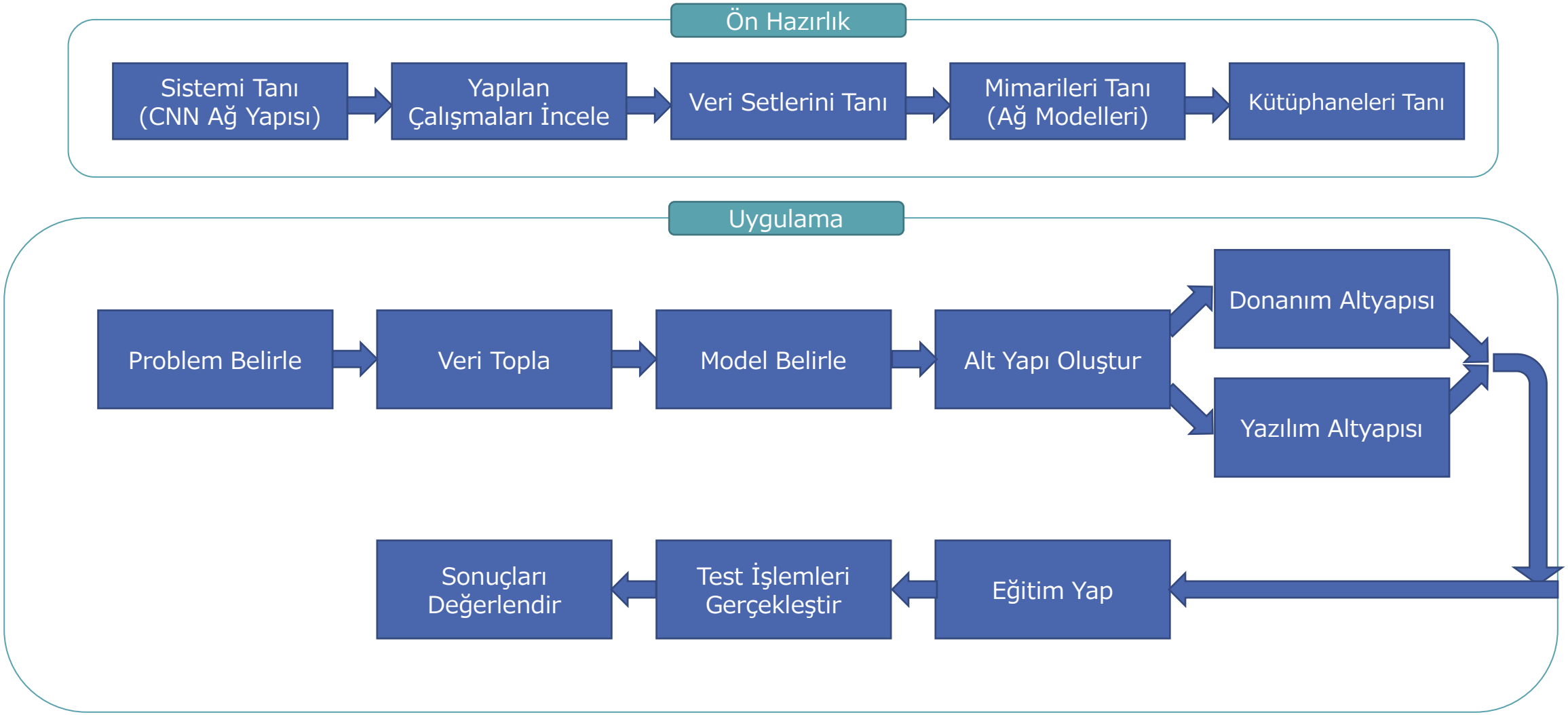
NESNE ALGILAMA VE SINIFLANDIRMA

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Dr.Günay TEMÜR

Algoritma

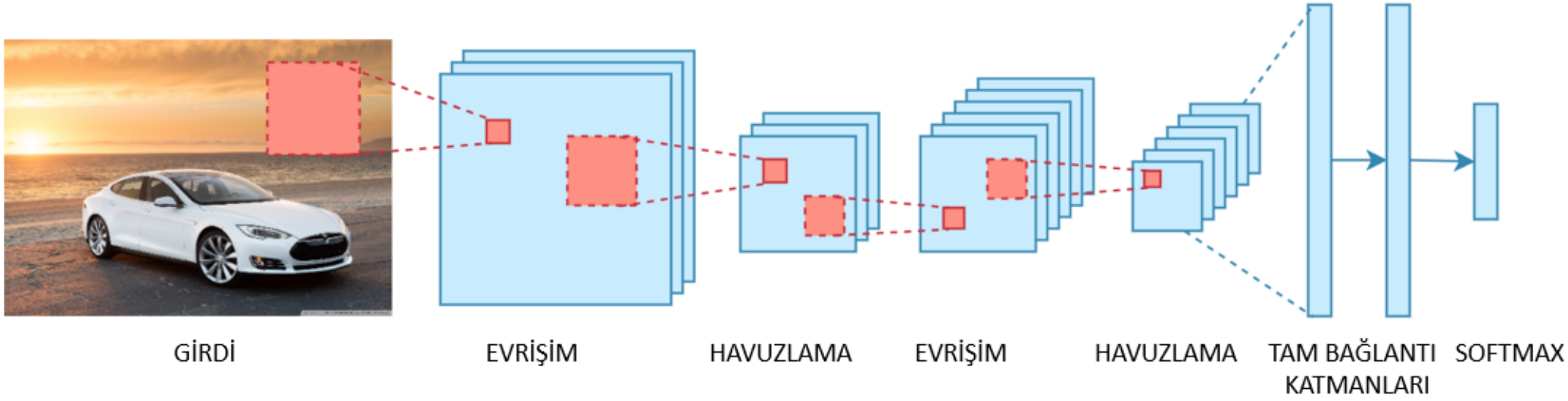

Ön Hazırlık
Sistemi Tanı (CNN Ağ Yapısı)
Yapılan Çalışmaları İncele
Veri Setlerini Tanı
Mimarileri Tanı (Ağ Modelleri)
Kütüphaneleri Tanı
Uygulama
Problem Belirle
Veri Topla
Model Belirle
Alt Yapı Oluştur
Donanım Altyapısı
Yazılım Altyapısı
Eğitim Yap
Test İşlemleri Gerçekleştir
Sonuçları Değerlendir

Evrişimli Sinir Ağları

Derin öğrenme ya da derin sinir ağı kavramı çok katmanlı yapay sinir ağlarını ifade etmektedir. En popüler derin sinir ağlarından biri evrişimli sinir ağları olarak görülmektedir.

