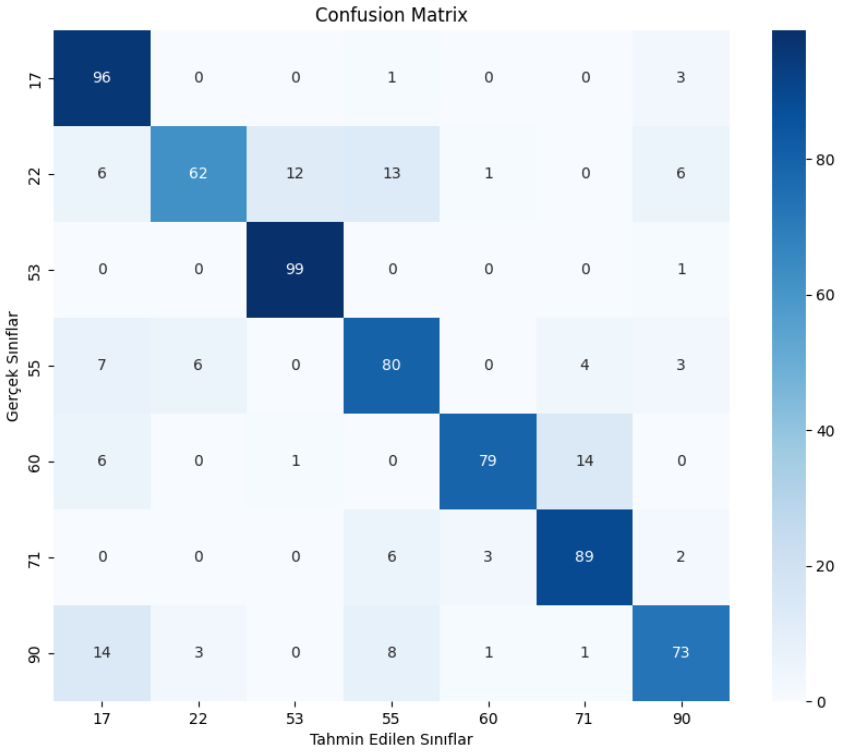


Resim3. – test ve doğruluk grafikleri

Model eğitimini yaparken epoch sayısı 200 alınmıştı. Ama grafiklerde de görüldüğü üzere yaklaşık 77 civarında sonlanmış eğitim. Bunun sebebi aşırı uyumu önlemek adına early_stopping kullanılmasından kaynaklanıyor. Train ve validation grafiklerinin birbirinden uzaklaşması da bir nevi aşırı uyumun göstergesidir. Bu şekilde de en uygun eğitim yapılmış oldu.

4.Adım

Bu adımda da model eğitildikten sonra test verisi ile test edilecekti ve ardından karmaşıklık matrisinin elde edilmesi gerekiyordu. En son kodlarda da belirtildiği gibi sonuçlar elde edildi. Aşağıda da karmaşıklık matrisinin görüntüsü mevcuttur.



Resim4. – Karmaşıklık(Confusion) Matrisi

5.Adım

Bu son aşamada ise predict fonksiyonu ile her sınıftan bir görüntü belirlenip onlar kullanılarak modelin test işlemi yapıldı. Ekran çıktıları aşağıdaki gibidir.

Resim5. – Örnek Test

KODLAR

1.Adım Kodu

```python
from keras.datasets import cifar100
import numpy as np
import matplotlib.pyplot as plt

(train_images, train_labels), (test_images, test_labels) = cifar100.load_data(label_mode='fine')

class_names = [
    'castle',
    'clock',
    'orange',
    'otter',
    'plain',
    'sea',
    'train'
]

classes = [17, 22, 53, 55, 60, 71, 90]

fig, axes = plt.subplots(len(classes), 10, figsize=(20, 16))

plt.subplots_adjust(left=0.07   , hspace=0.1, wspace=0.1)

for i, cls in enumerate(classes):
    indices = np.where(train_labels.flatten() == cls)[0]
    random_indices = np.random.choice(indices, 10, replace=False)

    y_position = axes[i, 0].get_position().y0 + axes[i, 0].get_position().height / 2

    fig.text(0.02, y_position, f'{class_names[i]}', fontsize=14, verticalalignment='center', transform=fig.transFigure)

    for j, idx in enumerate(random_indices):
        ax = axes[i, j]
        ax.imshow(train_images[idx], aspect='auto')
        ax.axis('off')

plt.show()
```

