Derin Öğrenmeye Giriş

Derin öğrenme (Deep Learning), bilgisayarlara verileri insan beyninin işleyişinden esinlenerek analiz etmeyi öğreten bir yapay zekâ yöntemidir. Bu teknoloji, çok katmanlı yapay sinir ağlarını kullanarak karmaşık veri setleri içindeki gizli kalıpları çözer. Derin öğrenmeyi "derin" kılan şey, verinin işlendiği yapay sinir ağlarındaki katman sayısıdır. Katman sayısı arttıkça modelin öğrenme kapasitesi de doğru orantılı olarak artar.

Temel Özellikleri ve Yetenekleri:

Derin öğrenme modelleri; resim, metin, ses ve video gibi diğer veri türlerindeki karmaşık desenleri tanıyarak doğru tahminler ve öngörüler üretir. Tipik olarak insan zekâsı gerektiren görevleri otomatikleştirmek için kullanılır. Örnekler:

- ***Karmaşık Desen Tanıma:*** Geleneksel algoritmaların veya insan gözünün fark edemeyeceği mikroskobik detayları ve çok boyutlu ilişkileri yakalar.
- ***Kendi Kendine Öğrenme:*** Veri arttıkça tahmin ve öngörü doğruluğunu otomatik optimize eder.
- ***İleri Seviye Otomasyon:*** Normal şartlarda insan zekâsı, deneyimi ve karar verme mekanizması gerektiren karmaşık görevleri gerçekleştirir.

Derin Öğrenme Nasıl Çalışır?

Derin öğrenmenin temelinde, insan beyninin mimarisini taklit eden Yapay Sinir Ağları (YSA) yer alır. Bu sistemin işleyişi şu temel prensiplere dayanır:

1. Nöron Yapısı: İnsan beyni, bilgiyi öğrenmek ve işlemek için birbirine bağlı milyarlarca nöronun (sinir hücresi) oluşturduğu devasa bir ağ kullanır. Benzer şekilde, derin öğrenme sinir ağları veya yapay sinir ağları, bilgisayar içinde birlikte çalışan birçok yapay nöron katmanından oluşur.

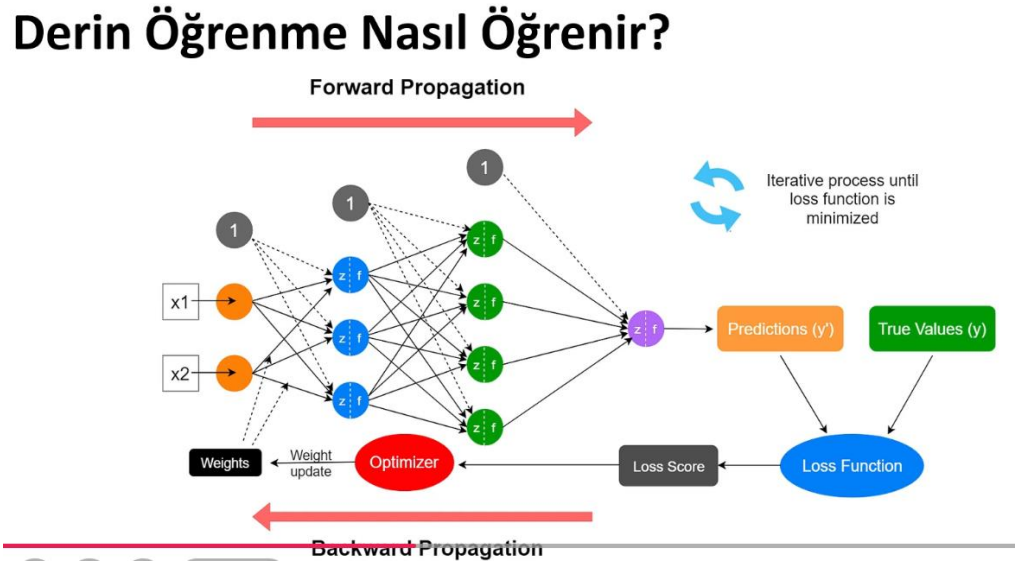
2. Katmanlı Mimari: Yapay sinir ağları, tek bir blok yerine üst üste gelen katmanlardan oluşur. Her katman verinin farklı bir özelliğini öğrenir:

- **Giriş Katmanı:** Ham verinin (örneğin bir resmin pikselleri) sisteme girdiği yerdir.
- **Gizli Katmanlar:** "Derin" terimi tam olarak buradan gelir. Verideki desenlerin (kenarlar, şekiller, dokular) analiz edildiği çok sayıda ara katmandır.
- **Çıkış Katmanı:** Yapılan analizler sonucunda nihai kararın (örneğin "Bu bir kedi resmidir") verildiği katmandır.

3. Yapay Nöronlar ve Düğümler (Nodes): Her katmanda, biyolojik nöronların görevini üstlenen düğümler bulunur. Bu düğümler aslında birer matematiksel fonksiyon gibi çalışır:

- **Veri İşleme:** Her düğüm, gelen veriyi belirli bir ağırlıkla çarpar ve toplar.
- **Karar Verme:** Eğer elde edilen sonuç belirli bir eşik değerini aşarsa, bilgi bir sonraki katmana aktarılır. Bu işleme "aktivasyon" denir.

Özetle: Derin öğrenme, veriyi milyonlarca küçük matematiksel işleme böler ve bu işlemlerin sonucunda tıpkı bir insanın zamanla öğrenmesi gibi, sistemin hata payını minimize ederek en doğru sonucu bulmasını sağlar.



Bu şema, Yapay Sinir Ağı'nın (ANN) veriden nasıl anlam çıkardığını gösteren aşamaları özetler:

1. Temel Kavramlar ve Formüller: Öğrenme süreci başlamadan önce, her bir nöronda (şemadaki renkli daireler) gerçekleşen matematiksel işlemi anlamalıyız.

- **Girdi (Input- x):** Modele sunulan ham verilerdir (örn: piksel değerleri).
- **Ağırlıklar (Weights- w):** Verinin ne kadar önemli olduğunu belirleyen katsayılardır. Öğrenme dediğimiz şey aslında bu w değerlerinin en doğru hale getirilmesidir.
- **Sapma (Bias- b):** Modelin esnekliğini artıran, toplama eklenen sabit bir sayıdır.
- **Aktivasyon Fonksiyonu (f veya σ):** Toplam sonucu (xw + b) alır ve onu bir karara dönüştürür. Bu, hücrenin "ateşlenip ateşlenmeyeceğini" belirler.
- **z1, z2, …:** Katmanlar arasındaki çıktıları temsil eder.

2. İleri Yayılım (Forward Propagation): Veri ilk katmandan girer, her katmanda ağırlıklarla çarpılıp aktivasyon fonksiyonlarından geçerek ilerler (h = f (xw+b)). Bu aşamanın sonunda model bir tahmin (y') üretir. Bu aşamada henüz "öğrenme" gerçekleşmez, sadece mevcut bilgilerle sonuç üretilir.

3. Hata Ölçümü: Kayıp Fonksiyonu (Loss Function): Model ürettiği tahmini (y'), olması gereken Gerçek Değer (y) ile karşılaştırır.

- Eğer loss düşükse: Model doğru yoldadır.
- Eğer loss yüksekse: Modelin tahminleri gerçeklikten uzaktır ve ağırlıkların ciddi şekilde güncellenmesi gerekir.

4. Geri Yayılım (Backward Propagation): Burası "öğrenmenin" gerçekleştiği yerdir. Model**,** loss fonksiyonundan aldığı hata sinyalini kullanarak en başa doğru gider. Her bir ağırlığın (w) hataya ne kadar sebep olduğunu hesaplar.

5. Optimizasyon ve Gradient Descent:

- **Parametrelerin Güncellenmesi:** Hata bulunduktan sonra, optimizer devreye girer. Ağırlıkları (w) ve bias (b) değerlerini, hatayı bir sonraki seferde azaltacak şekilde küçük adımlarla değiştirir.

- **Gradient Descent:** Hatayı bir dağın tepesi, en düşük hatayı ise vadinin tabanı gibi düşünün. Gradient Descent, "yokuş aşağı" inerek en düşük hata noktasını bulma yöntemidir.

Derin Öğrenme Donanım Gereksinimleri

Derin öğrenme modelleri, milyonlarca parametreyi eğitmek için devasa bir hesaplama kapasitesine ihtiyaç duyar. Standart bir bilgisayarın beyni olan CPU (Merkezi İşlem Birimi), karmaşık mantıksal görevlerde iyidir ancak derin öğrenmenin gerektirdiği yoğun matematiksel yük altında yavaş kalır.

- **Paralel İşleme Gücü:** GPU'lar (Grafik İşlem Birimleri), binlerce küçük çekirdekten oluşur. Bu yapı, sinir ağlarındaki matematiksel işlemleri ardışık yapmak yerine aynı anda (paralel) gerçekleştirmelerini sağlar.
- **Bellek Bant Genişliği**: Büyük veri yığınlarını (Big Data) işlemek için gerekli olan veri trafiğini hızla yönetirler.

Bulut Çözümlerinin Avantajları: Yüksek donanım maliyetlerinden kaçınmak isteyenler için bulut tabanlı platformlar (AWS, Google Cloud, Azure vb.) stratejik bir çözüm sunar:

- **Hız:** En yeni ve en güçlü GPU mimarilerine veya TPU'lara büyük yatırım maliyeti olmadan anında erişebilirsiniz.
- **Esneklik ve Ölçeklenebilirlik:** Projeniz büyüdükçe işlem gücünü saniyeler içinde artırabilir, işiniz bittiğinde ise kapasiteyi düşürerek yalnızca kullandığınız kadar ödeme yaparsınız.

Derin Öğrenmenin Karşılaştığı Zorluklar

- **Yüksek Kaliteli Veri İhtiyacı:** Modellerin öğrenmesi için sadece büyük veri değil, aynı zamanda doğru etiketlenmiş ve temiz "yüksek kaliteli" veri setleri gerekir.
- **İşlem Gücü ve Maliyet:** Donanım gereksinimleri, özellikle küçük ölçekli projeler için yüksek enerji ve satın alma maliyeti anlamına gelir.
- **Yorumlanabilirlik (Kara Kutu Problemi):** Derin öğrenme modellerinin bir karara "nasıl" vardığını açıklamak zordur; bu da güvenlik ve tıp gibi alanlarda güven sorunu yaratabilir.
- **Aşırı Öğrenme (Overfitting):** Modelin eğitim verilerini ezberlemesi ancak gerçek dünyadaki yeni verilerde başarısız olması durumudur.
- **Zaman:** Eğitim süreci, geleneksel yazılımların aksine anlık değildir. Modelin mimarisine ve veri büyüklüğüne bağlı olarak eğitim; saatler, günler, hatta haftalar sürebilir.

Derin Öğrenme Yöntemleri:

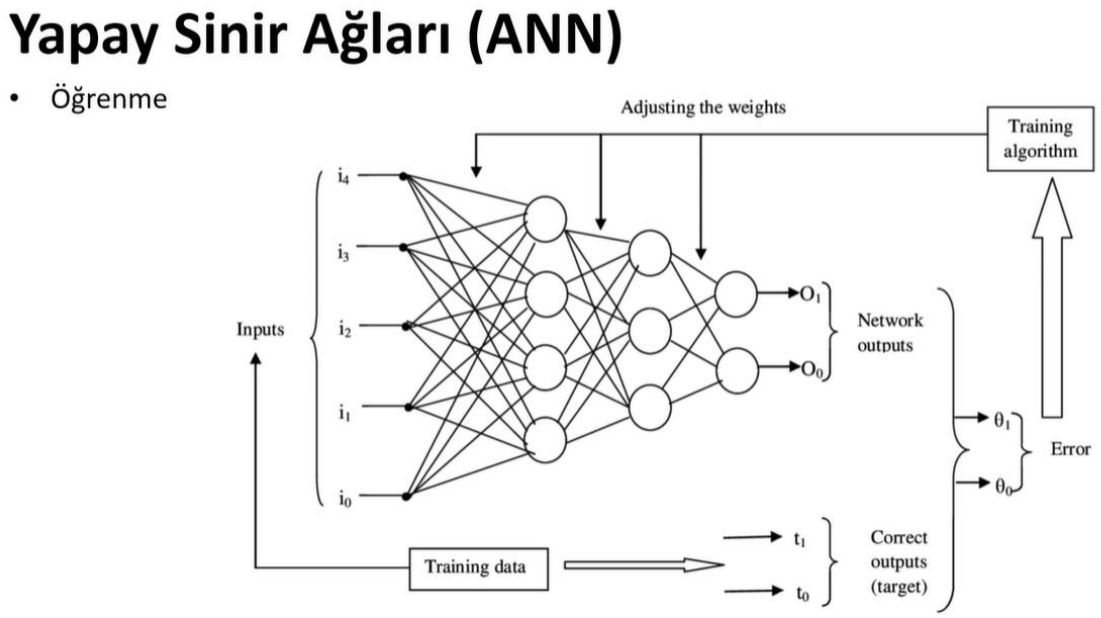
1. Yapay Sinir Ağları (ANN)
2. Evrişimsel Sinir Ağları (CNN)
3. Yinelemeli Sinir Ağları (RNN)
4. Uzun Kısa Süreli Bellek (LSTM)
5. Çekişmeli Üretici Ağlar (GANs)
6. Radyal Temelli Fonksiyon Ağları (RBFNs)
7. Kendini Düzenleyen Haritalar (SOMs – Self-Organizing Maps)
8. Derin İnanç Ağları (DBNs)
9. Otomatik Kodlayıcılar (Autoencoders)
10. Transfer Öğrenimi (Transfer Learning)
11. Artık Ağlar (ResNets)
12. Derin Pekiştirmeli Öğrenme
13. Transformers

1. Yapay Sinir Ağları (Artificial Neural Networks- ANN)

Tanım: Yapay Sinir Ağları, insan beyninin bilgi işleme mekanizmasını modelleyen, birbirine bağlı işlem birimlerinden (nöronlar) oluşan bir hesaplama sistemidir. Veriler arasındaki doğrusal olmayan karmaşık ilişkileri "öğrenme" yoluyla çözmek için tasarlanmıştır.

