# CNN Uygulaması

YAPAY ZEKA VE DERİN ÖĞRENME
2021 BAHAR

# CNN uygulaması

- Bu derin öğrenme uygulama örneğinde; sınıflandırma için temel bir derin öğrenme (CNN) ağının MATLAB ortamında nasıl oluşturulması gerektiği açıklanacaktır.

- Bu örnekte; derin öğrenme sınıflandırması için basit bir evrişimsel sinir ağının (CNN'in) nasıl oluşturulduğu ve eğitildiği gösterilecektir.

- CNN'ler derin öğrenme için temel araçlardır ve özellikle görüntü tanıma uygulamaları için uygulanabilmektedir.

# CNN uygulama adımları

- Bu uygulama örneği aşağıdaki adımların nasıl gerçekleştirdiğini göstermektedir.
- 1) Load and explore image data

(görüntü verilerinin yüklenmesi)

- 2) Define the network architecture

(ağ mimarisinin tanımlanması)

- 3) Specify training options

(eğitim ayarlarının yapılması)

- 4) Train the network

(ağın eğitilmesi)

- 5) Predict the labels of new data and calculate the classification accuracy

(sınıflandırma sonucu)

# 1) Load and explore image data

- Rakam verileri ‘image datastore (görüntü verideposu)’ olarak yüklenir.
- *imageDatastore* görüntüleri klasör adlarına göre otomatik olarak etiketler ve verileri bir *ImageDatastore* nesnesi olarak depolar.
- Bir görüntü veri deposu, belleğe sığmayan veriler dahil olmak üzere büyük görüntü verilerini depolamanıza ve bir evrişimsel sinir ağının eğitimi sırasında görüntü yığınlarını verimli bir şekilde okumanıza olanak tanır.

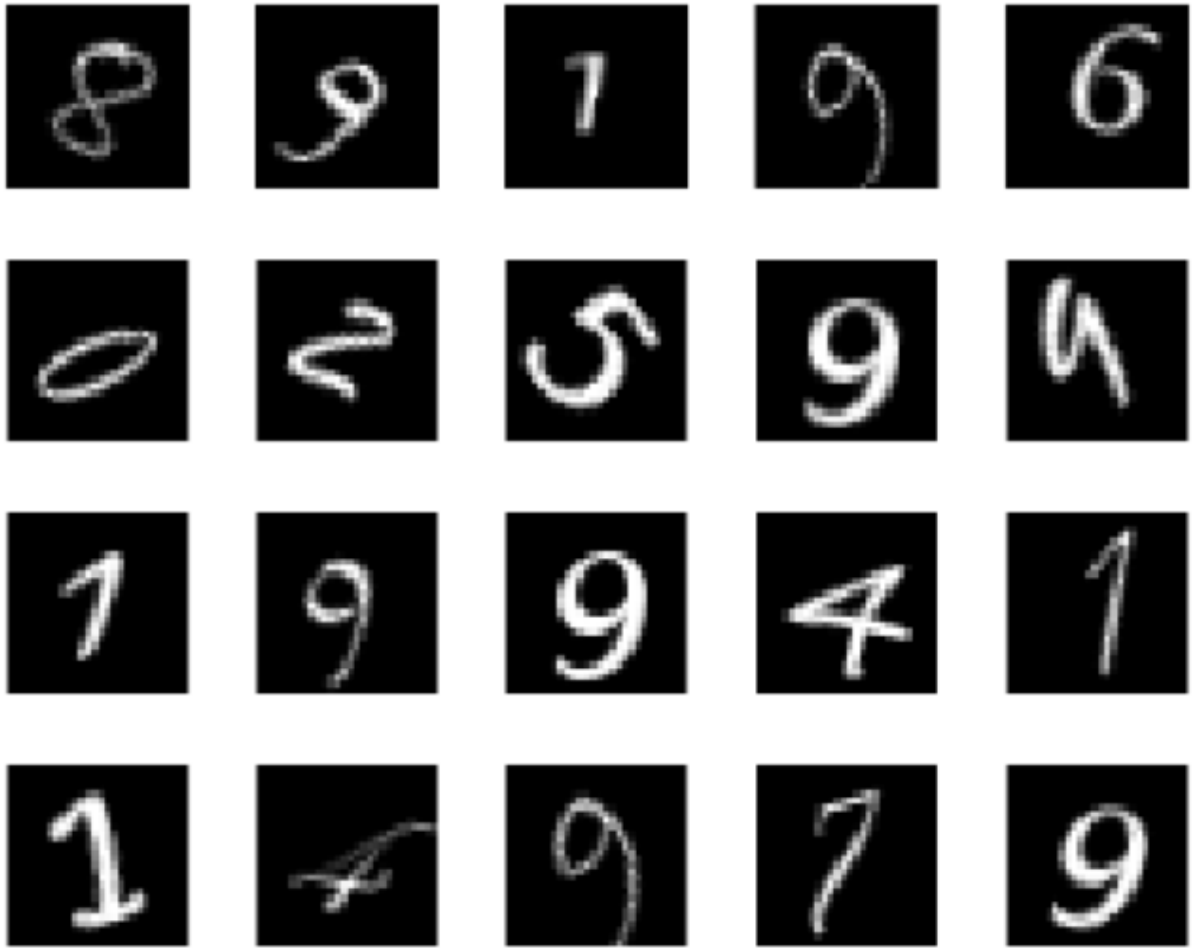
# 1) Load and explore image data

```
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet','nndemos', ...
    'nndatasets','DigitDataset');
imds = imageDatastore(digitDatasetPath, ...
    'IncludeSubfolders',true,'LabelSource','foldernames');
```

