VERİ ANALİZİNDE MAKİNE ÖĞRENMESİ YÖNTEMLERİNİN KULLANIMI

Editör: EYYÜP GÜLBANDILAR

BIDGE

BİDGE Yayınları

Veri Analizinde Makine Öğrenmesi Yöntemlerinin Kullanımı

Editör: Prof. Dr. Eyyüp GÜLBANDILAR

ISBN: 978-625-8995-65-7

1. Baskı
Sayfa Düzeni: Gözde YÜCEL
Yayınlama Tarihi: 2026-03-25
BİDGE Yayınları

Sertifika No: 71374
Yayın hakları © BİDGE Yayınları
www.bidgeyayinlari.com.tr - bidgeyayinlari@gmail.com
Krc Bilişim Ticaret ve Organizasyon Ltd. Şti.
Güzeltepe Mahallesi Abidin Daver Sokak Sefer Apartmanı No: 7/9 Çankaya / Ankara

ÖNSÖZ

Günümüzde veri, dijital dönüşümün en temel yapı taşı haline gelmiş; bu verinin anlamlandırılması ise modern bilimin ve endüstrinin öncelikli hedefi olmuştur. Klasik programlama yaklaşımlarının karmaşık ve dinamik veri yapıları karşısında yetersiz kaldığı günümüzde, Makine Öğrenmesi ve onun ileri bir aşaması olan Derin Öğrenme yöntemleri, sorunların çözümünde devrim niteliğinde imkânlar sunmaktadır. Bu kitap, **"Veri Analizinde Makine Öğrenmesi Yöntemlerinin Kullanımı"** başlığı altında, bu teknolojilerin tarımdan yazılım güvenliğine, deniz ekosisteminden nesne tespitine kadar uzanan geniş yelpazedeki uygulama alanlarını akademik bir perspektifle ele almaktadır.

Kitabın temel amacı, teorik algoritmaların gerçek dünya problemlerine nasıl uyarlandığını somut projeler ve kapsamlı analizler üzerinden göstermektir. Eser, birbirini tamamlayan üç ana bölümden oluşmaktadır:

Tarımsal Üretimde Akıllı Çözümler: İlk bölümde, derin öğrenme mimarilerinin tarımsal ürünlerin sınıflandırılmasındaki etkinliği incelenmektedir. Fındık meyvesinin karakteristik özellikleri üzerinden yürütülen çalışma, yapay sinir ağlarının manuel süreçleri nasıl optimize edebileceğini ve hata payını nasıl minimize edebileceğini ortaya koymaktadır.

Yazılım Güvenliği ve Mühendisliği: Veri analizinin sadece sayısal verilerle sınırlı kalmadığı, sistem mimarilerinin güvenliğinde de kritik bir rol oynadığı vurgulanmaktadır. Güvenli yazılım geliştirme yöntemleri ve alınması gereken önlemler, yazılım yaşam döngüsü içerisinde veriye dayalı güvenlik stratejilerinin önemini tartışmaktadır.

Gerçek Zamanlı Nesne Tespiti ve Entegrasyon: Kitabın son bölümü, YOLOv8 tabanlı derin öğrenme modellerinin su altı ekosistemindeki başarısına odaklanmaktadır. Gerçek zamanlı balık türü tanılama ve bu sistemlerin web tabanlı platformlara entegrasyonu, model başarısının son kullanıcıya ulaştırılmasındaki teknolojik süreci detaylandırmaktadır.

Bu çalışma, makine öğrenmesi yöntemlerini kendi disiplinlerine entegre etmek isteyen araştırmacılar, mühendisler ve öğrenciler için bir rehber niteliğindedir. Kitap boyunca sunulan analizler, sadece mevcut yöntemlerin başarısını değil, aynı zamanda verinin işlenmesi ve modelleme süreçlerindeki zorlukları ve çözüm yollarını da kapsamaktadır.

Bilginin hızla dönüştüğü bu çağda, bu kitabın veri analitiği ve yapay zekâ alanındaki literatüre katkı sağlamasını ve yeni akademik çalışmalara ilham vermesini temenni ediyoruz.

Prof.Dr.Eyyüp GÜLBANDILAR
Editör
Mart 2026

İÇİNDEKİLER

BÖLÜM 1

DERİN ÖĞRENME VE FINDIK MEYVESİNİN SINIFLANDIRILMASINDA KULLANIMI

ÖZGÜR TOMAK[1]

Giriş

Derin öğrenme (deep learning), yapay zekâ (AI) şemsiyesi altında yer alan makine öğrenmesinin bir alt alanı olarak, çok katmanlı yapay sinir ağları aracılığıyla veriden otomatik biçimde temsil (representation) öğrenmeyi hedefler. "Derin" ifadesi, ağın birden fazla gizli katmana sahip olmasına işaret eder; bu sayede model, ham girdiden başlayarak (piksel, ses dalgası, kelime dizisi, DNA dizisi vb.) giderek daha soyut ve görevle ilgili temsiller oluşturur. Bu yaklaşım, klasik makine öğrenmesinde sıklıkla gerekli olan elle özellik çıkarımı/özellik mühendisliği ihtiyacını azaltır ve sistemlerin veri ve hesaplama ölçeği arttıkça daha karmaşık örüntüleri öğrenebilmesini sağlamaktadır. (LeCun et al., 2015)

Derin öğrenmenin merkezindeki fikir hiyerarşik kavramlaşmadır: Modelin ilk katmanları genellikle basit örüntüleri yakalarken (ör. görüntüde kenar/yönelim), üst katmanlar bu basit

[1] Dr. Öğretim Üyesi, Mühendislik Fakültesi, Giresun Üniversitesi, Bilgisayar Mühendisliği Bölümü, Giresun, Türkiye, Orcid: 0000-0003-2993-6913

örüntülerin birleşiminden daha kompleks yapıları (motif → parça → nesne) temsil eder. Benzer hiyerarşiler konuşma ve metin gibi sıralı verilerde de görülür (seslerden hecelere, kelimelere ve cümlelere giden yapı). Bu nedenle derin öğrenme; görüntü, ses ve dil gibi "doğal sinyallerin" ham hâllerinden anlam çıkarma görevlerinde özellikle güçlüdür. (Kim, 2018)

Eğitim tarafında derin öğrenme çoğunlukla denetimli öğrenme senaryosunda ele alınır: Model, etiketli örneklerden bir kayıp/amaç fonksiyonunu minimize edecek şekilde öğrenir. Bu süreçte temel mekanizma, zincir kuralına dayanan geri yayılım (backpropagation) ile gradyanların katmanlar boyunca hesaplanması ve pratikte sıklıkla stokastik gradyan inişi (SGD) ve türevleriyle parametrelerin güncellenmesidir. Modern başarıların arkasında, bu öğrenme prosedürünün büyük veri üzerinde ölçeklenebilmesi, donanım (özellikle GPU) hızlanmaları ve düzenlileştirme yaklaşımlarının (ör. dropout gibi) pratikte etkili biçimde kullanılabilmesi bulunur. (Shinde & Shah, 2018)

Mimari açıdan derin öğrenme tek bir modelden ibaret değildir; farklı veri türleri ve görevler için özelleşmiş mimari aileleri öne çıkar. Evrişimli sinir ağları (CNN/ConvNet), yerel bağlantılar ve ağırlık paylaşımı sayesinde 1D–2D–3D "dizi/ızgara" yapılı verilerde (ses sinyali, görüntü, video, hacimsel tıbbi görüntüler) güçlü performans verir. Tekrarlayan ağlar (RNN) ve özellikle LSTM gibi kapılı yapılar, sıralı verilerde uzun bağımlılıkları öğrenmeyi kolaylaştırır; dil modelleme, makine çevirisi ve konuşma tanıma gibi görevlerde kritik rol oynar. Bunun yanında otoenkoderler ve üretici (generative) modeller, temsil öğrenmeyi denetimsiz/yarı denetimli bağlamlara taşırken; bellek/attention yaklaşımları ve pekiştirmeli öğrenme ile birleşimler, özellikle "daha aktif algı" ve "karar verme" gerektiren problemlerde önem kazanır.

Makine Öğrenmesinden Derin Öğrenmeye

Özellikle 2000'lerin sonu ve 2010'ların başında büyük veri + GPU + mimari/algoritma birleşimi, konuşma ve görüntü tanımada derin ağların hızla öne çıkmasına zemin hazırlamıştır.

Bu dönüşümün arka planında iki temel gözlem yer alır. Birincisi, geleneksel (konvansiyonel) makine öğrenmesi yöntemleri uzun süre ham veriyi doğrudan işleme konusunda sınırlı kalmış; yüksek performans çoğu zaman alan uzmanlığıyla tasarlanmış özellik çıkarıcılar (feature extractors) gerektirmiştir. İkincisi, büyük ölçekli veri ve hesaplama olanaklarının artmasıyla, çok katmanlı doğrusal olmayan modellerin temsil öğrenmeyi otomatikleştirerek görüntü, konuşma ve dil gibi alanlarda belirgin performans sıçramaları üretebildiği ortaya konmuştur.

Uzun yıllar boyunca pratik makine öğrenmesi sistemleri, ham girdiyi (ör. görüntü pikselleri) doğrudan modele vermekten ziyade, önce özellik vektörüne dönüştüren bir ön işleme ve özellik çıkarımı katmanına dayanmıştır. Ham verinin "uygun iç temsile" dönüştürülmesi, kayda değer mühendislik ve alan bilgisi gerektirir.

Ancak bu dönemdeki temel kısıt şudur: görüntü ve konuşma gibi problemler, girişteki küçük ama görev açısından kritik ayrıntılara duyarlı; buna karşın konum, aydınlatma, vurgu/aksan gibi alakasız değişimlere duyarsız (invariant) olmayı gerektirir.

Sinir Ağları, Temsil Öğrenme ve Derinlik

Derin öğrenme "deneyimden öğrenme" ve dünyayı "kavramlar hiyerarşisi" üzerinden açıklama fikrine dayanır; bu da bilgisayarın ihtiyaç duyduğu bilginin tümünün insan tarafından belirtilmesi gereğini azaltır. Bu perspektif, derin öğrenmenin yalnızca bir model ailesi değil, bir modelleme yaklaşımı olduğunu göstermektedir.

Derin ağların eğitimindeki ana mekanizma geri yayılımdır; Eğitimde pratikte en yaygın yaklaşım, küçük örnek grupları

üzerinden gradyan tahmini yapan stokastik gradyan inişi (SGD) ve varyantlarıdır. Tarihsel olarak derin ağlar bir dönem "optimizasyon zorluğu" nedeniyle geri plana düşmüştür: 1990'ların sonu ve 2000'lerin başında, gradyan inişinin kötü yerel minimumlara takılacağına dair yaygın bir kanaat olduğu aktarılır. Ancak daha sonraki kuramsal/ampirik bulgular, büyük ağlarda asıl sorunun çok sayıda "saddle point" etrafında yoğunlaşabileceğine ve iyi çözümlere ulaşmanın pratikte sanıldığı kadar zor olmayabildiğine işaret eder.

Bu çerçevede iki önemli hızlandırıcı ortaya çıkmıştır:

- Denetimsiz ön-eğitim (unsupervised pre-training) ve daha iyi başlatma (initialization): 2006 civarında CIFAR çevresindeki araştırmacılar, etiket gerektirmeden katman katman özellik öğrenmeye dayanan ön-eğitim yaklaşımlarını gündeme taşımış; daha sonra denetimli geri yayılımla ince ayar (fine-tuning) ile başarı elde edilmiştir. Bu strateji özellikle etiketli verinin sınırlı olduğu senaryolarda yararlı olmuştur.
- Mimari ve doğrusal olmayanlık seçimi (ör. ReLU) ile pratik eğitilebilirlik : ReLU gibi parçalı doğrusal aktivasyonların çok katmanlı ağları daha hızlı eğitebilir ve derin denetimli eğitimi kolaylaştırdırır; bu da belirli dönemlerde ön-eğitim ihtiyacını azaltan pratik bir kırılma olarak görülebilir. (LeCun et al., 2015)

Bilgisayarlı görüde ImageNet yarışması, derin evrişimli ağların "ana akım" hâline gelmesinde dönüm noktası olarak gözlemlenmektedir, yaklaşık bir milyon görüntü ve 1.000 sınıf ölçeğinde derin CNN'ler, önceki yaklaşımların hata oranlarını dramatik biçimde düşürmüştür. Bu başarıda GPU kullanımı, ReLU, dropout gibi düzenlileştirme yöntemleri ve veri artırma stratejileri birlikte anılır. Bu kırılma anları, DL'nin yalnızca "alternatif bir

model” değil, belirli problem sınıflarında varsayılan yaklaşım olmasının önünü açmıştır. (LeCun et al., 2015)

Derin öğrenmenin “algısal” görevlerden bilimsel veri analizine genişlemesi, ML’den DL’ye geçişin olgunlaşmış bir aşamasıdır. Derin öğrenmenin ön işleme bağımlılığının düşük olması ve büyük dizileme veri setlerinden yüksek seviyeli bilginin otomatik çıkarılması önemli bir avantajdır ve nadir bağımlılıklara bağlı aşırı uyum ve yüksek hesaplama maliyetini başlangıç zorlukları arasında bulunur. Bu örnek, derin öğrenmenin “temsil öğrenme” iddiasının yalnızca görüntü/sesle sınırlı olmadığını; uygun veri/etiketleme ve modelleme kurgusuyla bilimsel problemlerde de güçlü bir araç olabildiğini göstermektedir. (Rusk, 2016)

Makine öğrenmesinden derin öğrenmeye geçiş, birçok uygulamada pratik olarak DL lehine bir kayma yaratsa da, kuramsal açıdan DL; ML’nin temel kavramlarını (genelleme, aşırı uyum, regularizasyon, optimizasyon metodolojisi) devralır ve genişletir.

Matematiksel ve Pratik Altyapı

Derin öğrenme modelleri; çok sayıda parametre içeren, doğrusal olmayan dönüşümlerin bileşiminden oluşan hesaplama sistemleridir. Bu nedenle derin öğrenmeyi anlamak ve uygulamak, yalnızca “mimariyi bilmekten” ibaret değildir; modeli tanımlayan matematiksel nesneleri (vektör–matris–tensör), öğrenmeyi mümkün kılan optimizasyon mantığını (gradyan, geri yayılım, SGD) ve uygulamada başarıyı belirleyen metodolojik ayrıntıları (veri düzeni, değerlendirme, düzenlileştirme, hesaplama/verimlilik) birlikte gerektirir. Derin ağlar pratikte, çok katmanlı doğrusal dönüşümler (matris çarpımları) ve doğrusal olmayan fonksiyonların bileşimidir. Bu yüzden lineer cebir, yalnızca arka plan bilgisi değil, doğrudan “modelin dili”dir. (Kim, 2018)

Denetimli derin öğrenmede eğitim, bir amaç/kayıp fonksiyonunu (objective/cost) minimize etmeye dayanır. Bu

minimizasyonun motoru ise gradyan bilgisidir: ağın parametreleri küçük bir miktar değiştiğinde kaybın nasıl değiştiğini gösteren türevler. Geri yayılım özünde zincir kuralının (chain rule) çok katmanlı modüler yapılara uygulanmasıdır ve şu kavramlar kritik hale gelir. (LeCun et al., 2015)

- Türev / kısmi türev: tek bir parametrenin kayba etkisi
- Gradyan vektörü: tüm parametreler için "en hızlı artış" yönü
- Jacobian/Hessian sezgisi: özellikle optimizasyonun neden zor olabileceğini anlamak için (yüksek boyut, eğrilik, plato vb.)
- Hesaplama grafiği (computational graph): modern çerçevelerin otomatik türev aldığı yapı; ileri geçiş ve geri geçişin sistematikleşmesi

Bu bakış, "backprop bir formüldür" yaklaşımından ziyade, backprop'u genel bir hesaplama prensibi olarak kavramayı sağlamaktadır.

Derin öğrenme uygulamalarında çıktı çoğu zaman bir "skor" değil, olasılık yorumu yapılabilen bir niceliktir (özellikle sınıflandırma). Bu nedenle olasılık kuramı; modelleme, belirsizlik ve değerlendirme açısından temel rol oynar.

Pratikte en sık karşılaşılanlar:

- Koşullu olasılık ve Bayes sezgisi: "girdi verildiğinde sınıf olasılığı" yorumları
- Beklenen değer, varyans: kayıp fonksiyonlarının veri üzerinde ortalama davranışı
- Entropi ve çapraz entropi: sınıflandırmada yaygın kayıp ailesinin kavramsal temeli
- KL ayrışımı: dağılımlar arası farkın ölçülmesi; bazı düzenlileştirme/üretici model bağlamlarında kritik

Derin öğrenme eğitiminde çözüm, çoğu zaman kapalı formda değil; iteratif sayısal yöntemlerle bulunur. Derin öğrenmenin

“öğrenme” kısmı, çok sayıda parametreli bir fonksiyonun ayarlanmasıdır ve bu süreçte kavranması gereken ana fikirler aşağıda verilmiştir.

