# Örüntü Tanıma Final notları

**6.pdf**

**🧩 1. Kümeleme (Clustering)**

**📌 Genel Tanım**

* **Etiketsiz (etiketlenmemiş) verilerle** çalışılır.
* **Denetimsiz öğrenme** (unsupervised learning) yaklaşımıdır.
* Verilerin tespitinden çok **gruplandırılması** amaçlanır.

**📌 Ne Zaman Kullanılır?**

* **Müşteri segmentasyonu**
* **Anomali tespiti** (ör. sahtekârlık)
* **Veri yönlendirme** (routing)

**🔹 K-means Algoritması**

* Verileri **k** adet kümeye ayırır.
* Rastgele seçilen merkez noktalarına göre benzerlik baz alınarak veri gruplandırılır.
* Her iterasyonda merkezler yeniden hesaplanır.
* Belirli bir kritere ulaşıldığında (örneğin değişim kalmadığında) algoritma durur.

📍 **Kullanım durumu:**

* Veri kümen sabit ve önceden belirli (örneğin k = 3 gibi).
* Hızlı ve büyük veri setleri için uygundur.

**🔹 Hiyerarşik Kümeleme (ve Dendogram)**

* Veriler arasında **ağaç yapısı (dendogram)** ile küme ilişkisi kurulur.
* Her veri önce kendi kümesi sayılır.
* En yakın kümeler adım adım birleştirilir (alt kümeden üst kümeye).
* Tüm veriler birleşene kadar devam eder.

📍 **Kullanım durumu:**

* Küme sayısı **önceden bilinmiyorsa**.
* Veri setinin yapısı anlaşılmak isteniyorsa.
* Daha küçük veri setleri için daha uygundur (çünkü hesaplama maliyeti yüksektir).

**📉 2. PCA (Principal Component Analysis)**

**📌 Genel Tanım**

* **Boyut indirgeme** yöntemidir.
* Verideki **maksimum varyansı koruyarak** daha az boyutlu temsiller elde edilir.
* Özellikle **yüksek boyutlu verileri 2D/3D’ye düşürerek görselleştirme** imkânı sunar.

📍 **Ne Zaman Kullanılır?**

* Görselleştirme amacıyla.
* **Görüntü tanıma** (ör. yüz tanıma - Eigenfaces)
* **Veri sıkıştırma**.
* Özellikle **yüksek korelasyonlu değişkenler** varsa, bu değişken sayısını azaltmak için.

📌 Örnek:

* X ve Y değişkenleri arasında Y = 2X gibi bir ilişki varsa, aslında tek değişken yeterlidir.

**🧠 3. Pekiştirmeli Öğrenme (Reinforcement Learning)**

**📌 Genel Tanım**

* Ajan (agent), **çevre ile etkileşime girerek** öğrenir.
* Aksiyonlara göre ödül veya ceza alır.
* **Hedef: Kümülatif ödülü maksimize etmek**.

📍 **Kullanım durumu:**

* Kararların **sıralı etkiler** oluşturduğu senaryolarda.
* **Robotik** (ör: yoldan çıkmadan gitmek)
* **Oyunlar** (ör: satranç, Go)
* **Otonom sistemler**, adaptif sistemler

**🔄 Pekiştirmeli Öğrenme Döngüsü**

1. Agent (ajan)
2. Environment (çevre)
3. Action (aksiyon)
4. Observation (gözlem)
5. Reward (ödül)
6. Total reward (toplam ödül)

Anlık değil, **gelecekteki ödülleri** maksimize eden stratejiler aranır. Bazen kısa vadeli ödüller feda edilir.

metin, yazı tipi, ekran görüntüsü, beyaz içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

| **Yöntem** | **Amaç** | **Ne Zaman Kullanılır?** |
| --- | --- | --- |
| K-means | K kadar kümeye bölmek | K sabitse, hızlı ve büyük veri için |
| Hiyerarşik Kümeleme | Veri yapısını anlamak, dendogram üretmek | K bilinmiyorsa, küçük veri setlerinde |
| PCA | Boyut indirgeme ve görselleştirme | Yüksek boyutlu ve korelasyonlu verilerde |
| Pekiştirmeli Öğrenme | Etkileşimli öğrenme, ödül temelli | Oyun, robotik, otonom sistemler, sıra bağımlı kararlar |

**7.1 pdf**

**🧠 1. Sinir Ağları (Neural Networks)**

**📌 Perceptron**

* **En temel yapay sinir ağı birimi**.
* Girdi değerleri ağırlıklarla çarpılır, toplanır, bir eşik değerle karşılaştırılır ve çıktı üretilir.
* Yalnızca **doğrusal olarak ayrılabilen** problemleri çözebilir.

📍 **Ne zaman kullanılır?**

* Basit sınıflandırma problemleri.
* Derin yapılar gerektirmeyen, az boyutlu veri setleri.

