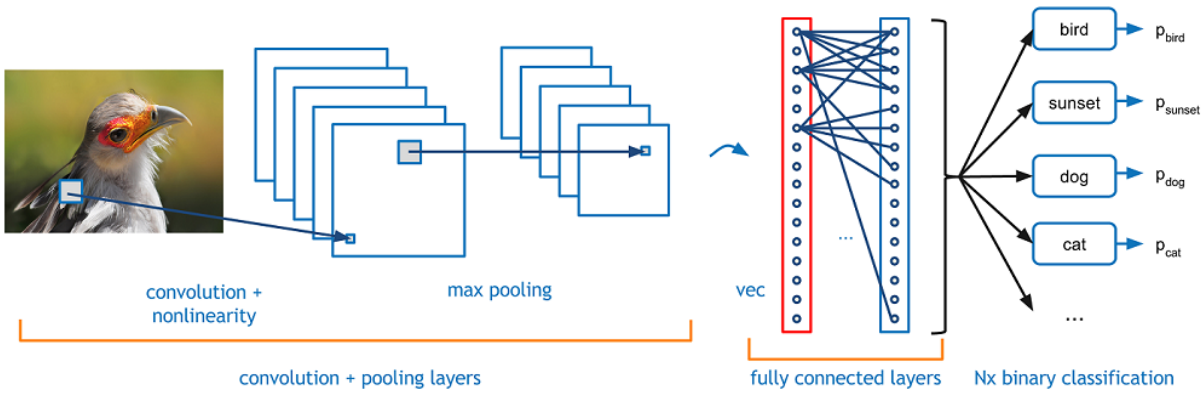
Uygulama 5. 1-Boyutlu Evrişimsel Sinir Ağı (CNN) ile Zaman Serisi Verilerinin Analizi

Evrişimli sinir ağları, temel olarak görüntüleri sınıflandırmak (örneğin gördüklerini isimlendirmek), benzerlikle kümelemek (fotoğraf arama) ve sahnelerde nesne tanıma yapmak için kullanılan derin yapay sinir ağlarıdır. Evrişimsel sinir ağları yapısı gereği giriş olarak resim ya da videolar alır. Tabi ki resimleri alırken ilgili formata çevrilmiş olması gerekir. Örneğin bir konvolüsyonel sinir ağına bir resim veriyorsak bunu matris formatında vermemiz gerekiyor. Bu uygulamada bu durumların aksine 1 boyutlu veriler olan zaman serilerine yönelik bir çalışma gerçekleştirilecektir.

ESA Yapısı



Şekil 1. ESA mimarisinin yapısı

1D-ESA Örnek Uygulaması (Yaprak Sınıflandırma)

Öncelikle gerekli kütüphaneler yüklenir.

```
#Yaprak sınıfandırması (1DESA)

#Gerekli kütüphanelerin yüklenmesi
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
```

Ardından sisteme veriler yüklenir

```
#Test ve eğitim verilerinin okunması
train = pd.read_csv("C:/Users/burak/Masaüstü/yaprak sınıflandırma/train.csv")
test = pd.read_csv("C:/Users/burak/Masaüstü/yaprak sınıflandırma/test.csv")
```

Daha sonra sınıfların etiketleri belirlenir.

```
#Sınıfların belirlenmesi ve etiketlenmesi
label_encoder = LabelEncoder().fit(train.species)
labels = label_encoder.transform(train.species)
classes = list(label_encoder.classes_)
```

Eğitim ve test verisi uygun bir şekilde ayarlanır.

```
#verilerin hazırlanması, özellik ve sınıf sayısının belirlenmesi
train = train.drop(["id", "species"],axis=1)
test = test.drop(["id"], axis=1)
nb_features = 192
nb_classes = len(classes)

#Standardization of train data
```

Ardından eğitim verisindeki veriler standartlaştırılır.

```
#Eğitim verisinindeki verilerin standartlaştırılması
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(train.values)
train = scaler.transform(train.values)
```