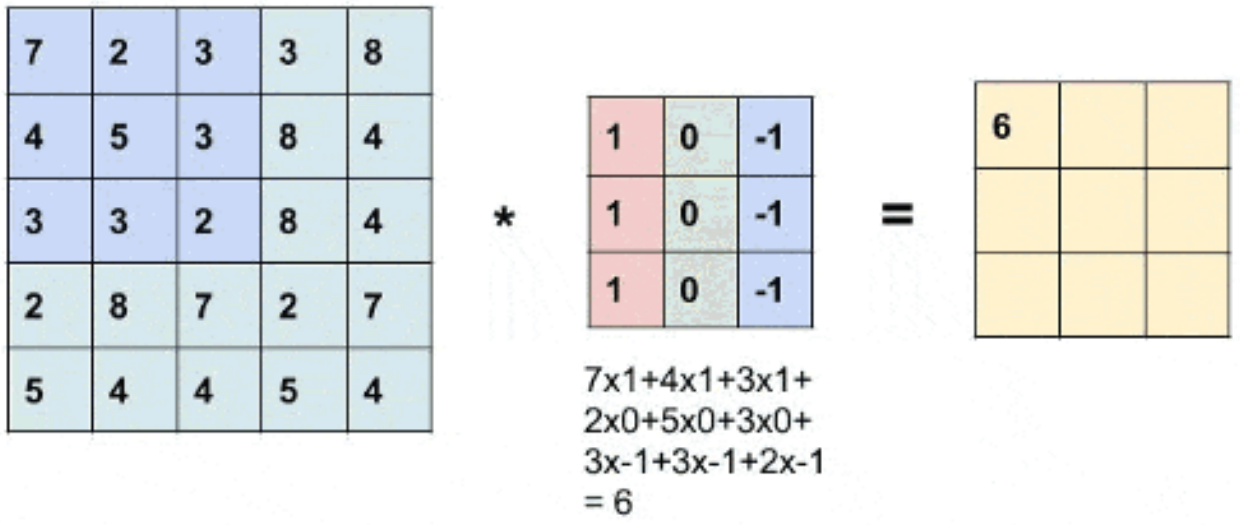
Evrişimli sinir ağları ile yapay sinir ağları arasındaki dikkate değer tek fark evrişimli sinir ağlarının öncelikle görüntü işlemede örüntü tanıma alanında kullanılmasıdır.

Evrişimli sinir ağları evrişim, havuzlama ve tam bağlantılı olmak üzere üç tür katmandan oluşmaktadır.



Evrişimli Katmanlar

Evrişimli Katman, CNN algoritmalarında görüntü üzerinde işlem yapan ilk katmandır. Bu katman için CNN'nin temel yapı taşı denilmesi yanlış olmaz. Görüntüler, içlerinde belirli değerler taşıyan piksellerden oluşan matrislerdir. Bu katmanda gerçekleşen olay görüntü matrisleri üzerinde dolaşan, görüntü matrislerinden daha küçük boyutlu filtre matrisleri ile görüntülerin belirli özelliklerini elde etmektir. Bu özellikler düşük veya yüksek seviyeli olabilirler. Örnek vermek gerekirse kullandığımız filtre görüntü üzerindeki kenarları algılayacak bir filtre olabilir.



Evrişimli Katmanlar

Görüntülerimiz tek bir özellikten oluşmadığı için genel olarak birden fazla filtre uygulanması gerekmektedir. Buradan CNN algoritmalarında çok sayıda evrişimli katman bulunduğu sonucuna ulaşılabilir. Ek olarak filtre işlemleri uygularken bilinmesi gereken iki terim daha vardır. Bu terimler Adım (Stride) ve Dolgu (Padding) terimleridir. Bunları kısaca açıklayacak olursak:

```
[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky
```

derin öğrenme modellerinde ağı eğitirken ağırlıkları ve aktivasyonları normalize etmek amacıyla kullanılan bir yöntemdir. **Batch normalization**, ağın daha hızlı eğitilmesine yardımcı olabilir, aşırı öğrenmeyi azaltabilir ve daha istikrarlı sonuçlar üretebilir

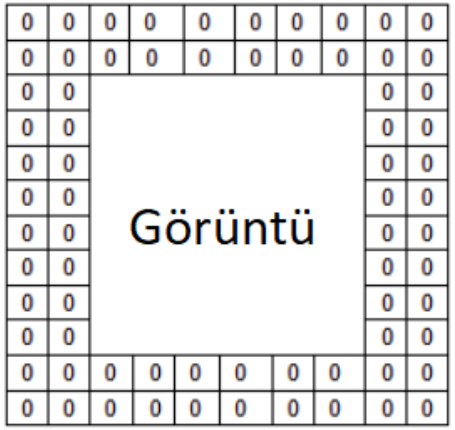
Filtreler, görüntünün belirli bir bölgesine uygulanır ve bu işlem, farklı özelliklerin çıkarılmasına yardımcı olur. Her filtre, belirli bir özelliği veya kalıbı tanımak için tasarlanmıştır.

Stride, bir konvolüsyon işlemi sırasında konvolüsyon penceresinin (filtrenin) ne kadar kaydırılacağını belirleyen bir parametredir.

Evrişimli Katmanlar

Adım değeri CNN modellerinde parametre olarak değiştirilebilen bir değerdir. Bu değer filtrenin ana görsel üzerinde kaç piksel boyunca kayacağını belirler. Eğer Adım değeri büyük bir değer olursa filtre görüntü üzerinde adım değerine göre kayacağı için ortaya çıkacak olan öznitelik haritası daha küçük boyutlu olacaktır.