Kullanım Alanı: Yapay sinir ağları, finansal öngörüler, görüntü tanıma, dil işleme ve oyun stratejileri gibi geniş uygulama yelpazesinde kullanılmaktadır.

Örnek: Bir yapay sinir ağı, sesli komutları tanıma uygulamasında kullanılabilir.



Bu görsel, bir Yapay Sinir Ağının (ANN) temel çalışma prensibini ve öğrenme sürecini özetliyor:

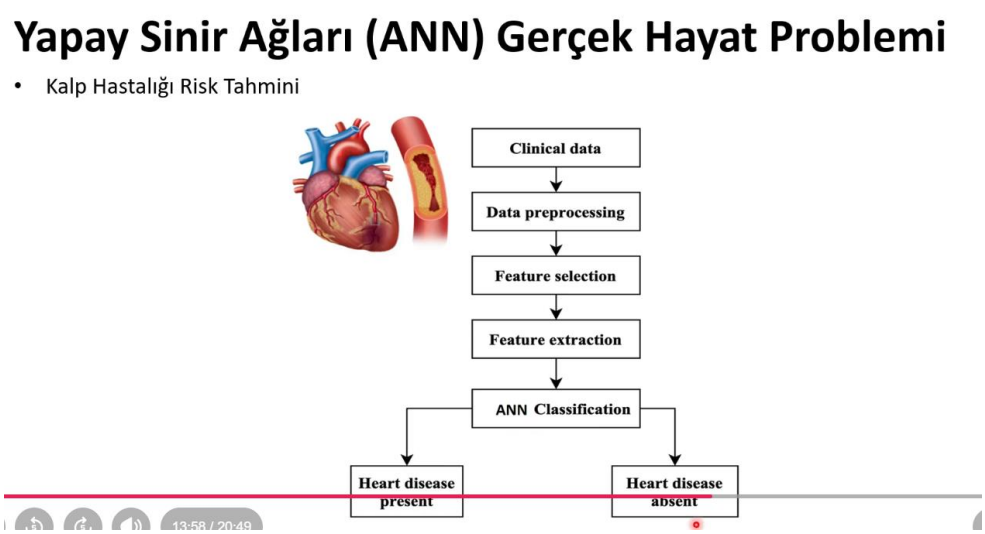
Yapay Sinir Ağları için öğrenme; **w (ağırlık)** ve **b (bias)** değerlerinin, en doğru tahmini yapacak şekilde optimize edilmesidir. Bu süreç temel olarak üç adımdan oluşan bir döngüdür:

1. **Forward Propagation (İleri Yayılım):** Girdi katmanından ($i_0 – i_4$) alınan veriler, her katmanda f (Xw + b) formülüyle işlenir. Burada **ReLU, Tanh** veya **Sigmoid** gibi aktivasyon fonksiyonları kullanılarak bir tahmin (ŷ) üretilir.
2. **Loss Calculation:** Üretilen tahmin (ŷ) ile gerçek hedef değer (target) karşılaştırılır. Aradaki fark, **MSE** (regresyon için) gibi yöntemlerle bir "Kayıp" (L) değeri hesaplanır.
3. **Backward Propagation ve Optimizasyon:** Hesaplanan hata, ağın sonundan başına doğru yayılır. Zincir Kuralı (Chain Rule) kullanılarak her bir ağırlığın hatadaki payı (türevi: **∂L / ∂w**) hesaplanır. Ardından Gradiyent İniş (Gradient Descent) formülü ile ağırlıklar güncellenir:

$$w = w - \alpha * \partial L / \partial w$$

- **α (LR- Learning Rate):** Öğrenme hızıdır. Notta daire içine alınmış "LR" budur. Adımların ne kadar büyük olacağını belirler.
- **∂L / ∂w:** Gradyan (türev) değeridir. Hatanın hangi yöne doğru azaldığını gösterir.

Özetle: Sistem bir tahmin yapıyor, ne kadar hata yaptığını ölçüyor ve bu hatayı azaltmak için Gradient Descent kullanarak ağırlıklarını (w) ve bias (b) değerlerini güncelliyor. Bu döngü binlerce kez tekrarlanarak ağın "öğrenmesi" sağlanıyor.



Yapay Sinir Ağının (ANN) "Kalp Hastalığı Risk Tahmini" gibi gerçek bir dünya problemine nasıl uygulandığını adım adım gösteriyor. Bu süreç, ham verinin bir karara dönüşme yolculuğudur.

1. Veri Girişi ve Hazırlık (Clinical Data):
Sürecin ilk aşaması, ham verinin modele uygun hale getirilmesidir:

- **Clinical Data:** Hastadan alınan yaş, cinsiyet, kolesterol düzeyi, EKG bulguları ve Vücut Kitle Endeksi (BMI) gibi temel parametrelerdir.
- **Data Preprocessing:** Ham verideki eksik değerlerin tamamlanması, hatalı girişlerin ayıklanması ve verinin normalize edilmesi aşamasıdır.
- **Feature Selection / Extraction:** Tahmin başarısını en çok etkileyen kritik değişkenlerin (örneğin; yaş ve kolesterol kombinasyonu) belirlenerek modelin işleyebileceği matematiksel formata getirilmesidir.

2. Model Yapısı (ANN Classification)

- **Mimari Detayı (1 Input → 3H → 1 Output):** Bu gösterim, modelin bir giriş katmanı, üç adet Gizli Katman (Hidden Layer) ve bir çıkış katmanından oluştuğunu belirtir. Katman sayısının artması, modelin verideki karmaşık ilişkileri çözme yeteneğini artırır.
- Sınıflandırma Mantığı: Bu bir regresyon (sayı tahmini) modeli değil, Sınıflandırma (Classification) modelidir. Çıkış katmanı iki kesin sonuç üretir:
  - Heart disease present: Hastalık var.
  - Heart disease absent: Hastalık yok (Sağlıklı).

3. Öğrenme Grafiği (Loss vs Epoch)

- **Loss (Kayıp/Hata):** Modelin tahminleri ile gerçek sonuçlar arasındaki farktır. Amacımız bu değeri sıfıra yaklaştırmaktır.
- **Epoch:** Modelin tüm veri setini kaç kez taradığını temsil eder.
- **Grafiğin Yorumu:** Eğrinin aşağı yönlü hareketi, modelin her epoch sonunda hatalarından ders çıkardığını gösterir. En sonda hata payı minimize olur ve model kararlı hale gelir.

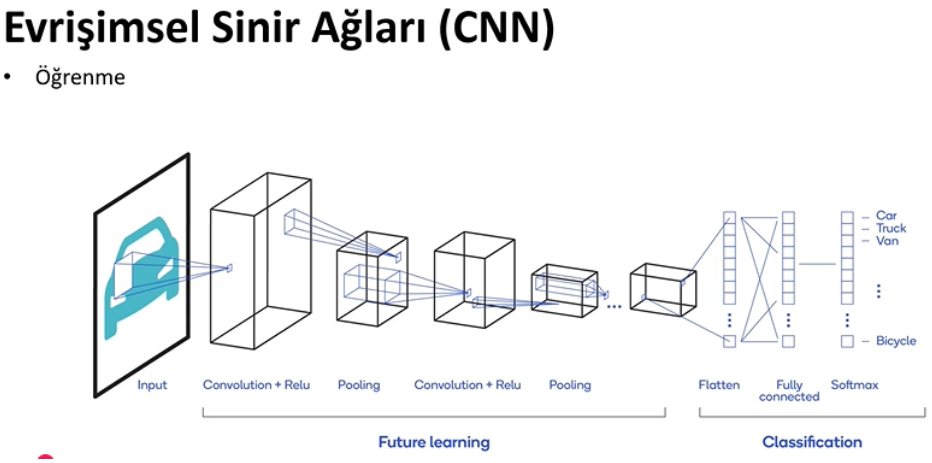
Küçük Bir Not
Önceki görselde gördüğümüz **Gradient Descent** ve **Backpropagation** mekanizmaları, tam olarak buradaki "Loss" (Hata) değerini düşürmek ve grafiği aşağı yönlü hareket ettirmek için arka planda çalışan matematiksel motorlardır.

2. Evrişimsel Sinir Ağları (Convolutional Neural Networks- CNN):

Tanım: Evrişimsel Sinir Ağları, görsel verileri doğrudan girdi olarak alıp işleyebilen, özellikle nesne algılama ve görüntü tanıma görevlerinde üstün başarı sergileyen bir derin öğrenme mimarisidir. Geleneksel yöntemlerin aksine, verideki öznitelikleri (kenarlar, dokular, şekiller) katmanlı yapısı sayesinde otomatik olarak öğrenir.

Kullanım Alanı: CNN'ler genellikle görsel veri analizi, yüz tanıma, otonom araçlar ve tıbbi görüntüleme gibi uygulamalarda yaygın olarak kullanılır.

Örnek: Bir şehir güvenlik sistemi, CNN modellerini kullanarak kamera görüntülerindeki nesneleri anlık olarak tanımlayabilir ve şüpheli bir durumu otomatik olarak yetkililere bildirebilir.



CNN mimarisi, görsel verileri analiz etmek için Özellik Öğrenme ve Sınıflandırma olmak üzere iki ana aşamadan oluşur.

1. Feature Learning: Bu aşamada model, ham piksellerden anlamlı desenleri otomatik olarak çıkarır:

- **Input:** Görüntü, sisteme renk kanallarından (RGB) oluşan bir sayısal matris olarak aktarılır.
- **Convolution (Evrişim) ve Filtreler:** Küçük filtreler (kernel) resim üzerinde gezinerek kenar, köşe ve doku gibi detayları yakalayıp **Özellik Haritaları (Feature Maps)** oluşturur.
- **Aktivasyon (ReLU):** $f(x) = \max(0, x)$ formülüyle çalışan ReLU, negatif değerleri sıfırlayarak ağa "lineer olmama" özelliği katar. Bu, modelin karmaşık desenleri öğrenmesi için kritiktir.
- **Pooling:** Veriyi sadeleştirir. Boyutu küçülterek işlem yükünü azaltır ve modelin en önemli detaylara odaklanmasını sağlayarak aşırı öğrenmeyi (overfitting) engeller.

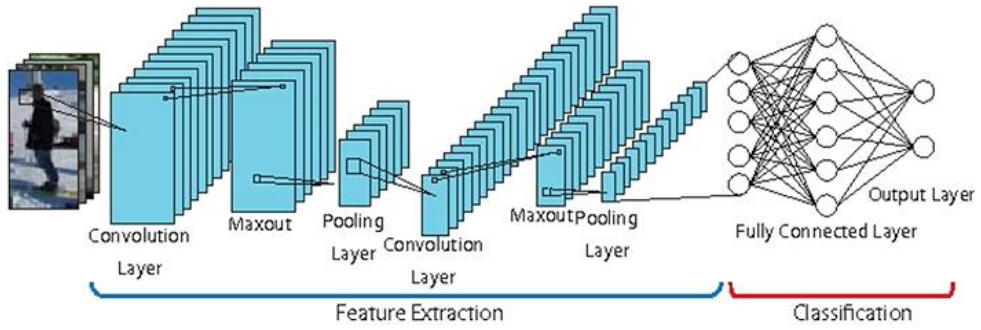
2. Classification: Özellikler çıkarıldıktan sonra, sistem bu bilgileri bir karara bağlamak için klasik sinir ağı yapısına geçer.