- Amaç/kayıp fonksiyonu: modelin “neye göre” öğrenmeye çalıştığını belirler
- Mini-batch yaklaşımı: tam veri gradyanı yerine örnek alt-kümeleri ile güncelleme (hesaplama ve genelleme açısından pratik)
- Öğrenme oranı (learning rate) sezgisi: çok küçükse yavaş, çok büyükse kararsız/dağılma riski
- Aktivasyon fonksiyonu seçimi ve eğitilebilirlik: ReLU gibi parçalı doğrusal aktivasyonlar derin ağların eğitimini pratikte hızlandırabilir.

Ayrıca aynı kaynak, eğitim zorluklarının her zaman “kötü yerel minimum” kaynaklı olmadığı; yüksek boyutlu kayıp yüzeyinde eyer noktalarının (saddle points) önemli olabildiği yönündeki bulgulara değinerek optimizasyon sezgisinin neden salt “basit konveks optimizasyon”dan farklılaştığını da ima eder.

Makine öğrenmesi temelleri (genelleme, underfitting/overfitting, bias–variance, regularizasyon) derin öğrenmeyi anlamak için önkoşuldur ve bu süreçte gerekli yöntem parçaları aşağıda verilmiştir. (Kim, 2018)

- Eğitim / doğrulama / test ayrımı: performansın gerçekçi ölçümü
- Hiperparametre seçimi: doğrulama seti üzerinden yapılmalı; test seti “son ölçüm” mantığıyla korunmalı
- Ölçüt seçimi: doğruluk (accuracy) her zaman yeterli değildir; dengesiz sınıflarda precision/recall gibi ölçütler gerekebilir.
- Ablasyon/baseline kültürü: “DL kullandım” demek yerine, hangi bileşenin katkı verdiğini göstermek

Derin öğrenmenin pratikte yaygınlaşması, veri/hesaplama ölçeği ve yazılım ekosistemiyle doğrudan ilişkilidir.(Shinde & Shah, 2018)

Temel Derin Öğrenme Mimarileri

Derin öğrenmede "mimari", modelin hesaplama grafiğinin (katmanların/bağlantıların düzeninin) ve bu grafikteki indüktif önyargıların (verinin yapısına uygun varsayımların) nasıl kurulduğunu ifade eder. Pratikte mimari seçimi çoğunlukla verinin yapısına göre yapılır:

- Sabit boyutlu vektörler → derin ileri beslemeli ağlar (MLP/FFN)
- Izgara/topolojik dizilimli veriler (1D sinyal, 2D görüntü, 3D hacim) → evrişimli ağlar (CNN/ConvNet)
- Sıralı veriler (metin, konuşma, zaman serisi) → tekrarlayan ağlar (RNN/LSTM)
- Temsil öğrenme / denetimsiz öğrenme → otoenkoderler ve bazı üretici modeller
- Karar verme / kontrol → derin pekiştirmeli öğrenme (Deep RL)

Derin ileri beslemeli ağlar (FFN/MLP), girdiden çıktıya doğru tek yönlü bilgi akışı olan, katman katman uygulanan doğrusal dönüşümler ve doğrusal olmayan aktivasyonlardan oluşur. FFN/MLP, derin öğrenmenin "temel taşı"dır; çünkü birçok mimari (ör. CNN) kavramsal olarak feedforward ağların daha yapılandırılmış bir özel hâli olarak görülebilir. Nitekim konvolüsyonel ağlar, nesne tanımada kullanılan "özel bir feedforward ağ" biçimi olarak da konumlandırılır. Güncel uygulamalarda ReLU (rectified linear unit) derin ağlarda eğitimi pratikte hızlandırır ve bazı durumlarda denetimsiz ön-eğitim ihtiyacını azaltır. (LeCun et al., 2015)

CNN'ler, "ızgara/topoloji" taşıyan veriyi (ör. 2D görüntü) işlemek üzere tasarlanmış özel mimarilerdir. CNN'ler bu tip veriler

için dört ana fikri kullanır: yerel bağlantılar (local connections), ağırlık paylaşımı (shared weights), havuzlama (pooling), çok katmanlı yapı (many layers)

RNN (Recurrent Neural Networks) , bir diziyi adım adım işlerken "gizli durum vektörü" ile geçmiş bilgiyi taşır. RNN, sıralı girdilerde her zaman adımında bir "state vector" tutar, ağ zaman içinde açıldığında (unfolding), çok derin bir feedforward ağ gibi görülebilir ve geri yayılım bu açılmış grafik üzerinden uygulanabilir. RNN eğitimi, uzun dizilerde gradyanların zaman içinde büyümesi veya sönmesi (exploding/vanishing gradients) nedeniyle zorlaşabilir.

LSTM (Long Short-Term Memory), RNN'lerin uzun bağımlılık öğrenme zorluklarını azaltmak için "bellek hücresi + kapılar" mantığıyla tasarlanmıştır. Bellek hücresinin kendi durumunu zaman adımında kopyalayan bir öz-bağlantı taşır ve bu akışın kapılarla kontrol edilerek bilginin ne zaman tutulacağı ve ne zaman silineceğini öğrenir.

Otoenkoderler (Autoencoder) , girdiyi çıktıda yeniden üretmeye çalışan bir ağdır. İçeride, girdiyi daha düşük boyutlu/özet bir "kod" temsiline dönüştüren bir gizli katman (latent code) bulunur. Otoenkoder, "girdiyi kopyalamaya çalışan" ağ olarak tanımlanabilir ve temsil öğrenmede temel bir yapı taşı olarak konumlandırılır.

Neden önemlidir?

- Denetimsiz biçimde "yararlı temsil" öğrenebilir
- Boyut indirgeme/sıkıştırma veya gürültüye dayanıklı temsil öğrenme (ör. denoising) gibi hedeflere uyarlanabilir
- Tarihsel olarak katman katman temsil öğrenmeye dayalı bazı ön-eğitim fikirleriyle de ilişkilidir (özellikle etiketli veri az olduğunda).

Ne zaman tercih edilir?

- Etiketli verinin sınırlı olduğu durumlarda temsili önce öğrenmek
- Anomali tespiti / veri sıkıştırma / özellik öğrenme
- Bir sonraki denetimli göreve aktarılabilecek ara temsiller öğrenmek

Üretici Çekişmeli Ağlar (GAN) yaklaşımında iki ağ birlikte eğitilir.

- Üretici (generator): Gerçeğe benzer örnekler üretmeye çalışır.
- Ayırt edici (discriminator): Gerçek ile üretilen örnekleri ayırt etmeye çalışır.

Bu "çekişmeli" kurgu, üretici modellemede güçlü bir çerçeve sunar (özellikle sentetik veri üretimi, örnekleme, veri artırma gibi senaryolarda).

Geri yayılım ve SGD

Derin öğrenmede eğitim süreci, özünde parametreleri (ağırlıklar ve bias'lar) ayarlayarak modelin ürettiği çıktılar ile hedef çıktılar arasındaki farkı azaltma problemidir. Denetimli öğrenmede model bir girdiyi alır, her sınıf için skor/olasılık üretir; ardından hedefle uyumsuzluğu ölçen bir amaç (objective) / maliyet (cost) fonksiyonu hesaplanır ve modelin iç parametreleri bu hatayı azaltacak yönde güncellenir.

Derin ağlar, basit doğrusal olmayan modüllerin bileşimidir. İleri geçişte (forward pass) her katman bir önceki katmanın temsilini alır ve dönüştürür. Bu "ağırlıklı toplam + doğrusal olmayanlık" şeklinde modüler bir yapı olarak ifade edilir.

Uygulamada bu yapı şu anlama gelir:

- Model, ham girdiden başlayarak ara temsilleri üretir.
- En üst katman, göreve uygun çıktı uzayıyla (sınıf olasılığı, skor, sürekli değer) ilişkilidir.

- Her katman, bir önceki temsildeki bilgiyi "daha ayrıştırılabilir" hâle getirecek şekilde dönüştürmeye çalışır; sınıflandırmada üst katmanlar ayrım için kritik sinyalleri güçlendirir, alakasız varyasyonları bastırır.

Eğitim sırasında modelin ürettiği çıktıların hedefle uyumu, bir amaç fonksiyonuyla ölçülür. Sınıflandırmada model her sınıf için skor üretir ve amaç fonksiyonu bu skorlar ile "istenen skor örüntüsü" arasındaki hatayı ölçer. Bu amaç fonksiyonu, parametre uzayında "tepecikli bir manzara" olarak düşünülebilir; gradyan ise bu manzarada en hızlı iniş yönünü verir. Amaç fonksiyonu çoğu zaman yalnızca performans ölçütü değil, regularizasyon terimlerini de içerdiği açıkça vurgulanır.

Uygulamada en yaygın yaklaşım stokastik gradyan inişi (SGD)'dir. Tüm veri kümesi yerine küçük örnek grupları (mini-batch) üzerinden hata ve gradyan hesaplanır; bu gradyan, tüm veri üzerindeki gerçek gradyanın "gürültülü" bir kestirimidir. Süreç, mini-batch'ler üzerinde defalarca yinelenir ve amaç fonksiyonu düşmeyi bırakana kadar devam eder. (LeCun et al., 2015)

Bu yaklaşımın pratik avantajları:

- Hesaplama verimliliği: Büyük veri üzerinde tam gradyan maliyetlidir; mini-batch bunu yönetilebilir kılar.
- Optimizasyon dinamikleri: Gürültü bazen "daha iyi" bölgelere kaçmayı kolaylaştırabilir (pratik gözlem olarak).
- Donanım uyumu: GPU gibi paralel donanımlar mini-batch matris işlemlerini verimli çalıştırır.

Derin ağ eğitiminde sonuçları belirleyen, çoğu zaman yalnızca mimari değil; eğitim kurulumudur. Bu bağlamda tipik kritik ayarlar aşağıda verilmiştir.

- Öğrenme oranı: Çok büyükse kararsızlık; çok küçükse aşırı yavaş yakınsama.

- Mini-batch boyutu: Gürültü seviyesi ve donanım verimliliği ile ilişkilidir.
- Eğitim süresi (epoch sayısı) ve erken durdurma: Overfitting'i kontrol etmekte kullanılmaktadır.
- Düzenlileştirme parametreleri: Amaç fonksiyonuna eklenen terimlerin (ör. ağırlık cezaları) şiddeti.

Derin ağların en güçlü yanı (yüksek temsil kapasitesi) aynı zamanda en temel riskini doğurur: aşırı uyum (overfitting). Model, eğitim verisindeki örüntüleri "öğrenmek" yerine, eğitim setine özgü tesadüfi ayrıntıları da ezberleyebilir; bu durumda eğitim hatası düşerken, daha önce görülmemiş veride (testte) performans zayıflar. Bu bölüm, genelleme kavramını netleştirip aşırı uyumu teşhis etme ve azaltma stratejilerini veri–model–optimizasyon eksenlerinde sistematik biçimde genişletir. Derin öğrenme literatüründe genelleme/overfitting/regularizasyonun, makine öğrenmesi temelleri içinde merkezi bir yere sahip olduğu özellikle vurgulanır. (Kim, 2018)

Fındık Kalitesinin Sınıflandırılması

Yapay zekâ teknolojileri, tarım sektöründe verimliliği artırmak ve üretim süreçlerini daha bilimsel temellere dayandırmak amacıyla giderek daha fazla kullanılmaktadır. Özellikle fındık üretiminde, ürünlerin kalite kontrolü, sınıflandırılması ve hastalıkların erken tespiti gibi birçok aşamada yapay zekâ tabanlı sistemlerden yararlanılabilir. Görüntü işleme ve makine öğrenmesi teknikleri kullanılarak fındıkların fiziksel özellikleri analiz edilebilir ve bu sayede ürünler kalite, boyut, renk ve şekil gibi kriterlere göre otomatik olarak sınıflandırılabilir. Bu yöntem, geleneksel insan tabanlı sınıflandırma süreçlerine kıyasla daha hızlı ve tutarlı sonuçlar sağlayabilmektedir.

Makine öğrenmesi algoritmaları sayesinde fındık üretiminde elde edilen büyük veri kümeleri analiz edilerek üretim süreçleri hakkında önemli çıkarımlar yapılabilir. Örneğin, farklı iklim

koşullarında, toprak türlerinde veya bakım yöntemlerinde yetişen fındıkların verim ve kalite değerleri karşılaştırılabilir. Bu analizler sonucunda hangi fındık türlerinin hangi çevresel koşullarda daha iyi yetiştiği belirlenebilir. Böylece üreticiler, üretim stratejilerini bilimsel veriler doğrultusunda planlayarak daha kaliteli ve verimli ürün elde etme imkânına sahip olabilirler.

Fındık toplama ve işleme süreçlerinde de yapay zekâ destekli otomasyon sistemleri kullanılabilir. Sensörler ve akıllı cihazlarla donatılmış makineler, fındıkların toplanması, ayrıştırılması ve işlenmesi gibi işlemleri otomatik olarak gerçekleştirebilir. Bu tür sistemler, insan gücüne olan bağımlılığı azaltırken aynı zamanda üretim süreçlerinin daha hızlı ve verimli bir şekilde yürütülmesini sağlamaktadır. Bu durum işletmeler açısından önemli ölçüde zaman ve maliyet tasarrufu sağlamaktadır.

Fındık meyvesinin sınıflandırılmasında en etkili yöntemlerden biri evrişimsel sinir ağlarıdır. Bu yöntem, özellikle görüntü verilerinin analizinde oldukça başarılı sonuçlar vermektedir. Bu süreçte ilk adım, fındık meyvelerine ait kapsamlı bir görüntü veri seti oluşturmaktır. Veri seti oluşturulurken farklı kalite sınıflarına, boyutlara ve renk özelliklerine sahip fındıkların görüntüleri toplanır. Daha sonra elde edilen veriler, modelin daha doğru öğrenme yapabilmesi için temizleme ve ön işleme aşamalarından geçirilir. Veri temizleme sürecinde hatalı veya eksik veriler ayıklanırken, normalize etme işlemi ile veriler belirli bir standart aralığa getirilir.

Görüntü verilerinin daha iyi analiz edilebilmesi için çeşitli görüntü işleme teknikleri uygulanabilir. Örneğin, görüntülerin yeniden boyutlandırılması (resizing), keskinleştirme (sharpening) veya gürültü azaltma gibi işlemler modelin daha doğru özellikler öğrenmesine yardımcı olur. Bu işlemler tamamlandıktan sonra veri seti genellikle iki temel bölüme ayrılır: eğitim seti ve test seti. Eğitim seti, sinir ağının örnekler üzerinden öğrenmesini sağlamak amacıyla kullanılmaktadırken; test seti ise eğitilmiş modelin performansını ve doğruluğunu değerlendirmek için kullanılmaktadır.

Modelin eğitim süreci boyunca doğruluk, kayıp değeri ve diğer performans ölçütleri sürekli olarak takip edilir. Bu değerlendirmeler sayesinde modelin öğrenme süreci optimize edilebilir ve gerekirse hiperparametre ayarlamaları yapılabilir. Eğitim süreci tamamlandıktan sonra elde edilen model, yeni fındık görüntülerini analiz ederek bunları belirlenen sınıflara otomatik olarak ayırabilir. Ayrıca modelin gerçek dünya verileri üzerindeki başarımını değerlendirmek amacıyla yeni veri setleri üzerinde tahminler yapılır ve elde edilen sonuçların doğruluğu ölçülür.

Yapay zekâ ve derin öğrenme yöntemleri kullanılarak geliştirilen sistemler fındık üretiminde kalite kontrolünü kolaylaştırabilir, üretim süreçlerini optimize edebilir ve hastalıkların erken tespit edilmesine yardımcı olabilir. Bu teknolojilerin tarım sektöründe yaygınlaşması, hem üreticilerin daha bilinçli kararlar almasını sağlayacak hem de tarımsal verimliliğin artırılmasına önemli katkılar sunacaktır.

ResNet-50

ResNet-50, derin öğrenme alanında görüntü sınıflandırma problemlerinde yaygın olarak kullanılan güçlü bir evrişimsel sinir ağı (Convolutional Neural Network – CNN) mimarisidir. Bu model, ImageNet veri tabanının bir alt kümesi üzerinde eğitilmiş önceden eğitilmiş (pre-trained) bir modeldir. ImageNet veri tabanı milyonlarca etiketlenmiş görüntü içeren büyük ölçekli bir veri kümesidir ve bilgisayarla görme çalışmalarında sıklıkla kullanılmaktadır. ResNet-50 modeli, yaklaşık bir milyondan fazla görüntü kullanılarak eğitilmiş ve 2015 yılında düzenlenen ImageNet Large Scale Visual Recognition Challenge (ILSVRC) yarışmasını kazanarak önemli bir başarı elde etmiştir. Böylece bu model bilgisayarlı görü alanında yaygın olarak kullanılan mimarilerden biri haline gelmiştir.

