VERİ BİLİMİ DERSİ

CNN ve Transfer Learning

Hafta 12 · Modül 12

Bilgisayarlı Görü, Evrişimli Ağlar ve Önceden Eğitilmiş Modeller

Dr. Murat Altun

Veri Bilimi ve Yapay Zekâ Eğitimi · 2026

6

Saat

3

Notebook

90%+

Hedef Doğruluk

İçindekiler

CNN Nedir? — Evrişimli Sinir Ağları

BÖLÜM 1

Convolutional Neural Network

CNN, görüntü verisi üzerinde uzmanlaşmış derin öğrenme mimarisidir. Piksel komşuluk ilişkilerini öğrenir, her katmanda daha soyut özellikler çıkarır. İnsan görme sisteminden ilham alır: kenarlar → şekiller → nesneler.

ANN vs CNN — Temel Farklar

Özellik	ANN (Dense)	CNN
Girdi	Düz vektör (1D)	2D/3D tensör (görüntü)
Bağlantı	Tam bağlantılı	Yerel bağlantılı (kernel)
Parametre	Çok fazla	Paylaşımlı ağırlıklar
Kullanım	Tablo verisi	Görüntü, video, ses

Neden Görüntü için İdeal?

1 **Uzamsal Hiyerarşi**

İlk katmanlar kenar, sonrakiler şekil, en derin katmanlar nesne tanır

2 **Parametre Paylaşımı**

Aynı filtre tüm görüntü üzerinde kayarak çok az parametre ile öğrenir

3 **Öteleme Değişmezliği**

Nesnenin konumu değişse de aynı filtreyle tespit edilir

CNN Katmanları — 4 Adımlı Pipeline

1

Conv2D

Evrişim katmanı: filtreler görüntüyü tarar, özellik haritaları (feature maps) üretir.
Girdi: (H×W×C) → Çıktı: (H'×W'×F)

2

MaxPool

Boyut küçültme: özellik haritalarını özetler, hesaplama maliyetini düşürür.
Girdi: (H×W) → Çıktı: (H/2×W/2)

3

Flatten

2D özellik haritalarını 1D vektöre dönüştürür. Sınıflandırma katmanlarına köprü görevi görür.

4

Dense

Tam bağlantılı katman: sınıflandırma yapar. Son katmanda softmax ile olasılık dağılımı üretir.

Filtre/Kernel — CNN Görüntüyü Nasıl Görür?

BÖLÜM 1

Hiyerarşik Özellik Çıkarımı

CNN'in her katmanı, bir önceki katmanın çıktısından daha soyut özellikler çıkarır. İlk katmanlar düşük seviyeli özellikleri (kenarlar, gradyanlar), orta katmanlar dokuları ve şekilleri, derin katmanlar nesneleri tanır.

Katman 1–2

Kenarlar & Köşeler

Yatay, dikey ve çapraz kenarları tespit eder. 3×3 kernel ile basit gradyan filtreleri.

Katman 3–4

Dokular & Şekiller

Kenarları birleştirerek daireler, dikdörtgenler, dokular oluşturur. Daha büyük alıcı alan.

Katman 5+

Nesneler & Yüzler

Yüz, araba, hayvan gibi yüksek seviyeli kavramları tanır. Semantik anlam.

Conv2D Katmanı — Parametreler

Temel Parametreler

`filters`	Kaç farklı özellik haritası üretilecek (ör: 32, 64, 128)
`kernel_size`	Filtre boyutu — genellikle (3,3) veya (5,5)
`strides`	Filtrenin kaydırma adımı — (1,1) standart, (2,2) hızlı
`padding`	'valid' (küçültür) veya 'same' (boyutu korur)
`activation`	'relu' en yaygın — negatif değerleri sıfırlar

```python
from tensorflow.keras.layers import Conv2D

# İlk evrişim katmanı
Conv2D(
    filters=32,
    kernel_size=(3, 3),
    strides=(1, 1),
    padding='same',
    activation='relu'
)
```

Padding ve Stride — Görsel Karşılaştırma

`padding='valid'`

Filtre sadece tam sığdığı yerlere uygulanır.
Çıktı boyutu küçülür:
(H−K+1) × (W−K+1)

`padding='same'`

Girdi sıfırlarla çevrelenir.
Çıktı boyutu girdiye eşit kalır:
H × W (stride=1 ise)

`stride=2`

Filtre 2 piksel atlayarak kayar.
Çıktı boyutu yarıya düşer:
Pooling'e alternatif.

MaxPooling — Boyut Küçültme

MaxPooling2D Nedir?