- **Flatten:** Çok boyutlu özellik haritaları, tek boyutlu bir sütun (vektör) haline getirilir.
- **Fully Connected Layer:** Klasik yapay sinir ağı gibi çalışır; tüm özellikler birbiriyle ilişkilendirilerek tahmin üretilir.
- **Softmax / Karar:** Çıkış katmanında olasılıksal bir hesaplama yapılır (Örn: %85 Araba, %10 Kamyon, %5 Bisiklet).

Özetle: Eğer veri bir tabloysa klasik **ANN**; veri bir görüntü veya karmaşık bir desen ise filtreleme yeteneği sayesinde **CNN** tercih edilir.

Evrişimsel Sinir Ağları (CNN) Gerçek Hayat Problemi

- Otonom Araçlarda Yaya Tespiti



Bu diyagram, otonom sürüş teknolojilerinde hayati önem taşıyan **Yaya Tespiti** probleminin bir CNN mimarisi üzerinden çözüm aşamalarını göstermektedir. Süreç temel olarak iki ana faza ayrılır:

1. Feature Extraction: Ham görsel verinin anlamlı desenlere dönüştürüldüğü aşamadır.

- **Giriş Katmanı:** Karlı sahnedeki yaya görseli, modele giren piksellerden oluşan ham veridir.
- **Convolution Layer (Evrişim Katmanı):** Görüntü üzerinde dolaşan matematiksel filtreler sayesinde kenarlar, köşeler ve dokular gibi düşük seviyeli özellikler ayırt edilir.
- **Maxout & Pooling:** Bu katmanlar veriyi sadeleştirir. Önemli bilgileri korurken işlem yükünü azaltır ve modelin nesneyi görüntünün farklı yerlerinde olsa bile tanımasını sağlar.
- **Hiyerarşik Öğrenme:** Şemada görülen tekrarlı katman yapısı, ağın derinleştikçe daha karmaşık detayları (önce çizgiler, sonra organlar, en son tüm bir vücut silueti) birleştirerek öğrenmesine olanak tanır.

2. Classification: Çıkarılan özelliklerin bir karara bağlandığı "akıl yürütme" aşamasıdır.

- **Fully Connected Layer (Tam Bağlantılı Katman):** Özellik çıkarımı aşamasından gelen 3 boyutlu veriler düzleştirilerek tek bir vektör haline getirilir. Buradaki her nöron, bir önceki katmandaki tüm nöronlarla bağlantılıdır.
- **Output Layer (Çıktı Katmanı):** En sondaki düğümler nihai sonucu üretir. Otonom araç için bu çıktı; "Bu bir yaya mı?" sorusuna %98 evet veya %2 hayır gibi bir olasılık değeri döndürür.

Özetle: Otonom araç kamerası görüntüyü aldığı anda CNN mekanizması devreye girer:

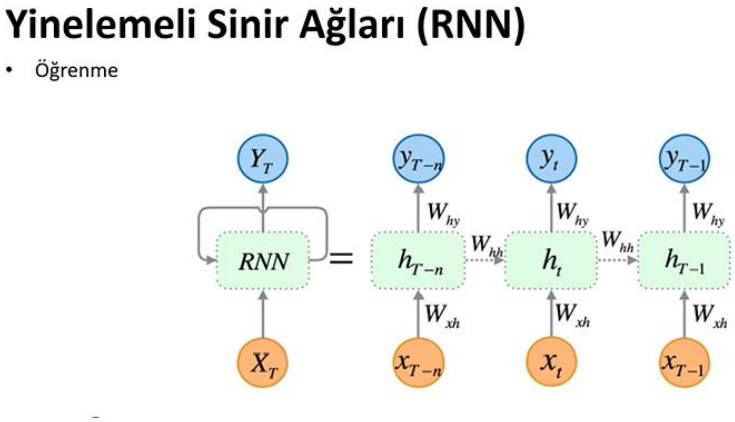
1. Evrişim (Convolution) ile temel hatlar belirlenir.
2. Havuzlama (Pooling) ile gereksiz gürültü ayıklanır.
3. Ağ, parçaları birleştirerek "burada bir insan figürü var" çıkarımında bulunur.
4. Son aşamada Sınıflandırma yapılarak aracın fren veya manevra sistemine anlık bilgi gönderilir.

3. Yinelemeli Sinir Ağları (Recurrent Neural Networks- RNN):

Tanım: Yinelemeli Sinir Ağları, ardışık ve zaman serisi verilerini işlemek üzere tasarlanmış bir derin öğrenme mimarisidir. Geleneksel sinir ağlarının aksine, RNN'ler geçmişteki bilgileri bir "bellek" mekanizması (hidden state) aracılığıyla tutarak, veriler arasındaki zamansal bağımlılıkları ve örüntüleri modelleyebilirler.

Kullanım Alanı: RNN'ler, doğal dil işleme, metin üretimi, konuşma tanıma ve finansal zaman serisi analizi gibi zaman bağımlı veri analizi gerektiren alanlarda kullanılır.

Örnek: Bir RNN modeli, bir metin belgesindeki dil yapısını analiz etmek ve ardından aynı dilde benzer metinler oluşturmak için kullanılabilir.



Bu görsel, RNN'lerin nasıl çalıştığını ve özellikle BPTT denilen öğrenme mekanizmasını anlatıyor.

1. Temel Bileşenler ve Akış: RNN, veriyi adım adım işler. Her adımda üç ana unsur bulunur:

- **X_t (Girdi):** Mevcut zaman adımındaki veri (örneğin dizideki bir kelime).
- **h_t (Hidden State):** Ağın hafızasıdır. Mevcut girdi (X_t) ile bir önceki adımın bilgisini (h_{t-1}) birleştirir.
- **y_t (Çıktı):** Ağın o zaman adımındaki tahmini.

2. Ağırlık Matrisleri (Parametreler): RNN'in en önemli özelliği, her adımda aynı ağırlıkları kullanmasıdır. Görseldeki W harfleri bunu temsil eder:

- **W_{xh}:** Girdiyi gizli duruma aktarır.
- **W_{hh}:** Geçmiş bilgiyi (hafızayı) bir sonraki adıma taşır.
- **W_{hy}:** Gizli durumdan çıktı üretir.

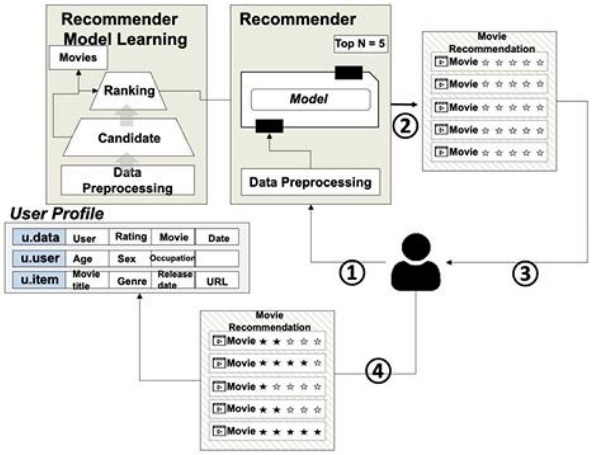
3. Öğrenme ve Kayıp Fonksiyonu (Loss): Her zaman adımında tahmin (ŷ) yapılır, gerçek değerle (y) karşılaştırılıp hata (L) hesaplanır. Total Loss: L(t) + L(t+1) = Loss ifadesi, tüm zaman adımlarındaki hataların toplanarak genel bir hata skoru oluşturulduğunu belirtir.

4. BPTT (Backpropagation Through Time): Hata hesaplandıktan sonra, gradyanlar (hatayı düzeltme sinyalleri) sondan başa doğru tüm zaman adımları boyunca geriye yayılır. Bu sayede ağ, sadece anlık girdiyi değil, geçmişteki bilgilerin (t-n) güncel tahmini nasıl etkilediğini de öğrenir.

Özetle: Bu görsel, bir RNN'in veriyi adım adım işlerken "hafızasını" nasıl kullandığını ve hataları geriye doğru yayarak geçmişteki bilgileri nasıl optimize ettiğini özetliyor.

Yinelemeli Sinir Ağları (RNN) Gerçek Hayat Problemi

- Film Öneri Sistemi



Bu diyagram, verinin kronolojik sırasını (izleme geçmişi) analiz ederek kişiselleştirilmiş tahminler yapan bir **Yinelemeli Sinir Ağı (RNN)** modelini özetlemektedir.

1. Modelin Hazırlanması (Recommender Model Learning):

Sol üstteki blok, sistemin "mutfağını" temsil eder. Ham veriler (film listeleri, kullanıcı tercihleri) önce ön işlemeden **(Data Preprocessing)** geçer. Ardından iki temel aşama gerçekleşir:

- **Candidate (Aday Belirleme):** Milyonlarca film arasından kullanıcının ilgisini çekebilecek birkaç yüz tanesi seçilir.
- **Ranking (Sıralama):** Bu adaylar, RNN algoritması ile kullanıcının geçmiş davranışlarına göre en yüksek puandan en düşüğe doğru sıralanır.

2. Öneri Süreci:

① **Kullanıcı Etkileşimi:** Kullanıcı sisteme giriş yapar veya bir eylemde bulunur. Sisteme gelen bu veri, modelin güncel durumuyla birleşir.

② **Öneri Üretimi:** "Recommender" bloğu, eğitilmiş modeli kullanarak kullanıcı için en iyi 5 filmi (**Top N = 5**) seçer.

③ **Kullanıcıya Sunum:** Seçilen bu filmler bir liste halinde kullanıcıya gösterilir.

④ **Geri Bildirim Döngüsü:** Kullanıcı bu önerilere bir puan verir veya izler (yıldızlarla gösterilen kısım). Bu yeni veri, "User Profile" kısmını günceller.

Neden RNN: Filmler söz konusu olduğunda sıradan bir yapay sinir ağı yerine RNN kullanılmasının sebebi zaman serisidir. RNN'ler geçmişteki seçimlerinizi bir "bellek" gibi tutarak, sadece genel tercihlerinizi değil, o anki "modunuzu" veya değişen zevklerinizi de anlayabilir.

Kullanılan Veri Yapıları (User Profile): Görselin sol altında yer alan tablo yapısı, sistemin hangi verileri eşleştirdiğini gösteriyor:

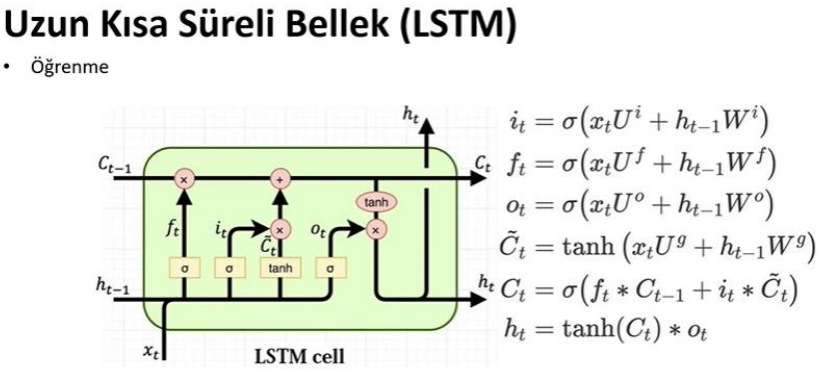
- **u.data:** Kullanıcının hangi filme, ne zaman, kaç puan verdiğini tutar.
- **u.user:** Kullanıcının yaşı, cinsiyeti ve mesleği gibi demografik bilgileri içerir.
- **u.item:** Filmin türü, çıkış tarihi ve linki gibi detayları barındırır.

4. Uzun Kısa Süreli Bellek (Long-Short Term Memory- LSTM):

Tanım: LSTM, geleneksel Yinelemeli Sinir Ağlarının (RNN) en büyük problemi olan "kaybolan gradyanlar" (vanishing gradients) ve kısa süreli bellek sorununu çözmek için tasarlanmış özel bir RNN mimarisidir. Standart sinir ağlarının aksine, verideki uzak geçmiş ile güncel bilgi arasındaki bağıntıyı koparmadan bilgiyi uzun süreler boyunca taşıyabilir.