ResNet-50 mimarisi adından da anlaşılacağı üzere 50 temel öğrenilebilir katmandan oluşan bir artık ağ (Residual Network)

mimarisine dayanmaktadır. Model, eğitim sürecinde öğrenmiş olduğu özellikleri kullanarak görüntüleri yaklaşık 1000 farklı nesne kategorisinden birine sınıflandırabilir. Bu kategoriler; kalem, klavye, araçlar, çeşitli nesneler ve birçok hayvan türü gibi çok geniş bir nesne yelpazesini kapsamaktadır. Modelin giriş katmanına verilen görüntülerin boyutu 224 × 224 piksel olacak şekilde ayarlanır. Bu doğrultuda modele verilecek görüntüler genellikle bu boyuta yeniden ölçeklendirilmektedir. (He et al., 2016)

Klasik bir evrişimsel sinir ağı mimarisinde giriş görüntüsü, ardışık şekilde düzenlenmiş evrişim katmanları, aktivasyon fonksiyonları ve havuzlama katmanları aracılığıyla işlenir. Bu süreçte her katman bir önceki katmandan aldığı çıktıyı dönüştürerek daha soyut özelliklerin öğrenilmesini sağlamaktadır. Ancak ağ derinliği arttıkça eğitim sırasında bazı problemler ortaya çıkabilir. Özellikle gradyan kaybolması (vanishing gradient) problemi, çok derin ağların etkili bir şekilde eğitilmesini zorlaştırır.

ResNet mimarisi, bu sorunu çözmek amacıyla artık bağlantı (residual connection) adı verilen özel bir yapı kullanır. Artık ağlarda, bir katmanın giriş verisi doğrudan sonraki katmanların çıktısına eklenerek kısa yol (skip connection) oluşturulur. Böylece ağ, yalnızca giriş ile çıktı arasındaki farkı öğrenmeye odaklanır. Bu yapı, modelin daha derin ağ mimarileri oluşturmasına ve öğrenme sürecinin daha kararlı olmasına olanak tanır.

Bir artık blokta giriş verisi x ile gösterildiğinde, blok içerisinde gerçekleştirilen evrişimsel işlemler sonucu elde edilen çıktı F(x) ile ifade edilir. Daha sonra bu çıktı, bloğun başlangıcındaki giriş verisi ile toplanarak nihai çıktı elde edilir. Bu işlem matematiksel olarak aşağıdaki Denklem 1'deki gibi ifade edilir.

$$H(x) = F(x) + x \quad (1)$$

Burada H(x), artık bloğun nihai çıktısını temsil etmektedir. Bu yapı sayesinde ağ, doğrudan dönüşümü öğrenmek yerine giriş ile

çıktı arasındaki artık fonksiyonu öğrenir. Bu yaklaşım, özellikle çok katmanlı ağlarda performansın düşmesini önleyerek modelin daha verimli bir şekilde eğitilmesini sağlamaktadır.

ResNet mimarisi, birden fazla artık bloğun ardışık olarak bir araya getirilmesiyle oluşturulmuştur. Bu bloklar sayesinde model, görüntülerin düşük seviyeli özelliklerinden (kenarlar, köşeler vb.) başlayarak daha karmaşık ve yüksek seviyeli özellikleri öğrenebilir. Ayrıca her evrişimsel katmandan sonra uygulanan Toplu Normalizasyon (Batch Normalization) işlemi, ağın eğitim sürecini daha kararlı hale getirir. Bu teknik, katmanlara verilen verilerin dağılımını düzenleyerek öğrenme sürecinin daha hızlı ve stabil olmasını sağlamaktadır.

ResNet mimarisinde genellikle tam bağlı katmandan (Fully Connected Layer) önce son aşamada Küresel Ortalama Havuzlama (Global Average Pooling) katmanı kullanılmaktadır. Bu katman, özellik haritalarındaki mekânsal boyutları azaltarak her özellik haritası için tek bir değer üretir. Böylece model, çok sayıda parametre içeren büyük bir tam bağlı katman yerine daha kompakt bir temsil elde eder. Bu yaklaşım hem modelin parametre sayısını azaltır hem de aşırı öğrenme (overfitting) riskini düşürmeye yardımcı olur.

Sonuç olarak ResNet-50 mimarisi, artık bağlantılar sayesinde çok derin sinir ağlarının etkin bir şekilde eğitilmesini mümkün kılan güçlü bir modeldir. Bu mimari, görüntü sınıflandırma, nesne tanıma ve birçok bilgisayarla görme uygulamasında yüksek doğruluk oranları elde edilmesini sağlamaktadır.

ShuffleNet

ShuffleNet, özellikle mobil cihazlar ve gömülü sistemler gibi sınırlı işlem gücüne sahip platformlarda kullanılmak üzere geliştirilmiş hafif ve verimli bir evrişimsel sinir ağı (Convolutional Neural Network – CNN) mimarisidir. (Zhang et al., 2018) Derin öğrenme modelleri genellikle yüksek hesaplama gücü ve büyük

miktarda bellek gerektirdiğinden, mobil cihazlarda doğrudan kullanılmaları zor olabilir. ShuffleNet mimarisi, bu soruna çözüm olarak tasarlanmış olup parametre sayısını ve hesaplama maliyetini önemli ölçüde azaltarak düşük donanım kapasitesine sahip sistemlerde bile yüksek performans sağlayabilmektedir. Bu özellikleri sayesinde ShuffleNet, mobil uygulamalar, gerçek zamanlı görüntü işleme sistemleri ve gömülü yapay zekâ çözümleri için oldukça uygun bir modeldir.

ShuffleNet mimarisi, doğruluk oranını korurken hesaplama maliyetini düşürmek amacıyla iki önemli teknik kullanmaktadır. Bunlar nokta bazlı grup evrişimi (pointwise group convolution) ve kanal karıştırma (channel shuffle) işlemleridir. (Zhang et al., 2018) Nokta bazlı grup evrişimi, geleneksel 1×1 evrişim işlemlerinin daha küçük gruplara bölünerek uygulanmasını sağlamaktadır. Bu yaklaşım, her grubun daha az parametre ile işlem yapmasına olanak tanır ve böylece modelin toplam hesaplama yükü önemli ölçüde azalır. Ancak grup evrişimleri kullanıldığında farklı gruplar arasında bilgi akışı sınırlı kalabilir. Bu sorunu çözmek amacıyla ShuffleNet mimarisinde kanal karıştırma işlemi kullanılmaktadır.

Kanal karıştırma işlemi, grup evrişimleri sonrasında elde edilen özellik haritalarındaki kanalların yeniden düzenlenmesini sağlamaktadır. Bu işlem sayesinde farklı gruplardan gelen özellikler arasında bilgi paylaşımı sağlanır. Böylece ağın öğrenme kapasitesi korunurken modelin hesaplama verimliliğini önemli ölçüde artırmaktadır. Kanal karıştırma işlemi, ShuffleNet mimarisinin en önemli yeniliklerinden biri olarak kabul edilmektedir.

ShuffleNet mimarisinin temel yapı taşı ShuffleNet bloğu olarak adlandırılan modüldür. Bu blok içerisinde derinlik yönlendirmeli ayrık evrişimler (depthwise separable convolutions) kullanılmaktadır. Bu evrişim türü, standart evrişimlere göre çok daha az hesaplama gerektirir. Depthwise separable convolution iki aşamadan oluşur: ilk aşamada her kanal ayrı ayrı işlenir (depthwise convolution), ikinci aşamada ise kanallar birleştirilerek yeni özellik

haritaları oluşturulur (pointwise convolution). Bu yaklaşım sayesinde giriş özellik haritaları daha küçük ve bağımsız parçalar halinde işlenir, böylece hesaplama verimliliği önemli ölçüde artırılır.

ShuffleNet biriminde kullanılan mimari yapıda ilk ve ikinci 1×1 evrişim katmanları grup evrişimleri ile değiştirilmiştir. Bu sayede modeldeki parametre sayısı azaltılmış ve hesaplama maliyeti düşürülmüştür. İlk 1×1 grup evrişiminden sonra kanal karıştırma işlemi uygulanarak farklı gruplardaki kanalların birbiriyle etkileşime girmesi sağlanır. Böylece ağın öğrenme kapasitesi korunmakta ve daha etkili özellik çıkarımı gerçekleştirilebilmektedir.

ShuffleNet biriminin bir diğer önemli özelliği de adım sayısı (stride) 2 olduğunda kullanılan özel yapıdır. Bu durumda ağın kısayol yoluna 3×3 ortalama havuzlama (average pooling) katmanı eklenir. Bu katman, özellik haritalarının boyutunu küçültürken önemli bilgilerin korunmasına yardımcı olur. Ayrıca klasik artık ağlarda kullanılan eleman bazında toplama işlemi (element-wise addition) yerine kanal birleştirme (channel concatenation) yöntemi tercih edilmiştir. Kanal birleştirme işlemi, hesaplama maliyetini çok fazla artırmadan kanal sayısının genişletilmesini sağlamaktadır.

ShuffleNet mimarisi ayrıca daha geniş özellik haritaları kullanabilme yeteneğine sahiptir. Bu durum özellikle küçük ve hafif ağlar için oldukça önemlidir. Çünkü küçük ağlarda genellikle sınırlı sayıda kanal bulunur ve bu da öğrenilebilecek bilgi miktarını azaltabilir. ShuffleNet mimarisi, kanal sayısını artırarak daha zengin özellik temsilleri oluşturabilir ve böylece küçük ağların performansını iyileştirebilir.

Sonuç olarak ShuffleNet, düşük hesaplama maliyeti ve yüksek verimlilik sağlayan bir derin öğrenme mimarisidir. Grup evrişimleri, kanal karıştırma mekanizması ve depthwise separable convolution gibi teknikler sayesinde model hem hafif hem de etkili bir yapı sunmaktadır. Bu özellikleri nedeniyle ShuffleNet, mobil cihazlarda çalışan görüntü sınıflandırma, nesne tanıma ve gerçek

zamanlı yapay zekâ uygulamaları için oldukça uygun bir mimari olarak kabul edilmektedir.

Veri Seti

Kullanılan görüntü verileri, 12 megapiksel çözünürlüğe sahip bir akıllı telefonun arka kamerası kullanılarak elde edilmiştir. Görüntülerin çekimi sırasında fındıkların sabit bir konumda ve uygun ışık koşullarında görüntülenebilmesi için özel olarak hazırlanmış bir platformdan yararlanılmıştır. Bu sistem sayesinde tüm görüntüler 3024 × 3024 piksel çözünürlükte ve benzer çekim koşullarında elde edilmiştir. Veri setinde yer alan her görüntüde yalnızca tek bir fındık bulunacak şekilde çekim yapılmıştır. Böylece görüntülerdeki nesne karmaşası önlenmiş ve sınıflandırma işlemi için daha temiz bir veri seti oluşturulmuştur.

Toplanan fındık örnekleri, alanında uzman bir kişi tarafından incelenmiş ve kalite durumlarına göre üç farklı sınıfa ayrılmıştır. Bu sınıflar kötü fındık, iyi fındık ve iç fındık olarak belirlenmiştir. Veri setinin oluşturulmasında toplam 519 adet kötü fındık, 535 adet iyi fındık ve 523 adet iç fındık kullanılmıştır. Her bir fındık örneğinin farklı açılardan ve farklı konumlardan görüntülenebilmesi amacıyla her fındık için 10 ayrı görüntü alınmıştır. Bu işlem sayesinde veri setinde yer alan görüntüler, fındıkların farklı perspektiflerini içerecek şekilde zenginleştirilmiştir. Sonuç olarak tüm sınıflar bir araya getirildiğinde toplamda 15.770 adet görüntüden oluşan kapsamlı bir veri seti elde edilmiştir.

Görüntülerin derin öğrenme modellerinde kullanılabilmesi için bazı ön işleme (preprocessing) adımları uygulanmıştır. İlk aşamada ham görüntüler, gereksiz arka plan alanlarını azaltmak ve veri setini standartlaştırmak amacıyla 1000 × 1000 piksel boyutunda kırpılmıştır. Ardından bu görüntüler, kullanılacak sinir ağı mimarilerinin giriş boyutlarına uygun hale getirmek için 224 × 224 piksel çözünürlüğe yeniden boyutlandırılmıştır. Bu işlem, görüntülerin model tarafından daha hızlı işlenmesini

sağlamaktadırken aynı zamanda tüm veri setinde boyut tutarlılığını sağlamaktadır. Bu ön işleme adımları sayesinde oluşturulan veri seti, derin öğrenme tabanlı sınıflandırma modellerinin eğitimi için uygun bir hale getirilmiştir. (Güneş, 2022) Kullanılan fındıklardan bazılarına ait görüntüler Şekil 1'de verilmiştir.

Kullanılan veri tabanı üç farklı sınıftan oluşmakta olup toplam 15.770 adet görüntü içermektedir. Veri setindeki görüntüler, fındıkların kalite durumlarına göre sınıflandırılmış ve derin öğrenme yöntemleri kullanılarak analiz edilmiştir. Çalışmada görüntü sınıflandırma işlemini gerçekleştirmek amacıyla ResNet-50 ve ShuffleNet olmak üzere iki farklı derin öğrenme mimarisi tercih edilmiştir. Model geliştirme ve eğitim süreçlerinin yürütülmesi için çalışma ortamı olarak MATLAB yazılımı kullanılmıştır. (Tomak et al., 2025)

Şekil 1. Kullanılan fındıklardan bazılarına ait görüntüler

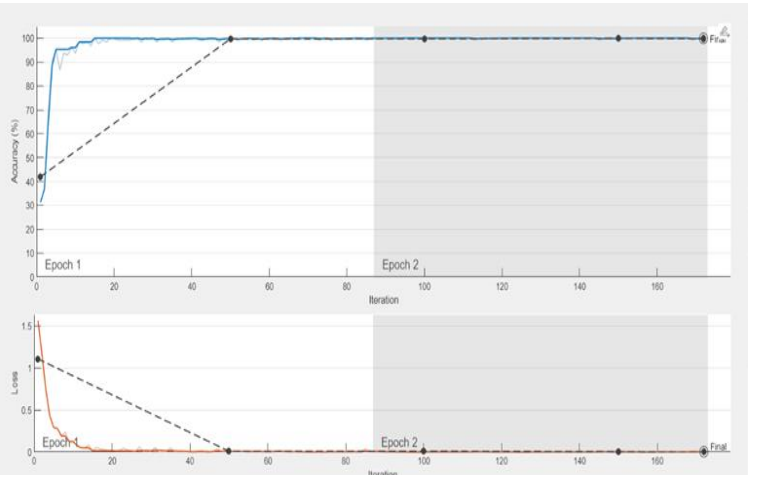
Kaynak: (Güneş, 2022)

Modelin eğitim ve test aşamalarının daha güvenilir sonuçlar verebilmesi için veri seti öncelikle rastgele şekilde karıştırılmıştır. Daha sonra veri seti iki farklı bölüme ayrılmıştır. Toplam veri setinin %30'luk kısmı eğitim (training) verisi olarak kullanılmaktadırken, kalan %70'lik kısmı test (testing) verisi olarak değerlendirilmiştir. Eğitim verileri, sinir ağlarının veri içerisindeki özellikleri öğrenmesini sağlamak amacıyla kullanılmaktadırken; test verileri

ise eğitilmiş modelin yeni ve daha önce görmediği veriler üzerindeki performansını ölçmek için kullanılmıştır.

Kullanılan iki farklı derin öğrenme algoritmasının performansı, eğitim ve test doğruluk oranları açısından karşılaştırılmıştır. Yapılan deneysel çalışmalar sonucunda ResNet-50 modeli eğitim aşamasında %99.92 doğruluk oranına, test aşamasında ise %99.97 doğruluk oranına ulaşmıştır. Bu modelin eğitim sürecindeki doğruluk değişimini gösteren grafik Şekil 2'de sunulmuştur.

Şekil 2. ResNet-50 doğruluk grafiği



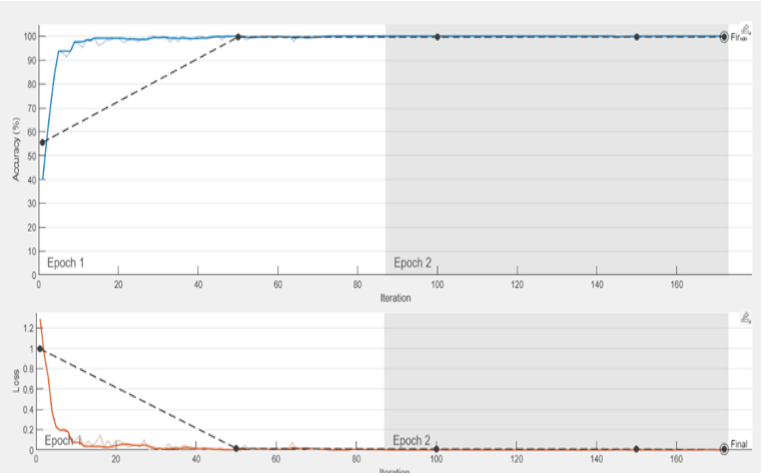
Kaynak: (Tomak et al., 2025)

Diğer taraftan ShuffleNet modeli ile yapılan deneylerde ise eğitim aşamasında %99.79 doğruluk, test aşamasında ise %99.94 doğruluk elde edilmiştir. ShuffleNet modelinin eğitim sürecindeki doğruluk performansı ise Şekil 3'te verilen grafik ile gösterilmiştir.