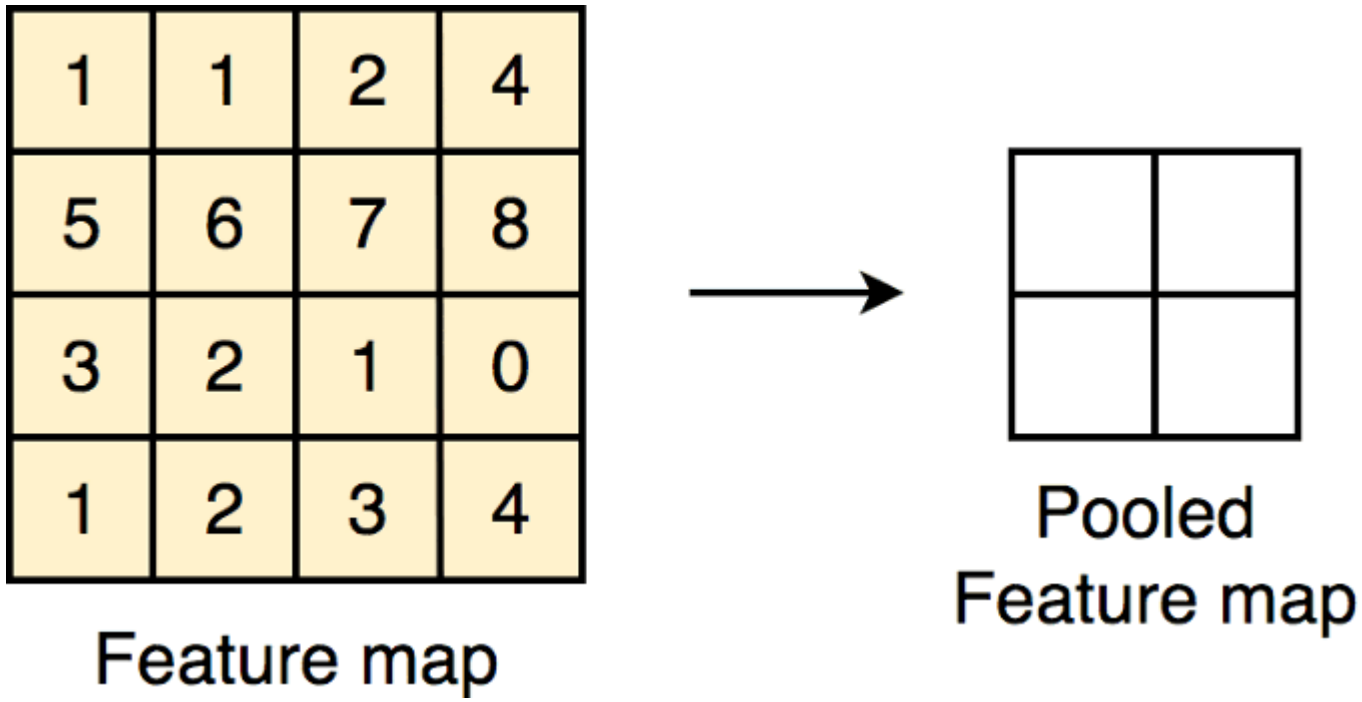
Bir görsele filtre uygulandığında boyutlardan dolayı çıktı orijinal görselden daha küçük olur. Bunu önlemek için kullanabileceğimiz yöntem ise padding yani dolgulamadır. Dolgulama işleminde görsele adete bir çerçeve olacakmış gibi dört taraftan da sıfırlar eklenir. Filtrenin boyutuna göre bu sıfır eklenen katmanlar artırılabilir.



Padding

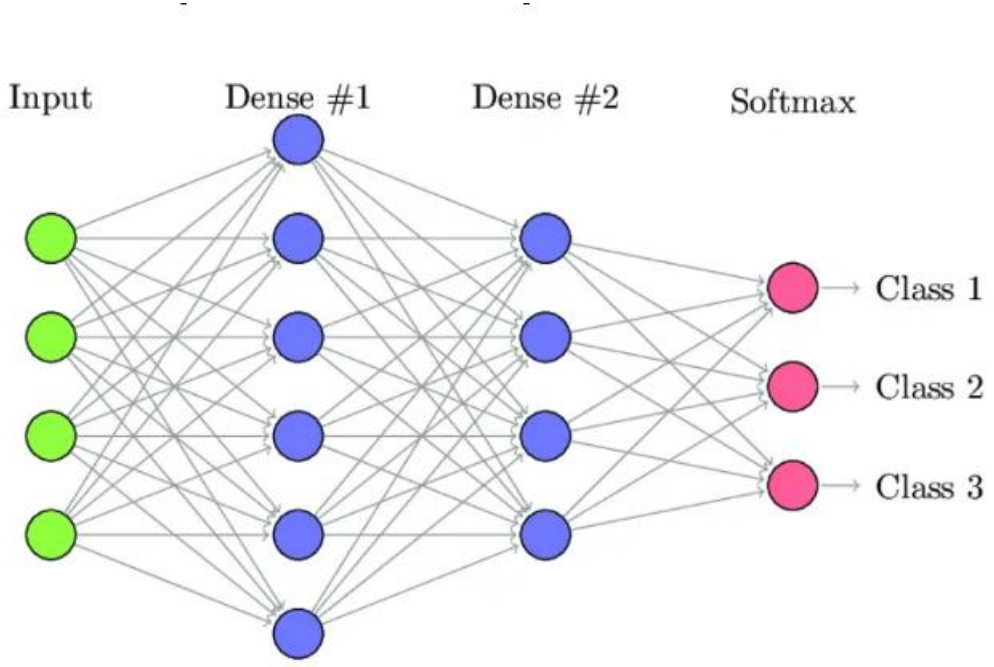
Havuzlama Katmanı (Pooling Layers)

Havuzlama katmanın amacı, görüntüler çok büyük olduğunda parametre sayısını azaltarak görüntüyü küçültmektir ve aşırı öğrenmeyi kontrol etmektir. Bu işlem, her bir haritanın boyutsallığını azaltır ama önemli bilgileri korur. 3 tür havuzlama yöntemi vardır. Maksimum, ortalama ve minimum havuzlama yöntemidir. CNN ağlarında genellikle maksimum havuzlama kullanılır.