Kullanım Alanı: Dil modelleri, metin analizi, otomatik metin tamamlama ve hisse senedi fiyat tahminleri gibi uzun vadeli bağımlılıkların olduğu alanlarda kullanılır.

Örnek: Bir LSTM modeli, uzun bir makaleyi baştan sona analiz ederek konunun bağlamını yitirmeden anlamlı otomatik özetler oluşturabilir.



LSTM, standart RNN'lerin "uzun vadeli hafıza kaybı" sorununu çözmek için tasarlanmış, **kapı (gate)** mekanizmalarına sahip özel bir mimaridir.

1. Temel Bileşenler

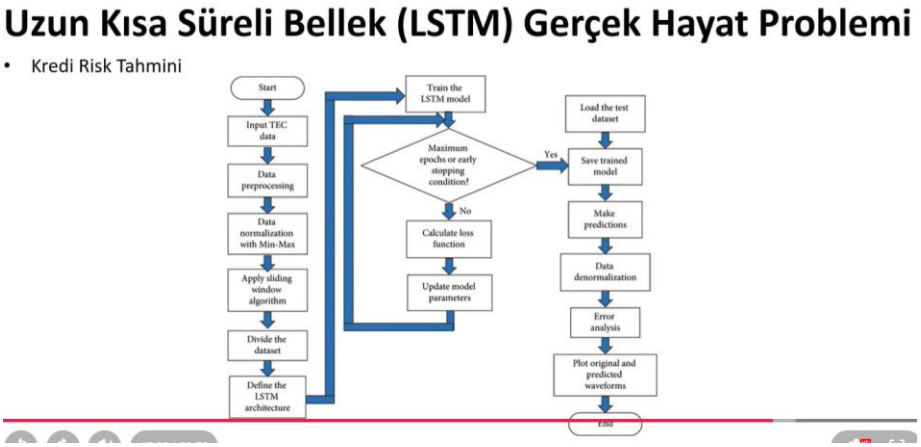
- **Cell State (C_t):** Hücrenin "uzun süreli belleği"dir. Bilgiler burada çok az değişiklikle akar, böylece model geçmişteki önemli bir bilgiyi çok sonraya taşıyabilir.
- **Hidden State (h_t):** Hücrenin o anki çıktısı ve kısa süreli belleğidir.
- **Girişler (x_t):** O anki zaman adımında modele giren yeni veri.

2. Kapı Mekanizmaları

- **Unutma Kapısı (Forget Gate- f_t):** Geçmişten gelen bilgilerin ne kadarının silineceğine karar verir. Nottaki [0, 1] ifadesi bunu simgeler; 0 ise tamamen unut, 1 ise tamamen tut demektir.
- **Giriş Kapısı (Input Gate- i_t):** Yeni gelen verinin ne kadarının belleğe ekleneceğine karar verir. Notta "bilgi ekle" olarak işaretlenmiş.
- **Çıkış Kapısı (Output Gate- o_t ve h_t):** Hücrenin o anki dış dünyaya ne cevap vereceğini belirler. C_t (güncel bellek) bir tanh filtresinden geçer ve çıkış kapısıyla çarpılarak nihai h_t sonucunu oluşturur.

3. Matematiksel Güncelleme Özeti

- **Karar Ver:** Eski belleği ne kadar unut? (f_t)
- **Aday Belirle:** Yeni bilgiden ne ekle? ($i_t * C'_t$)
- **Güncelle:** Eski belleği bul ve üzerine yeni bilgiyi ekle:
- $C_t = f_t * C_{t-1} + C'_t$
- **Yansıt:** Güncel belleği tanh filtresinden geçirerek çıktıyı (h_t) oluştur.



Görseldeki akış şeması, LSTM mimarisi kullanılarak hazırlanan bir "Kredi Risk Tahmini" modelinin nasıl oluşturulduğunu ve eğitildiğini adım adım anlatıyor. Bu süreç, zaman serisi verileriyle çalışan tipik bir makine öğrenmesi iş akışıdır.

1. Veri Hazırlama: Modelin öğrenmeye hazır hale getirildiği ön işleme aşamasıdır:

- **Input TEC data:** Sisteme giriş verileri (kredi geçmişi veya ekonomik göstergeler) yüklenir.
- **Data preprocessing:** Eksik verilerin tamamlanması veya hatalı verilerin temizlenmesi.
- **Data normalization with Min-Max:** Veriler, modelin daha hızlı ve kararlı öğrenmesi için genellikle 0 ile 1 arasına sıkıştırılır.
- **Apply sliding window algorithm:** LSTM'in geçmiş verileri hatırlayabilmesi için veriler "kayan pencere" yöntemiyle sıralı bloklar haline getirilir (Örn: Son 12 ayın verisine bakarak 13. ayı tahmin etmek).
- **Divide the dataset:** Veri seti; Eğitim (Training) ve Test olarak ikiye ayrılır.
- **Define the LSTM architecture:** Katman sayısı, hücre sayısı gibi modelin yapısı belirlenir.

2. Eğitim Döngüsü: Bu kısım, modelin hatasından ders çıkararak kendini geliştirdiği döngüdür:

- **Train the LSTM model:** Eğitim verileri modele verilir.
- **Maximum epochs or early stopping?** (Karar Mekanizması): Belirlenen tur sayısına (epoch) ulaşıldı mı veya model artık gelişmeyi bıraktı mı (**Early Stopping**)?
    - **Hayır (No) ise:** Kayıp fonksiyonu (Loss function) hesaplanır, model parametreleri (ağırlıklar) güncellenir ve eğitim devam eder.
    - **Evet (Yes) ise:** Eğitim tamamlanır ve en iyi model kaydedilir (**Save trained model**).

3. Test ve Değerlendirme: Modelin başarısının ölçüldüğü son aşamadır:

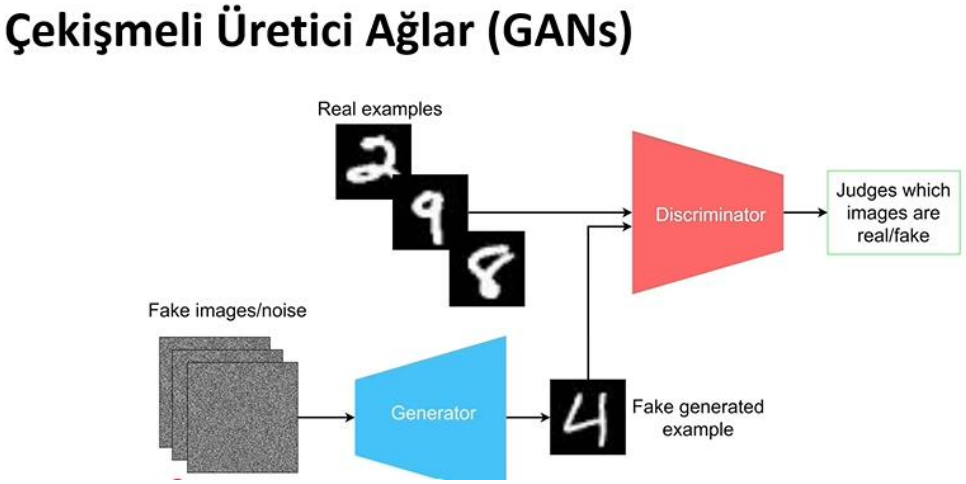
- **Load the test dataset:** Modelin daha önce hiç görmediği "test verileri" yüklenir.
- **Make predictions:** Model bu verilere bakarak kredi riski tahminleri yapar.
- **Data denormalization:** Normalleştirilmiş (0-1 arası) tahminler, gerçek dünya değerlerine (tekrar anlamlı rakamlara) geri dönüştürülür.
- **Error analysis:** Gerçek değerler ile tahmin edilen değerler arasındaki fark (hata payı) ölçülür.
- **Plot original and predicted waveforms:** Tahminler ile gerçek veriler grafik üzerinde karşılaştırılarak görsel bir analiz yapılır.

5. Çekişmeli Üretici Ağlar (Generative Adversarial Networks- GANs):

Tanım: Çekişmeli Üretici Ağlar, birbirine karşı çalışan iki yapay sinir ağı kullanılarak eğitilen bir derin öğrenme modelidir. Bu ağlardan biri veri üretir (generator – üretici), diğeri ise üretilen verinin gerçek mi yoksa sahte mi olduğunu değerlendirir (discriminator – ayırt edici).

Kullanım Alanı: GAN'lar, çeşitli alanlarda uygulama bulur, bunlar arasında görsel içerik oluşturma, video prodüksiyonu, yaratıcı sanat projeleri ve veri zenginleştirme yer alır.

Örnek: Bir GAN modeli, yüksek kaliteli, gerçekçi insan yüzü fotoğrafları üretmek için kullanılabilir.



Bu şema, Çekişmeli Üretici Ağlar (Generative Adversarial Networks- GANs) mimarisini anlatıyor.

1. Üretici (Generator): "Sahteci": Görevi, gerçek verilere (örneğin MNIST setindeki 28x28 piksellik rakamlar) o kadar çok benzeyen görüntüler üretmektir ki, kimse bunların sahte olduğunu anlamasın.

- **Giriş:** Rastgele gürültü (noise) alır.
- **Çıkış:** Yeni bir görüntü üretir (örneğin görseldeki '4' rakamı).

2. Ayırt Edici (Discriminator): "Dedektif": Görevi, önüne gelen görüntünün Gerçek mi yoksa Sahte mi olduğunu tahmin etmektir.

- **Giriş:** Hem gerçek veri setinden gelen örnekleri (Real examples) hem de Generator'ın ürettiği sahte örnekleri alır.
- **Çıkış:** Bir olasılık değeri döndürür (örneğin: "Bu %90 ihtimalle gerçektir").

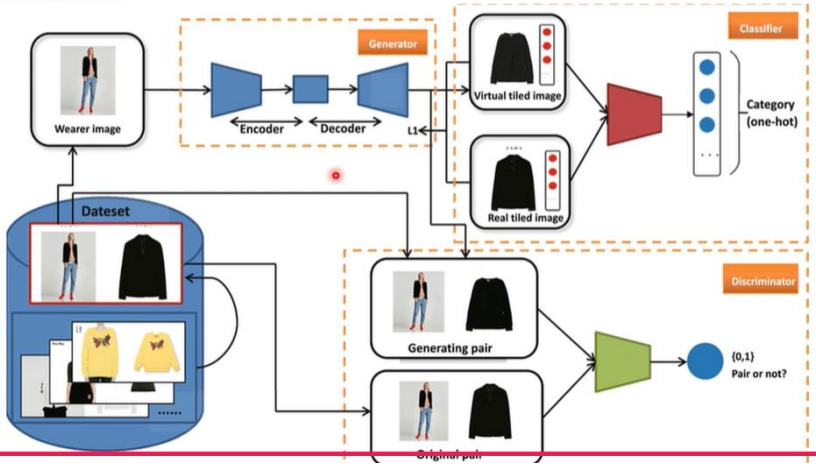
Bu Süreç Nasıl Çalışır? (Eğitim Döngüsü)

1. **Discriminator** gerçek ve sahte veriyi öğrenir. Gerçek rakamları gördüğünde "Gerçek", sahteleri gördüğünde "Sahte" demeye çalışır.
2. **Generator**, Discriminator'ı kandırmaya çalışır. Eğer Discriminator sahte bir görüntüye "Gerçek" derse, Generator kazanmış sayılır.
3. **Gelişim:** Discriminator kandırıldıkça daha dikkatli olmayı öğrenir; Generator ise Discriminator'ı kandıramadıkça daha gerçekçi görüntüler üretmeyi öğrenir.

Sonuç: Oyunun sonunda Generator o kadar başarılı olur ki, ürettiği sahte rakamlar gerçeklerinden ayırt edilemez hale gelir. Görseldeki el yazısı notta belirtildiği gibi; Generator "görüntü üretmeyi", Discriminator ise "gerçek ve sahte veriyi ayırt etmeyi" öğrenir.

Çekişmeli Üretici Ağlar (GANs) Gerçek Hayat Problemi

- Moda Ürünleri Tasarımı



Bu şema, GAN mimarisinin moda dünyasındaki çok özel bir uygulama alanını, yani "Kıyafet Tasarımı ve Sanal Deneme" (Virtual Try-On) sürecini açıklıyor. Sistem, bir kişinin fotoğrafını alıp üzerindeki kıyafeti analiz ederek, o kıyafetin tek başına nasıl görüneceğini (veya tam tersini) üretmeye çalışır.