Her pencere (pool_size) içindeki en büyük değeri seçer. Böylece:

- Özellik haritası boyutu küçülür (hesaplama azalır)
- En belirgin özellikler korunur
- Küçük öteleme farklılıkları tolere edilir
- Overfitting riski azalır

2×2 MaxPool Örneği

Girdi (4×4)

1	3	2	1
5	6	1	0
2	1	8	4
3	0	2	7

→

Çıktı (2×2)

6	2
3	8

÷4
Boyut Küçültme
(2×2 pool → ¼)

0
Öğrenilebilir
Parametre

Max
Havuzlama Tipi
(Average da var)

2×2
Standart
Pool Boyutu

İpucu: Stride=2 ile Conv2D, MaxPooling'in yerine kullanılabilir. Modern mimarilerde (ResNet) bu tercih edilir.

CIFAR-10 ile CNN — Veri Seti Tanıtımı

BÖLÜM 2

10	60K	32×32	RGB	50K/10K
Sınıf	Görüntü	Piksel	3 Kanal	Train/Test

10 Sınıf

Uçak (airplane)	Otomobil (automobile)
Kuş (bird)	Kedi (cat)
Geyik (deer)	Köpek (dog)
Kurbağa (frog)	At (horse)
Gemi (ship)	Kamyon (truck)

Önerilen CNN Mimarisi

`Conv2D(32, 3×3) + ReLU`	`(32, 32, 32)`
`MaxPooling2D(2×2)`	`(16, 16, 32)`
`Conv2D(64, 3×3) + ReLU`	`(16, 16, 64)`
`MaxPooling2D(2×2)`	`(8, 8, 64)`
`Conv2D(64, 3×3) + ReLU`	`(8, 8, 64)`
`Flatten`	`(4096,)`
`Dense(64) + ReLU`	`(64,)`
`Dense(10) + Softmax`	`(10,)`

CIFAR-10 — Tam Model Kodu


```
import tensorflow as tf
from tensorflow.keras import layers, models

# Veri yükleme ve normalizasyon
(x_train, y_train), (x_test, y_test) = \
    tf.keras.datasets.cifar10.load_data()
x_train, x_test = x_train/255.0, x_test/255.0

# Model oluşturma
model = models.Sequential([
  layers.Conv2D(32,(3,3),activation='relu',
    input_shape=(32,32,3)),
  layers.MaxPooling2D((2,2)),
  layers.Conv2D(64,(3,3),activation='relu'),
  layers.MaxPooling2D((2,2)),
  layers.Conv2D(64,(3,3),activation='relu'),
  layers.Flatten(),
  layers.Dense(64,activation='relu'),
  layers.Dense(10,activation='softmax')
])
```

```
# Derleme
model.compile(
  optimizer='adam',
  loss='sparse_categorical_
    crossentropy',
  metrics=['accuracy']
)
```

```
# Eğitim
history = model.fit(
  x_train, y_train,
  epochs=10, batch_size=64,
  validation_split=0.2)
```

Beklenen Sonuç

~70% doğruluk (10 epoch)
~75% (data augmentation ile)

Transfer Learning Nedir?

Analoji: Bisiklet Süren Çocuk → Motosiklet

Bisiklet sürmeyi öğrenen bir çocuk, motosiklete geçtiğinde sıfırdan başlamaz. Denge, yön kontrolü, fren refleksi gibi temel becerileri transfer eder. Transfer Learning de aynı mantıkla çalışır: Büyük bir veri setinde (ImageNet — 14M görüntü) eğitilmiş bir model, yeni ve küçük bir veri setine uyarlanır. Önceki bilgi, yeni göreve aktarılır.

1. Önceden Eğitilmiş Model (ImageNet)

1.2M görüntü
1000 sınıf
Milyonlarca parametre

→

2. Üst Katmanları Kaldır

Son Dense katmanları çıkarılır
(sınıflandırma kafası)

→

3. Yeni Katmanlar Ekle

Kendi veri setinize uygun Dense/Softmax katmanları eklenir

→

4. Fine-tune veya Freeze + Train

Alt katmanları dondur veya düşük lr ile tümünü eğit

Neden Transfer Learning?

100×

Daha Az Veri Gereksinimi

10×

Daha Hızlı Eğitim Süresi

95%+

Yüksek Doğruluk

Sıfırdan Eğitim vs Transfer Learning

Kriter	Sıfırdan CNN	Transfer Learning
Veri	10.000+ görüntü gerekli	100–1000 yeterli
Eğitim Süresi	Saatler / Günler	Dakikalar
GPU İhtiyacı	Güçlü GPU şart	CPU bile yeterli olabilir
Doğruluk	Veriye bağlı, değişken	Genellikle yüksek (%90+)
Zorluk	Mimari tasarım gerekli	Hazır model seç, uyarla