Daha sonra eğitim ve doğrulama verileri belirlenir.

```
#Eğitim verisinin eğitim ve doğrulama için ayarlanması
from sklearn.model_selection import train_test_split
X_train, X_valid, y_train, y_valid = train_test_split(train, labels, test_size=0.1)
```

Ardından etiketler kategorileştirilir.

```
#Etiketlerin kategorilerinin belirlenmesi
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_valid = to_categorical(y_valid)

#resize the input data
```

Daha sonra veriler 1DESA yapısına uygun olacak şekilde yeniden boyutlandırılır.

```
#Giriş verilerinin boyutlarının ayarlanması
X_train = np.array(X_train).reshape(891,192,1)
X_valid = np.array(X_valid).reshape(99,192,1)
```

Ardından ağın modeli oluşturulur.

```
#1DESA modelinin oluşturulması
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Conv1D, Dropout, MaxPooling1D, Flatten

model = Sequential()
model.add(Conv1D(512,1,input_shape=(nb_features,1)))
model.add(Activation("relu"))
model.add(MaxPooling1D(2))
model.add(Conv1D(256,1))
model.add(MaxPooling1D(2))
model.add(Dropout(0.25))
model.add((Flatten()))
model.add(Dense(2048, activation="relu"))
model.add(Dense(1024, activation="relu"))
model.add(Dense(nb_classes, activation="softmax"))
model.summary()
```

Model oluşturma işleminin ardından, model derlenir ve eğitilir.

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

#Modelin eğitilmesi
model.fit(X_train, y_train, epochs = 15, validation_data=(X_valid,y_valid))
```

Ardından ortalama değerler belirlenir.

```
#Ortalama değerlerin gösterilmesi
print(("Ortalama Eğitim Kaybı: ", np.mean(model.history.history["loss"])))
print(("Ortalama Eğitim Başarımı: ", np.mean(model.history.history["accuracy"])))
print(("Ortalama Doğrulama Kaybı: ", np.mean(model.history.history["val_loss"])))
print(("Ortalama Doğrulama Başarımı: ", np.mean(model.history.history["val_accuracy"])))
```

Son olarak sonuçlar grafikler üzerinde gösterilir.

```
#Değerlerin grafik üzerinde gösterilmesi
import matplotlib.pyplot as plt
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(15, 15))
ax1.plot(model.history.history['loss'], color='g', label="Eğitim kaybı")
ax1.plot(model.history.history['val_loss'], color='y', label="Doğrulama kaybı")
ax1.set_xticks(np.arange(20, 100, 20))
ax2.plot(model.history.history['accuracy'], color='b', label="Eğitim başarımı")
ax2.plot(model.history.history['val_accuracy'], color='r',label="Doğrulama başarımı")
ax2.set_xticks(np.arange(20, 100, 20))
plt.legend()
plt.show()
```

Uygulama 5. Değerlendirme Soruları

1) Verilen kodu inceleyerek kendi modelinizi oluşturunuz **(10p).**

2) Verilen kodu inceleyerek çapraz doğrulama işlemi yapınız ve buna göre başarımı değerlendiriniz **(30p).**

3) Verilen kodu inceleyerek F1-skor, kesinlik (precision), duyarlılık (sensitivity) ve özgüllük (specificity) değerlerini bulunuz **(20p)**.

4) Diyabet verisetini kullanarak kendi 1DESA modelinizi geliştiriniz. Geliştirmiş olduğunuz modelin doğrulama (accuracy), F1-skor ve kayıp değerlerini bulunuz. Bulduğunuz bu değerleri grafik üzerinde gösteriniz **(30p)**.

5) Verilen kodu inceleyerek farklı optimizasyon algoritmalarını kullanınız. Kullandığınız bu algoritmalar içerisinde en iyi optimizasyon algoritmasını belirleyiniz **(10p).**