1. Veri Seti ve Girdi (Dataset & Input): "Dateset" (Veri Seti) ile başlar. Burada iki tip veri var:

- **Wearer image:** Kıyafeti üzerinde taşıyan kişinin fotoğrafı.
- **Real tiled image:** Aynı kıyafetin düz bir zeminde çekilmiş profesyonel fotoğrafı. Bu, sistemin ulaşmaya çalıştığı "altın standart"tır.

2. Üretici Bölüm (Generator): Üst kısımdaki turuncu çerçeve, yeni görüntüler oluşturan kısımdır:

- **Encoder-Decoder:** Önce "Wearer Image"daki görsel veriyi sıkıştırarak anlamlandırır (Encoder), sonra bu veriden sadece kıyafeti ayıklayarak yeni bir görsel oluşturur (Decoder).
- **Virtual tiled image:** Jeneratör, kişinin üzerindeki kıyafete bakarak o kıyafetin sanki yere serilmiş gibi bir "sanal versiyonunu" üretir.

3. Sınıflandırıcı (Classifier): Üretilen kıyafetin doğru kategoride olup olmadığını kontrol eder:

- Üretilen sanal görüntü ile gerçek görüntü kıyaslanır.
- **Category (one-hot):** Sistemin, "Bu bir ceket mi, pantolon mu?" sorusuna doğru yanıt verip vermediği ölçülür.

4. Ayırt Edici (Discriminator- GAN'ın Kalbi): Yeşil bölge, sistemin "dedektifidir":

- **Generating pair:** Jeneratörün ürettiği çift.
- **Original pair:** Gerçek veri setindeki orijinal çift.
- **Pair or not? {0,1}:** Ayırt edici, önüne gelen görüntünün gerçek bir moda çekimi mi yoksa yapay zekâ tarafından uydurulmuş bir görsel mi olduğunu tahmin etmeye çalışır.

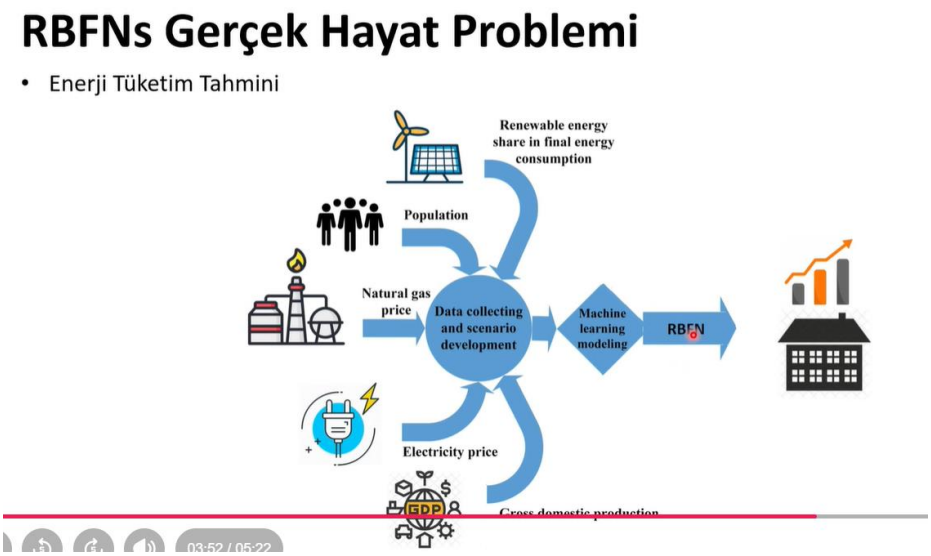
Özetle: Generator, Discriminator'ı (Ayırt ediciyi) kandıracak kadar gerçekçi kıyafet görselleri üretmeye çalışır. Discriminator ise hangisinin sahte olduğunu anlamaya çalışır. Bu çekişme sonucunda sistem, bir fotoğraftaki kıyafeti profesyonel bir tasarımcı gibi analiz edip yeniden oluşturmayı öğrenir.

6. Radyal Temelli Fonksiyon Ağları (Radial Basis Function Networks- RBFNs):

Tanım: Radyal Temelli Fonksiyon Ağları, giriş verilerini yüksek boyutlu bir uzaya taşıyarak doğrusal olmayan problemleri çözmek için tasarlanmış özel bir yapay sinir ağı türüdür. Klasik çok katmanlı algılayıcılardan (MLP) farkı, gizli katmanlarında aktivasyon fonksiyonu olarak genellikle Gauss fonksiyonu gibi "radyal simetrik" fonksiyonlar kullanmasıdır.

Kullanım Alanı: RBFN'ler, çeşitli alanlarda uygulanabilir; bunlar arasında zaman serisi tahmini, veri sınıfandırma, sistem modelleme ve veri sıkıştırma ve gürültü giderme gibi görevler bulunur.

Örnek: Bir RBFN modeli, bir pazarlama kampanyası başarısını tahmin etmek amacıyla kullanılabilir.



Bu model, Radyal Tabanlı Fonksiyon Ağları kullanarak karmaşık verilerden enerji tüketim tahmini üretme sürecini özetlemektedir.

1. Giriş Verileri (Girdi Parametreleri): Enerji tüketimini etkileyen ana faktörler toplanır. Bunlar modelin "öğrenmesi" gereken ham verilerdir:

- **Yenilenebilir Enerji Payı:** Toplam tüketim içindeki yeşil enerji oranı.
- **Nüfus (Population):** İnsan sayısı arttıkça doğal olarak enerji ihtiyacı da artar.
- **Doğal Gaz Fiyatı:** Alternatif enerji kaynaklarının fiyatı, elektrik kullanım alışkanlıklarını değiştirir.
- **Elektrik Fiyatı:** Fiyat artışı genellikle tasarrufu veya verimlilik arayışını tetikler.
- **GSYİH (Gross Domestic Product):** Ülkenin ekonomik üretimi; sanayi faaliyetleri ne kadar fazlaysa enerji tüketimi o kadar yüksektir.

2. Veri İşleme ve Modelleme

- **Veri Hazırlama:** İlk aşamada bu farklı kaynaklardan gelen veriler temizlenir ve modele uygun hale getirilir. Sonra devreye matematiksel algoritmalar girer.
- **RBFN:** Veriler arasındaki doğrusal olmayan karmaşık ilişkileri hızlıca çözer. Girdileri alır ve bunları yüksek boyutlu bir alana haritalayarak aralarındaki bağlantıları "anlar".

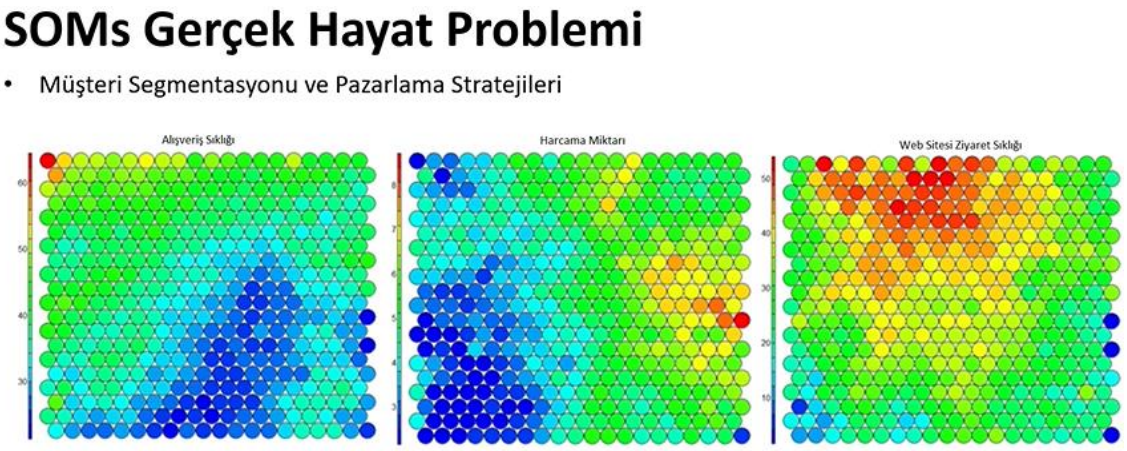
3. Çıktı (Tahmin): Model, geçmiş verileri ve girdi parametrelerini kullanarak; "Gelecek yıl nüfus şu kadar artarsa ve fiyatlar bu seviyede olursa, şehrin/ülkenin ne kadar enerjiye ihtiyacı olacak?" sorusuna yanıt verir.

7. Kendini Düzenleyen Haritalar (Self-Organizing Maps- SOMs):

Tanım: Veri setindeki karmaşık ilişkileri, benzerlikleri ve farklılıkları topolojik bir düzen içinde görselleştirmek için kullanılan bir **denetimsiz öğrenme** yöntemidir. Temel amacı, yüksek boyutlu verileri, aralarındaki geometrik ilişkileri koruyarak düşük boyutlu bir haritaya indirgemektir.

Kullanım Alanı: SOM'lar, veri görselleştirme, veri analizi, kümeleme, boyut indirgeme, desen tanıma gibi çeşitli uygulamalarda etkili bir şekilde kullanılabilir.

Örnek: Bir SOM modeli, bir müşteri veri setindeki benzer özelliklere sahip müşteri gruplarını belirlemek ve bu grupları görsel olarak analiz etmek için kullanılabilir.



Görselde, Self-Organizing Maps kullanılarak yapılmış bir "Müşteri Segmentasyonu" analizi yer alıyor. SOM, karmaşık verileri iki boyutlu bir harita üzerinde görselleştirerek benzer özelliklere sahip verileri (bu durumda müşterileri) birbirine yakın kümeleyen bir yapay sinir ağı türüdür.

Değişkenlerin Analizi

- **Alışveriş Sıklığı:** Sol üst köşedeki kırmızı alan, çok sık alışveriş yapan sadık bir kitleyi; sağ alt mavi alan ise seyrek alışveriş yapanları gösterir.
- **Harcama Miktarı:** Sağ orta bölgedeki sıcak renkler yüksek harcama yapan (VIP) grubu, sol alt ise harcaması çok düşük olan müşteri grubunu temsil eder.
- **Web Sitesi Ziyaret Sıklığı:** Üst orta bölgedeki yoğunluk, siteyi aktif kullanan ancak her zaman satın alma yapmayan "incelemeci" kitleyi işaret eder. Alt bölgeler ise web sitesini pek kullanmayan, muhtemelen doğrudan fiziksel mağazadan alışveriş yapan kitleyi temsil eder.

Segmentasyon ve Strateji Önerileri:

Müşteri Grubu	Özellikleri	Pazarlama Stratejisi
Şampiyonlar	Hem sık alışveriş yapan hem yüksek harcama yapanlar.	Özel sadakat programları ve kişiye özel VIP davetler.
Potansiyel Sadıklar	Siteyi çok ziyaret eden ama harcaması düşük olanlar.	Satın almaya ikna edecek indirim kuponları veya "sepette unutma" hatırlatıcıları.
Riskli Grup	Hem siteye az giren hem de az harcama yapanlar.	Yeniden kazanma (re-activation) kampanyaları ve "Seni Özledik" mailleri.

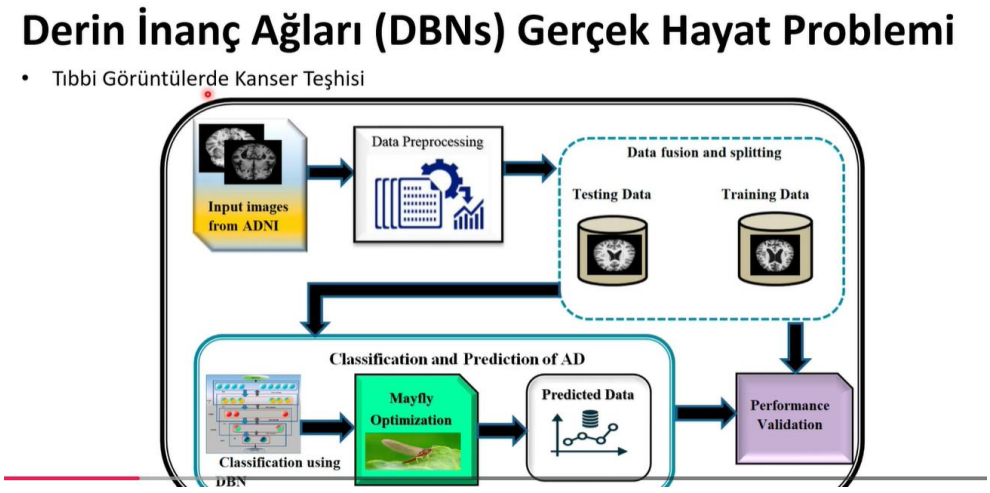
Özetle: SOM analizi, işletmenin "Kimin harcaması yüksek ama siteye az giriyor?" veya "Kim sadece bakıyor ama almıyor?" gibi soruları yanıtlayarak pazarlama bütçesini doğru yere harcamasını sağlar.