**🔁 Backpropagation (Geri Yayılım)**

* Sinir ağı **hataları öğrenmek için geri yayılım** yapar.
* Çıktıdan hesaplanan **kayıp (loss)** geri beslenir.
* **Ağırlıklar güncellenerek** öğrenme sağlanır.

📍 **Ne zaman önemlidir?**

* Tüm modern sinir ağı modellerinin temelidir.
* Derin yapılar için öğrenmeyi mümkün kılar.

**🌊 2. Deep Learning (Derin Öğrenme)**

**📌 Genel Tanım**

* Özellik mühendisliğini azaltır (yani veriden özellik çıkarma işini model öğrenir).
* Yeterli veri, çıktı, kayıp fonksiyonu ve model yapısı ile yüksek başarı sağlar.

📍 **Ne zaman kullanılır?**

* Özellikle **görüntü, ses, doğal dil** gibi karmaşık verilerde.
* Kendi kendine özellik çıkarımı gerektiren durumlarda.

**📈 Deep Learning’in Yükseliş Sebepleri**

* **Veri miktarındaki artış**
* **GPU / TPU** gibi güçlü donanımların yaygınlaşması
* **Benchmark’lar ve araçlar** (TensorFlow, PyTorch, Hugging Face vs.)

**♟ Uzay Küçültme & Karmaşıklık**

* Deep learning modelleri büyük uzaylarda zorlanabilir.
* Bu yüzden karmaşık problemler (örneğin **Go**, **Protein katlanma (AlphaFold)**) için sadeleştirme ve **boyut indirgeme** yapılır (örneğin 3D → 2D).

📍 **Uygulama Alanları:**

* Strateji oyunları
* Biyoinformatik
* Haritalama ve görselleştirme

**🔷 Convolutional Neural Network (CNN)**

**📌 Açıklama:**

* **Görüntü işlemede en yaygın kullanılan sinir ağıdır.**
* Evrişim (convolution) katmanları sayesinde uzamsal (görsel) ilişkileri öğrenir.

**🧠 Kullanıldığı Yerler:**

* Görüntü sınıflandırma (ImageNet, ResNet)
* Nesne tespiti (YOLO, Faster R-CNN, Mask R-CNN)
* Görüntü segmentasyonu
* Video analizi

**🔷 Recurrent Neural Network (RNN)**

**📌 Açıklama:**

* **Zaman bağımlı veriler** için geliştirilmiş ağlardır.
* Girdi ve çıktı zaman dizilerine (sequence) bağlıdır.
* Bellek özelliği taşır (önceki adımları hatırlar).

**🧠 Kullanıldığı Yerler:**

* Doğal dil işleme (metin tahmini, çeviri)
* Zaman serisi tahmini (ör: finansal veriler)
* Ses işleme (konuşma tanıma)

Alt türleri: LSTM, GRU, Bidirectional RNN, Seq2Seq

**🔷 Generative Modeller**

**📌 Açıklama:**

* **Yeni veri örnekleri üretmeyi** amaçlayan modellerdir.
* Özellikle **yapay görüntü**, **ses** ve **metin üretimi** için kullanılır.

**🧠 Örnekler:**

* **GAN (Generative Adversarial Network)**: Gerçekçi görseller üretir.
* **VAE (Variational Autoencoder)**: Veriden öğrenerek yeni örnekler türetir.
* **RBM / DBM**: Sık kullanılan generatif yapay sinir ağı türlerindendir.

📍 **Kullanım Alanları:**

* Görüntü üretimi (ör: yüz oluşturma)
* Veri artırma
* Yaratıcı yapay zekâ sistemleri (müzik, resim üretimi)

**🟧 Deep Belief Network (DBN)**

**📌 Açıklama:**

* Çok katmanlı olasılıksal sinir ağıdır.
* Özellikle **özellik çıkarımı (feature extraction)** için kullanılır.
* Genellikle **DBM (Deep Boltzmann Machine)** ve **Stacked Autoencoder** bileşenlerinden oluşur.

📍 **Kullanım Alanı:**

* Boyut indirgeme
* Anomali tespiti
* Özellik öğrenimi

**🟧 Transformer’lar**

**📌 Açıklama:**

* **Sıralı veri işlemek için geliştirilmiş modern bir mimari.**
* "Attention" mekanizması sayesinde çok uzun bağlamları etkili şekilde işler.

**🧠 Örnekler:**

* GPT, BERT, T5, RoBERTa, XLNet

📍 **Kullanım Alanları:**

* Makine çevirisi
* Metin üretimi
* Soru-cevap sistemleri
* Kodlama (ör. Codex)

Transformer mimarisi, günümüzde dil modellerinin temelidir.

**🟧 Autoencoder**

**📌 Açıklama:**

* Girdiyi sıkıştırıp (encode), tekrar üretmeyi (decode) öğrenen bir ağdır.
* Genellikle **boyut indirgeme** ve **gürültü temizleme** için kullanılır.