Önceden Eğitilmiş Modeller — Karşılaştırma

Popüler ImageNet Modelleri

Model	Yıl	Parametre	Top-1 Acc	Boyut (MB)	Hız	Avantaj
VGG16	2014	138M	71.3%	528	Yavaş	Basit mimari, öğretici — eğitim amaçlı ideal
ResNet50	2015	25.6M	76.0%	98	Orta	Skip connection: derin ağlarda gradyan kaybını çözer
InceptionV3	2015	23.8M	77.9%	92	Orta	Paralel filtreler: farklı ölçekleri aynı anda yakalar
MobileNetV2	2018	3.4M	71.3%	14	Hızlı	Depthwise separable conv: mobil cihazlar için hafif
EfficientB0	2019	5.3M	77.1%	29	Hızlı	Compound scaling: genişlik × derinlik × çözünürlük

Eğitim İçin

VGG16

Basit yapı, anlaşılması kolay, pedagojik

Genel Kullanım

ResNet50

Dengeli performans, en popüler seçim

Mobil/Edge

MobileNetV2

Hafif, hızlı, düşük kaynak tüketimi

Fine-tuning vs Feature Extraction

Feature Extraction

Strateji:

- Önceden eğitilmiş modelin TÜM katmanlarını dondur
- Sadece en üstteki sınıflandırma katmanlarını eğit
- Model, sabit bir özellik çıkarıcı olarak kullanılır

Ne Zaman?

✓ Çok az veri olduğunda (100–500 görüntü)
✓ Hedef veri seti, kaynak veriye benzediğinde
✓ Hızlı sonuç istendiğinde
✓ GPU kaynağı sınırlı olduğunda

Fine-tuning

Strateji:

- Önce Feature Extraction ile başla (birkaç epoch)
- Sonra bazı üst katmanları aç (unfreeze)
- Düşük learning rate ile tüm modeli eğit
- Temel özellikleri koruyarak yeni özellikleri öğret

Ne Zaman?

✓ Daha fazla veri olduğunda (1000+)
✓ Hedef veri seti, kaynaktan farklı olduğunda
✓ Maksimum doğruluk hedeflendiğinde
✓ Eğitim süresi sorun olmadığında

Karar Ağacı

Az veri + benzer domain → Feature Extraction | Çok veri + farklı domain → Fine-tuning | Emin değilsen → Feature Extraction ile başla, sonuç yetersizse Fine-tune

MobileNet ile Kedi/Köpek Sınıflandırma

Az veri (2000 görüntü) ile %95+ doğruluk!
MobileNetV2 + Feature Extraction stratejisi.

2K
Görüntü

95%+
Doğruluk

~2dk
Eğitim

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras import layers, models

# 1. Önceden eğitilmiş modeli yükle (ImageNet ağırlıkları)
base_model = MobileNetV2(
    weights='imagenet',
    include_top=
False,  # Sınıflandırma kafası YOK
    input_shape=(160, 160, 3)
)

# 2. Tüm katmanları dondur
base_model.trainable = False

# 3. Yeni sınıflandırma kafası ekle
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1, activation='sigmoid')
])

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

Data Augmentation — Veri Çoğaltma

Neden Veri Çoğaltma?

Az veri ile eğitim yaparken overfitting kaçınılmazdır. Data Augmentation, mevcut görüntülere rastgele dönüşümler uygulayarak sanal olarak veri setini büyütür. Model farklı açılardan aynı nesneyi görmeyi öğrenir.

```
from tensorflow.keras.preprocessing\
  .image
import ImageDataGenerator

datagen = ImageDataGenerator(
    rotation_range=20,
    horizontal_flip=True,
    zoom_range=0.2,
    width_shift_range=0.1)
```

Rotation

±20° rastgele döndürme. Nesne açısı değişse de tanınmalı.

`rotation_range=20`

Horizontal Flip

Yatay aynalama. Sol-sağ simetrisi olan nesneler için ideal.

`horizontal_flip=True`

Zoom

±%20 yakınlaştırma. Nesne boyutu değişse de tanınmalı.

`zoom_range=0.2`

Shift

Yatay/dikey kaydırma. Nesne konumu değişse de tanınmalı.

`width_shift_range=0.1`

Model Değerlendirme — Accuracy/Loss Grafikleri

BÖLÜM 3

Accuracy (Doğruluk) Grafiği

- Train accuracy sürekli artar
- Validation accuracy bir noktada düzleşir
- İkisi arasındaki fark KÜÇÜK → İyi model
- İkisi arasındaki fark BÜYÜK → Overfitting!
- Val accuracy düşmeye başlarsa eğitimi durdur

Loss (Kayıp) Grafiği

- Train loss sürekli düşer
- Validation loss bir noktada düzleşir
- Val loss artmaya başlarsa → Overfitting!
- Train loss = Val loss → İdeal durum
- EarlyStopping callback ile otomatik durdur