Tam Bağlantı Katmanları (Fully Connected Layers)

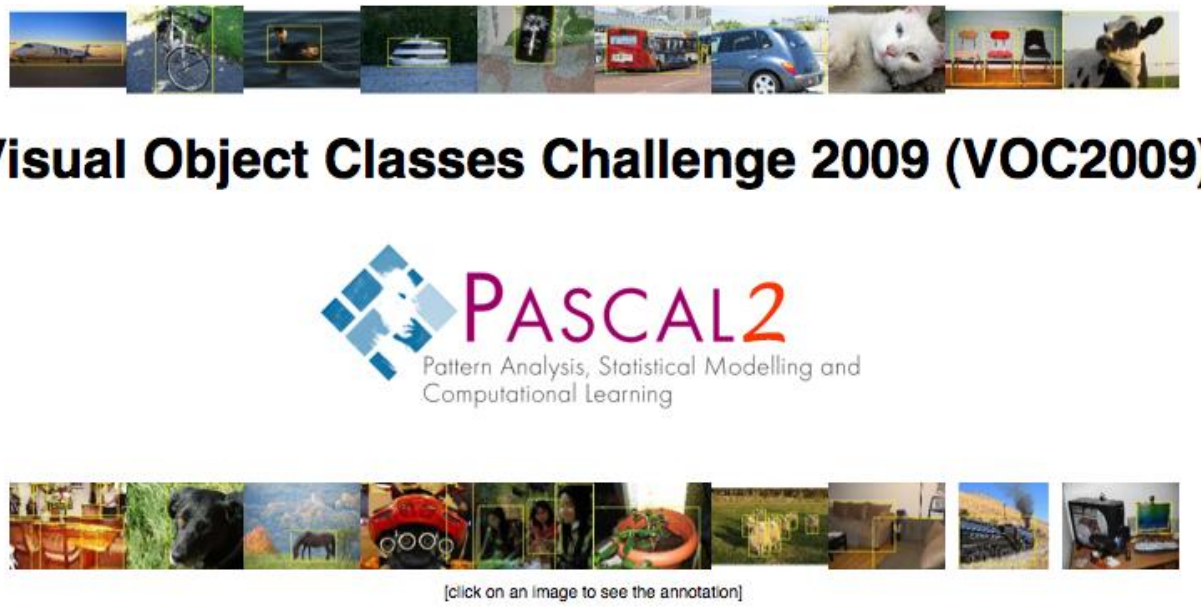
Evrişimli Sinir Ağlarının son adımı bu katmanda gerçekleşir. Tam Bağlantı Katmanları, mimarinin bir katmandaki bütün düğüm ve nöronlarının bir sonraki katmanlara bağlandığı bir yapay sinir ağ türüdür. Bu ağ türü hesaplama açısından karmaşık olmasının yanı sıra aşırı yüklenmeye meyillidir.



Veri Setleri

ImageNet veri seti 27 üst kategoriden ve bu kategorilerle ilişkili olarak en az 20binalt kategoriden oluşan yaklaşık 14 milyon tam çözünürlüklü görüntü barındıran görüntü veri kümesidir.

Veri Setleri



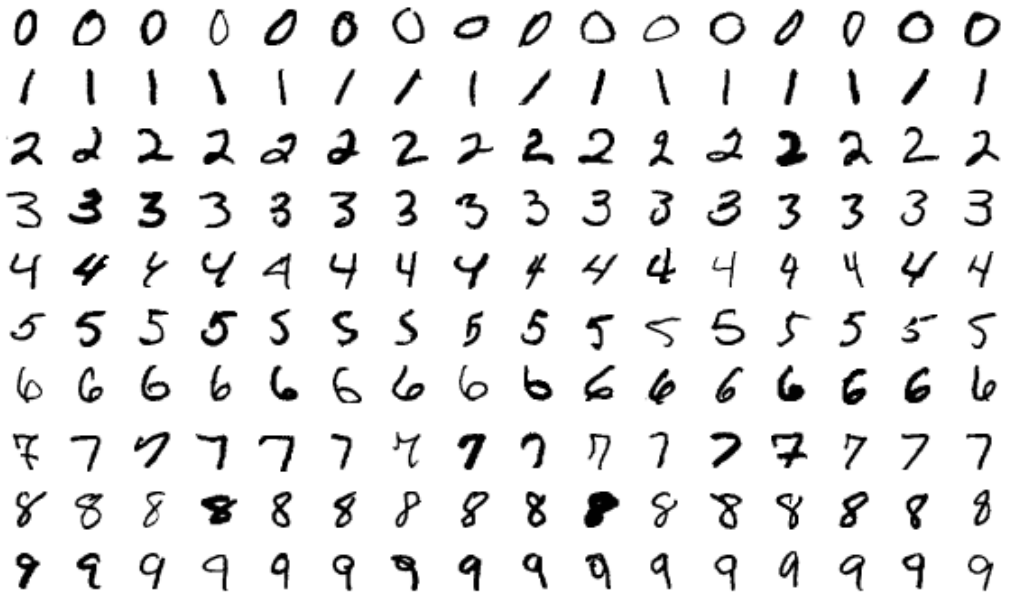
Pascal VOC veri seti 2005 –2012 yılları arasında geliştirilerek yayınlanmış her yıl aynı isimle düzenlenen yarışmada kullanılan ve en son haliyle toplam 20 nesne sınıfı olmak üzere 11.530 görüntüden ve 27.450 nesne etiketi verisinden oluşan görsel veri setidir.

Veri Setleri

MS COCO veri kümesi, 82 nesne kategorisi 5 binden fazla etiketli içeriğe sahip toplam 91 ortak nesne kategorisi içerir. Toplamda veri kümesinde 328 bin görüntüde 2 milyon 500 bin etiketli örneği bulunmaktadır. ImageNet veri setinin aksine, COCO daha az kategoriye ve kategori başına daha fazla örneğe sahiptir.