Elde edilen sonuçlar incelendiğinde her iki derin öğrenme modelinin de fındık görüntülerinin sınıflandırılmasında oldukça yüksek doğruluk oranlarına ulaştığı gözlemlenmektedir. Bununla birlikte ResNet-50 modeli, hem eğitim hem de test aşamasında ShuffleNet modeline kıyasla biraz daha yüksek doğruluk oranı elde ederek daha başarılı bir performans sergilemiştir. Bu durum, ResNet-

50 mimarisinin derin yapısı ve artık bağlantı (residual connection) mekanizmasının görüntü özelliklerini daha etkili şekilde öğrenebilmesi ile ilişkilendirilebilir.

Şekil 3. ShuffleNet doğruluk grafiği



Kaynak: (Tomak et al., 2025)

Tam bağlı katman parametrelerinin çoğu kapsandığından, aşırı miktarda uyum eğilimindedir. Aşırı miktarda uyumu azaltmanın bir yöntemi Dropout'tur. Her eğitim kademesinde, eğitim sırasında bazı nöronların geçici olarak devre dışı bırakılması olasılığı 1-p'dir veya p olasılıkla da tutulur, böylece indirgenmiş bir ağ kalır. Kaldırılan düğüme giden ve de gelen kenarlar kaldırılır. Bu aşamada, yalnızca indirgenmiş ağ verileri ile eğitim yapılır. Dropout, aşırı uyumu azaltır.

Kullanılan sınıflandırma algoritmalarının performansını değerlendirmek amacıyla karmaşıklık matrisi (Confusion Matrix) yöntemi kullanılmıştır. Karmaşıklık matrisi, bir sınıflandırma modelinin tahmin sonuçlarını gerçek değerlerle karşılaştırarak modelin ne kadar doğru tahmin yaptığını ayrıntılı biçimde analiz etmeye imkân sağlamaktadır. Bu yöntem sayesinde yalnızca genel doğruluk oranı değil, aynı zamanda modelin farklı sınıflar üzerindeki başarımı da değerlendirilebilmektedir.

Karmaşıklık matrisi dört temel bileşenden oluşmaktadır. TP (True Positive) değeri, modelin pozitif olarak tahmin ettiği ve gerçekte de pozitif olan durumları ifade eder. FP (False Positive) ise modelin pozitif olarak tahmin ettiği ancak gerçekte negatif olan örnekleri göstermektedir. TN (True Negative) değeri, modelin negatif olarak tahmin ettiği ve gerçekte de negatif olan durumları temsil ederken, FN (False Negative) ise modelin negatif olarak tahmin ettiği ancak gerçekte pozitif olan örnekleri ifade etmektedir (Tomak, 2019). Bu dört temel değer kullanılarak sınıflandırma modelinin performansını değerlendirmek için çeşitli ölçütler hesaplanmaktadır.

Model performansını değerlendirmek amacıyla doğruluk (accuracy), hassaslık (precision), özgünlük (specificity) ve F1-skoru (F1-score) gibi performans ölçütleri kullanılmıştır. Karmaşıklık matrisi Tablo 1'de gösterilmiş olup, bu ölçütlerin matematiksel ifadeleri ise Denklem 2–5 arasında verilmiştir. Bu ölçütler, modelin sınıflandırma performansını daha kapsamlı bir şekilde değerlendirmeye yardımcı olmaktadır.

Tablo 1 Karmaşıklık matrisi.

		Tahmini durum	
		Pozitif	Negatif
Gerçek Durum	Pozitif	TP, Gerçek Pozitif	FN, Yanlış Negatif
	Negatif	FP, Yanlış Pozitif	TN, Gerçek Negatif

Kaynak: (Tomak, 2019)

$$Doğruluk = \frac{TP+TN}{TP+FP+FN+TN} \quad (2)$$

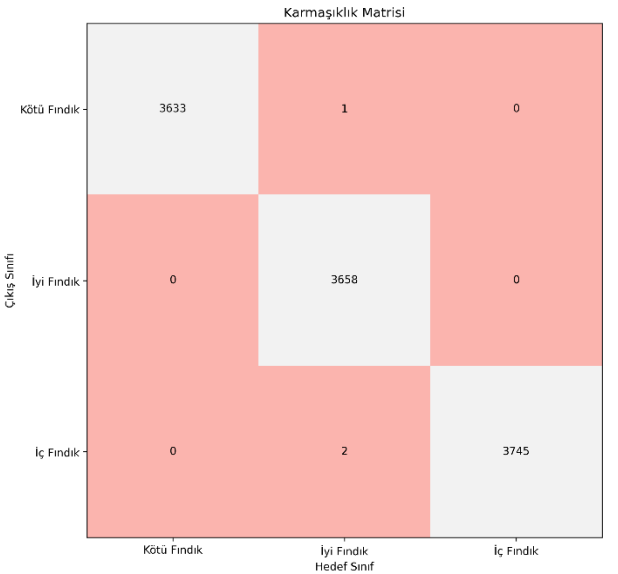
$$Hassaslık = \frac{TP}{TP+FN} \quad (3)$$

$$Özgünlük = \frac{TN}{TN+FP} \quad (4)$$

$$F1\ skor = 2 * \frac{Hassaslık * Özgünlük}{Hassaslık + Özgünlük} \quad (5)$$

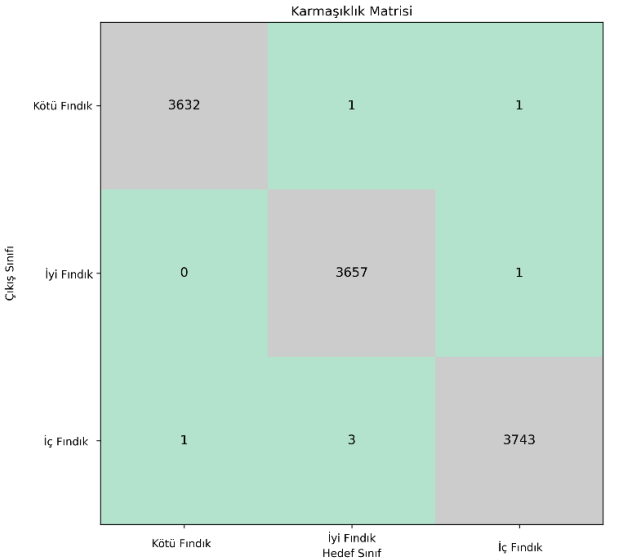
Çalışma kapsamında kullanılan derin öğrenme modellerinden ResNet-50 ve ShuffleNet için elde edilen karmaşıklık matrisleri grafiksel olarak sunulmuştur. ResNet-50 modeline ait karmaşıklık matrisi Şekil 4'te, ShuffleNet modeline ait karmaşıklık matrisi ise Şekil 5'te gösterilmektedir. Karmaşıklık matrislerinden elde edilen performans ölçütleri incelendiğinde, ResNet-50 modelinin oldukça yüksek bir başarı sergilediği gözlemlenmektedir. Bu model için hassaslık değeri 0.9997, özgünlük değeri 1, ve F1-skor değeri 0.9999 olarak hesaplanmıştır.

Şekil 4. ResNet-50 karmaşıklık matrisi



Kaynak: (Tomak et al., 2025)

Şekil 5. ShuffleNet karmaşıklık matrisi



Kaynak: (Tomak et al., 2025)

ShuffleNet modeli için elde edilen sonuçlar da oldukça yüksek performans değerlerine sahiptir. Bu modelde hassaslık değeri 0.9997, özgünlük değeri 0.9994 ve F1-skor değeri 0.9997 olarak belirlenmiştir. Sonuçlar genel olarak değerlendirildiğinde her iki modelin de fındık görüntülerinin sınıflandırılmasında oldukça başarılı olduğu gözlemlenmektedir. Ancak performans ölçütleri dikkate alındığında ResNet-50 modelinin ShuffleNet modeline kıyasla biraz daha yüksek başarı oranına sahip olduğu gözlemlenmektedir. Bu durum, ResNet-50 mimarisinin daha derin bir ağ yapısına sahip olması ve artık bağlantı (residual connection) mekanizması sayesinde daha güçlü özellik çıkarımı yapabilmesi ile açıklanabilir.

Fındıkların depolama sürecine alınmadan önce belirli bir temizleme ve ayıklama işleminden geçirilmesi büyük önem taşımaktadır. Bu aşamada fındıklar dikkatli bir şekilde incelenerek kötü fındıklar ve iç fındıklar diğer ürünlerden ayrılmalıdır. Eğer kötü

fındıklar ayıklanmazsa, elde edilen ürünün randıman değeri düşebilir ve bu durum hem kaliteyi hem de ekonomik değeri olumsuz etkileyebilir. Benzer şekilde, iç fındıkların temizlenmemesi durumunda depolama sürecinde bu fındıkların bozulma veya küflenme riski artabilir. Bu nedenle depolama öncesinde yapılan sınıflandırma ve ayıklama işlemleri, fındık kalitesinin korunması açısından kritik bir aşamadır.

Bu çalışmada, fındıkların otomatik olarak sınıflandırılabilmesi için iki farklı derin öğrenme algoritmasının performansı incelenmiştir. Bu amaçla ResNet-50 ve ShuffleNet mimarileri kullanılarak bir karşılaştırma yapılmıştır. Yapılan deneysel analizler sonucunda ResNet-50 modeli, eğitim aşamasında %99.92, test aşamasında ise %99.97 doğruluk oranı elde etmiştir. Diğer taraftan ShuffleNet modeli, eğitim sürecinde %99.79, test aşamasında ise %99.94 doğruluk oranına ulaşmıştır. Model performanslarının daha ayrıntılı şekilde değerlendirilmesi için karmaşıklık matrisi kullanılmış ve çeşitli performans ölçütleri hesaplanmıştır. Elde edilen sonuçlara göre ResNet-50 modeli, 0.9997 hassaslık, 1 özgünlük ve 0.9999 F1-skor değerlerine ulaşmıştır. ShuffleNet modeli ise 0.9997 hassaslık, 0.9994 özgünlük ve 0.9997 F1-skor değerleri elde etmiştir. Bu sonuçlar incelendiğinde ResNet-50 modelinin performans ölçütleri açısından çok küçük bir farkla daha iyi sonuçlar verdiği gözlemlenmektedir.

Bununla birlikte uygulama sürecinde modellerin çalışma hızları da değerlendirilmiştir. Yapılan gözlemler, kullanılan veri tabanı üzerinde gerçekleştirilen sınıflandırma işlemlerinde ShuffleNet modelinin ResNet-50 modeline kıyasla daha hızlı çalıştığını göstermiştir. Özellikle gerçek zamanlı sistemlerde hızlı sonuç elde etmek oldukça önemli bir faktördür. Bu nedenle doğruluk değerlerinin birbirine oldukça yakın olması ve ShuffleNet mimarisinin daha düşük hesaplama maliyeti ve daha hızlı çalışma süresi dikkate alındığında, fındık meyvesinin otomatik sınıflandırılması için ShuffleNet modelinin daha uygun bir seçenek

olabileceği sonucuna ulaşılmıştır. Gelecekte yapılacak çalışmalarda veri setinin daha geniş fındık türlerini içerecek şekilde genişletilmesi ve gerçek zamanlı sınıflandırma sistemlerinin geliştirilmesi önemli araştırma yönleri olarak değerlendirilebilir.

Kaynakça

Güneş, Engin (2022), "Hazelnuts", Mendeley Data, V2, doi: 10.17632/dvvx6kst3f.2

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Kim, K. G. (2016). Book review: Deep learning. Healthcare Informatics Research, 22(4), 351–354. https://doi.org/10.4258/hir.2016.22.4.351

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Rusk, N. (2016). Deep learning. Nature Methods, 13(1), 35. https://doi.org/10.1038/nmeth.3707

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. In 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (pp. 1–6). IEEE. https://doi.org/10.1109/ICCUBEA.2018.8697857

Tomak, Ö. (2019). Elektrokardiyografi sinyallerinde deneysel mod ayrıştırma ve geliştirilmiş karar ağaçları kullanarak aritmi tespiti. Karadeniz Fen Bilimleri Dergisi, 9(1), 103–110. https://doi.org/10.31466/kfbd.546569

Tomak, Ö., Dikbaş, M. C., & Külcü, S. (2025). Comparison of the performance of ResNet-50 and ShuffleNet deep learning algorithms in classification of hazelnut fruit. In 11th Azerbaijan Congress on Life, Engineering, Mathematical, and applied sciences

(pp. 55–63). https://doi.org/10.30546/19023.9789952-8573-5-1.2025.8988

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 6848–6856). https://doi.org/10.1109/CVPR.2018.00716

BÖLÜM 2

GÜVENLİ YAZILIM GELİŞTİRME YÖNTEMLERİ VE GÜVENLİK ÖNLEMLERİ

Murat Koca[1]
İsa Avcı[2]

Giriş

Sürekli gelişen teknoloji, günlük yaşam ve kurumsal süreçlerin ayrılmaz bir parçası hâline gelmiştir. Yazılım, donanımın belirli işlevleri yerine getirmesini sağlayan programlar, kütüphaneler ve yapılandırmaların bütünüdür. Yazılımı yalnızca kod geliştirme olarak değerlendirmek eksik kalır. Gereksinim analizi, mimari tasarım, doğrulama ve geçerleme, yayınlama ve bakım süreçlerini kapsayan planlı bir mühendislik faaliyetidir (Futcher & Von Solms, 2008). Yazılım geliştirme süreçleri genel olarak planlama, analiz, tasarım, geliştirme, test, yayınlama ile bakım ve güncelleştirme aşamalarından oluşur (Howard & Lipner, 2006).

Bireyler ve kuruluşlar günlük iş süreçlerinde yazılımlara giderek daha fazla bağımlı hâle geldiğinden, güvenli yazılım

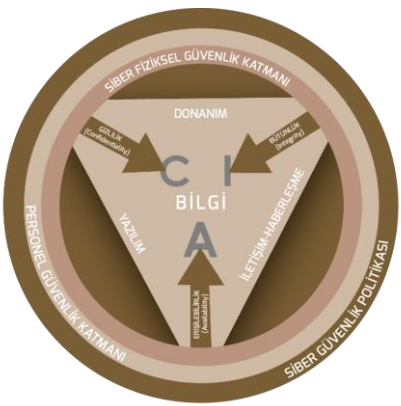
[1] Doktor Öğretim Üyesi, Van Yüzüncü Yıl Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği, Orcid: 0000-0002-6048-7645
[2] Doktor Öğretim Üyesi, Karabük Üniversitesi, Bilgisayar ve Bilişim Fakültesi, Bilgisayar Mühendisliği, Orcid: 0000-0001-7032-8018

ürünleri geliştirmek kritik önem taşır. Yazılım güvenliğinin etkisi, geliştirme sürecinde başlar. Uygulamanın kullanıcı tarafından kullanılmaya başlamasıyla devam eder ve bakım sürecinde sürekli olarak yönetilmesi gerekir. Sürekli değişen tehdit ortamı, güvenlik eğitimindeki eksiklikler, karmaşık yazılım mimarileri ve plansız geliştirme pratikleri, saldırganların hedeflerine ulaşmasını kolaylaştırabilmektedir. Bu nedenle güvenli yazılım geliştirme yaklaşımı son yıllarda belirgin biçimde önem kazanmıştır.

Bilgi güvenliğinin sağlanmasında temel çerçevelerden biri, gizlilik, bütünlük ve erişilebilirlik ilkelerinden oluşan CIA üçlüsüdür (Avcı, 2021; Hu et al., 2017; Kaufman, 2009; NIST, 2009). Bu bağlamda:

- Gizlilik, bilginin yalnızca yetkili kişilerce erişilebilir olmasını ve yetkisiz ifşa ile yetkisiz erişimin engellenmesini ifade eder.
- Bütünlük, bilginin doğruluk ve tutarlılığının korunmasını, yetkisiz değiştirme veya yok etmeye karşı güvence altına alınmasını ifade eder.
- Erişilebilirlik, yetkili kullanıcıların ihtiyaç duyduklarında hizmete ve veriye kesintisiz erişebilmesini ifade eder.

