Derin Öğrenmeye Giriş | 13:00

Date	TUESDAY
Tags	
Checkbox	☐

DERİN ÖĞRENMEYE GİRİŞ

📌 1956'da Dartmouth Konferansı, yapay zeka araştırmlarının başlangıcı olarak kabul edilir.

📌 1957'de Frank Rosenblatt, perceptronutanıttı, bu da makine öğrenmesinin erken bir örneğiydi.

📌 Makine öğrenimi, bilgisayarlara açıkça programlanmadan öğrenme yeteneği verir. (Arthur Samuel, 1959)

📌 2006'da Geoffrey Hinton ve diğerleri, derin öğrenme terimini popüler hale getirdi ve sinir ağlarının derin katmanlarını eğitmek için etkili yöntemler geliştirdi.

⚠ Veri miktarı arttıkça daha iyi genellem yapılır ve derin öğrenme algoritmalarının başarımı artar.

▼ **TENSÖRLER**

> **Üç veya daha fazla boyutlu bir sayı dizisidir.** Örneğin, 3×3×3 boyutunda bir küp. Tensörler, özellikle görüntü işleme gibi daha karmaşık veri yapılarını temsil etmek için kullanılır.

▼ **TÜRÜNE GÖRE 3 YAPAY ZEKA VARDIR**

▼ **DAR YAPAY ZEKA**

- Belirli bir görevi yerine getirmek için tasarlanmış yapay zeka türüdür.
- Satranç oynayan bir yapay zeka, satranç oyununu oynamak için tasarlanmış dar bir yapay zekadır.
- Dar yapay zeka, günümüzde yapay zekanın en yaygın kullanılan türüdür.

▼ **GENEL YAPAY ZEKA**

- İnsan benzeri bir zekayı taklit etmek için tasarlanmış yapay zeka türüdür.
- Genel yapay zeka henüz geliştirilme aşamasındadır, ancak gelecekte mümkün olabileceği düşünülmektedir.

▼ **SÜPER YAPAY ZEKA**

- İnsan zekasından daha üstün bir zekayı taklit etmek için tasarlanmış yapay zeka türüdür.
- Henüz yalnızca bilim kurguda var olan bir kavramdır, ancak potansiyel olarak dünyayı dönüştürücü bir etkiye sahip olabilir.

▼ **ML TEMEL OLARAK 3 KISMA AYRILIR**

▼ **Danışmalı/Denetimli Öğrenme**

- Özellik olarak adlandırılan parametreler ve her bir özellik için çıkışların tanımlandığı veri setleri kullanılarak model eğitilir (etiketli veri var).
- Böylece modelin eğitimde kullanılmayan benzer girişler için doğru bir çıkış elde etmesi sağlanır.

> **3D teensörler**, **zaman serisi verileri** için çok etkilidir.

> Beyinden gelen bir EEG (elektroensefalogram) sinyalini 3D tensör olarak kodlayabiliriz, çünkü 3 parametre olarak kapsüllenebilir.

- **Renkli Resim Verileri:** Genelde 3 boyutludur. (genişlik, yükseklik, renk kanalları)
- **Video Verileri:** 4 boyutlu olabilir. (zaman, genişlik, yükseklik, renk kanalları)
- **3B Konvolüsyon İşlemleri:** 3B görüntü işleme veya video analizinde kullanılır.
- **Batch İşlemleri:** Birden çok örneği aynı anda işlerken kullanılır.

📌 Eğer 1 milyon tweet kullanırsak bunu 3D şekilde tensörü olarak depolayacağız: **(1000000,140,128)**

📌 MNIST veri setinin 4D tensörü: **(60000,28,28,1)** (veri sayısı,genişlik,yükseklik,renk derinliği sayısı(sadece gri))

▼ **AKTİVASYON FONKSİYONLARI**

- **Yapay sinir ağlarında her bir nöronun çıktısını belirleyen** matematiksel fonksiyonlardır.
- Ağın doğrusal olmayan yapılar öğrenmesini sağlar ve derin öğrenme modellerinin **karmaşık örüntüleri yakalama yeteneğini artırır**.

▼ **Aktivasyon Fonksiyonlarının Önemi**

- Doğrusal Olmayan Yapıları Modelleme
- Gradyan Akışını Kontrol Etme
- Çıktı Aralığını Normalize Etme
- Ağın Öğrenme Kapasitesini Artırma

GRADİENT DESCENT

> Rastgele alınan değişkenlerle başlayarak global minimum değerine ulaşmayı

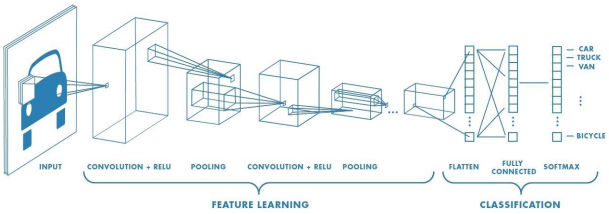
- Ev fiyatlarının tahmin edilmesi örnek olarak verilebilir (sayısal tahmin ya da ucuz/orta/pahalı vb).

▼ **Danışmasız/Denetimsiz Öğrenme**

- Veri setindeki benzerlikleri bularak çalışır. (etiketsiz veri var).
- Ev fiyatları örneğinde sadece evlerin özelliklerinin girildiği ama bunlara ait çıkışların, yani fiyatların belirtilmeden eğitimin gerçekleştirildiği düşünülebilir. (Veri sadece giriş, çıkış yok)
- Bu durumda model ile giriş parametrelerine bakarak Ev1 ile Ev3 birbirine benzer olduğu tespit edilebilir. Ancak evin fiyatı tahmin edilemez.

▼ **Takviyeli/Pekiştirmeli Öğrenme**

- Pekiştirmeli öğrenme;ajan(agent) adı verilen öğrenen bir sistem, durum (state), eylem (action) veödül(reward) unsurlarının etkileşimi üzerine kuruludur.Ajan, belirli bir durumda ödülü en üst düzeye çıkarmak için uygun işlemlerin yapılmasını hedefler.
- Öğrenme için veri seti yoktur ve ajan (agent) deneme yanılma ile öğrenir.
- Bir bebeğin ağlaması sonucunda şeker verilmesi -ödül. Zamanla, bebek şeker istediği her seferde ağlamayı öğrenir.



> Derin öğrenme modeli, karar verme mekanizmasından önce problemin gerekli özelliklerini çıkarma işlemini de kapsar.

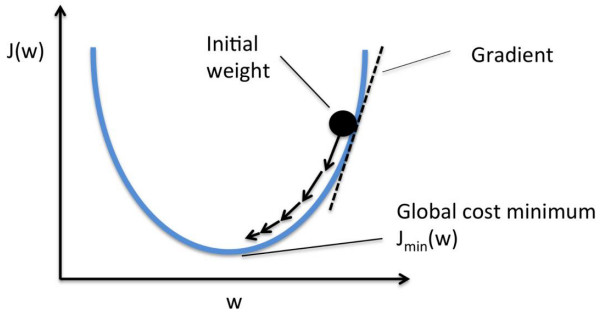
YAPAY SİNİR AĞLARI

> İnsan beyninin bilgi işleme yeteneğini taklit etmeyi amaçlayan bilgisayar sistemleridir.

hedefler.

Temel olarak, **GD** optimizasyonunu bir dağdan aşağıya (maaliyet fonksiyonu) bir vadiye gitmek isteyen (minimum maliyet) bir yürüyüşçü (ağırlık katsayısı) olarak görebiliriz.