8. Derin İnanç Ağları (Deep Belief Networks- DBNs):

Tanım: Derin İnanç Ağları (DBN), denetimsiz öğrenme prensibiyle çalışan, üst üste istiflenmiş Kısıtlanmış Boltzmann Makineleri'nden (RBM) oluşan çok katmanlı üretken yapay sinir ağı modelidir. Temel işlevi, verideki karmaşık yapıları hiyerarşik bir şekilde modelleyerek yüksek seviyeli özellik çıkarımı ve sınıflandırma yapmaktır.

***Kullanım Alanı*:** DBN'ler, ses tanıma, görüntü işleme, biyomedikal mühendislik ve doğal dil işleme gibi çeşitli alanlarda etkili bir şekilde kullanılabilir.

Örnek: Bir DBN modeli, tıbbi görüntü veri setlerinde lezyonları sınıflandırmak ve teşhis etmek amacıyla kullanılabilir.



Görseldeki akış diyagramı, Derin İnanç Ağları kullanılarak tıbbi görüntüler üzerinden bir hastalık teşhis sisteminin nasıl çalıştığını özetliyor. İlginç bir detay var: Başlıkta "Kanser Teşhisi" yazsa da diyagramın içindeki veriler (ADNI veri seti ve "AD" ifadesi) aslında Alzheimer Hastalığı teşhisine odaklandığını gösteriyor. Muhtemelen bu yöntem her iki alan için de genel bir örnek olarak sunulmuş.

1. Veri Girişi ve Ön İşleme (Data Input & Preprocessing): Süreç, beyin MR görüntülerinin sisteme yüklenmesiyle başlar. Data Preprocessing aşamasında; gürültü giderme, boyutlandırma ve görüntü netleştirme gibi işlemlerle ham veri, analiz edilebilir "temiz" bir formata getirilir.

2. Veri Hazırlığı (Data Fusion and Splitting): Hazırlanan veriler ikiye ayrılır:

- **Training Data (Eğitim Verisi):** Modelin öğrenmesi için kullanılan büyük veri kümesi.
- **Testing Data (Test Verisi):** Modelin performansını ölçmek için kullanılan bağımsız veri seti.

3. Sınıflandırma ve Tahmin: Bu bölüm sistemin karar mekanizmasıdır (beynidir de diyebiliriz):

- **Classification using DBN:** Derin İnanç Ağları, görüntüdeki patolojik özellikleri analiz eder.
- **Mayfly Optimization:** Mayfly algoritması, sistemin doğruluğunu artırmakta kullanılan bir optimizasyon tekniğidir. Yapay zekâ parametrelerini en iyi sonucu verecek şekilde düzenler.
- **Predicted Data:** Analiz sonucunda nihai teşhis tahmini üretilir.

4. Performans Doğrulama (Performance Validation): Bu aşamada, sistemin yaptığı tahminler gerçek sonuçlarla karşılaştırılır. Doğruluk oranı, hassasiyet ve hata payı gibi metrikler hesaplanarak modelin başarısı ölçülür.

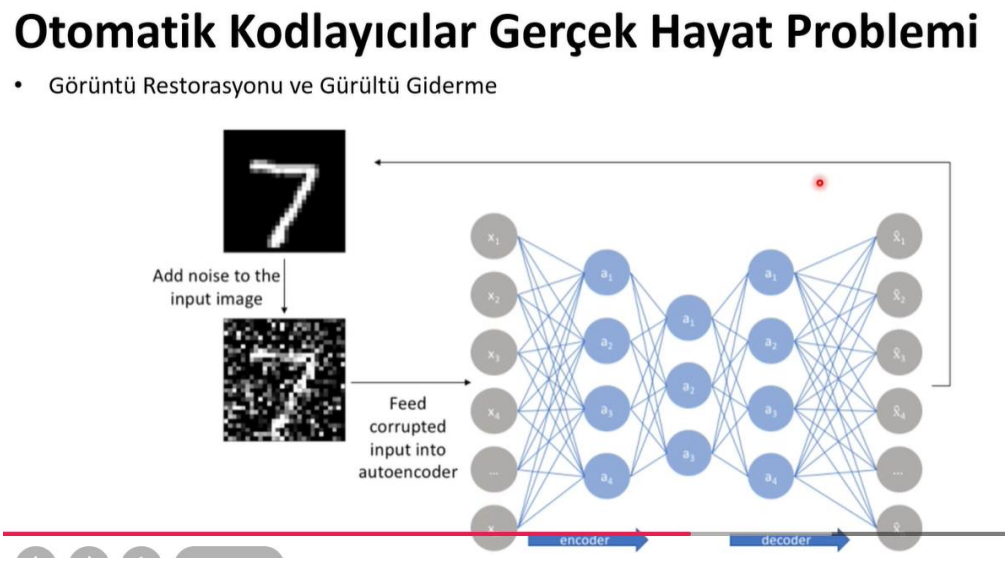
Özetle: Bu görsel, tıbbi görüntüleri alıp onları karmaşık matematiksel modellerden (DBN ve Mayfly) geçirerek doktorlara yardımcı olacak güvenilir bir teşhis sonucu üretme sürecini anlatıyor.

9. Otomatik Kodlayıcılar (Auto Encoders):

Tanım: Otomatik Kodlayıcılar, giriş verilerini denetimsiz bir şekilde öğrenerek önce düşük boyutlu bir temsile sıkıştıran (Encoder), ardından bu sıkıştırılmış veriden orijinal girişi yeniden inşa etmeye çalışan (Decoder) özel bir yapay sinir ağı mimarisidir. Temel amacı, veri içerisindeki gürültüyü ayıklayarak en anlamlı ve karakteristik özelliklerin bulunduğu bir bottleneck katmanı oluşturmaktır.

Kullanım Alanı: Otomatik Kodlayıcılar, veri sıkıştırma, görüntü restorasyonu, boyut azaltma, özellik çıkarımı, anomali tespiti gibi çeşitli uygulamalarda kullanılır.

Örnek: Bir otomatik kodlayıcı modeli, bir görüntüdeki önemli ve gizli özellikleri öğrenmek ve ardından bu bilgileri kullanarak görüntüyü yeniden oluşturmak için kullanılabilir.



Görselde, bozulmuş veriyi orijinal haline döndürmeyi öğrenen **Denoising Autoencoder** mekanizması açıklanmaktadır:

1. Veri Hazırlama (Gürültü Ekleme): Süreç, temiz bir giriş verisine ("7" rakamı) bilinçli olarak rastgele gürültü eklenmesiyle başlar. Amaç, yapay zekayı kirli veriden temiz veriyi ayırt etmeye zorlamaktır.

2. Encoder (Kodlayıcı): Kirli görüntü ağa girer ve daralan katmanlar boyunca **sıkıştırılır**. Bu "darboğaz" (bottleneck) noktasında ağ, gereksiz gürültüleri ayıklayarak verinin sadece en temel özelliklerini (rakamın formu, eğimi vb.) saklar.

3. Decoder (Kod Çözücü): Sıkıştırılmış olan öz bilgi, genişleyen katmanlar aracılığıyla tekrar orijinal boyutuna döndürülür. Ağ, elindeki kısıtlı bilgiyle görüntüyü gürültüsüz şekilde yeniden inşa eder.

4. Çıktı ve Öğrenme (Rekonstrüksiyon): Oluşturulan çıktı, orijinal temiz veriyle kıyaslanır. Aradaki fark (hata payı), ağın başarısını ölçer ve model bu hatayı minimize ederek öğrenmesini tamamlar.

Özetle Ne İşe Yarar?

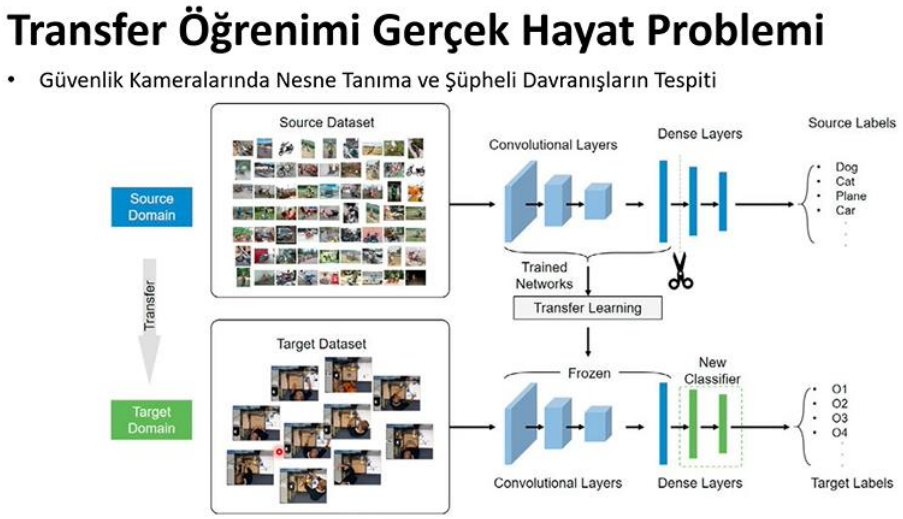
- **Eski Fotoğrafların Restorasyonu:** Çizilmiş veya kumlanmış eski fotoğrafları temizlemek.
- **Tıbbi Görüntüleme:** MR veya röntgen çekimlerindeki parazitleri yok ederek daha net teşhis koymak.
- **Veri Sıkıştırma:** Veriyi en küçük ama en anlamlı haliyle saklamak.

10. Transfer Öğrenimi (Transfer Learning):

Tanım: Transfer Öğrenimi, önceden büyük bir veri seti üzerinde eğitilmiş ve belirli yetenekler kazanmış bir modelin (pre-trained model), elde ettiği bu bilgileri (özellikleri, ağırlıkları) benzer veya ilişkili yeni bir görevde kullanması sürecidir. Temel amaç, sıfırdan öğrenmek yerine mevcut bilgi birikimini yeni bir alana transfer ederek öğrenme sürecini hızlandırmak ve performansı artırmaktır.

Kullanım Alanı: Transfer öğrenimi, genellikle sınırlı veri veya etiketli veri eksikliği durumlarında etkili bir şekilde kullanılır. Bilgisayarlı görü, doğal dil işleme ve tıbbi teşhis gibi alanlarda kullanılır.

Örnek: Bir transfer öğrenimi modeli, bir tür kanser teşhisinde kullanılan bir görüntü sınıflandırma modelinde elde edilen özellikleri, başka bir kanser türünün teşhisinde kullanmak için uygulanabilir.



1. Süreç Nasıl İşler?

- **Source Domain:** Model önce milyonlarca genel görselle (kedi, araba) eğitilir. Bu aşamada Evrişimli Katmanlar (Convolutional Layers) kenarları, renkleri ve formları tanımayı öğrenir.
- **Kesim İşlemi (Makas Simgesi):** Model eğitildikten sonra, genel nesneleri tanıma yeteneği kazanmış olan katmanlar alınır, ancak en sondaki sınıflandırma kısmı (etiketler) atılır.
- **Transfer Süreci:** Ortadaki ok ve makas simgesi, "bilginin transferini" temsil eder. Modelin genel görme yeteneği (bir nesneyi tanıma yetisi) korunur ve yeni göreve aktarılır.
- **Target Domain:** Model artık yeni görevi olan "Güvenlik Kamerası Analizi"ne odaklanır.

2. Uygulama Adımları:

- **Dondurulmuş Katmanlar (Frozen):** Kaynak alandan gelen katmanlar "dondurulur". Yani modelin temel görme yeteneği tekrar eğitilmez, olduğu gibi kullanılır.
- **Yeni Sınıflandırıcı:** Modelin sonuna sadece yeni göreve (örneğin: şüpheli hareket/normal hareket) odaklanan küçük bir katman eklenir ve sadece bu kısım eğitilir.

Neden Bu Yöntem Kullanılır?