Şekil 1 Bilgi Güvenliği Temel Unsurlar



Kaynak: Cuylaerts (2022)

Hassas veriler yetkisiz üçüncü kişilerin eline geçtiğinde gizlilik ihlali oluşur. Zafiyet nedeniyle içerik yetkisiz biçimde değiştirildiğinde bütünlük bozulur. Zafiyetler sonucu uygulamalara erişim engellenirse erişilebilirlik zarar görebilir. Bu risklerin azaltılmasında erişim denetimleri, kimlik doğrulama ve yetkilendirme kontrolleri temel rol oynar. Ayrıca, kayıt altına alma ve izleme mekanizmalarıyla olayların tespiti ve müdahalesi güçlendirilir (Avcı & Koca, 2023).

Günümüzde yazılım pek çok alanda kullanılmaktadır. E-devlet gibi uygulamalar üzerinden resmî işlemler yürütülmektedir. Günlük alışverişten çevrim içi hizmetlere kadar geniş bir yelpazede yazılımlara kişisel veriler emanet edilmektedir (Avcı et al., 2022; T.C. Cumhurbaşkanlığı Dijital Dönüşüm Ofisi Başkanlığı, 2019). Bu nedenle yazılımların emanet edilen verileri güvenli biçimde saklaması ve işlemesi gerekmektedir. Son dönem bulgular, küresel ölçekte veri ihlallerinin ortalama maliyetlerinin milyonlarca dolar seviyesinde seyrettiğini göstermektedir (Bonderud, 2024; IBM Security, 2021; IBM Security & Ponemon Institute, 2022, 2023). Bu tablo, güvenli geliştirme süreçlerine erken yatırımın yalnızca teknik değil, aynı zamanda finansal ve yönetişimsel bir gerekçe taşıdığını ortaya koymaktadır.

Güvenli yazılım, kasıtlı olarak istenmeyen işlevleri gerçekleştirmeye zorlanamamalı, saldırı altında dahi doğru biçimde çalışmaya devam etmeli, saldırı örüntülerini tanıyabilmeli ve bu örüntülere karşı dayanıklı olmalıdır (McGraw, 2004). Ayrıca saldırı sonrası kısa sürede toparlanabilmeli ve en az hasarla hizmet sürekliliğini sağlamalıdır (Cichonski et al., 2012).

Son yıllarda kriptografik güvenliğe yönelik tehditlerin yalnızca klasik siber saldırılarla sınırlı kalmadığı görülmektedir. Kuantum hesaplama teknolojilerindeki gelişmeler, mevcut asimetrik

şifreleme mekanizmalarının uzun vadede risk altına girebileceğini göstermektedir. Bu nedenle kuantum sonrası döneme hazırlık, yazılım güvenliği yaklaşımının bir parçası hâline gelmiştir. NIST tarafından yayımlanan FIPS 203 ML-KEM ve FIPS 204 ML-DSA standartları, bu dönüşümün temelini oluşturmaktadır (NIST, 2024a, 2024b, 2024c). Kurumların kriptografik altyapılarını yeni algoritmalara uyumlu hâle getirebilmesi için kripto çevikliği yaklaşımının benimsenmesi gereklidir (ENISA, 2023; TNO, 2024).

Yazılım güvenliği perspektifinden kuantum sonrası kriptografiye geçiş, yalnızca algoritma değişimi değildir. Tasarım, geliştirme, test ile bakım ve güncelleştirme aşamalarında güvenli kodlama politikalarıyla bütünleşik biçimde ele alınmalıdır. Ayrıca Yazılım Malzeme Listesi (SBOM) içinde kullanılan kriptografik ilkelere ilişkin meta verilerin izlenebilir biçimde yönetilmesi, geçiş sürecinin sürdürülebilirliği açısından kritik önem taşır (ENISA, 2023; TNO, 2024).

Bu bölümde güvenli yazılım geliştirme yaşam döngüsü, yaygın zafiyet türleri, tehdit modelleme ve risk analizi yaklaşımları, güvenlik önlemleri, güvenlik testleri, tedarik zinciri güvenliği ile SBOM ve kuantum sonrası kriptografi geçişi literatüre dayalı olarak ele alınmıştır.

Çalışmanın Amacı ve Katkıları

Bu çalışma, güvenli yazılım geliştirmeyi yaşam döngüsü odaklı bir yaklaşımla ele alarak SSDLC aşamalarını NIST Secure Software Development Framework (SSDF) pratikleri ile ilişkilendirmeyi amaçlamaktadır (Souppaya et al., 2022). Uygulama güvenliğinin doğrulanabilir hâle getirilmesi için OWASP Application Security Verification Standard (ASVS) gereksinimleri ile süreç olgunluğunun değerlendirilmesi için OWASP Software Assurance Maturity Model (SAMM) yaklaşımı birlikte ele alınmıştır (OWASP Foundation, 2020; OWASP Foundation, 2025a). Ayrıca

güncel risk sınıflandırması OWASP Top 10:2025 çerçevesi ile desteklenerek güvenlik önceliklendirme bakışı sağlanmıştır (OWASP Foundation, 2025b).

Çalışmanın tedarik zinciri boyutundaki katkısı, SBOM ve Vulnerability Exploitability eXchange (VEX) yaklaşımını birlikte ele alarak bileşen şeffaflığını ve zafiyetlerin ürün üzerindeki etkisinin yönetimini somutlaştırmasıdır (CISA, 2023, 2025; IBM Security & Ponemon Institute, 2025). Son olarak, PQC geçişin kripto çevikliği ilkeleri doğrultusunda yazılım yaşam döngüsüne nasıl entegre edilebileceği tartışılmış; ML-KEM ve ML-DSA gibi standartların geçiş sürecine etkisi değerlendirilmiştir (ENISA, 2023; IETF TLS, 2025; NIST, 2024a, 2024b; TNO, 2024).

Araştırma Yöntemi

Bu çalışma, standart temelli güvenli yazılım pratiklerini, uygulama doğrulama gereksinimlerini ve tedarik zinciri şeffaflığını tek bir uygulanabilir çerçevede birleştirmek amacıyla tasarlanmıştır. Yöntem iki aşamadan oluşmaktadır. Birinci aşamada NIST SSDF (SP 800-218), OWASP ASVS 5.0.0, OWASP SAMM v2 ve OWASP Top 10:2025 dokümanları incelenmiştir. SSDLC aşamalarında uygulanması beklenen güvenlik pratikleri belirlenmiş ve bu pratiklerin hangi çıktılarla doğrulanacağı çıkarılmıştır (OWASP Foundation, 2020; OWASP Foundation, 2025a, 2025b; Souppaya et al., 2022). İkinci aşamada SBOM'un üretimi ve tüketimi, benimsenme engelleri ve zafiyet etkisi yönetimi bağlamında güncel akademik çalışmalar ile kamuya açık rehberler derlenmiştir. Tedarik zinciri güvenliği için gereksinimler ve uygulanabilir kontrol adımları tanımlanmıştır (CISA, 2023, 2025; IBM Security & Ponemon Institute, 2025; Stalnaker et al., 2024). Bu iki aşamanın çıktıları birleştirilerek "bütünleşik güvenli yazılım çerçevesi" oluşturulmuştur. Çerçeve, CI ve CD süreçlerinde uygulanabilir kontrol adımları ve ölçütler ile desteklenmiştir.

Güvenli Yazılım Geliştirme Yaşam Döngüsü

Güvenli yazılım geliştirme yaşam döngüsü, yazılımın fikir aşamasından başlayarak kullanıcıların uygulamayı kullandığı süre boyunca geçen tüm aşamalarda güvenliğin sistematik biçimde ele alınmasını ifade eder. Güvenliği sonradan eklemek hem maliyeti artırır hem de tasarım kararlarını geri döndürmeyi zorlaştırır. Bu nedenle güvenlik, yaşam döngüsünün doğal bir parçası olmalıdır. NIST SSDF, OWASP SAMM ve OWASP ASVS birlikte kullanıldığında süreç olgunluğu, doğrulama gereksinimleri ve uygulama kontrolleri daha somut hâle getirilebilir (OWASP Foundation, 2020; OWASP Foundation, 2025a; Souppaya et al., 2022).

Bu çalışmada SSDLC adımları, planlama, analiz, tasarım, geliştirme, test, yayınlama ile bakım ve güncelleştirme olarak ele alınmıştır. Her aşama ayrıntılı biçimde planlanmalıdır. Çıktılar ve kanıtlar ile izlenebilir hâle getirilmelidir (Howard & Lipner, 2006; Souppaya et al., 2022). Güvenli yazılım yaşam döngüsü Şekil 2'de gösterilmektedir.

Şekil 2 Güvenli Yazılım Geliştirme Yaşam Döngüsü

Kaynak: Souppaya et al. (2022)

Planlama

Planlama aşaması, proje hedeflerinin, kapsamın, kaynakların ve yol haritasının belirlendiği başlangıç aşamasıdır. Bu aşamada güvenlik beklentileri netleştirilmelidir. Minimum güvenlik gereksinimleri, risk iştahı ve uyumluluk hedefleri tanımlanmalıdır (Souppaya et al., 2022). Tehdit modelleme teslimi, risk kabul kayıtları, üçüncü taraf bağımlılıkları ile SBOM üretim ve paylaşım kuralları planlama çıktıları arasında yer almalıdır (CISA, 2025; Souppaya et al., 2022).

Analiz

Analiz aşamasında sistemin işlevleri ayrıntılı biçimde belirlenir ve gereksinimler doğrulanabilir kriterlere dönüştürülür. Güvenlik gereksinimlerini yeterince tanımlamak için tehdide dayalı risk değerlendirmesi yapılmalıdır (Gollagi et al., 2021). Sisteme yönelik tehditlerin hem geliştirme sürecinde hem de yayınlandıktan sonra değişebileceği göz önünde bulundurulmalıdır. Bu kapsamda:

- Güvenliğin yazılım geliştiricisi tarafından ele alınacağını varsayılamaz.
- Güvenlik gereksinimlerini yeterince tanımlamak ve belirlemek için, sistem yayınlandıktan sonra karşılaşılabilecek tehditleri anlamak için, tehdide dayalı bir risk değerlendirmesi yapılmalıdır. Geliştirme ekibi, sistem geliştirme aşamasındayken ve yayınlandıktan sonra sisteme yönelik tehditlerin değişebileceğini anlamalıdır.
- Bir özellik istenen bir özellik değilse, yazılıma uygulanmaz ve test edilmez.

Yukarıda bahsedilen güvenlik gereksinimleri de şu şekilde sıralanabilir:

- Genel güvenlik gereksinimleri uygulanmalıdır.
- Birçok bilişim teknolojileri sistemleri “Kullanıcı Adı ve Parola”, “Erişim Kontrolleri”, “Giriş Doğrulama”, “Denetim” gibi standart güvenlik önlemlerine sahiptir.
- Test araçları, bütün kullanıcılar yazılımı kullanma konusunda uzman değildir. Kullanıcıların normal kullanım durumlarının yanı sıra hatalı ve kötüye kullanım durumları da kontrol edilmelidir.

Tehditlerin sistematik belirlenmesi için STRIDE veya iş etkisi odaklı PASTA yöntemi tercih edilebilir. Veri akış diyagramları ve güven sınırları üzerinden saldırı yüzeyi çıkarılmalı ve risk önceliklendirme yapılmalıdır (Kirtley, 2022; Microsoft, 2025).

Tasarım

Tasarım aşaması, analiz çıktılarının mimariye dönüştürüldüğü aşamadır. Mantıksal tasarımda sistemin işlevsel yapısı, fiziksel tasarımda ise bileşenler ve ayrıntılar tanımlanır. Tasarım kararları güvenlik üzerinde doğrudan etkilidir. Bu nedenle tasarım aşamasında güvenli varsayılanlar, katmanlı savunma, en az ayrıcalık ve izolasyon ilkeleri uygulanmalıdır (Gollagi et al., 2021; Jones & Rastogi, 2004; Saltzer & Schroeder, 1975).

Tasarım aşamasında dikkat edilmesi gereken ilkeler:

- Doğruluk tek başına güvenlik anlamına gelmez.
- Katmanlı güvenlik ek koruma sağlar ve saldırı maliyetini artırır.
- Yüksek riskli teknolojilerden kaçınmak riskleri azaltır.
- Az güvenilir işlevler izole edilmelidir.
- Yetkiler minimum düzeyde tutulmalıdır.

- Tersine mühendislik karşısında dayanıklılık artırılmalıdır.

Tasarım aşamasında güvenlik için yapılması gerekenler:

- Tasarım aşamasına güvenlik uzmanı dâhil edilmelidir.
- Tasarım dokümantasyonu, algoritma ve akış diyagramları, normal, hatalı ve kötüye kullanım durumları ve tehdit modeli çıktılarıyla desteklenmelidir.
- Tasarım bağımsız bir güvenlik incelemesinden geçmelidir.
- Güvenlik mekanizmaları modüler ve merkezileştirilebilir biçimde tasarlanmalıdır.

Web ve API uygulamalarında gereksinimler OWASP ASVS ile izlenebilir hâle getirilmeli ve kritik uygulamalarda daha yüksek doğrulama hedefleri benimsenmelidir (OWASP Foundation, 2025a). Ayrıca mimari düzeyde kripto çevikliği sağlanmalı; klasik mekanizmalara ek olarak kuantum sonrası kriptografi uyumlu anahtar değişimi ve imza süreçleri desteklenmelidir (ENISA, 2023; IETF TLS, 2025; NIST, 2024a, 2024b; TNO, 2024).

Yazılım Geliştirme

Geliştirme aşaması, tasarımda belirlenen mimariye uygun kodlama ve birim düzeyi doğrulama faaliyetlerini kapsar.

Kodlama

Programlama dili ve çerçeve seçimi, güvenlik açısından doğrudan etkili olabilmektedir. Kod okunabilir, bakımı kolay, standartlara uygun ve denetlenebilir olmalıdır. Kod karmaşıklığı azaltılmalı, güvenli kodlama standartları oluşturulmalı ve ekip eğitimleri planlanmalıdır (Ergasheva & Kruglov, 2020; Mohammed

et al., 2017). Güvenli kod geliştirme kapsamında şu adımlar öne çıkar (Mohammed et al., 2017):

- Geliştiricilere güvenli kod yazımı eğitimi verilmesi,
- Güvenliği kanıtlanmış bileşen ve kütüphanelerin tercih edilmesi,
- Girdi doğrulama, kimlik doğrulama, yetkilendirme ve kayıt altına alma mekanizmalarının doğru uygulanması,
- Kod gözden geçirme süreçlerinin işletilmesi.

PQC perspektifinden, kriptografik çağrılar soyut arayüzler üzerinden yürütülerek algoritma bağımsızlığı sağlanmalıdır. Bu yaklaşım kripto çevikliği hedefini destekler ve ileride yapılacak algoritma geçişlerini kolaylaştırır (ENISA, 2023; TNO, 2024). NIST tarafından standartlaştırılan ML-KEM ve ML-DSA gibi algoritmalara geçiş için gerekli altyapı kodlama aşamasında planlanmalı ve test edilebilir hâle getirilmelidir (NIST, 2024a, 2024b, 2024c).

Geliştirme sürecinde statik analiz, dinamik analiz, etkileşimli analiz ve bağımlılık analizi araçları CI ve CD hatlarına entegre edilmelidir. İmzalı çıktılar, tekrarlanabilir derleme, kod inceleme kayıtları ve SBOM üretimi standart hâline getirilmelidir (CISA, 2025; Souppaya et al., 2022).

Test

Test aşaması, yazılımın gereksinimleri karşılayıp karşılamadığını ve güvenlik zafiyetleri içerip içermediğini ortaya koyar. Testlerde amaç, yazılımın hatalı işlem yapması veya durmasına neden olabilecek güvenlik kusurlarının bulunmaması, beklenmeyen davranışların tespiti ve gömülü sırlar gibi tehlikeli yapıların ortaya çıkarılmasıdır (Gollagi et al., 2021; Mohammed et al., 2017).

Test sürecinde:

- Saldırı örüntülerine benzer girdiler ve akışlar uygulanmalı,
- Beklenmeyen girdiler ile hata yönetimi izlenmeli,
- Modüller arası etkileşimler ve kullanıcı etkileşimleri doğrulanmalı,
- Girdi doğrulama, hata kontrolü ve kayıt mekanizmaları test edilmelidir.

Ayrıştırıcılar ve protokoller gibi bileşenlerde fuzz testi yüksek bulgu verimine sahiptir; iş akışı ve tasarım kusurları için penetrasyon testleri gereklidir (Felderer et al., 2016; Oka, 2020). Test sonuçları kalite kapıları ile ilişkilendirilmeli; kritik bulgularda yayın süreci durdurulmalıdır (Souppaya et al., 2022).