Her adım, eğimin dikliği (eğim-gradient) ve yürüyüşçünün bacak uzunluğu (öğrenme oıranı-learning rate) tarafından belirlenir.



Yalnızca tek bir ağırlık katsayısına sahip bir maliyet fonksiyonu göz önüne alındığında, bu kavramı yukarıdaki gibi gösterebiliriz.

STOCHASTİC GRADIENT DESCENT (SGD)

GD optimizasyonunda, eğitim setindeki tüm veriler için maliyet hesaplanır. Bu nedenle, **Yığın (Batch) GD** olarak da adlandırlır.

- Çok büyük veri setleri olması durumunda, GD'yi kullanmakj hesaplama yükü oldukça artırabilir. Çünkü ağırlıkların bir kez güncellenmesi için tüm eğitim setinin hataya katkısının hesaplanması gerekiyor.
- Bu nedenle, eğitim seti büyüdükçe, algoritma yavaşlar, ağırlıkları uzun sürede günceller ve global minimuma daha yavaş yaklaşır.

SGD, bazen yinelemeli (iterative) veya çevrimiçi (online) GD oalrak da adlandırlır.

- Burada "stokastik" terimi, tek bir eğitim örneğine dayalı eğimin "gerçek" maliyet eğiminin "stokastik bir yaklaşımı" olmasıgerçeğinden kaynaklanmaktadır.

Ana Bileşenler: Nöronlar(yapay hücreler), katmanlar (giriş, gizli, çıkış) ve ağırlıklar (nöronlar arsı bağlantı gücü) yapay sinir ağlarının temel yapı taşlarıdır.

Nöronlar: Yapay nöronlar, gerçek nöronların işlevlerini taklit eden ve sinyalleri işleyen matematiksel fonksiyonlardır.

Katmanlar: Giriş, gizli ve çıkış katmanlarından oluşur. Her katman, belirlii bir işlemi gerçekleştirmek için nöronları içerir.

Ağırlıklar: Nöronlar arasındaki bağlantıların gücünü temsil eder. Ağırlıklar, öğrenme sürecindek sürekli olarak güncellenir.

YAPAY SİNİR AĞLARININ ÇALIŞMA MEKANİZMASI

İLERİ YAYILIM (FORWARD PROPAGATION), sinir ağındaki girdi katmanından (sol) çıktı katmanına (sağ) geçmenin yoludur.

📌 Çıktı katmanından girdi katmanına sağdan sola yani geriye doğru hareket etme işlemine **geri yayılım** denir.

Geri yayılım, minimum kayıp fonksiyonuna ulaşmak için ağırlıkları ayarlamak veya düzeltmek için tercih edilen yöntemdir.

HATA FONKSİYONU, gerçek çıktı ile beklenen çıktı arasındaki farkı ölçer. Ağırlıklar, bu hatayı minimize edecek şekilde ayarlanır.

OPTİMİZASYON Gradient Descent gibi optimizasyon yöntemleri, en düşük hata değerine ulaşmak için ağırlıkları günceller.

- Stokastik doğasından dolayı, küresel minimum maliyete giden yol GD'deki gibi "doğrudan" değildir, ancak maliyet yüzeyini 2 boyutlu bir alanda görselleştiriyorsak "zig-zag" gidebilir.

MINI BATCH GRADIENT DESCENT (MB-GD)

> Mini Toplu Gradyan İnişi (MB-GD), toplu GD ve SGD arasında bir uzlaşmadır.

> MB-GD'de modeli daha küçük eğitim örnek gruplarına göre güncelliyoruz; 1 örnekten (SGD) veya tüm n eğitim örneğinden (GD) gelen gradyanmı hesaplamak yerine, 1<k<n eğitim örneğinden gelen gradyanı hesaplıyoruz (yaygın bir mini batch boyutu k= 50dir.)

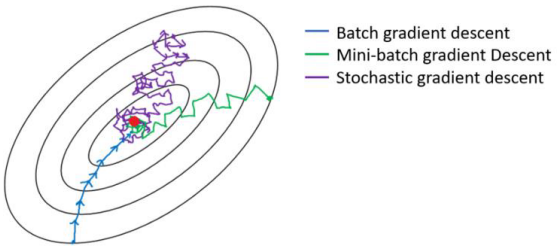
> MB-GD, ağırlıkları daha sık güncellediğimiz için GD'ye göre daha az yinelemde yakınsar; ancak MB-GD, vektörleştirilmiş işlemi kullanmamıza izin verir, bu da genellikle SGD'ye göre hesaplama performansı artışıyla sonuçlanır.

📌 **HESAPLAMA HIZLARI:** SGD > MB-GD > GD

GRADYAN TAHMİNİ DOĞRULUĞU: GD > MB-GD > SGD

YAKINSAMA KARARLILIĞI: GD > MB-GD > SGD

YEREL MİNİMUMLARDAN KAÇINMA: SGD > MB-GD > GD



UNDERFITTING & OVERFITTING

> **OVERFITTING,** algoritmanın **eğitim verisi üzerinden en alt kırılıma kadar çalışıp**, **sonuçları ezberlemesi** ve **sadece o veriler üzerinde başarı** elde edebilmesidir.

- Aşırı öğrenme problemi ile karşılaştığımızda, eğitim verisi ile kurduğumuz modeli, test verisi üzerinde çalıştırdığınızda muhtemelen sonuçlar **eğitim verisine göre çok düşük** olacaktır.
- Modelimizin amacı, **her şeyi tahmin etmesi değil genel bir doğru elde etmesi** yani genel bir örüntü bulmasıdır ve bu genel doğrunun (örüntünün) sonraki verilere de uygulanabilliyor olmasdır.
- **Algoritma çok karmaşık ise veri içindeki örünütüyü bulmak yerine, öğrenme süreci gürültüyü ezberlemek ile sonuçlanabilir.**

📌 Öğrencinin sınava girmeden önce bilgileri çok iyi bir şekilde ezberleyip, sınavda farklı türde sorular ile karşılaştığında sınavda başarısız olması gibi.

📌 Aşırı öğrenme problemi olan modeller yüksek varyans problemi içerebilir.

OVERFITTING - AŞIRI ÖĞRENME SORUNU NASIL ÇÖZÜLÜR

▼ **DEĞİŞKEN SAYISINI AZALTMAK**

> Aşırı Öğrenmeyi önlemek için **korelasyon**, **eksik değer** ve **aykırı değer analizleri** büyük önem taşır.

- Veri setinde yüksek korelasyona sahip bağımsız değişkenlerin varlığı, aynı bilgiyi taşımaları nedeniyle hem yanlılığa hem de aşırı öğrenmeye neden olabilir.

> Doğru açıklayıcı değişkenleri bularak basit bir model kurmak mantıklıdır.

▼ **DAHA FAZLA VERİ EKLEMEK**

▼ **GD, SGD, MBGD AVANTAJLARI VE DEZAVANTAJLARI**

▼ **GRADIENT DESCENT**

▼ AVANTAJLARI

- Tüm veri setini kullanarak dah doğru gradyan hesaplaması yapar.
- Genellikle daha kararlı ve pürüzsüz bir yakınsama sağlar

▼ DEZAVANTAJLARI

- Büyük veri setleri için hesaplama açısından pahalıdır.
- Her iterasyonda tüm veri setiini işlediği için yavaştır.
- Yerel minimumlara takılma riski vardır.

▼ **STOCHASTIC GRADIENT DESCENT (SGD)**

▼ AVANTAJLARI

- Her iterasyonda sadece bir örnek kullanıldığı için çok hızlıdır.
- Büyük veri setleri için uygundur.
- Yerel minimumlaradan kaçma potansiyeli daha yüksektir.