- Veri deposundaki bazı görüntüler görüntülenir.

```
figure;
perm = randperm(10000,20);
for i = 1:20
    subplot(4,5,i);
    imshow(imds.Files{perm(i)});
end
```

# 1) Load and explore image data



# 1) Load and explore image data

- Her kategorideki resimlerin sayısı hesaplanır. *labelCount*, her bir etiketin sahip olduğu görüntü sayısını ve etiketlerin sayısını içeren bir tablodur. Veri deposu, toplam 10000 görüntü olmak üzere; 0-9 rakamlarının her biri için 1000 görüntü içerir. *OutputSize* değişken olarak ağınızın, tam bağlı katmanındaki sınıfların sayısını belirtilir.

```
labelCount = countEachLabel(imds)
```

labelCount=*10×2 table*

| Label | Count |
|---|---|
| 0 | 1000 |
| 1 | 1000 |
| 2 | 1000 |
| 3 | 1000 |
| 4 | 1000 |
| 5 | 1000 |
| 6 | 1000 |
| 7 | 1000 |
| 8 | 1000 |
| 9 | 1000 |

# 1) Load and explore image data

- Ağın giriş katmanında görüntülerin boyutu belirtilir. *digitData* ile ilk görüntünün boyutu kontrol edilir. Her bir görüntü 28x28x1 pikseldir.

```
img = readimage(imds,1);
size(img)
```

```
ans = 1×2

    28    28
```

# 1) Load and explore image data

- Eğitim ve doğrulama set'leri belirtilir. Verileri eğitim ve doğrulama veri setlerine bölünür, böylece eğitim setindeki her kategori 750 resim içerir ve doğrulama seti her etiketten kalan resimleri içerir.
- *splitEachLabel;* veri deposu *digitData* 'yı, *trainDigitData* ve *valDigitData* olarak iki yeni veri deposuna böler.

```
numTrainFiles = 750;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
```

# 2) Define the network architecture

- Evrişimsel sinir ağı mimarisi tanımlanır.
- Image Input Layer: Bu katmanda 28x28x1 piksel boyutlu görüntü bulunur. Bu değerler sırasıyla; yükseklik, genişlik ve kanal boyutuna karşılık gelir. Rakam verileri gri tonlamalı görüntülerden oluşur, bu nedenle kanal boyutu (renk kanalı) 1'dir. Renkli bir görüntü için kanal boyutu RGB değerlerine karşılık gelen 3'tür.

```
layers = [
    imageInputLayer([28 28 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,16,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,32,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(10)
    softmaxLayer
    classificationLayer];
```

# 2) Define the network architecture

- Convolutional Layer: Evrişim katmanındaki ilk argüman olan *filterSize*, eğitim fonksiyonunun görüntüleri tararken kullandığı filtrelerin yüksekliği ve genişliğidir. Bu örnekte, 3 sayısı filtre boyutunun 3x3 olduğunu gösterir. Filtrenin yüksekliği ve genişliği için farklı boyutlar belirtilebilir. İkinci argüman olan *numFilters*, girdinin aynı bölgesine bağlanan nöronların sayısı olan filtre sayısıdır. Bu parametre, özellik haritalarının sayısını belirler.

# 2) Define the network architecture

- Max Pooling Layer: Maksimum havuzlama katmanı, bağımsız değişken *poolSize* tarafından belirtilen dikdörtgen girdi bölgelerinin maksimum değerlerini döndürür. Bu örnekte, dikdörtgen bölgenin boyutu [2,2] 'dir.