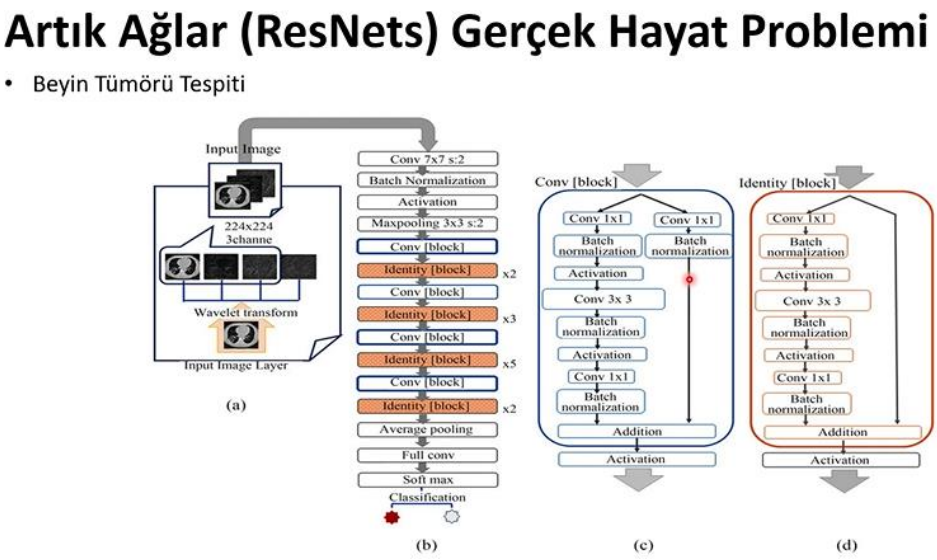
- **Az Veriyle Başarı:** Şüpheli davranışlara ait elinizde çok az görüntü olsa bile, model genel görmeyi zaten bildiği için hızlıca öğrenir.
- **Hız ve Tasarruf:** Sıfırdan eğitim haftalar sürerken, bu yöntemle saatler içinde sonuç alınır; donanım maliyeti düşer.

11. Artık Ağlar (Residual Networks- ResNets):

Tanım: ResNets, derin sinir ağlarında katman sayısı arttıkça eğitimin zorlaşması ve performansın düşmesi sorununu gidermek için geliştirilmiş bir mimaridir. Temel yeniliği, "Artık Bağlantılar" veya "Atlamalı Bağlantılar" kullanarak, bilginin bir veya birden fazla katmanı atlayarak doğrudan daha derin katmanlara aktarılmasını sağlamaktır.

Kullanım Alanı: ResNets, görüntü sınıflandırma, nesne tespiti, görüntü segmentasyonu, video analizi ve tıbbi görüntüleme gibi görevlerde yaygın olarak kullanılır.

Örnek: Bir ResNet modeli, bir görüntüdeki farklı nesneleri tanımlamak ve sınıflandırmak için kullanılabilir.



Bu mimari, derin ağlarda görülen gradyan yok olması (vanishing gradient) problemini "atlama bağlantıları" (skip connections) ile çözerek tıbbi görüntülemede yüksek doğruluk sağlar.

1. Giriş ve Ön İşleme (Input Image Layer)

- **Giriş:** 224x224 boyutunda, 3 kanallı beyin MR görüntüleri kullanılıyor.
- **Wavelet Transform:** Standart ResNet'ten farklı olarak, görüntüdeki detay ve kenar bilgilerini belirginleştirmek için dalgacık dönüşümü uygulanmıştır.

2. Ana Mimari Akışı (ResNet Omurgası)

- **Kök Katmanlar:** 7x7 Conv, Batch Norm ve Maxpooling ile temel özellikler yakalanır.
- **Tekrarlayan Bloklar:** Görseldeki x2, x3, x5 ifadeleri, blokların derinliği artırmak için kaç kez üst üste bindiğini gösterir.
- **Sınıflandırma:** En altta Average Pooling ve Softmax katmanları ile nihai teşhis yapılır.

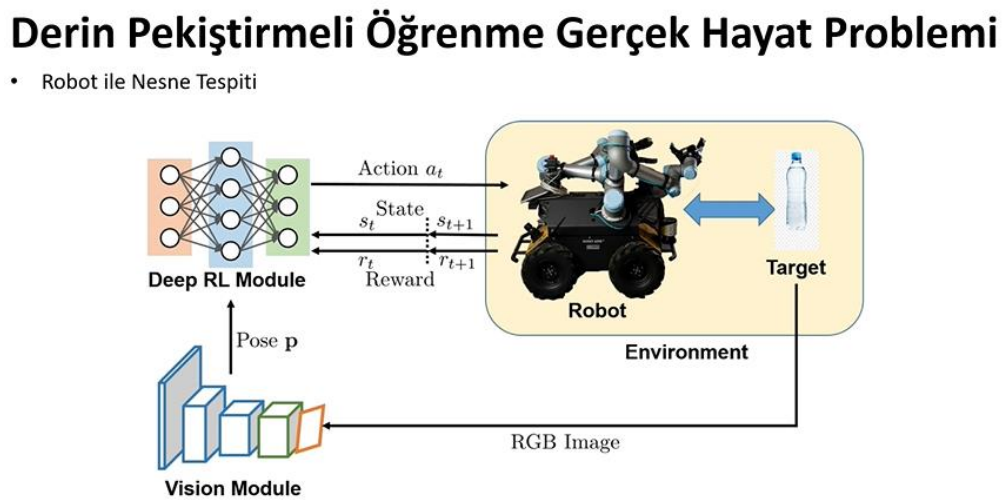
Özellik	Conv [block] (Mavi)	Identity [block] (Turuncu)
Giriş-Çıkış Boyutu	Girişin boyutu ve kanal sayısı değişir (genelde küçülür).	Giriş ve çıkış boyutu **aynıdır**.
Kısa Yol (Shortcut)	Atlanan yolda bir Conv 1x1 vardır. Bu, boyutları eşitlemek içindir.	Doğrudan bir çizgi vardır. Giriş, hiçbir değişikliğe uğramadan çıkışa eklenir.
Amacı	Özellik haritalarını küçültmek ve derinliği artırmak.	Bilginin kaybolmadan (degrade olmadan) derin katmanlara akmasını sağlamak.

12. Derin Pekiştirmeli Öğrenme:

***Tanım**:* Derin Pekiştirmeli Öğrenme, bir yapay zekâ ajanının karmaşık bir çevrede (environment) en yüksek ödülü (reward) toplamak amacıyla, kendi deneyimlerinden öğrenerek karar verme sürecini optimize etmesidir.

Kullanım Alanı: Derin Pekiştirmeli Öğrenme, oyun stratejileri, robotik kontrol, finansal ticaret, otomatik araçlar ve daha birçok alanda uygulanabilir.

***Örnek**:* Bir derin pekiştirmeli öğrenme modeli, bir otonom aracın çevresini algılayarak güvenli bir şekilde yön bulmasını ve trafik koşullarına uygun hareket etmesini sağlamak için kullanılabilir.



1. Ana Bileşenler

- **Robot & Ortam (Environment):** Robot, bir hedef nesneye ulaşmaya çalışıyor. Ortam, robotun içinde bulunduğu ve etkileşime girdiği fiziksel dünyadır.
- **Vision Module:** Robotun "gözü" gibi çalışır. Hedefin ve çevrenin görüntüsünü (RGB Image) alır. Genellikle bir CNN kullanılarak görüntüden anlamlı veriler (pozisyon- Pose p) çıkarır.
- **Deep RL Module (Derin Pekiştirmeli Öğrenme Modülü):** Robotun beynidir. Görüntüden gelen pozisyon bilgisini ve mevcut durum verilerini alarak hangi hareketi yapacağına karar verir.

2. Döngüsel İşleyiş (Dönüt Mekanizması)

1. **Gözlem (State s_t):** Robot o anki durumunu (nerede olduğunu, hedefi ne kadar gördüğünü) algılar.
2. **Aksiyon (Action a_t):** Beyin (RL Modülü), duruma bakarak bir hareket emri verir (Örn: "10 cm ileri git" veya "Sağa dön").
3. **Ödül (Reward r_t):** Yapılan hareket hedefe yaklaştırıyorsa robot "+" puan, uzaklaştırıyor veya hata yapıyorsa "-" puan alır. Robotun amacı zamanla toplam puanı (ödülü) maksimize etmeyi öğrenmektir.
4. **Yeni Durum (s_{t+1}):** Hareketten sonra robot yeni bir pozisyona geçer ve döngü en baştan tekrar başlar.

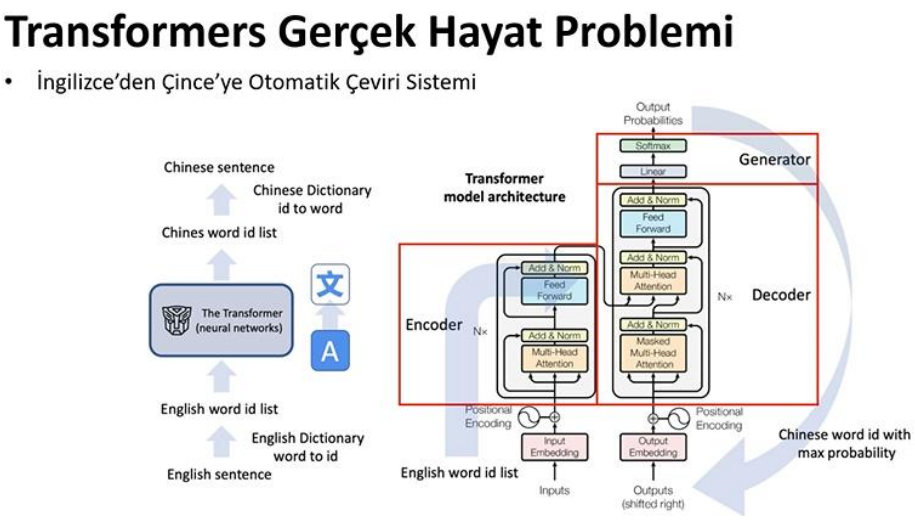
Özetle: Bu yapı sayesinde robot, başlangıçta nasıl hareket edeceğini bilmese bile, deneme-yanılma yoluyla ve topladığı ödüller sayesinde hedefe en kısa ve en güvenli yoldan nasıl ulaşacağını kendi kendine öğrenir. Not: Buradaki kritik nokta, "Vision Module" ile "Deep RL Module" arasındaki iş birliğidir. Biri ne gördüğünü söyler, diğeri ise bu görülen şeye karşı ne yapılması gerektiğine karar verir.

13. Transformers:

Tanım: Transformerlar, ardışık verileri (özellikle metinleri) işlemek için dikkat mekanizmasını (Attention Mechanism) kullanan bir derin öğrenme mimarisidir. Geleneksel modellerin aksine, veriyi sırayla değil, paralel işleyerek kelimeler arasındaki uzak ilişkileri daha etkili bir şekilde kavrayabilir.

Kullanım Alanı: Transformers modelleri, dil modelleme, metin çevirisi, metin özetleme ve sınıflandırma, konuşma sentezi gibi çeşitli doğal dil işleme alanlarında yaygın olarak kullanılır.

Örnek: Bir Transformers modeli, bir metindeki anlamsal bağlamları analiz ederek ve anlam çıkararak, metni başka bir dile çevirmek için kullanılabilir.



Bu şema, Transformer mimarisinin bir "İngilizce'den Çince'ye çeviri" problemi işleyişini anlatıyor.

1. Hazırlık: Kelimeden Sayıya: Bilgisayarın metni işleyebilmesi için kelimeler önce **Sözlük (Dictionary)** yardımıyla sayısal kimliklere (**ID**) dönüştürülür. Artık cümle, modelin üzerinde işlem yapabileceği bir sayı dizisidir.

2. İşleme: Encoder ve Decoder (Kalp): Modelin asıl "düşünme" aşaması burada gerçekleşir:

- Encoder: İngilizce cümleyi analiz eder.

- **Input Embedding:** Sayıları, kelimenin anlamını temsil eden karmaşık vektörlere dönüştürür.
- **Positional Encoding:** Kelimelerin cümle içindeki sırasını modele öğretir.
- **Multi-Head Attention:** Cümledeki kelimeler arasındaki ilişkileri kurar.

- Decoder: Çince karşılığı oluşturmaya başlar.

- **Masked Multi-Head Attention:** Gelecekteki kelimeleri "görmeden", o ana kadar yazılan kelimelere göre bir sonraki kelimeyi tahmin eder.
- **Encoder-Decoder Attention:** Encoder'dan gelen bilgiyi kullanıp doğru Çince kelimeyi seçer.