▼ DEZAVANTAJLARI

- Gradyan tahmini gürültülüdür, bu da yakınsamayaı dalgalı hale getirebilir.
- Optimum çözüme ulaşmak için daha fazla iterasyon gerektirebilir.
- Öğrenme oranının dikkatli ayarlanması gerekir.

▼ **MB-GD**

▼ AVANTAJLARI

- GD ve SGD arasında bir denge sağlar.
- SGD'den daha kararlı yakınsama, GD'den daha hızlı hesaplama sunar
- Paralel işleme için uygundur.

▼ DEZAVANTAJLARI

- Batch boyutunun doğru seçilmesi gerekir
- SGD kadar hızlı değildir, GD kadar kararlı da değildir.

VARYANS

- Eğer aşırı öğrenme problemi, eğitim verisinde az veri olmasından, dolayısıyla tek tip veri olmasından kaynaklanıyor ise daha fazla çeşitli veri eklemek gerekir.
- Burada engele takılmamak için veri hazırlığını dikkatli yapmak, eğitim verisi ve test verisi ayrımını dikkatli incelemekte fayda var.

▼ **ÇAPRAZ DOĞRULAMA**

- Sonuçları doğru değerlendirmek için eğitim verisi ve test verisinin benzer özelliklere sahip olaması gerekir. K katmanlı çapraz doğrulma ile raslantısallık azaltılarak sonuç metriklerinin tutarlılığı sağlanmaktadır.
- K Katmanlı çapraz doğrulama yöntemi kullanılarak verinin tüm parçalarının eğitim ve test verisinde yer almasıyla daha doğru bir öğrenme süreci oluşturmak

▼ **ERKEN DURDURMA**

- İki hatanın birbirinden ayrılamay başladığı nokta itibariyle aşırı öğrenme başlamış demektir.
- Eğitim verisi ile test verisi hataları arsındaki fark açıklığı belli bir seviyeye geldiğinde eğitimi durdur.
- Bazı algoritmalar bunu sürekli kontrol ederek otomatik yapar.

▼ **DÜZENLİLEŞTİRME**

- Modelin karmaşıklığını azaltmak için kullanılan tekniktir. Bunu kayıp fonksiyonunu cezalandırarak yapar. Yani modelde ağırlığı yüksek olan değişkenlerin ağırlığını azaltarak bu değişkenlerin etki oranını azaltır.

▼ **SEYRELTME**

> Sinir ağı içerisinde yer alan bazı nöronların rastgele olarak ortadan kaldırılmasında seyreltme (dropout) katmanı kullanılmaktadır

- Sinir ağında nöronların yüzde kaçı ortadan kaldırılacağı kullanıcı tarafından belirlenmektedir (0 ile 1 arasında bir değer ile belirlenir). Böylece ağınoverfittingolması önlenerek performansın artması sağlanmaktadır.
- Basitliği ve etkili olması nedeniyle, günümüzde çeşitli mimarilerde,

> Model eğitim **veri setinde iyi performans gösterdiğinde**, ancak bir test veri kümesi veya doğrulama veri kümesi gibi, **eğitilmemiş bir veri kümesinde iyi performans göstermediğinde** ortaya çıkar.

📌 Varyans, gerçek değerden tahmin edilen değerin ne kadar dağınık olduğunu söyler.

BIAS

> **Gerçek değerlerden tahmin edilen değerlerin ne kadar uzak olduğudur**. Tahmin edilen değerler gerçek değerlerden uzaksa, bias yüksektir.

📌 **Yüksek bias**'a sahip bir modelin **çok basit** olduğunu söyleyebiliriz.

📌 **Yüksek varyansa** sahip bir model, veri noktalarının çoğuna uymaya çalışır ve bu da modeli **karmaşık yapar** ve **modellenmesini zorlaştırır**.

📌 **OVERFITTING** ⇒ Yüksek Varyans, Düşük Bias

UNDERFITTING ⇒ Düşük Varyans, Yüksek Bias

📌 **Yüksek Bias Düşük Varyans : Model Tutarlıdır, Ortalama Hata Oranı Yüksektir**

Yüksek Bias Yüksek Varyans: Modeller Hem Hatalı Hem De Tutarsızdır

Düşük Bias Düşük Varyans: Modeller Ortlama Olarak Doğru Ve Tutarlıdır

Düşük Bias Yüksek Varyans: Modeller Bir Dereceye Kadar Doğrudur Ancak Ortalamada Tutarsızdır

genelliklefullyconnectedkatmanından sonradropoutkatmanı kullanılmaktadır.

> **UNDERFITTING**, modelin verilerdeki temel örüntüleri yaklamak için çok basit olması ve bu nedenle kötü performans göstermesi olarak tanımlanır.

- Bu modeller eğitim verilerini çok yakından takip etmek yerine, eğitim verilerinden alınan dersleri yok sayar ve firdiler ile çıktılar arasındaki temel ilişkiyi öğrenemez.

📌 Öğrencinin sınava hiç çalışmadan girip sınavdan da doğal olarak kötü not alması gibi.

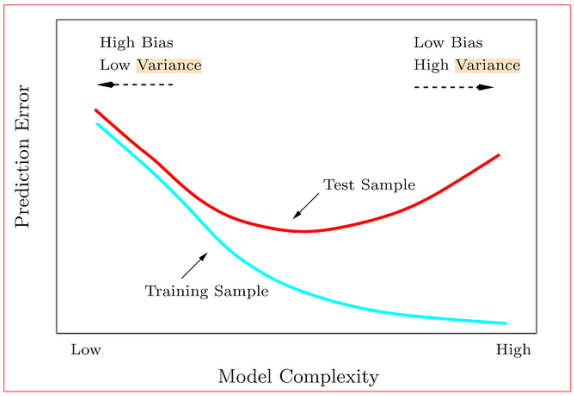
📌 Az öğrenme durumunda model hem eğitim hem de test verilerinde başarısız bir performans gösterir.

AZ ÖĞRENME SORUNU NASIL ÇÖZÜLÜR

> Az öğrenmeye sebep olan başlıca nedenler modelin basit yapısı, değişken sayısının yetersiz olması, yetersiz veri, gürültülü veri olarak sayılabilir.

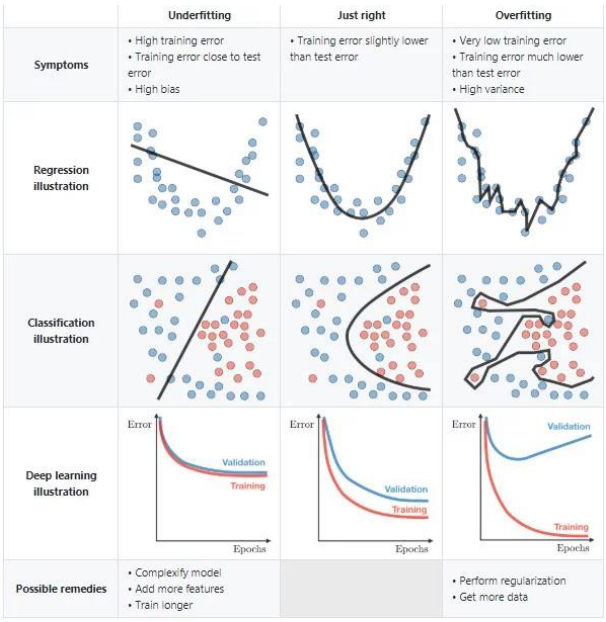
> Az öğrenme, **aşırı öğrenmeye göre önüne geçilmesi daha kolay bir sorundur.**

- Eğer sorun modelin basit yapısı ise bu sorunu gidermek için **modelimizin kapasitesini artırmamız gerekir**. Modelimizin kapasitesini modelin yapısında (örneğin katmanlardaki nöron sayılarını yada katman sayılarını arttırmak) değişikliklere giderek artırabiliriz.
    - Eğer problem veri sayısının azlığından ya da özellik sayısının azlığından kaynaklanıyorsa **veri sayısını ya da özellik sayısını arttırmak** çözüm olacaktır.