Overfitting Tespiti ve Çözümleri

1 Data Augmentation

Veri çeşitliliğini artırarak modelin ezberleme yerine genellemesini sağlar

2 Dropout

Eğitimde rastgele nöronları kapatarak bağımlılıkları kırar (0.2–0.5)

3 EarlyStopping

Val loss artmaya başlayınca eğitimi otomatik durdurur (patience=5)

4 Regularization

L2 regularization ile büyük ağırlıkları cezalandırır, modeli basitleştirir

Gerçek Dünya: Maske Tespiti Uygulaması

COVID-19 Maske Tespiti — Transfer Learning ile

Gerçek bir pandemi döneminde ortaya çıkan ihtiyaç: Maskeli ve maskesiz yüzleri ayırt eden bir model. MobileNetV2 + Transfer Learning ile sadece 1000 görüntü kullanarak %98+ doğruluk elde edilebilir.

1

Veri Toplama

Kaggle'dan maskeli/maskesiz yüz görüntüleri indirme
(~1000 görüntü yeterli)

2

Ön İşleme

Yeniden boyutlandırma (224×224)
Normalizasyon (0-1)
Data Augmentation

3

Model Kurulumu

MobileNetV2 (frozen)
+ GlobalAveragePooling
+ Dense(1, sigmoid)

4

Eğitim

Binary crossentropy
Adam optimizer
5 epoch yeterli

5

Değerlendirme

Accuracy: %98+
Confusion Matrix
Sınıf bazlı F1-score

Haftalık Notebook'lar

1
NB

CNN Temel

`hafta12_cnn_temel.ipynb`

CIFAR-10 ile sıfırdan CNN modeli oluşturma. Conv2D, MaxPooling, Flatten ve Dense katmanlarını adım adım ekleme, model.summary() ile parametreleri inceleme, ~70% doğruluk elde etme.

`Conv2D · MaxPooling · Flatten · Dense · model.summary() · history plot`

2
NB

Transfer Learning

`hafta12_transfer_learning.ipynb`

MobileNetV2 ile kedi/köpek sınıflandırma. Feature Extraction stratejisi, katman dondurma, yeni sınıflandırma kafası ekleme, Data Augmentation ile doğruluğu artırma.

`MobileNetV2 · Feature Extraction · Fine-tuning · ImageDataGenerator`

3
NB

Maske Tespiti

`hafta12_maske_tespiti.ipynb`

Gerçek dünya uygulaması: COVID-19 maske tespiti. Kaggle veri seti, ön işleme pipeline, MobileNetV2 ile Transfer Learning, confusion matrix ve sınıf bazlı metrikler.

`Kaggle veri · Pipeline · Confusion Matrix · Classification Report`

Ödev ve Kaynaklar

Haftalık Ödev

1. Kendi veri setinizi hazırlayın: En az 2 sınıf, sınıf başına 200+ görüntü (Kaggle veya Google Images)
2. Transfer Learning ile sınıflandırma modeli kurun: MobileNetV2 veya ResNet50 kullanarak Feature Extraction uygulayın
3. Data Augmentation ekleyin: rotation, flip, zoom, shift dönüşümleri ile veri çeşitliliğini artırın
4. %90+ doğruluk hedefleyin: Accuracy/Loss grafiklerini çizin, overfitting varsa önlem alın
5. Confusion Matrix ve Classification Report oluşturun: Sınıf bazlı precision, recall, F1-score raporlayın

Faydalı Kaynaklar

TensorFlow CNN Eğitimi
tensorflow.org/tutorials/images/cnn

Transfer Learning Guide
tensorflow.org/tutorials/images/transfer_learning

Keras Applications
keras.io/api/applications/
(tüm önceden eğitilmiş modeller)

Kaggle Datasets
kaggle.com/datasets
(cats-vs-dogs, face-mask)

CS231n (Stanford)
cs231n.stanford.edu
(CNN teorisi derinlemesine)

Teslim: Hafta 13 dersi öncesi · Jupyter Notebook (.ipynb) + PDF rapor · Google Colab linki de kabul edilir

Hafta 12 — Özet

1. CNN, görüntü verisinde uzamsal ilişkileri öğrenen özel bir derin öğrenme mimarisidir
2. Conv2D → MaxPool → Flatten → Dense pipeline'ı ile özellik çıkarımı ve sınıflandırma yapılır
3. Transfer Learning, büyük modellerin bilgisini küçük veri setlerine aktarmanın en etkili yoludur
4. MobileNetV2 gibi hafif modeller, az veri ile bile %95+ doğruluk sağlar
5. Data Augmentation + EarlyStopping + Dropout: overfitting'e karşı üçlü kalkan

"Görmek, inanmaktır. Bilgisayarların görmesini sağlamak ise mühendisliktir."

Dr. Murat Altun · Veri Bilimi ve Yapay Zekâ Eğitimi · 2026