3. Generator ve Çıkış: Son aşama, tahmin edilen veriyi tekrar insan diline dönüştürmektir.

- **Linear & Softmax:** Modelin elindeki binlerce kelimeden en yüksek olasılıklı olanı seçmesini sağlar.
- **Chinese word id with max probability:** En yüksek olasılıklı kelime ID'sini seçer.
- **Chinese Dictionary (id to word):** Bu ID'yi tekrar gerçek bir Çince kelimeye dönüştürür ve sonuç cümlesini oluşturur.

Derin Öğrenme Modellerinin Karşılaştırılması

Model	Kullanım Alanları	Derinlik	Kullanım Kolaylığı	Özellikler
Yapay Sinir Ağları (ANN)	Genel sınıflandırma, regresyon, zaman serisi analizi	Genellikle sığ (5-20 katman)	Orta, temel bir yapı	Temel yapay sinir ağı, genellikle tamamen bağlı katmanlar
Evrişimsel Sinir Ağları (CNN)	Görüntü sınıflandırma, nesne tespiti, görüntü segmentasyonu	Derin (10-100+ katman)	Yüksek, birçok kütüphane desteği	Konvolüsyon, havuzlama, derin özellik öğrenme
Tekrarlayan Sinir Ağları (RNN)	Zaman serisi analizi, dil modelleme, konuşma tanıma	Orta (10-50 katman)	Orta, dil modeli ve sekans işleme	Zaman bağımlılığı, sıralı veri işleme
Uzun Kısa Süreli Bellek (LSTM)	Zaman serisi tahmini, dil modelleme, konuşma tanıma	Derin (10-50+ katman)	Orta, RNN'lerin geliştirilmiş hali	Uzun vadeli bağımlılıkları öğrenme, unutma ve hatırlama mekanizmaları
Çekişmeli Üretici Ağlar (GANs)	Görüntü sentezi, veri artırma, yaratıcı içerik üretimi	Derin (genellikle 2-10 katman)	Zor, deneme yanılma gerektirebilir	Üretici ve ayırt edici ağlar, veri üretimi
Derin Pekiştirmeli Öğrenme (DRL)	Oyun stratejileri, robotik kontrol, otonom araçlar	Derin (10-100+ katman)	Zor, genellikle çok sayıda parametre	Çevresel geri bildirim ile öğrenme, eylem-ödül mekanizması
Ön Düzenleyi Ağlar (SOMs)	Veri görselleştirme, gruplama, desen tanıma	Sığ (genellikle 1-3 katman)	Kolay, görselleştirme odaklı	Benzerlik tabanlı kümeleme, düşük boyutlu veri haritalama
Radyal Temelli Fonksiyon Ağları (RBFNs)	Tahmin, sınıflandırma, veri sıkıştırma	Orta (5-20 katman)	Orta, klasik yapılar	Radyal tabanlı çekirdek, lokalizasyon özelliği
Derin İnanç Ağları (DBNs)	Görüntü tanıma, ses tanıma, doğal dil işleme	Derin (10-50 katman)	Orta, genellikle pre-training gerektirir	Katmanlı probabilistik model, öznitelik çıkarımı
Otomatik Kodlayıcılar (Autoencoders)	Veri sıkıştırma, görüntü restorasyonu, boyut azaltma	Derin (5-20+ katman)	Orta, genellikle denetimsiz öğrenme	Veriyi sıkıştırma ve yeniden oluşturma
Transformers	Dil modelleme, çeviri, metin sınıflandırma, konuşma sentezi	Derin (10-100+ katman)	Yüksek, birçok güçlü kütüphane	Öz-dikkat mekanizması, paralel işlem, uzun vadeli bağımlılıkları öğrenme

Derin Öğrenme Kullanım Alanları:

Görüntü Tanıma ve İşleme: Bilgisayarların pikselleri anlamlandırmasını sağlar.

- *Yüz Tanıma:* Güvenlik ve kilit açma sistemlerinde kullanılır.
- *Medikal Analiz:* Tıbbi sonuçlarda gözden kaçırabileceği tümör veya anormallikleri saptar.

Doğal Dil İşleme (NLP): İnsan dilinin yapısını, tonunu ve bağlamını çözümler.

- *Çeviri:* Kelime kelime değil, anlam bazlı (Google Çeviri gibi) dönüşüm yapar.
- *Sentiment (Duygu) Analizi**:*** Bir yorumun "mutlu" mu yoksa "şikâyet" mi içerdiğini anlar.

Otonom Araçlar ve Drone'lar: Sensörlerden gelen verileri anlık işleyerek karar verme sürecidir.

- *Yol Analizi:* Nesneleri (yaya, trafik ışığı, engel) birbirinden ayırarak aracın güvenle ilerlemesini sağlar.

Ses ve Konuşma Tanıma: Ses dalgalarını dijital veriye, oradan da anlamlı komutlara dönüştürür.

- *Siri/Alexa:* "Işıkları kapat" dediğinizde niyetinizi anlayan derin öğrenme mimarisidir.

Sağlık Hizmetleri: Veri madenciliği yaparak tıbbı daha "öngörülebilir" hale getirir.

- *Erken Teşhis**:*** Genetik verileri, geçmiş vakaları tarayıp hastalık risklerini önceden tahmin eder.

Finans ve Risk Yönetimi: Milyonlarca işlem içindeki aykırılıkları bulur.

- *Fraud (Dolandırıcılık):* Normal harcama alışkanlıklarınızın dışındaki şüpheli bir işlemi milisaniyeler içinde durdurur.

E-ticaret ve Kişiselleştirme: Satış stratejilerini tamamen "size özel" hale getirir.

- *Öneri Sistemleri**:*** "Bunu alan bunu da aldı" mantığının ötesinde, sizin ilgi alanlarınızı modelleyerek nokta atışı ürün sunar.

Özetle: Derin öğrenme, verinin olduğu her yerde tahminleme ve sınıflandırma yaparak hayatımızı otonomlaştırıyor.

Günlük Hayatta Derin Öğrenme Kullanan Yazılımlar

- **Google Translate:** Nöral Makine Çevirisi (NMT) ile kelime değil, bağlam odaklı akıllı çeviri.
- **Tesla Otonom Sürüş:** Kameralardan gelen görüntüleri gerçek zamanlı işleyerek nesne tanıma, şerit takibi ve karar verme süreçlerini yönetir.
- **Apple Face ID:** Yüzdeki 30.000'den fazla noktayı analiz ederek derinlik haritası oluşturur.
- **Amazon:** Satın alma geçmişinizi ve göz atma alışkanlıklarınızı analiz ederek "sizin için en doğru" ürünleri tahmin eder.
- **Netflix:** İzleme süresinden, tür tercihlerinden yola çıkarak içerik kütüphanesini özelleştirir.
- **Google Photos:** Yüz, nesne ve mekân tanıma sayesinde etiketsiz, akıllı arama.
- **Adobe Photoshop:** "Neural Filters" özelliği sayesinde eski fotoğrafları onarır, yüz ifadelerini değiştirir ve karmaşık nesne seçimlerini otomatikleştirir.

Derin Öğrenme Veri Kaynakları

1. Açık Veri Setleri (Open Datasets): Herkesin erişimine açık, genellikle akademik araştırmalar ve yarışmalar için kullanılan hazır setlerdir.

- **Kaggle:** Veri bilimi yarışmalarının merkezidir (Örn: Titanic, Konut Fiyatları).
- **ImageNet / CIFAR:** Nesne tanıma, görüntü sınıflandırma için standartlaşmış kütüphanelerdir.
- **UCI Repository:** Makine öğrenmesi için klasikleşmiş veri setlerini barındırır.

2. Özel Veri Setleri (Proprietary Datasets): Şirketlerin veya kurumların kendine sakladığı, gizlilik içeren ve ticari değer taşıyan verilerdir.

- **Finans & Sağlık:** Banka işlemleri veya hastane kayıtları gibi dışarıya kapalı veriler.
- **Müşteri Verileri:** Şirketlerin kendi kullanıcı tabanlarından elde ettiği alışkanlık verileri.

3. Simülasyon Verileri: Gerçek dünyada veri toplamanın tehlikeli veya maliyetli olduğu durumlarda, dijital ortamda üretilen verilerdir.

- **Otonom Sürüş:** Bir aracın kaza anı verisini toplamak yerine, bu durumu simülatörde (Unity, CARLA gibi) binlerce kez canlandırmak.
- **Fizik Simülasyonları:** Uzay veya mühendislik testleri.

4. Web Kazıma (Scraping): İnternet üzerindeki sayfaların taranarak verilerin otomatik çekilmesidir.

- **Uygulama:** E-ticaret sitelerinden fiyat takibi yapmak veya haber sitelerinden metin analizi için veri toplamak.

5. Sensör ve IoT Verileri: Nesnelerin interneti (IoT) cihazlarından gelen anlık ve sürekli akış halindeki verilerdir. **Örnekler:** Akıllı saatlerin nabız verileri, şehirlerdeki trafik yoğunluğu sensörleri veya fabrika makinelerinin sıcaklık ölçümleri.

6. Kullanıcı Tarafından Oluşturulan İçerik (UGC): İnsanların dijital platformlarda gönüllü olarak paylaştığı ham verilerdir. **Kaynaklar:** Sosyal medya postları (X, Instagram), blog yazıları ve forumlardaki tartışmalar. Özellikle doğal dil işleme (NLP) için hazinedir.

7. Veri Artırma (Data Augmentation): Eldeki mevcut veriyi çeşitli yöntemlerle çoğaltarak veri setini yapay olarak büyütme tekniğidir.

- **Görüntü/Ses:** Bir fotoğrafı döndürmek, rengini değiştirmek veya ses kaydına gürültü ekleyerek modelin daha dayanıklı hale gelmesini sağlamak.

Yapay Zekada Devrim: Google Brain ve AlphaGo'nun Hikayesi

1. Google Brain: Derin Öğrenmenin Doğuşu:

Google Brain, 2011 yılında Jeff Dean ve Andrew Ng gibi isimler tarafından kurulan bir araştırma projesidir. Temel amacı, devasa verileri kullanarak deep learning algoritmalarını geliştirmekti.

- **Neden Önemli?** O dönemde yapay zekanın devasa veri setleri ve devasa işlemci güçleriyle (ölçeklenebilirlik) neler yapabileceği henüz bir soru işaretiydi.
- **Meşhur "Kedi" Deneyi:** Araştırmacılar, binlerce bilgisayarı birbirine bağlayarak devasa bir yapay sinir ağı kurdular ve bu ağa 10 milyon YouTube videosu izlettiler. Sisteme "kedi"nin ne olduğu söylenmemesine rağmen, algoritma videolardaki ortak kalıpları analiz ederek "kedi" kavramını kendi kendine keşfetti.
- **Sonuç:** Bugün kullandığımız Google Çeviri, fotoğraf gruplandırma ve sesli asistanların temeli bu projeyle atıldı.

2. AlphaGo: Strateji ve "Sezgi"nin Zaferi:

AlphaGo, Google'ın satın aldığı DeepMind ekibi tarafından geliştirilen, antik Çin oyunu Go'yu oynamak için tasarlanmış bir yapay zekadır.

- **Neden Go?** Satrançtan çok daha karmaşık olan Go oyununda olası hamle sayısı, evrendeki atom sayısından daha fazladır. Bu yüzden sadece hesaplama gücü yetmiyor, bir tür "sezgi" gerekiyordu.
- **Tarihi Başarı:** 2016 yılında, AlphaGo dünya şampiyonu Lee Sedol'u 4-1 yenerek dünyayı şok etti. Bu, yapay zekanın sadece mantık değil, yaratıcılık ve strateji gerektiren bir alanda insanı geçebileceğinin kanıtıydı.
- **Teknik:** AlphaGo, hem geçmişteki insan maçlarını analiz ederek hem de kendi kendine milyonlarca kez oynayarak (**pekiştirmeli öğrenme- reinforcement learning**) ustalaştı.

Aralarındaki Fark Nedir?

Özellik	Google Brain	AlphaGo (DeepMind)
Odak Noktası	Genel amaçlı algoritmalar ve tüketici servisleri.	Karmaşık problem çözme ve oyun teorisi.
Öğrenme Methodu	Veri madenciliği ve Derin Öğrenme.	Pekiştirmeli Öğrenme (Deneme-Yanılma).
Sembolik Başarı	Kedi videolarını kendi kendine tanıması.	Lee Sedol'u (Dünya Şampiyonu) yenmesi.
Temel Katkı	Algılama ve Sınıflandırma (Görsel/İşitsel).	Strateji ve Karar Verme.

Bu iki olay, yapay zekanın sadece bir laboratuvar deneyi olmadığını, gerçek dünyada insan zekasını taklit edebileceğini (ve hatta geçebileceğini) tüm dünyaya gösterdi.