Source: Elements of Statistical Learning by Trevor Hastie, Robert Tibshirani and Jerome Friedman

Model Karmaşıklığı arttıkça eğitim seti üzerinde hatalı tahmin oranı azalmakta ancak test veri seti üzerinde tahmin hatası artmakta



▼ Eğer model **Yüksek Bias**'a sahipse

- Modelin eğitim setinin hata oranı yüksektir.
- Test / doğrulama veri seti hata oranı eğitim seti ile benzer oranda yüksektir.

▼ Eğer model **Yüksek Varyans**'a sahipse

- Modelin eğitim setinin hata oranı düşüktür.
- Modelin test/doğrulama veri setinin hata oranı yüksektir.

▼ **Yüksek Bias** problemini çözmek için

- Daha fazla veri eklemek
- Daha fazla değişken eklemek
- Regularization

▼ **Yüksek varyans** problemini çözmek için

> Overfitting sorununu çözme yöntemlerini uygulamamız gereklidir.

KATMANLAR

▼ **TAM BAĞLANTILI (YOĞUN) KATMAN**

- Sinir ağlarındaki en yaygın katman türlerinden biridir.
- Her düğüm, **önceki ve sonraki katmandaki her düğümle ağırlıklar aracılığıyla bağlantılıdır**. Bu yoğun bağlantı, ağın özellikler arasındaki **karmaşık ilişkileri öğrenmesini sağlar.**
- **Sadece yoğun katmanlardan oluşan bir ağa** tam bağlantılı veya çok katmanlı algılayıcı (**MLP**) denir.

- .isna = eksik veya boş değer kontrol eder
- .isnull = dizi benzeri bir nesne için eksik değerleri algılar
- .sum = toplama
- .fillna = NULL değerler yerine belirtilen değerle değişir.
- .map = dizinin içindeki elemanlarala işlem yapmayı sağlar
- dense = 'Dense', katman türüdür. Dense, çoğu durumda çalışan standart bir katman türüdür. Yoğun bir katmanda, önceki katmandaki tüm düğümler mevcut katmandaki düğümlere bağlanır.
- fit = eğitim
- len = eleman sayısı (lenght)
- argmax = yinelebilir bir değişkendeki maksimum değer indeksini döndüren yerleşik bir fonksiyondur.
- shape = dizinin boyutlarını verir
- evaluate = bir ifadeyi argüman olarak ayrıştırmak ve daha sonra Python programı içerisinde yürütmek için kullanılır.
- accuracy = doğruluk oranı
- loss = kayıp oranı
- convert = dönüştürme yapar

> **İdeal Kullanım**, TBK, her özelliğin diğer tüm özelliklerle iletişime girmesi gereken durumlarda özellikle yararlıdır; bu nedenle **sınıflardırma görevlerinin son aşamaları** için idealdir.

▼ AVANTAJLARI

- **Çok Yönlü Uygulama: Görünütler**, **metinler**, **sayısal veriler** gibi **çeşitli veri türlerine uygulanabilir** ve çok amaçlı uygulamalar için **yüksek uyum** sağlar.
- **Gelişmiş Tahmin Gücü:** Tüm özellikleri bir araya getirerek, yoğun katmanlar ağın karmaşık özellik etkileşimlerine dayalı olarak kesin ve güvenilir tahminler yapma yeteneğini artırır.

▼ DEZAVANTAJLARI

- **Ağır Parametre Yükü:** Çok sayıda parametre içerir, bu da büyük ağlarda **bellek kullanımı** ve **hesaplama maliyetini** artırır.
- **Aşırı Uydurma (Overfitting):** Parametre sayısının yüksek olması, **özellikle küçük veri kümeleri** üzerinde eğitildiğinde, yoğun katmanların **kolayca aşırı uyum sağlamasına** neden olabilir.

▼ KONVOLÜSYONEL KATMAN

> Konvolüsyonel katman, görüntü sınıflandırma gibi mekansal veri içeren görevlerde esastır.

- Konvolüsyonel katmanlar, **girdiyi tarayan** ve **yerel bölgelere odaklanan filtreler** (küçük matrisler) kullanır.
  - Bu yaklaşım, **kenarlar**, **dokular** ve **şekiller** gibi desenleri yakalamalarını sağlar ve bu da onları görüntü verisi için verimli hale getirir.
- Birden fazla konvolüsyonel katmana sahip bir ağ, **konvolüsyonel sinir ağı (CNN)** olarak adlandırılır. CNN'ler, düşük seviyeden yüksek seviyeye özellikleri kademeli olarak yakaladıkları için **görüntü tabanlı görevler için son derece etkilidir.**

▼ AVANTAJLAR

📌 **batch size** = modelin ağırlık gücellemesi yapılmadan önce kaç adet örneği işlediğini belirtir 2 nin katlarıdır. (değeri çok küçük olursa eğitim süresi uzayabilir; çok büyük olursa modelin eğitim kalitesi düşebilir)

📌 **epoch** = veri setini baştan kaç kere tanıttığını verir (epoch sayısı arttıkça model daha iyi öğrenebilir ama çok fazla olması overfitting e yatkın hale gelir)

📌 **sequential** = Modelin katmanlarını sırayla eklenmesini sağlar

📌 **relu fonk** = negatif değerleri sıfıra, pozitif değerleri ise aynen bırakır

📌 **Fully Connected Katman =** Öğrenilen özellikleri bir sınıfa veya çıktı değerine dönüştürür. Öğrenilen özellikleri sınıflandırmak için son katmanda kullanılır.

📌 **softmax fonk** = her nöronun çıktı değerini olasılığa dönüştürür ve toplam oalsılık 1 olur, birden fazla sınıf arasından bir tanesini seçmek için kullanılır

📌 **RMSPROP** = optimizasyon algoritmasıdır

📌 **categorical_crossentropy** = modelin doğru tahmnin yapmasını için eğitim gerçekleştirilir

- **Parametre Verimliliği:** Tüm nöronları birbirine bağlamak yerine, giriş üzerinde **kayan filtreler** kullanarak evrimsel katmanlar daha az parametreye ihtiyaç duyar, bu da **bellek kullanımını ve hesaplamayı azaltır**.
- **Yer Değiştirme Bağışıklığı:** **Evrimsel işlemler,** ağların desenleri giriş içindeki konumlarından bağımsız olarak tanımasını sağlar ve bu da **genelleme yeteneğini ve dayanıklılığını artırır.**

▼ **DEZAVANTAJLAR**

- **Yerel Desenlerle Sınırlıdır: Evrimsel katmanlar,** duyarlı oldukları alanlarda **yerel desenlere odaklanır**, bu da **küresel bilgiyi** aynı derecede **etkili bir şekilde yakalayamayabilecekleri anlamına gelir.**
- **Eğitimde Zorluk:** Özellikle derin evrimsel ağlar **eğitilmesi zor olabilir** ve genelleme yapabilmek için büyük miktarda etiketlenmiş veriye ihtiyaç duyabilir.