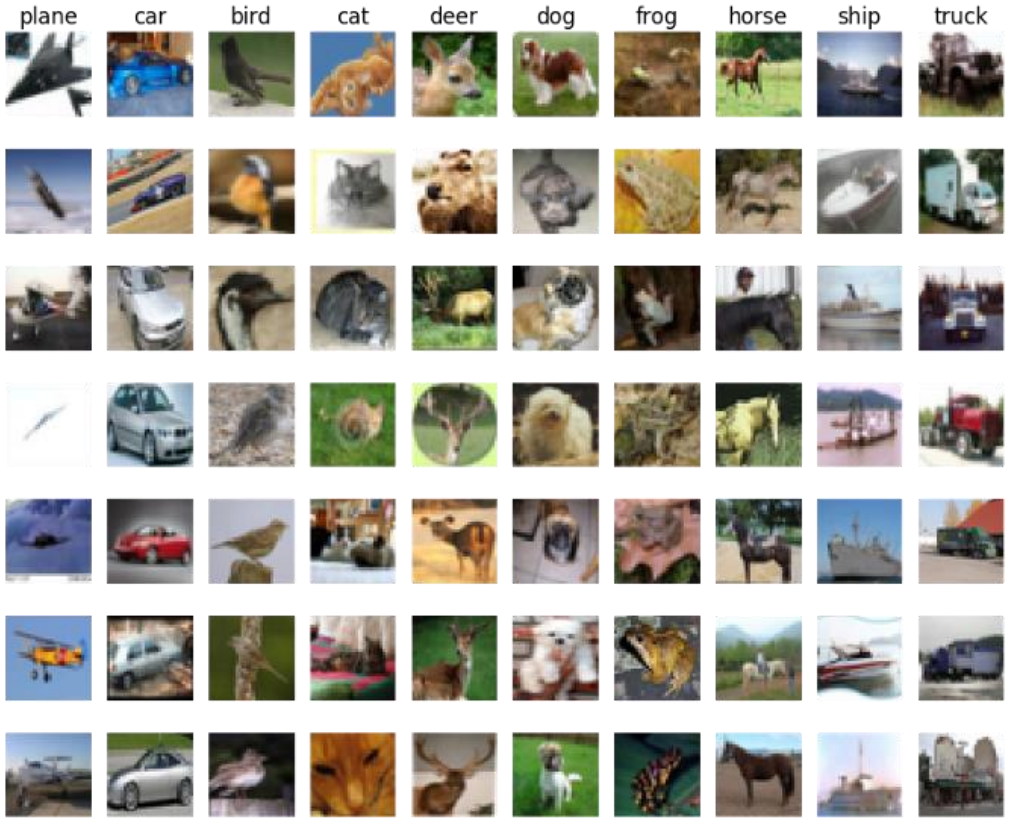
Veri Setleri

MNIST



MNIST, el yazısıyla yazılmış rakamlardan oluşan ve 60 bin eğitim, 10 bin test olmak üzere toplamda 70 bin görüntüden oluşan bir veri setidir. Görüntüler 28x28 piksel boyutlarında olup gri seviyelidir.

Veri Setleri



CIFAR, 80 milyon küçük boyutlu görüntü içeren veri setidir. İki farklı versiyonu olan veri setinde;

CIFAR-10 veri seti her sınıfta 6000 görüntü bulunan toplam 10 sınıftan oluşmaktadır. Görüntü boyutları 32x32 piksel boyutlarında olup 50 bin görüntü eğitim, 10 bin görüntü test kümesi olarak ayrılmıştır

CIFAR-100 veri seti ise her sınıfta 600 görüntü bulunan toplam 100 sınıftan oluşmaktadır