**🧠 Türleri:**

* VAE (Variational Autoencoder)
* DAE (Denoising Autoencoder)
* Sparse, Contractive Autoencoder

📍 **Kullanım Alanları:**

* Anomali tespiti
* Görüntü gürültü giderme
* Özellik çıkarımı

| **Algoritma** | **Ne Zaman Kullanılır?** |
| --- | --- |
| CNN | Görüntü, video, uzamsal veri analizi |
| RNN | Zaman serisi, dil verisi, ardışık işler |
| Generative | Yeni veri üretimi, yaratıcı işler, artırılmış veri |
| DBN | Özellik çıkarımı, ön eğitim, boyut indirgeme |
| Transformer | Metin işleme, doğal dil anlama/üretme |
| Autoencoder | Veri sıkıştırma, anomali tespiti, gürültü temizleme |

**📊 3. Veri Toplama & Önişleme**

**🔹 Veri Kaynakları**

* [HuggingFace Datasets](https://huggingface.co/datasets)
* [Kaggle Datasets](https://www.kaggle.com/datasets/)
* [Google Dataset Search](https://datasetsearch.research.google.com/)
* [Papers with Code](https://paperswithcode.com/datasets)

📍 **Ne zaman?**

* Model eğitmeden önce veri gerekiyorsa buradan toplanır.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**🔧 Önişleme (Preprocessing)**

* Verinin model için **temizlenmesi, dönüştürülmesi ve uygun hale getirilmesi**.
* Eksik veri, normalizasyon, encoding vs. işlemleri içerir.

📍 **Ne zaman önemli?**

* Modelin başarısı büyük ölçüde önişlemeye bağlıdır.
* Kirli veri → düşük performans.

**🏋️ 4. Eğitim Aşamaları**

**🟩 Sıfırdan Eğitim (From Scratch)**

* Model baştan eğitilir.
* Çok veri, çok zaman ve güçlü donanım gerekir.

📍 **Ne zaman?**

* Yeni bir problem için özel model gerekiyorsa.
* Elinde yeterli veri ve kaynak varsa.

**🟦 Finetuning (İnce Ayar)**

* Mevcut bir modeli kendi verinle biraz daha eğitmek.
* **PEFT** ve **LoRA** gibi yöntemler ile daha az parametreyle ince ayar yapılabilir.

📍 **Ne zaman?**

* Hazır bir model var, kendi veri setine uyarlamak istiyorsan.
* Verin azsa, eğitim süresini kısaltmak istiyorsan.

**🟨 One-shot / Zero-shot Learning**

* **One-shot:** Tek bir örnekle öğrenme
* **Zero-shot:** Hiç örnek görmeden görevi yapma (genel bilgiye dayalı)

📍 **Ne zaman?**

* Veri çok azsa veya hiç yoksa.
* Büyük, genel modeller kullanıldığında (örneğin GPT, BERT).

| **Yöntem** | **Ne Zaman Kullanılır?** |
| --- | --- |
| Perceptron | Basit doğrusal sınıflandırmalar |
| Backpropagation | Tüm modern sinir ağları için temel öğrenme yöntemi |
| Deep Learning | Karmaşık veriler, özellik çıkarımı gereken durumlar |
| Uzay Küçültme | Çok büyük problem alanlarında (ör: AlphaFold, strateji oyunları) |
| Veri Toplama | Model eğitimi öncesinde uygun veri setleri gerekiyorsa |
| Önişleme | Kirli ya da uygunsuz veri ile çalışırken |
| Sıfırdan Eğitim | Yeterli veri, zaman ve kaynak varsa |
| Finetuning / LoRA / PEFT | Önceden eğitilmiş modellerle çalışma durumlarında |
| One-shot / Zero-shot | Az veriyle veya hiç veri olmadan genelleme gereken durumlar |

**7.2 Pdf**

**🧠 1. Derin Öğrenme ve Temelleri**

**Açıklama:**

* Derin öğrenme, veriyle öğrenen sistemlerdir.
* Geleneksel yöntemler: **RBM, DBN, Hopfield, SOM** gibi klasik yaklaşımlar.

**Ne zaman kullanılır?**

* Geleneksel yapay zeka tarihçesini anlamak istiyorsan veya alternatif mimariler araştırıyorsan.

**📌 2. Uygulama Alanları**

**Açıklama:**

* Görüntü işleme
* Doğal dil işleme
* Ses ve sinyal işleme
* Kontrol ve otonom sistemler

**Ne zaman kullanılır?**

* Gerçek dünyadaki uygulamalarda hangi sinir ağı yapısını seçeceğini belirlerken.

**📈 3. Derin Öğrenmenin Yaygınlaşma Sebepleri**

**Faktörler:**

* Büyük veri setlerinin ulaşılabilir olması
* Hesaplama gücünün artması (GPU vb.)
* Kolay model üretimi ve deneme-yanılma imkanı
* WEKA gibi araçlarla görsel modelleme

**Ne zaman kullanılır?**

* Neden günümüzde derin öğrenmenin bu kadar popüler olduğunu anlamak için.