▼ **DROPOUT KATMANI**

> Derin öğrenmede **aşırı öğrenme (overfitting) sorununu azaltmaya yönelik** olarak yaygın şekilde kullanılan bir düzenleme (regularization) tekniğidir.

- Aşırı öğrenme, modelin eğitim verisi üzerinde çok iyi performans gösterirken yeni, görülmemiş verilerde başarısız olmasına neden olur.
- Dropout, her eğitim adımında belirli bir oranla **nöronları rastgele** "devre dışı bırakarak" (geçici olarak kaldırarak) çalışır.

> Bu sayede modelin belirli **nöronlara aşırı derecede bağımlı hale gelmesi önlenir** ve daha sağlam, genellenebilir özellikler öğrenmesi sağlanır.

> Dropout katmanları en yaygın olarak **yoğun (tam bağlantılı) katmanlarda** kullanılır. Bununla birlikte:

📌 **One-hot encoding**, her kategoriyi ayrı bir sutun olarak temsil ederken, **label-encoding** her kategoriye bir sayı atar.

📌 **Fine-Tuning =** Önceden geniş bir veri kümesiyle eğitilmiş bir modelin, daha küçük ve belirli bir veri kümesinde yeniden eğitilmesi işlemidir.

📌 **Transfer Learning** = Bir görev veya veri kümesini aracılığıyla elde edilen bilginin, başka bir ***ilgili*** görev veya farklı veri kümesindeki model performansı iyileştirmek için kullanılan bir makine öğrenimi tekniğidir.

26.11.2024 (9.Hafta)

- LTSM ve RNN zamana bağlı verilerle kullanılır.
- Önişlemede özellik seçimi CNN in içinde yapıldığından fazldana bir özelliğe gerek yoktur.
- GRU, LTSM den daha hızlı ama onun kadar başarılı olamayabilir. Bu bir dezavantajdır.
- Embedding sunumundaki kısımlar önemli-Word2Vec vb.
- Pytorch ile yapılan örnekler sınavda çıkmaz, ama keras a bağlı kodalr çıkabilir.

03.12.2024 (10.Hafta)

▼ **Konular**

- RNN Konusu
- Derin Öğrenme Donanım Gereksinimler - Cloud
- Hüsamaeddin sunumundaki lojistik reg, xgboost vb. kısmını geçebilrisin dedi hoca
  - Beyaz olanlar hatalı, mavi olanlar doğru
  - GNN

10.12.2024 (11.Hafta)

▼ **Konular**

Çok çıkışlı model yapıları

- Rol Head

> - **Konvolüsyonel Sinir Ağları (CNN)**: Dropout katmanı, CNN yapılarında tam bağlantılı katmanlardan sonra kullanılabilir.
> - **Tekrarlayan Sinir Ağları (RNN)**: RNN yapılarında katmanlar arasında seçici olarak kullanılabilir ve özellikle büyük sıralı veri modellerinde genelleştirmeyi iyileştirir.

▼ **AVANTAJLARI**

- **Aşırı Öğrenmeyi Önler**
- **Yedeklemeyi ve Sağlamlığı Teşvik Eder**

▼ **DEZAVANTAJLARI**

- **Eğitim Süresini Uzatabilir**
- **Modelin Kapasitesini Azaltabilir**

RNN

📌 RNN'ler insan beynindeki nöron aktivitesini simüle eden
modellerin geliştirilmesinde kullanılır.

📌 RNN'ler, 1980'lerde araştırmacılar David Rumelhart, Geoffrey Hinton ve Ronald J. Williams
tarafından tanıtıldı .

📌 RNN'ler, doğal dil ve zaman serisi analizi gibi sıralı verilerin işlenmesinde ilerlemeler için temel
oluşturdu

▼ RNN giriş ve çıkış yapılarına bağlı olarak 3 farklı türe ayrılır

▼ BİRE ÇOK

Model tek bir giriş (bir sabit veri) alır ve buna bağlı olarak sıralı bir çıktı üretir.

- Cross loss fonk
- PyTorch sınavda sorumlu değiliz
- 1D convulation
- GRU
- Konvolüsyon filtrelerinin görselleştirilmesi
- Time step önceki adımlara bakmak
- Üretken derin öğrenme modelleri GANs, VAEs, vb.

GAN

> **İki modelin** karşılıklı olarak öğrenip geliştiği bir yapıdır. Genellikle **görsel üretimi** için kullanılır.

GAN EĞİTİM SÜRECİ

- **Generator:** Rastgele bir gürültü (noise) alır ve sahte veriler üretir. Amaç, gerçek veriye benzer veriler üretmektir.
- **Discriminator:** Hem gerçek hem de sahte verileri alır ve **gerçek mi yoksa sahte mi?** sorusuna yanıt verir.
- **İki Ağı Eğitim:** Generator, Discriminator'ı kandırmaya çalışırken, Discriminator ise sahte verileri doğru şekilde ayırt etmeye çalışır.

LOSS FONKSİYONU

- **Discriminator Loss**: Gerçek veriler için 1, sahte veriler için 0 etiketleriyle çalışarak geröek ve sahte cerileri ayırt etmeyi öğrenir.
- **Generator Loss**: Generator, sahte verileri üretirken Dsicriminator'ı kandırmaya çalışır. Generator'ın kaybı, Discriminator'ın sahte veriyi gerçek olarak sınıflandırmaya çalıştığı hatayı minimize eder.

TOPLAM LOSS

- **Total Loss** = Discriminator Loss + Generator Loss
- Bu kayıp, her iki ağı karşılıklı olarak geliştirmeye çalışır ve zamanla daha gerçekçi veriler üretilmesini sağlar.

GAN Avantajları

- **Yüksek Kaliteli Veri Üretimi:** Gerçekçi ve **keskin veriler** üretir, özellikle görüntü üretiminde çok etkilidir.

Bu türün en açık örneklerinden biri, bir veya daha fazla kelimeden oluşan başlığa göre makale oluşturulmasıdır.

▼ ÇOKTAN BİRE

Çoklu girişlerin tek bir çıktıya dönüştürüldüğü bir model türüdür.

Birden fazla giriş (kelime) olan yorumları olumlu veya olumsuz (tek kelime) olarak değerlendirmede gördüğümüz durumdur.

▼ ÇOKTAN ÇOĞA

Sıralı girişlerin sıralı çıkışlarla eşleştirildiği bir model türüdür.

Giriş Dizisinin her bir öğesi, çıkış dizisinin her bir öğesiyle ilişkilendirilir ve her iki dizinin de uzunlukları genellikle aynı olabilir veya farklı olabilir.

▼ RNN Kullanım Alanları

1. Doğal Dil İşleme
2. Zaman Serisi Analizi
3. Görüntü İşleme
4. Ses işleme

▼ RNN AVANTAJLARI

- Bir önceki örnek ile ilişki kurar. Bu sayede girdiler unutulmadan ilerlenir.
- Kullanım alanı çok geniştir. (Metin, ses, sınıflandırma prob., regresyon vb modellerde kullanılır.)

▼ RNN DEZAVANTAJLARI

- Gradient vanishing/exploding problemleri
- Uzun girdileri işlemekte zorlanır

1D Konvolüsyon

> Verinin tek bir boyutunda kayan bir filtre (kernel) ile yapılan işlemdir.

Filtre, verinin üzerinde işlem yaparak önemli özellikleri çıkaran bir tür matematiksel yapıdır ve bu yapının içindeki parametreler, modelin eğitimi sırasında öğrenilir.