Nesne Algılama Mimarileri

AlexNet

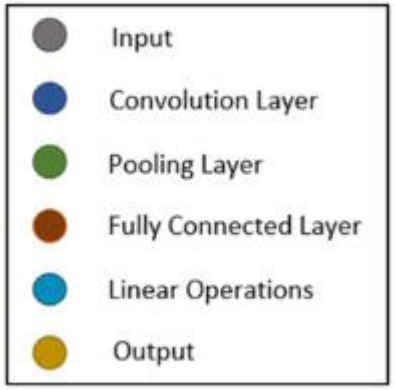
VGG-16

GoogLeNet

ResNet-50

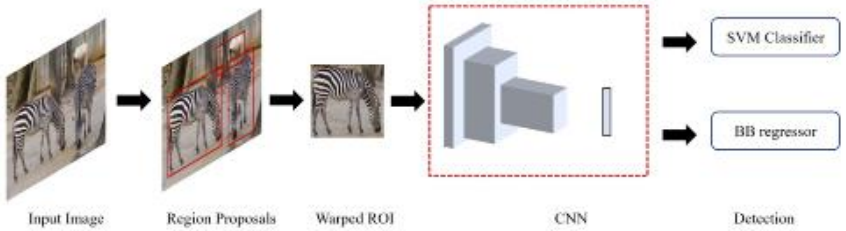
CSPResNeXt-50

EfficientNet

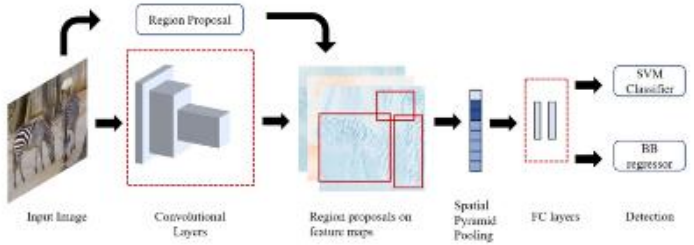


Nesne Algılama Algoritmaları: Two-stage detectors

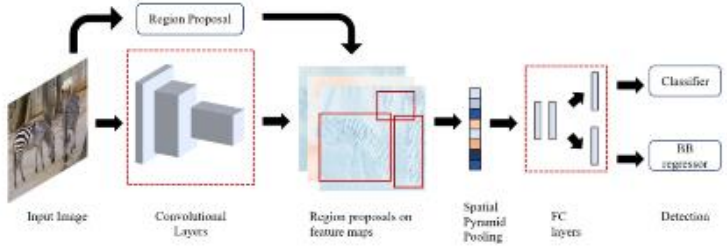
RCNN



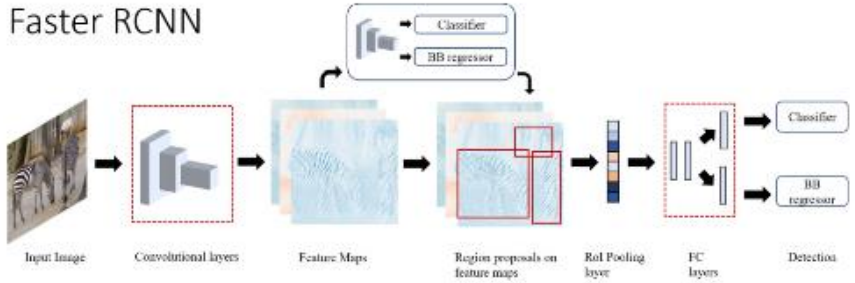
SPP-Net



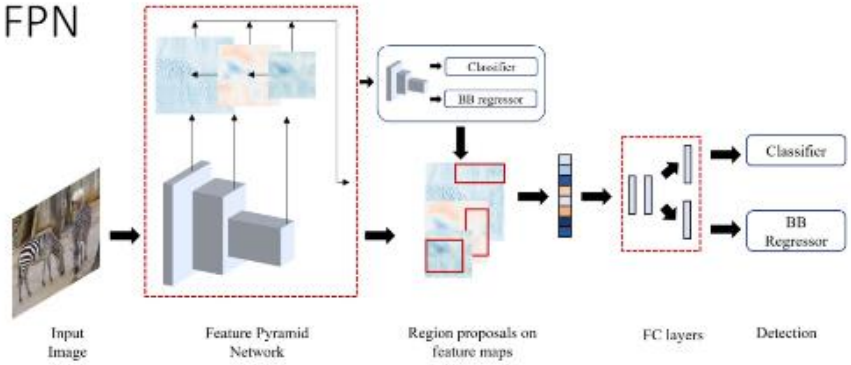
Fast RCNN



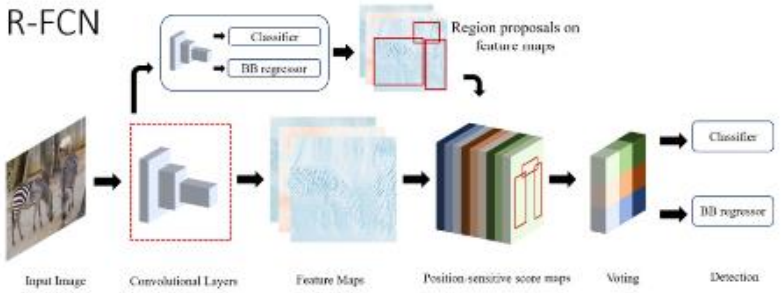
Faster RCNN



FPN

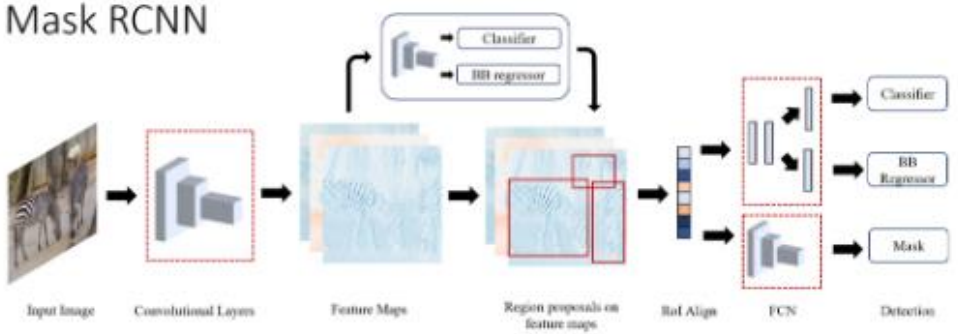


R-FCN

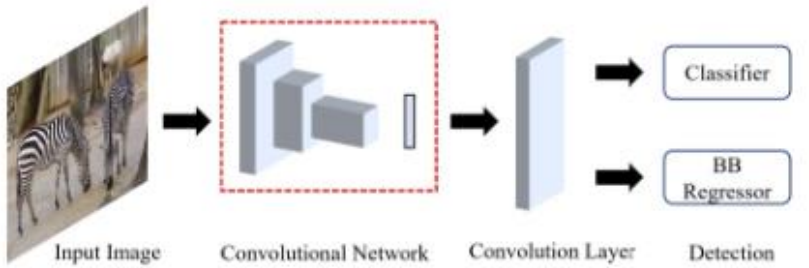


Nesne Algılama Algoritmaları: Single stage detectors

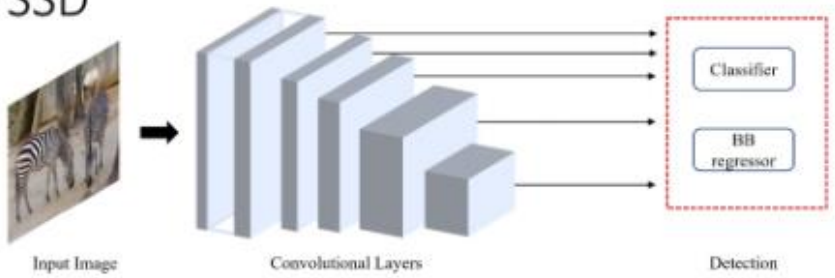
Mask RCNN



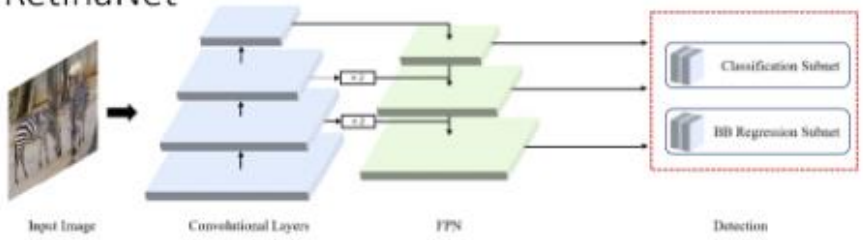
YOLO



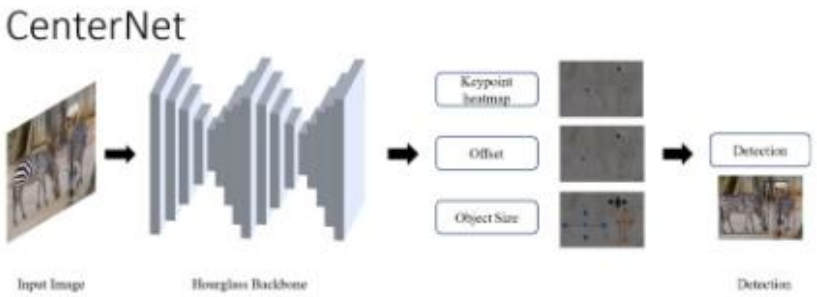
SSD



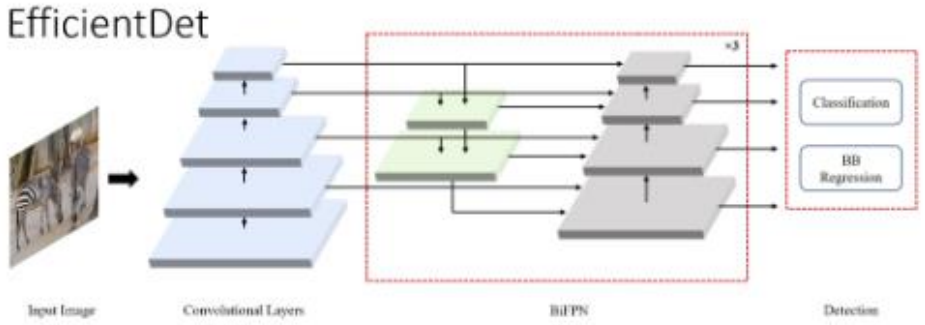
RetinaNet



CenterNet



EfficientDet



Nesne Algılama Algoritmaları : Single stage detectors

"Single-stage detectors" (Tek Aşamalı Algılayıcılar), bilgisayarlı görü sistemlerinde nesne tespiti yaparken kullanılan bir yöntemdir. Bu yöntem, nesneleri tespit etmek için yalnızca bir aşamayı içerir. Tek aşamalı algılayıcılar, nesneleri hızlı bir şekilde ve tek bir aşamada tespit etmek için tasarlanmıştır.

Tek aşamalı algılayıcılar genellikle şu özelliklere sahiptir:

Hızlı İşlem: Tek aşamalı algılayıcılar, özellikle gerçek zamanlı uygulamalar için hızlı çalışma yeteneğine sahiptir. Bu, hızlı nesne tespiti gereken uygulamalarda tercih edilen bir özelliktir.

Düşük Bellek Kullanımı: Tek aşamalı algılayıcılar, daha düşük bellek kullanımıyla çalışabilirler, bu da daha hafif ve daha hızlı modeller oluşturmayı mümkün kılar.

Basitlik: Tek aşamalı algılayıcılar, tasarım olarak daha basit olabilir ve daha az karmaşık bir yapıya sahip olabilirler.