Resim6. – 1.Adım Kodu

2.Adım Kodu

```python
from keras.datasets import cifar100
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Activation, GlobalAveragePooling2D
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
import numpy as np

(x_train, y_train), (x_test, y_test) = cifar100.load_data(label_mode='fine')

classes = [17, 22, 53, 55, 60, 71, 90]
class_map = {k: i for i, k in enumerate(classes)}

y_train_mapped = np.array([class_map[y[0]] for y in y_train if y[0] in classes])
y_test_mapped = np.array([class_map[y[0]] for y in y_test if y[0] in classes])

x_train_filtered = np.array([x for i, x in enumerate(x_train) if y_train[i][0] in classes])
x_test_filtered = np.array([x for i, x in enumerate(x_test) if y_test[i][0] in classes])

x_train_last = x_train_filtered.astype('float32') / 255
x_test_last = x_test_filtered.astype('float32') / 255

y_train_categorical = to_categorical(y_train_mapped, num_classes=len(classes))
y_test_categorical = to_categorical(y_test_mapped, num_classes=len(classes))
```

Resim7. – 2.Adım Kodu

```python
model = Sequential([
    Conv2D(32, (3, 3), padding='same', input_shape=(32, 32, 3)),
    Activation('relu'),
    MaxPooling2D((2, 2)),

    Conv2D(64, (3, 3), padding='same'),
    Activation('relu'),
    MaxPooling2D((2, 2)),

    Conv2D(128, (3, 3), padding='same'),
    Activation('relu'),
    GlobalAveragePooling2D(),

    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(classes), activation='softmax')
])

early_stopping = EarlyStopping(monitor='val_loss', patience=10)

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

# Modeli eğit
history = model.fit(x_train_last, y_train_categorical, epochs=200, validation_data=(x_test_last, y_test_categorical), batch_size=50,callbacks=
[early_stopping])

# Modeli değerlendir
loss, accuracy = model.evaluate(x_test_last, y_test_categorical)
print(f'Test Doğruluğu: {accuracy*100:.2f}%')
```

Resim8. – 2.Adım Kodu Devamı

3.Adım Kodu

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10)

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

# Modeli eğit
history = model.fit(x_train_last, y_train_categorical, epochs=200, validation_data=(x_test_last, y_test_categorical), batch_size=50,callbacks=
[early_stopping])

# Modeli değerlendir
loss, accuracy = model.evaluate(x_test_last, y_test_categorical)
print(f'Test Doğruluğu: {accuracy*100:.2f}%')

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='train accuracy')
plt.plot(history.history['val_accuracy'], label='validation accuracy')
plt.title('train and validation accuracy')
plt.xlabel('epoch')
plt.ylabel('accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='train loss')
plt.plot(history.history['val_loss'], label='validation loss')
plt.title('train and test loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend()

plt.show()
```

Resim9. – 3.Adım Kodu

4.Adım Kodu

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10)

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

history = model.fit(x_train_filtered, y_train_categorical, epochs=100, validation_data=(x_test_filtered, y_test_categorical), batch_size=50,
callbacks=[early_stopping])

loss, accuracy = model.evaluate(x_test_filtered, y_test_categorical)
print(f'Test Doğruluğu: {accuracy*100:.2f}%')

predictions = model.predict(x_test_filtered)
predicted_classes = np.argmax(predictions, axis=1)

true_classes = y_test_mapped

cm = confusion_matrix(true_classes, predicted_classes)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
plt.title('Confusion Matrix')
plt.ylabel('Gerçek Sınıflar')
plt.xlabel('Tahmin Edilen Sınıflar')
plt.show()
```

Resim10. – 4.Adım Kodu

5.Adım Kodu

```
early_stopping = EarlyStopping(monitor='val_loss', patience=10)

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

# Modeli eğit
history = model.fit(x_train_last, y_train_categorical, epochs=100, validation_data=(x_test_last, y_test_categorical), batch_size=50,callbacks=
[early_stopping])

# Modeli değerlendir
loss, accuracy = model.evaluate(x_test_last, y_test_categorical)
print(f'Test Doğruluğu: {accuracy*100:.2f}%')

unique_classes = np.unique(y_test_mapped)
indices_per_class = {class_id: np.where(y_test_mapped == class_id)[0][5] for class_id in unique_classes}

selected_images = np.array([x_test_filtered[indices_per_class[class_id]] for class_id in unique_classes])
selected_labels = np.array([y_test_mapped[indices_per_class[class_id]] for class_id in unique_classes])

predictions = model.predict(selected_images)

fig, axes = plt.subplots(1, len(unique_classes), figsize=(15, 2))
for i, ax in enumerate(axes):
    ax.imshow(selected_images[i].astype('uint8'))
    ax.title.set_text(f'Class: {classes[unique_classes[i]]}\nPred: {np.argmax(predictions[i])}')
    ax.axis('off')

plt.show()

for i, prediction in enumerate(predictions):
    print(f"{classes[unique_classes[i]]}. Sınıf İçin Çıkış Vektörleri:\n{prediction}\n")
```

Resim11. – 5.Adım Kodu