> 1D konvolüsyon modeli, sıralı verilerdeki önemli desenleri keşfetmek için kullanılır.

- **Çeşitlendirilmiş Çıktılar:** Farklı ve **çeşitli veriler** üretebilir, yaratıcı uygulamalarda kullanışlıdır (sanat, moda vb.).
- **Transfer Öğrenme:** Önceden eğitilmiş GAN'ler, **veri setlerini artırma** ve yeni veri üretme konusunda etkilidir.
- **Veri Artırma:** Az veriyle **gerçekçi sahte veriler** üreterek modelin genel performansını iyileştirir.

VAE

> **Otomatik Kodlayıcıları**n (Autoencoders) bir çeşididir. Amaç, verileri daha düşük boyutlu bir latent uzaya sıkıştırırken, bu sıkıştırılmış temsilden yeni veriler **üretmek**tir.

📌 **Encoder:** Veriyi daha küçük bir latent uzaya sıkıştırır.

📌 **Latent Space:** Verilerin sıkıştırılmış temsilleridir.

📌 **Decoder:** Latent uzaya yerleştirilen bu temsillerden verileri yeniden oluşturur.

VAE nin LOSS FONKSİYONLARI

1. Rekonstrüksiyon Hatası
2. KL Divergence

Özellik	GAN (Generative Adversarial Network)	VAE (Variational Autoencoder)
Eğitim Yapısı	**Adversarial (Rekabetçi)**: Generator ve Discriminator birbirine karşı çalışır.	**Otomatik Kodlayıcı**: Encoder ve Decoder arasında bir yapı vardır.
Çıktı Kalitesi	Yüksek kaliteli ve gerçekçi veriler üretir. **(Keskin, Net)**	Genellikle **bulanık** çıktılar, kalite düşük olabilir.
Latent Uzay	Latent uzayda **düzgünlük** yoktur, çıktılar karmaşık olabilir.	Latent uzay sürekli ve **düzenli** bir şekilde yapılandırılır.
Modelin Karmaşıklığı	**Daha karmaşık**: Generator ve Discriminator arasındaki dengeyi bulmak zordur.	**Daha basit**: Tek bir model, daha stabil bir eğitim süreci sunar.
Üretim Yeteneği	Gerçekçi veriler oluşturmak için çok etkilidir.	Yüksek kaliteli veri üretiminde genellikle **daha sınırlıdır**.

> Sıralı veriler ise aşağıdaki şekilde örneklendirilebilir:

- **Zaman Serisi Verileri**: Bir hava durumu verisi, her saat başı ölçülen sıcaklık değerlerini içerebilir. Burada her bir sıcaklık ölçümü, önceki ölçümlerle bir ilişki içindedir.
- **Metin Verileri** : Kelimeler veya harfler sırasına göre düzenlenmiş verilerdir. Bir cümledeki kelimelerin sırası, cümlenin anlamını belirler.
- **Ses Verisi**: Zaman içinde değişen ses dalgalarından oluşan verilerdir. Bir müzik parçasında melodinin sırası veya bir konuşmadaki kelimelerin sırası, anlamın doğru algılanabilmesi için önemlidir.

> 1D Kovulüsyon, sıralı veri işleme problemlerinde derin öğrenmenin önemli bir parçasıdır.

▼ **Aşağıdaki Nedenlerden Dolayı Geleneksel Yöntemlerin Aksine 1D Konvolüsyon Tercih Edilmektedir**

1. **Yerel Bağımlılıkları Öğrenme**: Zaman verisi veya metin gibi sıralı verilerde bağımlılıkları yakalar ve anlamlar oluşturur.
2. **Transfer Öğrenme**: Önceden eğitilmiş 1D Konvolüsyon modelleri, yeni sıralı veri problemlerine adapte edilebilir.
3. **Derinleştirilmiş Ağlar**: Birden fazla katman kullanılarak daha karmaşık örüntüleri kolaylıkla çözümlenebilir.

▼ **Conv1D Katmanı**

> Sıralı verinin üzerinde kayan bir filtre kullanarak veriden özellikler çıkarır. Bu, verideki yerel ilişkileri anlamaya yardımcı olur.

1. filters: Çıkışta kullanılacak filtre sayısı; modelin öğrenebileceği özellik sayısını belirler.
2. kernel_size: Filtrenin genişliği; her adımda filtre penceresinin kaç öğe ile işlem yapacağını belirler.
3. activation: Aktivasyon fonksiyonu (ör relu); çıkışın doğrusal olmayan hale gelmesini sağlar.

Özellik	GAN (Generative Adversarial Network)	VAE (Variational Autoencoder)
Eğitim Zorlukları	Zorlu: Dengesiz eğitim ve mode collapse gibi problemler yaşanabilir.	Daha **kararlı eğitim** süreci, ancak **yeniden yapılandırma hatası** olabilir.
Veri Çeşitlendirmesi	Veriyi çok **çeşitli** şekilde üretebilir.	Çeşitlendirme bazen sınırlı olabilir, daha **homojen** veriler üretilir.
Modelin Kontrolü	Modelin çıktısı daha **zor kontrol edilir**.	Modelin çıktısı daha **kontrol edilebilir**.

DİĞER ÜRETKEN DL MODELLERİ

▼ **Normalizing Flows**

- Verilerin doğru bir şekilde modele aktarılmasını sağlar.
- Çok daha kontrollü ve çeşitli veri üretimlerine olanak tanır.

▼ **Autoregressive Modeller**

- PixelCNN gibi modeller, her pikseli ardışık olarak tahmin eder.
- GPT-3 ve BERT gibi dil modelleri, metin üretmek için autoregressive yöntemlerini kullanır.

▼ **Energy-Based Models (EBM)**

- Bu modeller, veri üzerinden enerji fonksiyonları öğrenir ve gerçek veriye en yakın “enerji”yi bulmaya çalışır.

📌 **VERİ DENGESİNİ SAĞLAMA TEKNİKLERİ**

- **Oversampling**: Azınlık sınıfına ait örneklerin sayısını çoğaltarak veri dengesini sağlama yöntemidir. Bu, örnekleri kopyalayarak veya yeni örnekler sentezleyerek (SMOTE gibi) gerçekleştirilebilir.
- **Undersampling**: Çoğunluk sınıfına ait örneklerin sayısını azaltarak veri dengesini sağlama yöntemidir. Ancak bu yöntem, veri kaybına neden olabileceği için dikkatli kullanılmalıdır.
- **SMOTE (Synthetic Minority Over-sampling Technique)**: Azınlık sınıfına ait örnekler arasında yeni sentetik örnekler oluşturarak oversampling yapan bir tekniktir.

4. strides (isteğe bağlı): Filtrenin kayma adımı; filtrenin her adımda ne kadar kayacağını belirler.

▼ **MaxPooling1D Katmanı**

> Verinin boyutunu küçültmek için kullanılır. Böylece modelin daha küçük boyutlarda önemli özellikleri öğrenmesini sağlar.

1. pool_size: Havuzlama penceresinin büyüklüğüdür. Her adımda kaç veri öğesinin havuzlanacağına karar verir.
2. strides: (isteğe bağlı): Havuzlama işlemi için kayma adımı. Bu, pencerenin her adımda ne kadar kayacağını belirler.
3. padding (isteğe bağlı): Veri kenarlarıyla ilgili nasıl işlem yapılacağını belirler (valid veya same). valid padding kenarları keserken same paddingverinin boyutunu korur.