Yayınlama

Yayınlama aşaması, yazılımın üretim ortamına alınması ve güvenli varsayılanların doğrulanması süreçlerini kapsar. Yayınlamadan önce:

- Varsayılan geliştirici ayarları ve test yapılandırmaları kaldırılmalı,
- Hata ayıklama kodları kapatılmalı,
- Açıklama satırlarında hassas bilgi bulunmadığı doğrulanmalı,
- Varsayılan hesaplar ve test hesapları silinmeli,
- Parolalar ve sırlar secret management yaklaşımıyla yönetilmeli,

- İşletim sistemi, web sunucusu ve veri tabanı güvenlik yapılandırmaları güçlendirilmelidir (Dodson et al., 2020; Gollagi et al., 2021).

TLS 1.3 uç noktaları ve kod imzalama süreçleri için hibrit konfigürasyonlar doğrulanmalı; yalnızca klasik ilkelere bağımlı bileşenler için risk kabul kaydı oluşturulmalıdır (ENISA, 2023; IETF TLS, 2025; NIST, 2024a, 2024b; TNO, 2024). Ayrıca CSP, HSTS ve karşılıklı TLS gibi güvenli varsayılanlar doğrulanmalıdır (OWASP Foundation, 2025a).

Yayınlandıktan sonra:

- Yayınlanan sistem düzenli olarak izlenmeli,
- Kullanım ve güvenlik olayları kayıt altına alınmalı,
- Anomali tespiti ve müdahale süreçleri işletilmelidir (Cichonski et al., 2012; Souppaya et al., 2022).

Bakım ve Güncelleştirme

Bakım ve güncelleştirme aşaması, yazılımın üretimde kullanıldığı süreçte hataların giderilmesi, yamaların uygulanması, yeni özelliklerin eklenmesi ve güvenlik iyileştirmelerinin sürdürülmesini kapsar (Dodson et al., 2020; Gollagi et al., 2021). Bu aşamada:

- Bağımlılık yamaları ve sürüm güncellemeleri takip edilmelidir.
- Her düzeltme sonrası güvenlik etkisi yeniden değerlendirilmelidir.
- Büyük sürümlerde güvenlik analizleri tekrarlanmalıdır.
- Tüm değişiklikler kayıt altına alınmalı ve denetlenebilir hâlde tutulmalıdır.

Zafiyet duyurularına yanıt süreleri için hizmet seviyesi hedefleri belirlenmeli, otomatik bağımlılık güncellemeleri ve periyodik güvenlik gözden geçirmeleri uygulanmalıdır (OWASP Foundation, 2025b). Kripto yaşam döngüsü politikası kapsamında anahtar rotasyonu, algoritma yükseltmesi ve kuantum sonrası kriptografiye kademeli geçiş periyodik olarak gözden geçirilmelidir (ENISA, 2023; TNO, 2024).

Yazılım Geliştirmede Yaşanan Güvenlik Sorunları ve Risk Modelleme

Yazılım kusurları, kötüye kullanılabilecek açıklar olarak ortaya çıkar. Bu kusurlar, yazılımın çalıştığı ortamın ve kullanılan protokollerin eksikliklerinden, mimari tasarım hatalarından veya temel bileşenlerdeki zayıflıklardan kaynaklanabilir. Yazılımlarda karşılaşılan güvenlik hataları genel olarak uygulama hataları ve mimari hatalar olmak üzere iki grupta ele alınır. Güncel risk alanları erişim kontrolü, kriptografik hatalar, enjeksiyon ve tedarik zinciri zayıflıkları etrafında yoğunlaşmaktadır. OWASP Top 10:2025 bu çerçeveyi güncel biçimde sunmaktadır (OWASP Foundation, 2025b).

Uygulama hatalarına örnekler:

- Buffer Overflow,
- Güvensiz sistem değişkenleri,
- Güvensiz sistem çağrıları,
- Güvensiz girdi işleme.

Mimari hatalara örnekler:

- Hatalı şifreleme tasarımı veya uygunsuz kripto seçimleri,
- Bölümleme ve izolasyon hataları,
- Ayrıcalıklı blok koruma hataları,

- Mantıksız erişim kontrolleri,
- Güvenli olmayan kalıtım ve metot geçersiz kılma tasarımları.

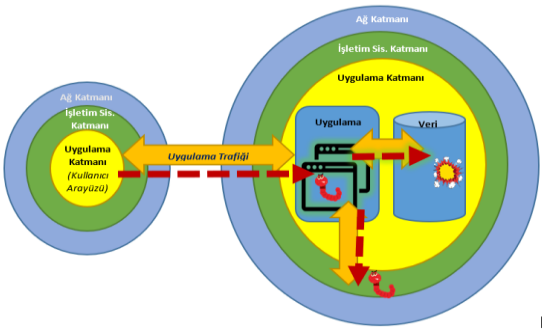
Yazılım Geliştirmede Güvenlik Zafiyetleri

Yazılım güvenlik açığı, saldırganın sistem denetimini ele geçirmesine veya yetkisiz işlem yapmasına izin verebilecek bir hatadır. Saldırganlar aşağıdaki yöntemlerle sisteme sızmaya çalışabilir:

- **Session Hijacking:** Oturum açmış başka bir kullanıcının bilgilerini ele geçirebilir (Baitha & Vinod, 2018).
- **Command Injection:** TextBox'tan veri tabanına zarar verebilecek komutlar girebilir (Stasinopoulos et al., 2019).
- **Cross Site Scripting:** Diğer kullanıcıların bilgilerini çalmak için sunucu üzerinde script çalıştırabilir (Rodríguez et al., 2020).
- **Buffer Overflows:** Yazılımı bellek taşması hatasına düşürüp zararlı kod yükleyerek sunucuyu ele geçirebilir (Kiriansky & Waldspurger, 2018).
- **Denial of Service:** Belirli bir kullanıcıyı ya da tüm sistemi işlem yapamaz hale getirebilir (Khalaf et al., 2019).

Saldırganlar genellikle yazılımın bulunduğu sunucuya erişmek için uygulamalara Şekil 3'teki gibi saldırılar gerçekleştirir. Bu şekilde sunucu kaynaklarını kendi amaçları için kullanabilir veya sunucu üzerinde bulunan veri tabanına erişerek verileri ele geçirebilir.

Şekil 3 Uygulama Saldırı Topolojisi



Yazılım Geliştirmede Güvenlik Zafiyetlerinin Maliyeti

Saldırılar sonucu oluşan veri ihlalleri yüksek maliyetlere neden olmaktadır (IBM, 2019). IBM raporlarına göre küresel ortalama ihlal maliyeti 2024 yılında 4,88 milyon ABD doları seviyesindeyken, 2025 raporunda ortalamanın 4,44 milyon ABD doları olarak raporlandığı görülmektedir; sektör ve bölge kırılımında maliyetler belirgin biçimde değişebilmektedir (Bonderud, 2024; IBM Security & Ponemon Institute, 2025). Bu bulgular, güvenli geliştirme süreçlerine erken yatırımın gerekçesini güçlendirmektedir. Güvenli geliştirme yaklaşımı izlenmeden yürütülen projelerde karşılaşılacak güvenlik açıklarının giderilmesi, geriye dönük iş yükü doğurur ve maliyet artışına yol açar. Literatürde, hataların yaşam döngüsünün ileri aşamalarında düzeltilmesinin maliyetinin arttığı vurgulanmaktadır (Stecklein, 2004). Bu nedenle güvenlik kontrollerinin erken aşamalara taşınması gereklidir.

Şekil 4, 2021 ile 2025 dönemi için ülke ve bölge düzeyinde ortalama veri ihlali maliyetlerini karşılaştırmalı olarak sunmaktadır. Bulgular, ABD'nin yüksek maliyet bandında yer aldığını ve bazı bölgelerde maliyetlerin dalgalanabildiğini göstermektedir (IBM Security, 2021; IBM Security & Ponemon Institute, 2022, 2023, 2025; Bonderud, 2024).

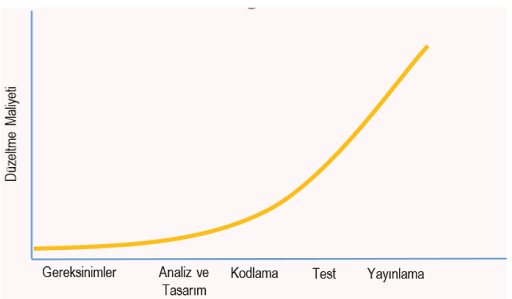
Şekil 4 Ülke ve Bölgeye Göre Veri İhlalinin Ortalama Maliyeti

	2021($ Million)	2022($ Million)	2023($ Million)	2024($ Million)	2025($ Million)
USA	9,05	9,44	9,48	9,36	10,22
Orta Doğu	6,93	7,46	8,07	8,75	7,29
Kanada	5,4	5,64	5,13	4,66	4,84
Almanya	4,89	4,85	4,67	5,31	4,03
Japonya	4,69	4,57	4,52	4,19	3,65
Birleşik Krallık	4,67	5,05	4,21	4,53	4,14
Fransa	4,57	4,34	4,08	4,17	3,73
Güney Kore	3,68	3,57	3,48	3,62	2,84
İtalya	3,61	3,74	3,86	4,73	3,44
Güney Afrika	3,21	3,36	2,79	2,78	2,37
Avustralya	2,82	2,92	2,7	2,78	2,55
Asya	2,71	2,87	3,05	3,23	3,67
İskandinavya	2,67	2,08	1,91	0	0
Latin Amerika	2,56	2,8	3,69	4,16	3,81
Hindistan	2,21	2,32	2,18	2,35	2,51
Türkiye	1,91	1,11	0	0	0
Brazilya	1,08	1,38	1,22	1,36	1,22
Benelüks (Belçika, Hollanda, Lüksemburg)	0	0	0	5,9	6,24

Kaynak: IBM Security (2021); IBM Security & Ponemon Institute (2022, 2023, 2025); Bonderud (2024)

Güvenli bir yazılım geliştirme yaklaşımı kullanılmadan yürütülen projelerde karşılaşılacak sorunlar, geriye dönük iş yükleri oluşturur ve bazı durumlarda yazılımın baştan yazılmasına kadar gidebilecek maliyetleri tetikleyebilir. Yapılan araştırmalar, hataların geç aşamalarda düzeltilmesinin maliyeti belirgin biçimde artırdığını göstermektedir (Stecklein, 2004).

Şekil 5 Geliştirme Aşamaları ve Düzeltme Maliyetleri

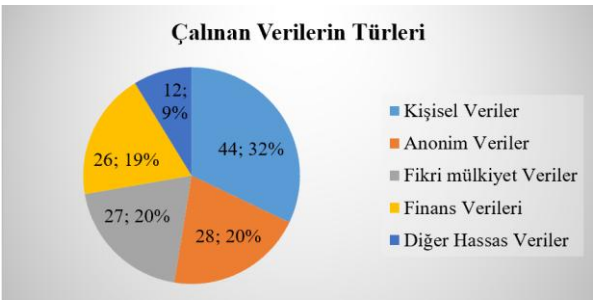


Kaynak: Stecklein (2004)

Yazılım Geliştirmede Ele Geçirilen Veri Türleri

Veri ihlallerinde ele geçirilen veri türleri maliyet üzerinde doğrudan etkilidir. Kimlik verileri, iletişim bilgileri ve kurumsal sırlar gibi farklı veri türleri farklı etki profilleri oluşturur. Bu nedenle veri sınıflandırması ve veri işleme politikaları, tehdit modelleme çıktılarıyla ilişkilendirilmelidir. Diğer çalınma yöntemlerine göre maliyet frekansları da Şekil 6'da görülmektedir.

Şekil 6 Çalınan Verilerin Türleri



Güvenli Yazılım Geliştirmede Tehdit Modelleme ve Risk Analizi

Yazılım geliştirirken oluşabilecek tehditler öngörülmeli ve modellenmeli; karşılaşılabilecek riskler analiz edilmelidir. Bu şekilde gelecekte ortaya çıkabilecek riskler büyük ölçüde önlenebilir. Tehdit modelleme sürecinde temel adımlar şu şekilde özetlenebilir:

Uygulama modelini oluşturun (DFD, UML vb.):

- Korunması gereken verilerin bir listesini oluşturun.

Her bir saldırıda hedef alınabilecek noktaları kategorize edin:

- Sahte bilgi oluşturma,
- Bilgileri bozma,

- Başvuruları reddetme,
- Bilgileri ifşa etme,
- Hizmetleri durdurma,
- Yetkileri yükseltme.

Tehditleri risklerine göre sıralayın:

- Risk = Potansiyel x Hasar,
- Hasar potansiyeli,
- Tekrarlanabilirlik,
- Kötüye kullanılabilirlik,
- Etkilenen kullanıcılar vb.

Risk analizi, yazılımın mimari tasarım aşamasında başlar. Bu aşamada yapılan mimari risk analizinde aşağıdaki hususlar dikkate alınmalıdır:

- Değerlendirme işlemi sistem tasarımcılarına bırakılmamalıdır,
- Basit bir tasarım çizilmelidir,
- Riskleri kategorize etmek için hipotezler üretilmelidir,
- Riskleri önem sırasına konulmalıdır,
- Bütün bu işlemler yapılacak olan işle ilişkilendirilmelidir,
- Muhtemel hasarlara karşı çözümler önerilmelidir,
- Bu işlemleri tekrar tekrar uygulanmalıdır.

Değerlendirme yalnızca tasarımcıya bırakılmamalıdır. Bağımsız güvenlik incelemeleri ve dış göz değerlendirmeleri ile desteklenmelidir. STRIDE, veri akış diyagramı ve güven sınırları üzerinden tehdit sınıflandırması için uygun bir yöntemdir. PASTA

ise iş etkisi analiziyle başlayıp risk önceliklendirmeye uzanan bir yaklaşım sunar (Kirtley, 2022; Microsoft, 2025). Ayrıca penetrasyon testleri proje ekibi dışından kişilerce yürütüldüğünde kör noktaları azaltır (Gollagi et al., 2021).

Güvenli Yazılım için Alınması Gereken Önlemler

Yazılımda güvenlik, geliştirme aşamasında uygulanmaya başlar. Böylece klasik yaşam döngüsü yaklaşımı, güvenlik odaklı yaşam döngüsüne dönüşür (Fujdiak et al., 2019; Gollagi et al., 2021). Bu dönüşüm, zafiyetleri azaltırken düzeltme maliyetlerini de düşürür (Ergasheva & Kruglov, 2020; Stecklein, 2004). Süreç olgunluğu için OWASP SAMM, uygulama doğrulaması için OWASP ASVS ve süreç pratikleri için NIST SSDF birlikte değerlendirilebilir (OWASP Foundation, 2020; OWASP Foundation, 2025a; Souppaya et al., 2022). Yazılım döngüsünde güvenlik ile ilgili hazır ürün çözümleri ve güvenli geliştirme metodolojileri arasındaki ilişki Tablo 1'de özetlenmiştir.

Tablo 1 Güvenlik Aşamaları ve Önlemleri

	Hazır Ürün	**Güvenli Yazılım Metotları Geliştirme**
İhtiyaçlar	-	Güvenli geliştirme danışmanlığı
Tasarım	-	Güvenli tasarım inceleme
Uygulama	Otomatik yazılmış kodlar için Statik analiz	Tehdit modelleme Güvenlik test planlama
Test	Güvenlik sorunlarını bulmak için özel test durumları geliştir	Uygulama güvenlik testi Web-uygulaması güvenlik testi
Yayınlama	Network tarayıcıları Uygulama tarayıcıları	Ağ penetrasyon testi
Bakım	Yazılım yamaları	Güvenlik denetimi Mevzuata uygunluk denetimi 3rd Party doğrulama

Eğitim ve Farkındalık

Günümüzde yazılım konusu her zamankinden çok daha büyük ilgi çekmektedir. Ancak 4000'den fazla yazılım geliştiriciyle yapılan bir ankete göre, geliştiricilerin yarısından azının güvenlik açıklarını tespit edebildiği raporlanmıştır (Gasiba et al., 2020). Güvenlik açıkları içeren yazılımlar kritik altyapılarda konuşlandırıldığında riskin etkisi daha da büyümektedir. Bu nedenle yazılım geliştirme ekiplerinde güvenlik uzmanı bulunması önemlidir. Ekipte güvenlik uzmanı bulunamayan durumlarda ise geliştiricilerin güvenlik konularında bilgi sahibi olmaları ve gerekli eğitimleri almaları gerekir. Ayrıca geliştiricilere ve kullanıcılara yönelik farkındalık eğitimlerinin düzenlenmesi zorunlu bir ihtiyaç olarak öne çıkmaktadır.