**❌ 4. Derin Öğrenmede Başarı Mitleri**

* Daha fazla veri ≠ daha iyi model
* Daha büyük model ≠ daha iyi sonuç
* Ham veri ≠ kaliteli sonuç

**Ne zaman kullanılır?**

* Model başarısızsa, sebebin "daha fazla veri/model" değil "uygun yapı ve temizlik" olduğunu anlamak için.

**🔗 5. Yapay Sinir Ağı (YSA) Temelleri**

**Yapılar:**

* Nöron (ax + b)
* Ağırlıklar (w)
* Aktivasyon fonksiyonları (sigmoid, ReLU vs.)

**Ne zaman kullanılır?**

* Sinir ağının nasıl çalıştığını, veri nasıl işlendiğini öğrenmek için.

**⚙️ 6. Lineer ve Non-Lineer Katmanlar**

* Lineer: ax
* Affine: ax + b
* Non-lineer: Sigmoid, Tanh, ReLU, Leaky ReLU

**Ne zaman kullanılır?**

* Kompleks problemleri çözmek için **non-lineer** yapılar şart. Sadece lineer yapı yeterli değildir.

**🔄 7. Öğrenme – Ağırlık Güncelleme**

* Kayıp fonksiyonu: Binary için Cross-Entropy
* Çok sınıflı için: Softmax + Cross-Entropy

**Ne zaman kullanılır?**

* Model eğitilirken "hatalar nasıl ölçülüyor?" sorusunu yanıtlamak için.

**🧱 8. Katmanlı Derin Yapılar (N-A-L)**

* N: Nöron
* A: Aktivasyon
* L: Kayıp
* V, T, P: Girdi, tahmin, sonuç katmanları

**Ne zaman kullanılır?**

* Derin öğrenme modelini katmanlı olarak planlarken.

**🧭 9. Boyut Arttırımı ve Karmaşıklık**

* Hyperplane’lerle ayrım
* Derinlik arttıkça karmaşıklık da artar

**Ne zaman kullanılır?**

* Veriler basit şekilde ayrılamıyorsa boyut arttırımı ile çözüm aranır.

**🔧 10. Model Özellikleri ve Pratik Bilgiler**

**Öne çıkan ipuçları:**

* Sigmoid yerine ReLU genelde tercih edilir
* Gradient Descent kullanılır
* Adam optimizer genelde varsayılan olarak işe yarar
* Overfit’i azaltmak için:
  + L1/L2 Regularizasyon
  + Dropout
  + Early stopping
  + Noise ekleme

**Ne zaman kullanılır?**

* Model aşırı öğrenmişse (training accuracy yüksek, test accuracy düşük)
* Eğitim başarısızsa ya da çok yavaşsa

**🧪 11. Model Geliştirme İpuçları**

* Ağırlıklar 0 olmamalı
* Çok büyük değerler de problem yaratır
* Eğitim başarısızsa:
  + Veri veya tanım hatalı olabilir
  + Accuracy yeterli metrik olmayabilir

**Ne zaman kullanılır?**

* Eğitim sırasında "neden öğrenmiyor?" sorusuna yanıt ararken.

**🔍 12. Model Takibi ve Parametre Ayarı**

* Ağırlık normları izlenmeli
* Şekil (shape) uyuşmazlığına dikkat
* Aynı anda yalnızca 1 parametre değiştirilmeli (kontrollü deney)

**Ne zaman kullanılır?**

* Model tuning sürecinde etkili deney tasarımı yaparken.

**8.pdf**

**🖼️ 1. Görüntü Tanıma – İşleme**

**Nerede kullanılır?**

* Sağlıkta hastalık teşhisi
* Otonom araçlarda çevre algılama
* Güvenlik sistemlerinde yüz, hareket ve obje tanıma
* OCR (Optik Karakter Tanıma) ve el yazısı okuma

**📂 2. Görüntü Sınıflandırma**

**Açıklama:**

* Eski yöntemler: manuel özellik çıkarımı, kural tabanlı sistemler
* Yeni: derin öğrenme sayesinde otomatik özellik çıkarımı

**Ne zaman kullanılır?**

* Büyük veri kümesinde (örneğin ImageNet) nesne sınıflandırma yapılmak istendiğinde

**🧮 3. Görüntülerin Sinir Ağına Verilmesi**

**Sorun:**

* Görseller matris (2D), ama sinir ağı vektör (1D) ile çalışır.

**Neden dikkat edilmeli?**

* Matris → vektör dönüşümde lokalite (komşuluk bilgisi) kaybolur.
* Bu, görüntüdeki desenlerin anlaşılmasını zorlaştırır.

**🧠 4. CNN ile Lokal Özelliklerin Korunması**

**Nasıl çözülür?**

* **Kernel (filtre)** kullanılarak yerel (lokal) özellikler çıkarılır.
* Her kernel → farklı bir özelliği öğrenir (kenar, doku vb.)
* Çıktılar → **feature map** olarak kaydedilir.

**Ne zaman kullanılır?**

* Görsellerdeki detayları yakalamak ve uzamsal bilgiyi korumak istiyorsan.