▼ **Flatten Katmanı**

> Çok boyutlu veriyi tek boyutlu bir vektöre dönüştürür. Bu işlem, veriyi tam bağlı (Dense) katmanlara uygun hale getirir.

▼ **Danse Katmanı**

> Önceki katmanlardan gelen çıktıları alır ve her bir girdiyle tam bağlantılı olarak çalışarak nihai sonuca ulaşır. Bu katman, öğrenilen özellikleri kullanarak modelin tahminini yapar.

1. **units**: Çıkıştaki nöron sayısı. Bu modelin çıktısının boyutunu belirler.
2. activation: Aktivasyon fonksiyonu
3. **use_bias** (isteğe bağlı): Bias (önyargı) parametresinin kullanılıp kullanılmayacağını belirler. (True veya False) Varsayılan olarak True'dur.

📌 **NOTE**

- **Shuffle**: Bir dizinin içerisindeki değerlerin yerlerini değiştirir.
- **Unsqueeze**: Bir tensördeki belirtilen konuma bir boyutlu yeni bir boyut ekler.
- Görüntü için HWC formatında, Derin Öğrenme için CHW formatında veri gerek.
- HxWxC (Yükseklik, genişlik, kanal sayısı)
- TxD (kelime sayısı, kelime gömme boyutu)

Çok Çıkışlı Sinir Ağları

> Çok çıkışlı sinir ağları, birden fazla bağımsız veya ilişkili çıktıyı aynı anda tahmin edebilen ağlardır.
>
> Standart sinir ağlarından farkı, birden fazla görev için eş zamanlı öğrenim sağlayacak şekilde yapılandırılmış olmalarıdır.

▼ **Neden Çok Çıkışlı Sinir Ağı Kullanılır?**

- **Kaynak Verimliliği**: Birden fazla görev için ayrı modeller kullanmak yerine, tek bir model ile verimlilik sağlanır.
- **Bilgi Paylaşımı**: Ortak katmanlar sayesinde bir görevde öğrenilen bilgiler diğer görevlerde de kullanılabilir.
- **Zaman ve Maliyet Tasarrufu**: Eğitim ve tahmin süreçlerinde zaman kazandırır ve işlem maliyetlerini düşürür.

Embedding

> "Embedding" terimi, genellikle makine öğrenimi ve doğal dil işleme (NLP) alanlarında kullanılan bir kavramdır. Bir veriyi, genellikle yüksek boyutlu bir uzayda daha düşük boyutlu bir temsile dönüştürmek anlamına gelir.

Örnek Uygulama:

📌 **HİPERPARAMETRE OPTİMİZASYONU**

- **Grid Search:** Tüm olası hiperparametre kombinasyonlarını deneyerek en iyi performansı veren seti bulma yöntemi.
- **Random Search:** Hiperparametre uzayında rastgele noktalar seçerek daha verimli bir arama yapma yöntemi.
- **Bayesian Optimization:** Önceki deneyimleri kullanarak yeni hiperparametre kombinasyonları öneren bir yöntem.
- **Otomatik Hiperparametre Ayarlama:** AutoML gibi araçlar kullanarak hiperparametre optimizasyonunu otomatikleştirme.

Çok Çıkışlı Sinir Ağları

📌 Faster R-CNN, nesne tespiti için kullanılan gelişmiş bir modeldir ve nesnelerin türlerini ve konumlarını aynı anda tahmin eder. Model, üç ana bileşene sahiptir: **Backbone, Region Proposal Network (RPN), ve Region of Interest (ROI) Heads,**

Bu bileşenlere entegre edilmiş **Feature Pyramid Network (FPN)**, modelin farklı çözünürlüklerden özellik çıkarma yeteneğini artırır.

Netflix: Kullanıcıların izleme geçmişi ve beğenileri kullanılarak, film ve dizi önerileri yapılır. Kullanıcı embedding'leri, izledikleri içeriklerin türlerine göre oluşturulurken, ürün embedding'leri (film/dizi) içerik türü, oyuncular ve izlenme oranları gibi özelliklere dayanarak oluşturulur.

Embedding Katmanlarının Eğitim Stratejileri:

1- Pre-tranied Embeddings (Önceden Eğitilmiş): Örneğin, World2Vec veya GloVe gibi önceden eğitilmiş embeddinglerin kullanımı.

2- Fine-tuning: Belirli bir görev için mevcut embeddinglerin optimize edilmesi.

3- Custom Training: Veriye özel embedding matrisinin sıfırdan eğitilmesi.

📌 **TF-IDF**

Amaç:

- Belirli bir kelimenin bir dokümandaki önemini ölçmek

Nasıl Çalışır:

- **TF (Term Frequency):** Kelimenin bir dokümanda kaç kez geçtiğini ölçer.
- **IDF (Inverse Document Frequency):** Kelimenin genel olarak tüm dokümanlarda ne kadar yaygın olduğunu ölçer.

📌 **WORD2VEC**

Amaç:

- Kelimeleri matematiksel vektörlere dönüştürerek semantik anlamlarını modellemek.

Nasıl Çalışır:

- Sinir ağı kullanarak kelimeler arasındaki ilişkileri öğrenir.

Konvolüsyon Filt. Görselleştirme Yöntemleri

📌 BACKBONE (Özellik Çıkarım Katmanları)

Görevi:

- Görüntüden temel özellik haritalarını (feature maps) çıkarmaktır.
- Görüntüyü düşük seviyeli bilgiden (kenar, doku gibi) yüksek seviyeli bilgilere (nesne yapıları) dönüştürür.

Kullanılan Yapılar:

- ResNet, VGG veya diğer CNN mimarileri: ResNet-50 veya ResNet-101 gibi önceden eğitilmiş modeller sıkça tercih edilir.

FPN ile Entegrasyon

- Backbone'dan çıkarılan özellik haritaları FPN aracılığıyla işlenir.
- FPN, farklı ölçeklerdeki nesneleri algılamak için çoklu çözünürlük seviyelerinden bilgi toplar.

📌 Feature Pyramid Network (FPN)

FPN Nedir?

- FPN, Backbone'un farklı seviyelerinden gelen özellikleri birleştirerek çok ölçekli (multi-scale) özellik haritaları oluşturur.

Görevi:

- Küçük ve büyük nesneleri tespit etmek için farklı çözünürlüklerdeki bilgileri birleştirir.
- Küçük nesnelerin algılanabilirliğini artırır.

Çalışma Mantığı

- **Çalışma Mantığı:**

1. **Bottom-Up Pathway:** Backbone'dan gelen düşük çözünürlüklü özellikler, hiyerarşik olarak işlenir.
2. **Top-Down Pathway:** Düşük çözünürlüklü bilgileri yukarıya (yüksek çözünürlüklü katmanlara) taşır.
3. **Lateral Connections:** Farklı çözünürlük seviyelerini birleştirerek zengin özellikler oluşturur.

- **FPN Kullanımı:**
  - Çıkardığı çoklu çözünürlükteki özellik haritaları, RPN ve ROI Heads tarafından kullanılır.

📌 Filtre Ağırlıklarının Görselleştirilmesi

- Konvolüsyon filtrelerinin öğrenilen ağırlıkları, modelin girişte hangi temel özelliklere odaklandığını anlamak için görselleştirilir.
- Bu yöntem, filtre ağırlıklarını bir görüntüye dönüştürerek, matris değerlerini renkli ya da gri tonlama formatında sunar.