Örneğin, "YOLO" (You Only Look Once) ve "SSD" (Single Shot MultiBox Detector) gibi nesne tespiti algoritmaları, tek aşamalı algılayıcılar olarak kabul edilirler. Bu algoritmalar, hızlı ve etkili nesne tespiti için kullanılırlar. Tek aşamalı algılayıcılar, özellikle gerçek zamanlı nesne tespiti ve uygulamalarında popülerdir.

Nesne Algılama Algoritmaları : Two-stage detectors

Two-stage detectors" (İki Aşamalı Algılayıcılar), bilgisayarlı görü sistemlerinde kullanılan bir nesne tespit yöntemini ifade eder. Bu yöntemde, nesneleri tespit etmek için iki ayrı aşama kullanılır. İki aşamalı bir nesne tespit sistemi genellikle şu şekilde çalışır:

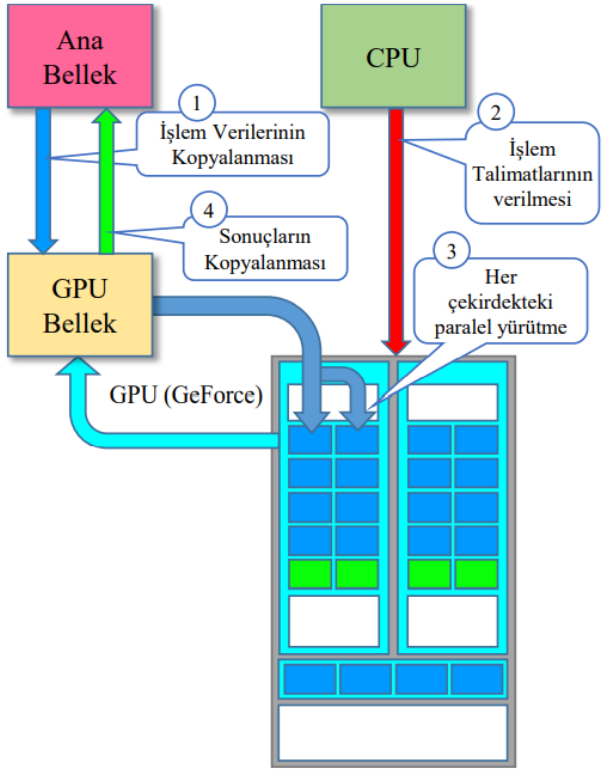
Bölge Önerisi (Region Proposal): İlk aşamada, görüntüde potansiyel nesne bölgesi önerileri üretilir. Bu bölge önerileri, görüntüde muhtemelen nesne bulunan bölgeleri belirlemek için kullanılır. Bu aşamada yaygın olarak "Selective Search," "EdgeBoxes," "Region Proposal Network (RPN)," gibi yöntemler kullanılır.

Nesne Tanıma (Object Recognition): İkinci aşamada, her bir bölge önerisi ayrı ayrı bir nesne tespiti modeline veya sınıflandırıcıya (örneğin, bir derin öğrenme ağı) iletilir. Bu model, her bölgeyi belirli bir nesne sınıfına ait olup olmadığını belirlemek için kullanılır. Ayrıca, nesnenin konumu ve sınıfı tespit edilir.

İki aşamalı algılayıcılar, özellikle nesne tespiti ve sınıflandırma görevlerinde kullanılırlar. İlk aşama, bölgeleri belirlemek ve sayılarını azaltmak için kullanılırken, ikinci aşama, her bölgeyi daha ayrıntılı bir şekilde inceleyerek nesne tespiti ve sınıflandırma yapar. Bu yöntem, özellikle karmaşık nesneleri tespit etmek için etkilidir ve özelleştirilmiş nesne tespiti görevlerinde yaygın olarak kullanılır.