**🔧 5. Kaynak Kısıtları ve Optimizasyon**

**Teknikler:**

* **Padding**: Çıktı boyutunu korumak
* **Stride**: Kaydırma miktarını artırmak (bilgi kaybı riskli)
* **Dilated Convolution**: Bilgi kaybını azaltmak için sıçramalı kernel kullanımı

**🧊 6. 3B Görüntüler ve Kanal Kullanımı**

**Sorun:**

* Renkli görüntüler 3 boyutlu (RGB), her kanal ayrı bilgi taşır.

**Çözüm:**

* **Depthwise Convolution**: Her kanala ayrı kernel uygula → daha verimli

**🧹 7. Pooling Katmanı**

**Amaç:**

* Boyutu küçültmek, gereksiz bilgiyi azaltmak

**Türler:**

* **Max pooling**: En büyük değeri alır
* **Avg pooling**: Ortalama alır
* **Min pooling**: En küçük değeri alır

**🧩 8. En Temel CNN Mimarisi**

* C: Convolution
* P: Pooling
* L: Loss
* N: Nöron
* A: Aktivasyon
* V: Girdi
* P: Çıktı

**🧱 9. Örnek CNN Modelleri**

| **Model** | **Özellikler / Kullanım** |
| --- | --- |
| **LeNet-5** | İlklerden, basit görevler |
| **AlexNet** | GPU ile devrim yarattı |
| **GoogLeNet** | İnception bloklarıyla verimli |
| **VGG** | Daha derin ama büyük filtre |
| **ResNet** | Residual bağlantılar ile çok derin modeller |
| **UNet** | Segmentasyon için kullanılır |
| **MobileNet** | Mobil cihazlar için optimize |
| **DenseNet** | Katmanlar arası bağlantılar |
| **AmoebaNet** | AutoML ile elde edilen model |

**🎯 10. Object Detection (Nesne Tespiti)**

* **RCNN**: Bölge önerme + sınıflandırma
* **Mask R-CNN**: Segmentasyon + tespit bir arada
* **UNet**: Görüntü segmentasyonu için ideal

**🧠 11. İpuçları ve Uyarılar (Tips & Tricks)**

**Dikkat edilmesi gerekenler:**

* Her katman aslında bir sınıflayıcıdır.
* Derinleştirmek başarıyı garanti etmez.
* Ağırlıklar çok küçük veya uniform olmamalı.
* **Exploding/Vanishing Gradient** riski: çözüm → **Batch Normalization**
* Karmaşıklık artarsa:
  + Depthwise convolution kullan
  + Her kanala ayrı kernel uygula
* Kernel karmaşıklığı fazla olursa, SVM benzeri yapıdan kaçın.

**Uyg6**

Bu Jupyter notebook'u Python'da görüntü işleme konusunu ele alıyor. İşte konu bazlı özet:

**🎯 Hangi Kütüphaneyi Ne Zaman Kullanmalı?**

**Pillow (PIL) - Basit İşlemler İçin**

* ✅ **Ne zaman:** Fotoğraf düzenleme, temel transformasyonlar, web için görüntü hazırlama
* ✅ **Avantajları:** Kolay kullanım, hızlı öğrenme
* ❌ **Sınırları:** Gelişmiş görüntü analizi yapamaz

**OpenCV - Profesyonel İşlemler İçin**

* ✅ **Ne zaman:** Bilgisayarla görme, video işleme, kenar tespiti, yüz tanıma
* ✅ **Avantajları:** Güçlü algoritmalar, gerçek zamanlı işleme
* ❌ **Zorlukları:** Daha karmaşık, öğrenme eğrisi yüksek

**📋 Konu Bazlı Özet**

**1. Temel İşlemler (Pillow)**

* **Görüntü açma/kaydetme:** Image.open(), save()
* **Boyutlandırma:** resize() - kesin boyut, thumbnail() - oran koruyarak
* **Format dönüştürme:** convert("L") gri ton, convert("RGB") renkli

**2. Görüntü Transformasyonları**

* **Döndürme:** rotate(45) - 45 derece çevir
* **Kırpma:** crop((x1,y1,x2,y2)) - belirli alanı kes
* **Çevirme:** transpose() - yatay/dikey çevir

**3. Görsel Efektler**

* **Filtreler:** Bulanıklaştırma, kenar bulma (ImageFilter)
* **Renk ayarları:** Parlaklık, kontrast (ImageEnhance)
* **Özel efektler:** Sepia filtresi (manuel kodlama)

**4. Renk Kanalları**

* **RGB ayrıştırma:** split() ile R,G,B kanallarını ayır
* **YCbCr dönüşümü:** Parlaklık ve renk bilgilerini ayrı işle
* **Kanal manipülasyonu:** Tek renk kanalıyla çalışma

**5. Çizim ve Metin**

* **Şekil çizme:** Dikdörtgen, çember (ImageDraw)
* **Metin ekleme:** Görüntü üzerine yazı yazma
* **Görüntü birleştirme:** Yan yana veya üst üste yapıştırma