📌 Özellik Haritalarının Görselleştirilmesi

- Belirli bir konvolüsyon filtresinin bir giriş görüntüsüne nasıl tepki verdiğini gösterir.
- Bu haritalar, görüntünün hangi kısımlarının model için önemli olduğunu anlamaya yardımcı olur.
- Görselleştirme sırasında bir katmandaki her filtrenin aktivasyonu ayrı ayrı incelenir.

📌 Deconvolution Networks ve Guided Backpropagation

- Bir görüntünün modelde nasıl işlendiğini tersine mühendislik yaparak analiz eder.
- **Deconvolutional Networks,** aktivasyonları tersine çevirerek girişle eşleştirirken,
- **Guided Backpropagation** ise sadece pozitif gradyanları izler.
- Bu sayede modelin girişte hangi özelliklere dikkat ettiğini net şekilde ortaya koyar.
- Bu teknik, özellikle görüntü sınıflandırmada detaylı analiz sağlar.

📌 Region Proposal Network (RPN)

RPN Nedir?

- Aday nesne bölgelerini (region proposals) belirleyen bir alt ağdır.

Görevi:

- Backbone ve FPN'den gelen özellik haritalarını kullanarak potansiyel nesne konumlarını belirler.

Her bölge için:

1. Sınır Kutusu Koordinatları (x,y,w,h),
2. Nesne Olasılığı (Pobject) tahmin eder.

- **Çalışma Mantığı:**

1. Backbone ve FPN'den gelen özellik haritalarını alır.
2. Kayar pencere (sliding window) yöntemiyle bölgeleri tarar.
3. Anchor boxes kullanarak, her bölge için potansiyel nesne tahminleri üretir.

- **Özet:**
  - RPN, hangi bölgelerde nesne olduğunu tahmin eder ve bu bilgiyi ROI Heads'e gönderir.

📌 Region of Interest (ROI) Heads

ROI Heads Nedir?

- RPN tarafından belirlenen bölgeleri işler ve her bölge için nesne sınıfını ve hassas sınır kutusu koordinatlarını tahmin eder.

Görevi:

- Her aday bölgenin sınıfını tahmin etmek (örneğin, kedi, araba)
- Daha kesin sınır kutuları üretmek.

Çalışma Mantığı

1. Backbone ve FPN'den gelen özellik haritaları ile RPN'den gelen bölgeleri birleştirir.
2. **ROI Align** yöntemiyle bu bölgeleri hassaslaştırır.
3. Nihai tahminleri (sınıf ve sınır kutusu) üretir.

- **Özet:**
  - ROI Heads, aday bölgeleri son tahminler için işleyen bileşendir.

📌 Gradient-Based Methods: Grad-CAM

- Modelin bir sınıf tahmini yaparken hangi görüntü alanlarına odaklandığını anlamaya olanak tanır.
- Aktivasyon haritalarını ve sınıfa özel gradyan bilgilerini birleştirerek bir ısı haritası oluşturur.
- Bu ısı haritası, modelin karar mekanizmasını görselleştirir ve özellikle hata analizi ve model açıklanabilirliğinde oldukça popülerdir.

Grad-CAM Görselleştirmesi: Sağ tarafta, sınıf aktivasyon haritası görüntüyle çakıştırılmış olarak gösterilir. Haritada kırmızı ve sarı bölgeler modelin o sınıfa karar verirken dikkat ettiği alanları temsil eder.

📌 SONUÇ

Konvolüsyon filtrelerinin görselleştirilmesi, derin öğrenme modellerini anlamayı ve geliştirmeyi kolaylaştırır. Bu yöntemler sayesinde:

- Modellerin hangi özellikleri öğrendiği görülebilir,
- Hatalar analiz edilerek performans artırılabilir,
- Modelin karar mekanizmaları şeffaf hale gelir.

Doğru görselleştirme yöntemi seçildiğinde, hem modelin gücü artırılır hem de sonuçların daha iyi yorumlanması sağlanır.

📌 En Küçük Kareler

Doğrusal Regresyonunda hedefimiz, dikey ofsetleri en aza indiren çizgiyi bulmaktır.

Faster R-CNN'in Genel Çalışma Akışı

1. **Görsel Girdi:** Model, bir görüntüyü alır ve Backbone'a gönderir.
2. **Özellik Çıkarımı:** Backbone, görüntüden temel özellik haritalarını çıkarır.
3. **Çok Ölçekli Özellikler (FPN):** FPN, farklı çözünürlüklerdeki özellikleri birleştirir.
4. **Aday Bölgeler (RPN):** RPN, potansiyel nesne bölgelerini belirler.
5. **Son Tahmin (ROI Heads):** ROI Heads, her bölgenin sınıfını ve sınır kutularını tahmin eder.

📌 GRU

> GRU'nun temel amacı, geleneksel RNN'lerin
> karşılaştığı vanishing gradient (kaybolan
> gradyan) ve exploding gradient (patlayan
> gradyan) problemlerini çözmektir.

- GRU, geleneksel RNN'lere göre daha etkili çalışır çünkü daha az parametreye sahip olup, daha verimli bir şekilde öğrenme sağlar.
- GRU'nun temel yapı taşları sıfırlama ve güncelleme kapılarıdır.

📌 Kayıp (Loss) Fonksiyonları

- **MSE:**

Hesaplanan çıkışlarla beklenen çıkışların farklarının karelerinin
toplamının eleman sayısına bölümü ile hesaplanır.

- **MAE:**

Hesaplanan çıkışlarla beklenen çıkışların farklarının mutlak
değerlerinin toplamının eleman sayısına bölümü ile hesaplanır.

- **Binary Cross Entropy:**

İkili sınıflandırma için kullanılır

- **Categorical Cross Entropy**

📌 Vanishing Gradient Problem

Öğrenme eylemini sağlayan yapının yani gradyan fonksiyonunun (bir learning rate ile loss function'ı 0'a doğru yakınsamaya çalışan yapı) işlevsizleşmesine vanishing gradient problemdir.

📌 **AKTİVASYON FONKSİYONLARI**

Sigmoid Fonk

- Çıktı aralığı: (0, 1)
- Gradyan vanishing (gradyan yok olması) problemi yaşayabilir

Kullanım alanları:

- İkili sınıflandırma problemlerinde çıkış katmanı
- Eski yapay sinir ağı modellerinde yaygın kullanım

Tanh (Hiperbolik Tanjant) Fonk

- Çıktı aralığı: (-1, 1)
- Sigmoid'e göre daha güçlü gradyanlar

Kullanım alanları:

- RNN ve LSTM gibi tekrarlayan ağlarda
- Gizli katmanlarda

ReLU

- Çıktı aralığı: [0, ∞)
- Seyrek aktivasyon sağlar
- Negatif değerler için gradyan sıfır (dying ReLU problemi)

Kullanım Alanları:

- Çoğu derin öğrenme modelinde varsayılan seçim
- Özellikle CNN'lerde yaygın kullanım

Leaky ReLU

- ReLU'nun bir varyasyonu
- Negatif değerler için küçük bir eğim sağlar
- Dying ReLU problemini azaltır
- ReLU'nun alternatifi olarak, özellikle dying ReLU problemi yaşanan durumlarda

---

ELU

- Negatif girdiler için yumuşak doygunluk

- ReLU'ya göre gürültüye daha dayanıklı
- Derin ağlarda, özellikle hızlı öğrenme ve yüksek doğruluk gerektiren durumlarda

Softmax

- Çıktıları olasılık dağılımına dönüştürür
- Tüm çıktıların toplamı 1'dir
- Çok sınıflı sınıflandırma problemlerinde çıkış katmanı