Proje Döngüsü Yönetimi

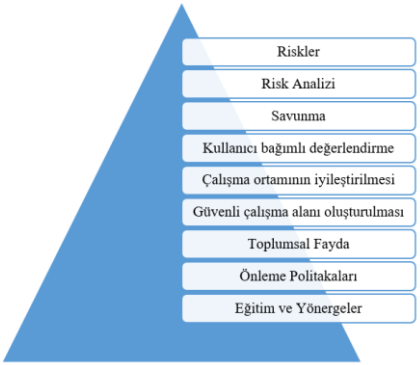
Proje döngüsü yönetimi, sağlıklı, sürdürülebilir, yenilikçi ve güvenli yazılım geliştirme için temel bir çerçevedir. Bir projenin başlangıcından geliştirilmesine, tamamlanmasına ve yürütülmesine kadar tüm ilerleyişini bütünleşik biçimde ele alır. Proje kapsamında amaç, hedef, kazanımlar, çıktılar ve faaliyetler; bağımlılıklar ve sorumluluklar ile zamanlama, bütçe ve doğrulama kaynakları gibi göstergeler üzerinden planlanmalıdır. Bu bölümde tanımlanan yönergeler doğrultusunda güvenli yazılım geliştirme faaliyetleri yürütülür (Souppaya et al., 2022).

Güvenlik ve Tasarım Prensiplerinin Belirlenmesi

Tasarım kararları güvenliği doğrudan etkiler. Tasarımcılar, sistem tasarımlarını oluştururken güvenlik tedbirlerini dikkate almalıdır (Jones & Rastogi, 2004). Güvenlik prensipleri, riskin ortadan kaldırılması veya azaltılması hedefiyle belirlenmelidir. Erişim denetimi modelleri (DAC, MAC, RBAC, ABAC), acil durum senaryoları, en az ayrıcalık ve güvenli varsayılanlar birlikte ele alınmalıdır (OWASP Foundation, 2025a). Gereksinimler, doğrulama

kriterleriyle eşleştirilerek denetlenebilir hâle getirilmelidir (OWASP Foundation, 2025a). Güvenlik ve tasarım prensipleri önem sırasına göre Şekil 7'de gösterilmiştir.

Şekil 7 Güvenlik ve Tasarım Prensipleri



Ürün Risk Analizi ve Değerlendirmesi

Güvenlik riski değerlendirmesi, uygulamalardaki temel güvenlik kontrollerini tanımlar, değerlendirir ve uygular. Aynı zamanda uygulama güvenlik kusurlarını ve güvenlik açıklarını önlemeye odaklanır. Risk değerlendirmesi yapmak, bir kuruluşun uygulama portföyünü saldırgan bakış açısından bütünsel olarak görmesine ve bilgiye dayalı kaynak tahsisi yapmasına katkı sağlar (Canedo et al., 2022; Jimoh et al., 2022; Koca & Avci, 2024). Değerlendirme sürecinde saldırı alanları, kodlama alanları, yetkilendirme ve gizlilik alanları analiz edilir ve ardından tehditlerin modellenmesine geçilir. Bu aşamada senaryoların tanımlanması, dış bağımlılıkların listelenmesi, güvenlik varsayımlarının belirlenmesi, tehdit tiplerinin sınıflandırılması, risklerin çıkarılması ve önlemlerin planlanması hedeflenir.

Risk değerlendirmeleri kabul, azalt, transfer veya kaçın kararlarıyla sonuçlanmalıdır. Her karar için kanıt ve sahiplik atanmalı; yeniden tehdit modelleme tetikleyicileri (büyük mimari

değişiklik, yeni veri kategorisi, yeni entegrasyon vb.) açık biçimde tanımlanmalıdır (Kirtley, 2022).

Güvenlik Dokümanları ve Araçlarının Geliştirilmesi

Yazılım belgeleri, bir yazılımın nasıl çalıştığını, neden oluşturulduğunu ve nasıl kullanılmasının amaçlandığını açıklayan yazılı çıktılardır (Karim et al., 2016). Yazılımın karmaşıklığına bağlı olarak belgeler; genel kullanım, işlev ve özelliklerin ayrıntılı açıklamalarını içermelidir. Dokümantasyonda öğrenme odaklı eğitimler, yol gösterici kılavuzlar, çalışma mantığını açıklayan yönergeler ve yardımcı belgeler yer almalıdır. Dokümantasyona mimari karar kayıtları (ADR), tehdit modeli çıktıları, SBOM (SPDX veya CycloneDX) ve güvenlik güncelleme politikası eklenmesi önerilir (Souppaya et al., 2022).

Güvenli Kodlama Politikalarının Geliştirilmesi

Güvenli yazılım geliştirme, kodun güvenli olmasının yanı sıra kullanıcı dostu bir ürün ortaya koymayı da gerektirir (Gasiba & Lechner, 2019). Geliştirme yaşam döngüsü boyunca güvenlik açıkları izlenmelidir. Yayınlamadan önce ve yayın sonrasında tespit edilen bulgular kontrol edilip giderilmelidir (Gasiba et al., 2020). Bu nedenle güvenli kodlama politikalarının belirlenerek uygulamaya konulması gerekir. Politika geliştirme kapsamında; derleyici ve destekleyici araç sürümlerinin güncel tutulması, arabellek güvenlik kontrollerinin eksiksiz yapılması, kaynak kod analiz araçlarının kullanılması, yasaklı işlevlerden kaçınılması, gereksiz hazır kütüphane kullanımının azaltılması ve güvenli kodlama kontrol listelerinin işletilmesi önemlidir. Politikalar; yasaklı API'ler, girdi doğrulama stratejisi, kripto politikası (algoritma, mod, anahtar yönetimi), gizli yönetimi ve kod inceleme onayı kriterlerini içermeli ve gereksinimler ASVS ile eşlenmelidir (OWASP Foundation, 2025a).

Güvenli Test Politikalarının Geliştirilmesi

Siber güvenlik açıklarını ve diğer potansiyel güvenlik kusurlarını ortaya çıkarmak ve güvenlik gereksinimlerinin doğru uygulandığını doğrulamak için test yöntemleri geliştirilmelidir (Khakzad et al., 2017). Penetrasyon testi manuel bir süreçken, fuzz testi otomatik bir süreçtir (Felderer et al., 2016; Oka, 2020). Fuzz testinde hedefi bozmayı amaçlayan çok sayıda farklı girdi denenir; kapsamı artırmak için hedefin farklı bileşenleri (kod, ikili kitaplıklar, arayüzler) analiz edilebilir. Penetrasyon testleri ise bilinen (veya sıfır gün) zafiyetlerden yararlanarak saldırı yüzeyine erişim sağlamaya çalışır.

Test politikalarında ayrıca çalışma zamanı doğrulaması, tehdit modellerinin düzenli gözden geçirilmesi ve saldırı yüzeyinin yeniden değerlendirilmesi gibi faaliyetler de kapsanmalıdır. SAST, DAST, IAST, SCA ile fuzz ve penetrasyon testi kombinasyonu, CI ve CD süreçlerinde kalite kapısı olarak konumlandırılmalı; başarısızlık durumunda derleme ve yayın durdurulmalıdır (Souppaya et al., 2022).

Yazılım Tedarik Zinciri Güvenliği ve SBOM

Modern yazılım ekosistemlerinde üçüncü taraf bileşenler ve açık kaynak bağımlılıkları, güvenlik risklerini kurumsal sınırların dışına taşır. Bu nedenle tedarik zinciri güvenliği; bileşen kaynağının doğrulanması, bütünlük güvencesi, zafiyet yönetimi ve izlenebilirlik boyutlarını kapsayan bir yönetişim gerektirir (Souppaya et al., 2022). SBOM, bu risklerin yönetiminde temel bir çıktı olarak konumlanır. Her sürüm için SPDX veya CycloneDX formatlarında üretilen SBOM; bileşen adı, sürümü, lisans bilgisi, zafiyet referansları ve mümkünse kriptografik meta verileri içerecek şekilde yönetilmelidir (CISA, 2025; CycloneDX, 2025; ENISA, 2023; Souppaya et al., 2022; SPDX, 2024; TNO, 2024).

Tedarik zinciri güvenliği kapsamında önerilen uygulamalar:

- Derleme çıktılarında tekrarlanabilir derleme ve imzalı artefakt dağıtımı (OpenSSF, 2025; Souppaya et al., 2022),
- Politika temelli bağımlılık yönetimi ve kritik zafiyetlerde derleme engelleme eşikleri (Souppaya et al., 2022),
- SBOM'un her yayınla birlikte versiyonlanması ve paydaşlarla paylaşılması (CISA, 2025; ENISA, 2023; Souppaya et al., 2022; TNO, 2024),
- Kripto çevikliği ve kuantum sonrası kriptografi geçişinin izlenebilir biçimde yönetilmesi (ENISA, 2023; OWASP Foundation, 2025a; TNO, 2024),
- Yayın ve bakımda güvenli varsayılanların doğrulanması ve zafiyet duyurularına yanıt hedeflerinin işletilmesi (OWASP Foundation, 2025a, 2025b; Souppaya et al., 2022).

Sonuç olarak SBOM'un kurumsal süreçlere yerleşmesi ve SSDF uyumlu kontrollerin CI ve CD hatlarına bağlanması, zafiyet görünürlüğünü artırırken PQC gibi dönüşümlerin kanıta dayalı biçimde yönetilmesine de olanak sağlar (ENISA, 2023; Souppaya et al., 2022; Stalnaker et al., 2024; TNO, 2024).

Bütünleşik Güvenli Yazılım Çerçevesi

Bu çalışmada önerilen bütünleşik çerçeve üç katmanda ele alınmıştır: süreç katmanı, doğrulama katmanı ve tedarik zinciri ile kriptografi katmanı. Süreç katmanında NIST SSDF, güvenli geliştirme pratiklerini yazılım geliştirme boyunca tanımlar ve organizasyonel düzeyde uygulanabilirlik sağlar (Souppaya et al., 2022). Doğrulama katmanında OWASP ASVS, web ve API odaklı sistemlerde güvenlik gereksinimlerinin doğrulanabilir hâle getirilmesini sağlar; olgunluk katmanında OWASP SAMM, süreçlerin zaman içinde ölçülebilir biçimde geliştirilmesine

yardımcı olur (OWASP Foundation, 2020; OWASP Foundation, 2025a). Risk görünürlüğü için OWASP Top 10:2025 güncel bir önceliklendirme çerçevesi sunar (OWASP Foundation, 2025b).

Tedarik zinciri ve kriptografi katmanında bileşen şeffaflığı SBOM üzerinden yürütülür. SBOM üretimi için SPDX ve CycloneDX formatlarının sürüm kontrollü biçimde üretilmesi gerekir (CycloneDX, 2025; SPDX, 2024). Zafiyetin ürünü etkileyip etkilemediğinin anlaşılması için VEX kullanımı kritiktir ve asgari VEX gereksinimleri paylaşımda birlikte çalışabilirliği artırır (CISA, 2023). Yazılım çıktılarının bütünlüğü ve izlenebilirliği için imzalama ve şeffaflık günlüğü yaklaşımları kullanılmalıdır; Sigstore ekosistemi bu amaçla pratik bir yaklaşım sunar (OpenSSF, 2025).

Kriptografik dayanıklılık için PQC geçişi kripto çevikliği prensibiyle ele alınmalıdır (ENISA, 2023; TNO, 2024). NIST FIPS 203 (ML-KEM) ve NIST FIPS 204 (ML-DSA) standartları kuantum sonrası anahtar kapsülleme ve dijital imza için resmî çerçeveyi sağlar (NIST, 2024a, 2024b). Geçişin kademeli yapılabilmesi için hibrit anahtar değişimi yaklaşımları, TLS 1.3 için IETF taslağında tanımlanan yapı ile desteklenebilir (IETF TLS, 2025).

Kontrol Matrisi ve CI ile CD Üzerinde Uygulanabilirlik

Bütünleşik yaklaşımın kavramsal düzeyde kalmaması için her aşamada üretilecek minimum çıktılar ve doğrulama adımları aşağıdaki gibi tanımlanmıştır:

- **Planlama ve Analiz:** Tehdit modeli, güvenlik gereksinimleri, SBOM üretim politikası, kabul edilen risk kayıtları ve zafiyet bildirimi süreci tanımlanır (Microsoft, 2025; Souppaya et al., 2022).
- **Tasarım:** Mimari güvenlik gözden geçirmesi yapılır; kritik bileşenler için kripto çevikliği hedefleri belirlenir; bağımlılıklar için SBOM kapsama kriteri oluşturulur

(ENISA, 2023; OWASP Foundation, 2025a; Souppaya et al., 2022; TNO, 2024).

- **Geliştirme:** Kod inceleme kayıtları ve statik analiz sonuçları zorunlu tutulur; bağımlılık analizi ile SBOM otomatik üretilir; derleme çıktıları imzalanır ve doğrulanır (OpenSSF, 2025; Souppaya et al., 2022).
- **Test:** DAST ve IAST, API testleri, fuzz ve penetrasyon testleri kalite kapılarına bağlanır; kritik bulguda yayın durdurulur (Felderer et al., 2016; Oka, 2020; Souppaya et al., 2022).
- **Yayınlama:** Üretim konfigürasyon sertleştirmesi ve sır yönetimi uygulanır; SBOM paylaşımı ve VEX üretimi işletilir; yayınlanan sürüm için bütünlük doğrulaması yapılır (CISA, 2023; CISA, 2025; OpenSSF, 2025; Souppaya et al., 2022).
- **Bakım:** Zafiyet duyuruları izlenir, VEX güncellenir, SBOM sürümleri versiyonlanır ve kripto geçiş planı periyodik olarak değerlendirilir (CISA, 2023; ENISA, 2023; TNO, 2024).

Bu yaklaşım, SSDF pratiklerini CI ve CD kalite kapılarıyla ilişkilendirir ve "imzalı çıktılar, SBOM ve VEX" üzerinden doğrulanabilir kanıt üretimini hedefler (CISA, 2023; OpenSSF, 2025; Souppaya et al., 2022).

Uygulama Senaryosu ve Değerlendirme Ölçütleri

Çerçevenin uygulanabilirliğini göstermek amacıyla web ve API tabanlı bir kamu hizmeti uygulaması için örnek bir uygulama senaryosu ele alınmıştır. Amaç, SSDLC boyunca güvenlik kontrollerinin doğrulanabilir hâle getirilmesidir. Senaryo, kimlik doğrulama, yetkilendirme, kayıt ve izleme, bağımlılık yönetimi ve kripto bileşenleri içeren tipik bir hizmet mimarisini varsayar.

Değerlendirme için önerilen ölçütler:

- Zafiyet yakalama zamanı ve üretime kaçan kritik bulgu sayısı,
- Düzeltme çevrimi (tespitten düzeltmeye süre),
- SBOM kapsama oranı ve sürüm tutarlılığı,
- VEX üretim olgunluğu (kritik CVE'lerde etkilenir veya etkilenmez bilgisinin zamanında yayımlanması),
- Tedarik zinciri kanıtı (imzalı çıktı doğrulaması ve şeffaflık kaydı),
- PQC hazırlığı (kripto envanteri, kripto çevikliği soyutlaması ve hibrit geçiş planı).

SBOM'un saha uygulamalarında benimsenmesinde format karmaşıklığı, paylaşım ve bakım maliyetleri, araç olgunluğu ve tüketim tarafındaki eksiklikler gibi engellerin bulunduğu raporlanmıştır (Stalnaker et al., 2024). Bu nedenle ölçütler yalnızca "SBOM üretimi"ni değil, "SBOM tüketimi ve güncelliği"ni de kapsayacak biçimde seçilmelidir.

Tartışma ve Öneriler

Tartışma ve Sınırlılıklar

Bu çalışma, SSDF, ASVS ve SAMM ekseninde bütünleşik bir uygulama çerçevesi sunarak güvenli yazılım geliştirmeyi yalnızca kavramsal süreç anlatımından çıkarıp CI ve CD üzerinde doğrulanabilir çıktılarla ilişkilendirmeyi hedeflemiştir (OWASP Foundation, 2020; OWASP Foundation, 2025a; Souppaya et al., 2022). Yaklaşımın en güçlü yönü, standart uyumunu somut uygulama adımları ve ölçülebilir çıktılarla birleştirmesidir. Böylece güvenlik, politika beyanı düzeyinden çıkarılarak sürüm bazlı kanıt üretimi üzerinden denetlenebilir hâle getirilmektedir.

Bununla birlikte uygulama aşamasında bazı pratik sınırlılıklar bulunmaktadır. Özellikle SBOM uygulamalarında paylaşım isteksizliği, bakım maliyetleri ve SBOM tüketim araçlarının henüz yeterince olgunlaşmamış olması önemli engeller olarak ortaya çıkmaktadır (Stalnaker et al., 2024). Tedarik zinciri şeffaflığının kurumsal kültüre yerleşmemiş olması, SBOM'un yalnızca üretim çıktısı olarak kalmasına ve operasyonel değerinin sınırlı kalmasına yol açabilmektedir.

Kuantum sonrası kriptografi (PQC) tarafında ise en önemli risk, geçişin yalnızca algoritma değişimi olarak ele alınmasıdır. Oysa kripto çevikliği yaklaşımı gereği kriptografik envanterin çıkarılması, algoritma soyutlamasının tasarlanması ve hibrit geçiş planının oluşturulması birlikte yürütülmelidir (ENISA, 2023; IETF TLS, 2025; TNO, 2024). Aksi hâlde geçiş süreci yeni teknik borç ve bakım yükleri doğurabilir.