**6. OpenCV ile Gelişmiş İşlemler**

* **Kenar tespiti:** Canny algoritması
* **Bulanıklaştırma:** Gaussian blur
* **Gerçek zamanlı gösterim:** imshow(), waitKey()

**🚀 Pratik Kullanım Senaryoları**

| **İhtiyaç** | **Çözüm** | **Kütüphane** |
| --- | --- | --- |
| Web sitesi için foto resize | thumbnail() | Pillow |
| Instagram filtresi | Sepia, renk ayarları | Pillow |
| Güvenlik kamerası analizi | Kenar tespiti, hareket | OpenCV |
| E-ticaret ürün fotoğrafı | Boyutlandırma, kalite | Pillow |
| Yüz tanıma sistemi | Cascade classifiers | OpenCV |
| Fotoğraf galerisi düzenleme | Rotate, flip, crop | Pillow |

Bu notebook, görüntü işleme dünyasına adım atmak isteyenler için mükemmel bir başlangıç rehberi. Temel işlemlerden başlayıp profesyonel düzeye kadar ilerliyor.

CNN EĞİTİM MODEL DÖNGÜ

**PyTorch CNN Projesi - Önemli Notlar**

**🧠 Model Oluşturma (SimpleCNN)**

**Model Mimarisi**

Girdi (32x32x3 renkli görüntü)

↓

📊 Özellik Çıkarma Katmanları:

• Conv2d (3→16 kanal) + ReLU + MaxPool (32x32→16x16)

• Conv2d (16→32 kanal) + ReLU + MaxPool (16x16→8x8)

↓

🎯 Sınıflandırma Katmanları:

• Flatten (32x8x8 = 2048 boyutuna düzleştir)

• Linear (2048→128) + ReLU

• Linear (128→sınıf\_sayısı)

↓

Çıktı (sınıf skorları)

**Katman Açıklamaları**

* **Conv2d**: Görüntüdeki desenleri algılar (kenar, şekil vs.)
* **ReLU**: Negatif değerleri 0 yapar, modele doğrusal olmayanlık katar
* **MaxPool2d**: Görüntü boyutunu küçültür, önemli bilgileri korur
* **Linear**: Tam bağlantılı katman, son karar verme aşaması

**🏃‍♂️ Eğitim Döngüsü (Training Loop)**

**Ana Eğitim Fonksiyonu**

def train(model, device, loader, criterion, optimizer, epoch):

model.train() # Eğitim moduna geç

for batch\_idx, (data, target) in enumerate(loader):

# 1️⃣ Veriyi GPU/CPU'ya taşı

data, target = data.to(device), target.to(device)

# 2️⃣ Gradyanları sıfırla

optimizer.zero\_grad()

# 3️⃣ İleri besle (forward pass)

output = model(data)

# 4️⃣ Kaybı hesapla

loss = criterion(output, target)

# 5️⃣ Geri yayılım (backward pass)

loss.backward()

# 6️⃣ Ağırlıkları güncelle

optimizer.step()

**Doğrulama Fonksiyonu**

def validate(model, device, loader, criterion):

model.eval() # Değerlendirme moduna geç

with torch.no\_grad(): # Gradyan hesaplama

# Tahmin yap ve doğruluğu ölç

pred = output.argmax(dim=1)

correct += pred.eq(target).sum().item()

**⚙️ Temel Ayarlar**

**Hyperparametreler**

* **Batch Size**: 32 (aynı anda işlenen görüntü sayısı)
* **Epochs**: 10 (veri kümesini kaç kez gezeceği)
* **Learning Rate**: 0.001 (öğrenme hızı)

**Veri Hazırlığı**

transform = transforms.Compose([

transforms.ToTensor(), # Görüntüyü tensor'e çevir

transforms.Normalize(...) # Piksel değerlerini normalize et

])

**Model Bileşenleri**

model = SimpleCNN(num\_classes) # Model oluştur

criterion = nn.CrossEntropyLoss() # Kayıp fonksiyonu (sınıflandırma için)

optimizer = optim.Adam(model.parameters(), lr=0.001) # Optimizatör

**🔄 Ana Eğitim Döngüsü**

for epoch in range(1, epochs + 1):

train(model, device, train\_loader, criterion, optimizer, epoch) # Eğit

validate(model, device, val\_loader, criterion) # Doğrula

**💾 Model Kaydetme**

torch.save(model.state\_dict(), 'simple\_cnn.pth') # Eğitilmiş ağırlıkları kaydet

**🎯 Özet**

Bu kod 32x32 piksel renkli görüntüleri (CIFAR gibi) sınıflandırmak için basit bir CNN eğitir. Model önce görüntülerden özellik çıkarır, sonra bu özellikleri kullanarak sınıflandırma yapar. Eğitim sırasında model sürekli kendini iyileştirir ve sonunda eğitilmiş haliyle kaydedilir.