# 3) Specify training options

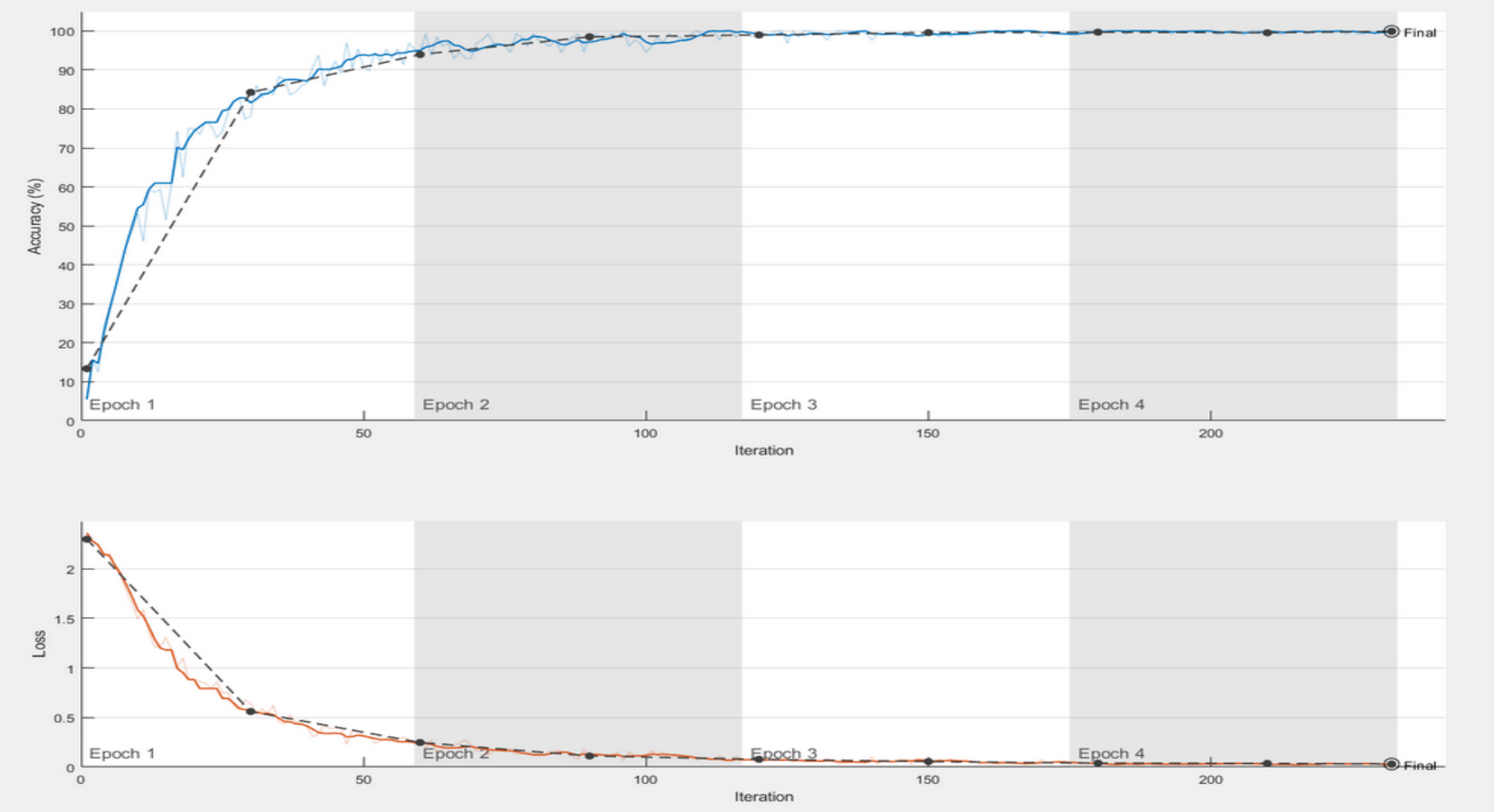
- Ağ yapısını tanımladıktan sonra eğitim seçeneklerini aşağıdaki gibi belirtilir.

```
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.01, ...
    'MaxEpochs',4, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
```

# 4) Train the network

- *Layers* tarafından tanımlanan mimari kullanılarak ağ eğitilir.

```
net = trainNetwork(imdsTrain,layers,options);
```



# 5) Predict the labels of new data and calculate the classification accuracy

- Eğitimli ağı kullanarak doğrulama verilerinin etiketleri tahmin edilir ve son doğrulama doğruluğu hesaplanır. Doğruluk, ağın doğru şekilde tahmin ettiği etiketlerin oranıdır. Bu örnekte, tahmin edilen etiketlerin %99'undan fazlası doğrulama set'inin gerçek etiketleriyle eşleşir.

```
YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation)
```

```
accuracy = 0.9988
```

# Kaynaklar


- https://www.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html (Son erişim tarihi: 04.10.2020)