Kütüphaneler

CUDA : CUDA (Compute Unified Device Architecture), Nvidia tarafından geliştirilen bir paralel hesaplama platformu ve uygulama programlama arayüzü (API) modelidir. C, C ++ ve Python gibi programlama dilleriyle çalışacak şekilde tasarlanmıştır.




```python
import torch

# GPU kullanılabilir mi kontrol et
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

# Rastgele veri oluştur
n = 10000
a = torch.rand(n).to(device)  # GPU'ya taşı
b = torch.rand(n).to(device)  # GPU'ya taşı

# İşlemi GPU üzerinde yap
result = a + b

# Sonucu GPU'dan CPU'ya taşı
result_cpu = result.to("cpu")

# Sonucu yazdır
print("Sonuç: ", result_cpu)
```

Bu örnek, CPU yerine GPU'da hesaplama yapmak için PyTorch'u kullanır. **to()** yöntemi ile tensörleri GPU'ya taşır ve sonucu GPU'dan CPU'ya taşır. Bu, PyTorch'un GPU hızlandırmayı kullanarak basit bir işlemi nasıl gerçekleştireceğini gösteren basit bir örnektir.

Kütüphaneler

cuDNN: NVIDIA CUDA® Derin Sinir Ağı kütüphanesi (cuDNN), derin sinir ağlarının temelleri için GPU hızlandırmalı bir kütüphanedir. cuDNN, ileri ve geri konvilsüyon, havuzlama, normalleştirme ve aktivasyon katmanları gibi standart görevler için yüksek düzeyde ayarlanmış uygulamalar sağlar.

Tensorflow: Başlangıçta Google tarafından Google Beyin projesinin bir parçası olarak geliştirilen Tensorflow, 2015'in sonlarında açık kaynak haline getirilmiştir. Tensorflow bir Python kütüphanesidir.

OpenCV: OpenCV (Açık Kaynak Kodlu Bilgisayarla Görme Kütüphanesi), açık kaynak kodlu bir bilgisayarla görme ve makine öğrenmesi yazılımı kütüphanesidir. OpenCV, bilgisayarla görme uygulamaları için ortak bir altyapı sağlamak ve ticari ürünlerde makine algısının kullanımını hızlandırmak için oluşturulmuştur.

Kütüphaneler

PyTorch: Torch kütüphanesine dayanan açık kaynaklı bir makine öğrenme kütüphanesidir, bilgisayarla görme ve doğal dil işleme gibi uygulamalar için kullanılır. Öncelikle Facebook'un AI Araştırma laboratuvarı (FAIR) tarafından geliştirilmiştir.

NumPy: NumPy Python programlama dili için oluşturulan bir kütüphanedir. Büyük, çok boyutlu diziler ve matrislerin yanı sıra geniş bir yüksek seviye matematiksel fonksiyon koleksiyonuna bu diziler üzerinde çalışabilmeleri için ek destek verir.

MatplotLib: Matplotlib, Python programlama dili ve sayısal matematik uzantısı NumPy için bir grafik kütüphanesidir. Grafikleri Tkinter, wxPython, Qt veya GTK + gibi genel amaçlı GUI araç takımlarını kullanarak uygulamalara gömmek için nesne yönelimli bir API sağlar.

Kütüphaneler : numpy

python Copy code

```python
import numpy as np
import cupy as cp

# Rastgele iki matris oluşturma
n = 1000  # Matris boyutu
A = np.random.rand(n, n).astype(np.float32)
B = np.random.rand(n, n).astype(np.float32)

# Matrisleri GPU'ya kopyalama
A_gpu = cp.array(A)
B_gpu = cp.array(B)

# Matrisleri toplama işlemi
C_gpu = cp.add(A_gpu, B_gpu)

# Sonucu CPU'ya kopyalama
C = cp.asnumpy(C_gpu)

# İlk 5×5 bölümünü yazdırma
print(C[:5, :5])
```

Bu örnekte, önce CPU üzerinde iki rastgele matris oluşturulur. Daha sonra, bu matrisler GPU'ya kopyalanır ve GPU üzerinde matris toplama işlemi yapılır. Sonuç matrisi, GPU'dan CPU'ya kopyalanarak ekrana yazdırılır.

CUDA kütüphanesini kullanırken uygun donanım ve yazılım gereksinimlerine dikkat etmek önemlidir.

Kütüphaneler : OpenCV

```python
import cv2
import numpy as np

# Resmi yükleme
image = cv2.imread('sample.jpg')

# Resmi görüntüleme
cv2.imshow('Orjinal Resim', image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Resmi Gri Tonlamaya Dönüştürme
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Gri Tonlama Resmini Görüntüleme
cv2.imshow('Gri Tonlama', gray_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Resmin Üzerine Çizim Yapma (Örnek olarak bir dikdörtgen)
cv2.rectangle(image, (100, 100), (300, 300), (0, 255, 0), 2)

# Çizim Yapılmış Resmi Görüntüleme
cv2.imshow('Çizim Yapılmış Resim', image)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Resmi Kaydetme
cv2.imwrite('sample_with_rectangle.jpg', image)
```

cv2.imread ile bir resim yüklenir ve cv2.imshow ile görüntülenir.

cv2.cvtColor ile resim gri tonlamaya dönüştürülür ve yine cv2.imshow ile görüntülenir.

cv2.rectangle ile resmin üzerine bir dikdörtgen çizilir ve yine cv2.imshow ile görüntülenir.

Son olarak, cv2.imwrite ile resim diskte kaydedilir.

Bu örnek, OpenCV kullanarak resim yükleme, görüntüleme, işleme ve kaydetme işlemlerini gösterir

Kütüphaneler : MatplotLib

```python
import matplotlib.pyplot as plt

# Örnek veriler
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

# Basit bir çizgi grafiği çizme
plt.plot(x, y)

# Eksen başlıkları ve grafik başlığı ekleme
plt.xlabel('X Ekseni')
plt.ylabel('Y Ekseni')
plt.title('Basit Bir Çizgi Grafiği')

# Grafiği gösterme
plt.show()
```

Bu örnekte:

x ve y listeleri, çizgi grafiği çizmek için kullanılacak örnek verileri temsil eder.

plt.plot ile çizgi grafiği çizilir.

plt.xlabel, plt.ylabel ve plt.title ile eksen başlıkları ve grafik başlığı eklenir.

plt.show ile grafiği görüntülenir.

Bu örnek, Matplotlib kütüphanesini kullanarak basit bir çizgi grafiği nasıl çizeceğinizi ve grafik üzerine nasıl başlık ve etiket ekleyeceğinizi gösterir.

BİTTİ ☺