**🔢 Tensor ve Veri Yapıları - Sözel Açıklama**

**Skalar (0 boyut)**: Tek bir bilgi parçası. Mesela yaşın "25", boy "1.75" veya sıcaklık "36.5" gibi. Sadece bir sayı, başka hiçbir boyutu yok.

**Vektör (1 boyut)**: Bir liste gibi düşün. Mesela bir öğrencinin notları: Matematik 85, Fizik 90, Kimya 78. Bu üç not yan yana dizilmiş [85, 90, 78]. Ya da bir evin özellikleri: oda sayısı, yaşı, metrekaresi. Hep tek sıra halinde.

**Matris (2 boyut)**: Excel tablosu gibi. Satırları ve sütunları var. Mesela bir sınıftaki tüm öğrencilerin notları:

Mat Fiz Kim

Ahmet 85 90 78

Ayşe 92 88 95

Mehmet 76 82 89

Her satır bir öğrenci, her sütun bir ders.

**Tensor (3+ boyut)**: Artık karmaşık hale geliyor. Mesela bir renkli fotoğraf:

* Yükseklik (pikseller yukarı-aşağı)
* Genişlik (pikseller sağa-sola)
* Renk (kırmızı, yeşil, mavi)

Ya da bir video düşün: fotoğraflar + zaman boyutu. Yani 4 boyutlu tensor.

**Özetle**: Skalar bir nokta, vektör bir çizgi, matris bir düzlem, tensor ise 3D küp ve daha fazlası!

**MODEL YAŞAM DÖNGÜSÜ**

**Makine Öğrenmesi Model Yaşam Döngüsü - Kısa Versiyon**

**🎯 1. Problem Tanımla**

"Ne yapmak istiyorum?" - Spam tespiti, fiyat tahmini vs.

**📊 2. Veri Topla**

Mümkün olduğunca çok ve kaliteli veri bul

**🔍 3. Veriyi İncele**

* Grafik çiz, istatistik çıkar
* Eksik/hatalı değerleri tespit et

**🧹 4. Veriyi Temizle**

* Eksik değerleri doldur
* Aykırı değerleri düzelt
* Sayısal hale getir

**⚙️ 5. Özellik Mühendisliği**

* Yeni özellikler yarat
* Gereksizleri çıkar

**📈 6. Model Seç ve Eğit**

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

**📋 7. Test Et**

* Train (%70), Validation (%15), Test (%15)
* Accuracy, precision gibi metriklere bak

**⚡ 8. Hiperparametre Ayarla**

Learning rate, tree depth gibi ayarları optimize et

**🚀 9. Deploy Et**

API yap, servise çevir, canlıya al

**👀 10. İzle ve Güncelle**

Performansı takip et, gerekirse yeniden eğit

**🔄 Özet: Döngü**

**Veri → Temizle → Eğit → Test → Deploy → İzle → Tekrar**

**Toplam süre**: 2-8 hafta (proje büyüklüğüne göre)

**SINAV SORULARI**

**1. Teorik Bilgi Soruları**

**Muhtemel Sorular:**

* Perceptron neyi çözemez ve neden sınırlıdır?
* Backpropagation nedir, nasıl çalışır?
* CNN ile RNN arasındaki fark nedir? Hangi veri tipleri için kullanılır?
* Transformer mimarisinin avantajı nedir? Neden günümüzde tercih ediliyor?

**2. Görüntü İşleme ve CNN**

**Muhtemel Sorular:**

* Convolution katmanı neden kullanılır? Ne işe yarar?
* Max pooling ile average pooling farkı nedir?
* ResNet’i diğer CNN mimarilerinden ayıran özellik nedir?

**3. PCA (Principal Component Analysis)**

**Muhtemel Sorular:**

* PCA’nın amacı nedir ve ne zaman kullanılır?
* Y = 2X gibi bir örnekte neden tek boyut yeterlidir?

**4. Öğrenme Yöntemleri: Supervised, Unsupervised, Reinforcement**

**Muhtemel Sorular:**

* K-means denetimli midir? Ne zaman tercih edilir?
* Pekiştirmeli öğrenmede ajan nasıl karar verir?
* Reward mekanizması nedir?

**5. Transformer, Autoencoder ve Generative Modeller**

**Muhtemel Sorular:**

* Transformer mimarisi neden LSTM’den daha iyi sonuç verir?
* Autoencoder ile PCA arasında nasıl bir benzerlik vardır?
* GAN nasıl veri üretir? Generator ve Discriminator rolleri nedir?

**6. Eğitim Aşamaları ve Overfitting**

**Muhtemel Sorular:**

* Overfitting nedir ve nasıl önlenir?
* Finetuning ne zaman tercih edilir?

**7. Görüntü İşleme Kütüphaneleri (Pillow, OpenCV)**

**Muhtemel Sorular:**

* Pillow ile bir görseli nasıl griye çevirirsin?
* OpenCV’de kenar tespiti nasıl yapılır?

**SORULAR ve CEVAP FORMATLARI**

**1. (Klasik)**  
PCA (Principal Component Analysis) nedir? Hangi durumlarda ve neden tercih edilir? Örnek vererek açıklayınız.

**2. (Kısa Tanım)**  
K-means algoritmasının temel çalışma adımlarını sırasıyla yazınız.

**3. (Doğru/Yanlış – Açıklamalı)**  
“Perceptron, doğrusal olmayan sınıflandırma problemlerini başarıyla çözer.”  
Bu ifade doğru mu, neden?

**4. (Karşılaştırmalı)**  
CNN ve RNN mimarilerini karşılaştırınız. Hangi tür verilerde hangisi tercih edilir ve neden?

**5. (Açıklamalı Seçmeli)**  
Aşağıdakilerden hangisi overfitting problemini azaltmak için kullanılmaz? Neden?  
A) Dropout  
B) L1/L2 regularizasyon  
C) Daha büyük model kullanmak  
D) Early stopping

**6. (Kavram Tanımı)**  
“Backpropagation” nedir? Sinir ağlarında ne işe yarar?

**7. (Uygulama)**  
Bir RGB görseli CNN ağına vermek istiyorsun. Görseldeki lokal bilgileri (kenar, doku gibi) koruyarak çalışacak bir model nasıl olmalıdır? Hangi katmanları kullanırsın?

**8. (Kısa Açıklama)**  
Transformer mimarisinin temel avantajı nedir? RNN’lere göre farkı nedir?

**9. (Kod veya Açıklama)**  
PyTorch kullanarak aşağıdaki görev için bir yapı kur:  
→ Girdi: 32x32 RGB görsel  
→ Katmanlar: 2 Conv + 2 MaxPool → Flatten → Dense (sınıf sayısı kadar)  
→ Hangi bileşenleri kullanırsın, mimariyi özetle

**10. (Örnekli)**  
Pekiştirmeli öğrenmede bir ajanın çevresiyle etkileşimi nasıl olur? Bir oyun örneği ile açıklayınız.

**1. KOD – CNN Eğitim Döngüsü**  
train.py içinde bir CNN modeli için eğitim döngüsünü (training loop) PyTorch ile yazınız.  
Girdi: RGB 32x32 görseller  
Çıktı: 10 sınıf  
Yapılması gerekenler: Model tanımı, eğitim fonksiyonu, optimizer, loss, forward, backward, step sırası

**2. KAVRAM – Backpropagation**  
Backpropagation nedir? Hangi problemle ilgilenir?  
Bu yöntemde ağırlıklar nasıl güncellenir, hangi kayıp fonksiyonu ile birlikte çalışır?

**3. SÜREÇ – Model Yaşam Döngüsü**  
Bir makine öğrenmesi modelini oluşturmak için aşağıdaki adımları sırasıyla açıklayınız:  
Problem tanımı  
Veri toplama  
Ön işleme  
Özellik mühendisliği  
Model eğitimi  
Test ve değerlendirme  
Deploy ve izleme

**4. KOD – KNN ile Sınıflandırma**  
Scikit-learn kullanarak basit bir KNN modeli kurup fit ve predict işlemlerini gösteriniz.  
Girdi: X, y veri seti  
Amaç: K=3 ile sınıflandırma yapmak

**5. SOZEL – Tensor ve Vektör**  
Tensor nedir? Vektör nedir? Aralarındaki farkları yazınız.  
Hangi yapay öğrenme yöntemlerinde tensor kullanılır? Görsel veri ile nasıl ilişkilidir?

**6. KOD – CNN Model Tanımı**  
Aşağıdaki mimariye uygun basit bir CNN modeli PyTorch ile yazınız:  
Conv2d(3→16) → ReLU → MaxPool  
Conv2d(16→32) → ReLU → MaxPool  
Flatten → Linear(2048→128) → ReLU → Linear(128→10)

**7. KAVRAM – CNN Ne Zaman Kullanılır?**  
CNN modelleri hangi veri türlerinde tercih edilir?  
Bir görüntü sınıflandırma problemi için neden CNN seçilir, avantajı nedir?

**8. UYGULAMA – Iron Value ve Vektör**  
Iron value ve vektör nedir? Nerelerde kullanılırlar?  
Bunlar hangi veri yapısında temsil edilir, öğrenme modelleri bu bilgileri nasıl işler?  
(Hafta2.pdf sayfa 14-15'teki tanımlara bakılarak net sorulmuş gibi düşünebilirsiniz.)

**9. SEÇMELİ – Overfitting Çözümü**  
Aşağıdakilerden hangisi overfitting problemini azaltmak için doğrudan çözüm sağlar?  
A) Öğrenme oranını artırmak  
B) Dropout katmanı eklemek  
C) Daha fazla epoch ile eğitim  
D) Eğitim verisini rastgele karıştırmamak

**10. SOZEL – Her Şey Ne İçin Kullanılır?**  
Aşağıdaki yöntemlerin her biri ne işe yarar, nerede kullanılır?  
Kısa kısa yazınız:  
PCA  
K-means  
Autoencoder  
RNN  
Transformer  
GAN  
Finetuning