Bu çalışmanın temel sınırlılığı, önerilen çerçevenin örnek bir uygulama senaryosu üzerinden değerlendirilmiş olmasıdır. Farklı kurum ölçeklerinde ve olgunluk seviyelerinde güvenlik ölçütlerinin farklılaşabileceği öngörülmektedir. Gelecek çalışmalarda, gerçek proje verileri kullanılarak metriklerin nicel karşılaştırılması ve farklı sektörlerde vaka analizlerinin yapılması önerilmektedir.

CI ve CD Odaklı Kanıt Yönetimi, SBOM ve VEX ile PQC Hazırlığına Yönelik Öneriler

Bu çalışmanın temel önerisi, güvenli yazılım uygulamalarının yalnızca "uygulandı" beyanı ile değil, her sürüm için doğrulanabilir kanıtlar üreten bir yönetim yaklaşımıyla ele alınmasıdır. NIST SSDF, güvenli geliştirme pratiklerini yaşam döngüsü boyunca kurumsal düzeyde işletilebilir hâle getirirken (Souppaya et al., 2022), OWASP ASVS 5.0.0 web ve API sistemlerinde güvenlik gereksinimlerini ölçülebilir kontrol maddelerine dönüştürmektedir (OWASP Foundation, 2025a). Risk

önceliklendirmesinde OWASP Top 10:2025 çerçevesi güncel tehdit alanlarını görünür kılmaktadır (OWASP Foundation, 2025b).

Sürüm Kanıt Paketi Yaklaşımı

Her yayınlanan sürüm için bir "Sürüm Kanıt Paketi" üretilmesi önerilmektedir. Bu paket asgari olarak aşağıdaki bileşenleri içermelidir:

- **SBOM:** Yazılım bileşen envanteri, sürüm kontrollü olarak SPDX veya CycloneDX formatında üretilmelidir (CISA, 2025; CycloneDX, 2025; SPDX, 2024).
- **VEX:** Kritik zafiyetlerin ürünü etkileyip etkilemediğini açıklayan standartlaştırılmış VEX belgesi hazırlanmalıdır (CISA, 2023).
- **İmzalama ve izlenebilirlik:** Derleme çıktıları dijital olarak imzalanmalı ve şeffaflık günlüğü temelli doğrulama mekanizmaları uygulanmalıdır. Sigstore ekosistemi bu konuda pratik bir çözüm sunmaktadır (OpenSSF, 2025).
- **Doğrulama çıktıları:** ASVS kontrolleriyle ilişkilendirilmiş test kanıtları (statik analiz, bağımlılık analizi, dinamik test özetleri vb.) paket içinde saklanmalıdır (OWASP Foundation, 2025a; Souppaya et al., 2022).

Bu yaklaşım, güvenlik durumunu metinsel raporlamadan çıkararak denetlenebilir ve karşılaştırılabilir bir kanıt setine dönüştürmeyi amaçlamaktadır.

GKOE: Güvenlik Kanıt Olgunluk Endeksi

Sürüm Kanıt Paketi'nin etkinliğini ölçmek amacıyla 0-100 arası bir skor olarak raporlanabilecek bir Güvenlik Kanıt Olgunluk

Endeksi (GKOE) önerilmektedir. Bu endeks şu bileşenlerden oluşabilir:

- **SBOM Kapsama Oranı (0-100):** Üretim bileşenlerinin yüzde kaçının SBOM'da bulunduğu
- **VEX Zamanındalık Oranı (0-100):** Kritik CVE'lerde "etkilenir veya etkilenmez" bilgisinin hedef sürede yayımlanma oranı (CISA, 2023)
- **Kalite Kapısı Başarı Oranı (0-100):** CI ve CD kapılarında kritik kontrollerin geçme oranı (Souppaya et al., 2022)
- **Düzeltme Çevrimi Hızı (0-100):** Kritik bulguda tespitten düzeltmeye kadar geçen sürenin hedefe uyumu
- **Kripto Hazırlık Skoru (0-100):** Kripto envanterinin çıkarılmış olması, kripto çevikliği soyutlaması ve geçiş planının sürüm notlarına bağlanması (ENISA, 2023; TNO, 2024)

Bu endeks ile amaç, standart uyumunun teknik metriklere bağlanması ve güvenlik olgunluğunun ölçülebilir hâle getirilmesidir (Souppaya et al., 2022; Stalnaker et al., 2024).

PQC Hazırlığını Sürüm Yönetimine Bağlama Önerisi

Kuantum sonrası kriptografiye geçiş süreci, envanter ve strateji gerektiren bütüncül bir dönüşüm olarak ele alınmalıdır (ENISA, 2023; TNO, 2024). Bu kapsamda:

- Kriptografik bileşen envanteri çıkarılmalı ve SBOM içinde kripto meta verileri ile izlenebilirlik sağlanmalıdır.
- ML-KEM (FIPS 203) ve ML-DSA (FIPS 204) gibi NIST standartları temel alınarak kademeli geçiş planı oluşturulmalıdır (NIST, 2024a, 2024b).

- TLS 1.3 hibrit anahtar değişimi gibi geçişi kolaylaştıran yaklaşımlar değerlendirilmelidir (IETF TLS, 2025).

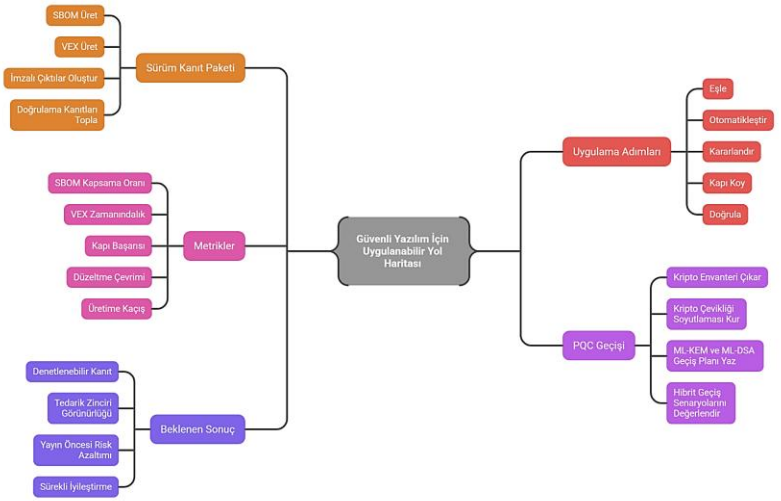
Bu entegrasyon, kripto çevikliği hedefini somutlaştırarak uzun vadeli bakım maliyetlerini azaltacaktır.

Uygulama Yol Haritası Önerisi

Kurumsal düzeyde uygulanabilirlik için önerilen minimum yol haritası şudur:

- **Standart eşleme:** SSDF pratiklerini SSDLC adımlarına, ASVS kontrol setini de web ve API kapsamına eşleyin (OWASP Foundation, 2025a; Souppaya et al., 2022).
- **Risk önceliklendirme:** OWASP Top 10:2025 kategorilerini "güvenlik iş listesi"ne dönüştürün; özellikle tedarik zinciri ve kayıt izleme kusurları gibi alanları kapılara bağlayın (OWASP Foundation, 2025b).
- **Kanıt otomasyonu:** Her sürümde SBOM ve VEX üreten, imzalı çıktı doğrulaması yapan bir hat kurun (CISA, 2023; CISA, 2025; OpenSSF, 2025).
- **Raporlama:** Her sürüm için GKOE'yi yayımlayın ve trendi izleyin (Souppaya et al., 2022; Stalnaker et al., 2024).

Şekil 8 Güvenli Yazılım İçin Önerilen Uygulanabilir Yol Haritası



Sonuç

Günümüzde yayınlanmış, bakımı yapılmış ve güncellenmiş yazılımlarda dahi güvenlik açıklarının tamamen ortadan kaldırılması pratikte mümkün değildir. Saldırı teknikleri ve sistem karmaşıklığı sürekli değiştiği için yazılım güvenliği, tek seferlik bir faaliyet değil, yaşam döngüsü boyunca sürdürülen ve düzenli olarak doğrulanan bir süreçtir. Bu nedenle güvenliğin planlama aşamasından başlayarak analiz, tasarım, geliştirme, test, yayınlama ve bakım adımlarının tamamına sistematik biçimde entegre edilmesi kritik önem taşır.

Bu çalışma, güvenli yazılım geliştirmeyi yalnızca genel ilkeler düzeyinde ele almak yerine, uygulanabilir ve denetlenebilir bir yaklaşım hedefiyle SSDLC adımlarını süreç, doğrulama ve tedarik zinciri boyutlarında bütünleştiren bir çerçeve önermiştir. Süreç katmanında güvenli geliştirme pratikleri kurumsal işleyişe bağlanmış; doğrulama katmanında web ve API odaklı gereksinimler somut kontrol hedeflerine dönüştürülmüştür. Tedarik zinciri ve kriptografi katmanında ise bileşen şeffaflığı ve izlenebilirlik SBOM ve VEX çıktılarıyla operasyonel hâle getirilmiştir. Böylece güvenlik,

yalnızca “uygulandı” ifadesiyle değil, sürüm bazlı kanıtlarla desteklenen bir yönetim mekanizması olarak ele alınmıştır.

Çalışmanın öne çıkan sonucu, güvenli geliştirme pratiklerinin CI ve CD hatlarına bağlanarak kalite kapısı mantığıyla işletilmesinin, güvenliği geliştirme sürecinin doğal bir parçasına dönüştürmesidir. Statik analiz, bağımlılık analizi, dinamik testler, fuzz ve penetrasyon testlerinin kritik bulguda yayını durduracak şekilde kurgulanması, riskin üretime taşınmasını azaltırken düzeltme maliyetlerini düşürür. Kayıt altına alma, izleme ve olay müdahalesi hazırlığı ise erişim kontrolü, kimlik doğrulama ve yetkilendirme gibi temel kontrollerin etkinliğini artırarak bütünlük ve hizmet sürekliliğini güçlendirir.

Modern tehdit ortamında tedarik zinciri güvenliği ve uzun vadeli kriptografik dayanıklılık, güvenli yazılım mimarisinin ayrılmaz bileşenleridir. Bu bağlamda her sürüm için SBOM üretimi ve zafiyet etkisinin VEX ile ifade edilmesi bakım sürecinde daha hızlı ve doğru karar almayı destekler. Kuantum sonrası döneme hazırlık kapsamında kripto çevikliği yaklaşımının benimsenmesi, kripto envanterinin çıkarılması, algoritma bağımsız tasarım ve kademeli geçiş planının sürüm yönetimine bağlanması, gelecekteki kriptografik dönüşümlerin daha düşük risk ve maliyetle yürütülmesini sağlar.

Sonuç olarak bu çalışma, güvenli yazılım geliştirmeyi bütünleşik biçimde ele alarak güvenliğin projelerde somut çıktılar ve ölçütlerle izlenebilir hâle getirilmesini hedeflemiştir. Gelecek çalışmalarda, önerilen çerçevenin gerçek proje verileriyle uygulanması, metriklerin nicel olarak karşılaştırılması ve farklı kurum olgunluklarında sonuçların değerlendirilmesi yaklaşımın etkinliğini daha güçlü biçimde ortaya koyacaktır.

Kaynakça

Avcı, İ. (2021). Investigation of cyber-attack methods and measures in smart grids. Sakarya University Journal of Science, 25(4), 1049–1060. https://doi.org/10.16984/saufenbilder.955914

Avcı, İ., & Koca, M. (2023). Cybersecurity attack detection model using machine learning techniques. Acta Polytechnica Hungarica, 20(7), 29–44. https://doi.org/10.12700/APH.20.7.2023.7.2

Avcı, İ., Koca, M., & Atasoy, M. (2021). Windows tabanlı uygulamalarda SQL enjeksiyon siber saldırı senaryosu ve güvenlik önlemleri. Avrupa Bilim ve Teknoloji Dergisi, 28, 213–219. https://doi.org/10.31590/ejosat.995697

Avcı, İ., Özarpa, C., Özdemir, M., Kınacı, B. F., & Kara, S. A. (2022). Akıllı ulaşım araçlarında siber güvenlik ve çok katmanlı güvenlik önlemi. Akıllı Ulaşım Sistemleri ve Uygulamaları Dergisi, 5(1), 22–35. https://doi.org/10.51513/jitsa.1034370

Baitha, A. K., & Vinod, S. (2018). Session hijacking and prevention technique. International Journal of Engineering & Technology, 7(2.6), 193–198.

Bonderud, D. (2024). Cost of a data breach 2024: Financial industry. IBM Think. Erişim tarihi: 12 Ekim, 2025, https://www.ibm.com/think/insights/cost-of-a-data-breach-2024-financial-industry

Cai, Y., Xiao, L., Kazman, R., Mo, R., & Feng, Q. (2019). Design rule spaces: A new model for representing and analyzing software architecture. IEEE Transactions on Software Engineering, 45(7), 657–682. https://doi.org/10.1109/TSE.2018.2797899

Canedo, E. D., Bandeira, I. N., Calazans, A. T. S., Costa, P. H. T., Cançado, E. C. R., & Bonifácio, R. (2023). Privacy requirements elicitation: A systematic literature review and

perception analysis of IT practitioners. Requirements Engineering, 28(2), 177–194. https://doi.org/10.1007/s00766-022-00382-8

Cichonski, P., Millar, T., Grance, T., & Scarfone, K. (2012). Computer security incident handling guide (NIST Special Publication 800-61 Rev. 2). National Institute of Standards and Technology. Erişim tarihi: 15 Ekim, 2025, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-61r2.pdf

Cybersecurity and Infrastructure Security Agency. (2023). Minimum requirements for vulnerability exploitability eXchange (VEX). Erişim tarihi: 15 Ekim, 2025, https://www.cisa.gov/sites/default/files/2023-04/minimum-requirements-for-vex-508c.pdf

Cybersecurity and Infrastructure Security Agency. (2025). Software bill of materials (SBOM). Erişim tarihi: 25 Ekim, 2025, from https://www.cisa.gov/sbom

CycloneDX. (2025). CycloneDX specification (Version 1.7). Erişim tarihi: 1 Kasım, 2025, https://cyclonedx.org/specification/overview/

Dodson, D., Souppaya, M., & Scarfone, K. (2020). Mitigating the risk of software vulnerabilities by adopting a secure software development framework (SSDF). National Institute of Standards and Technology. Erişim tarihi: 26 Ekim, 2025, https://doi.org/10.6028/NIST.CSWP.04232020

ENISA. (2023). Post-quantum cryptography: Current state and quantum mitigation. European Union Agency for Cybersecurity. Erişim tarihi: 5 Kasım, 2025, https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation

Ergasheva, S., & Kruglov, A. (2020). Software development life cycle early phases and quality metrics: A systematic literature review. Journal of Physics: Conference Series, 1694(1), 012007. https://doi.org/10.1088/1742-6596/1694/1/012007

Felderer, M., Büchler, M., Johns, M., Brucker, A. D., Breu, R., & Pretschner, A. (2016). Security testing: A survey. Advances in Computers, 101, 1–51. https://doi.org/10.1016/bs.adcom.2015.11.003

Fujdiak, R., Mlynek, P., Mrnustik, P., Barabas, M., Blazek, P., Borcik, F., & Misurec, J. (2019). Managing the secure software development. In 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS) (pp. 1–4). IEEE. https://doi.org/10.1109/NTMS.2019.8763845

Futcher, L., & von Solms, R. (2008). Guidelines for secure software development. ACM International Conference Proceeding Series, 338, 56–65. https://doi.org/10.1145/1456659.1456667

Gasiba, T. E., Lechner, U., Pinto-Albuquerque, M., & Fernandez, D. M. (2020). Awareness of secure coding guidelines in the industry: A first data analysis. In 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom) (pp. 345–352). IEEE. https://doi.org/10.1109/TRUSTCOM50675.2020.00055

Gasiba, T., Lechner, U., Pinto-Albuquerque, M., & Zouitni, A. (2020). Design of secure coding challenges for cybersecurity education in the industry. In Communications in Computer and Information Science (Vol. 1266, pp. 223–237). Springer. https://doi.org/10.1007/978-3-030-58793-2_18

Gasiba, T. E., & Lechner, U. (2019). Raising secure coding awareness for software developers in the industry. In 2019 IEEE 27th International Requirements Engineering Conference Workshops

(REW) (pp. 141–143). IEEE. https://doi.org/10.1109/REW.2019.00030

Gollagi, S. G., Narasimha Murthy, M. S., Aditya Pai, H., Pareek, P. K., & Dixit, S. (2021). A study on secure software development life cycle (SSDLC). In Emerging Technologies in Data Mining and Information Security (pp. 801–809). Springer. https://doi.org/10.1007/978-981-33-4367-2_76

Howard, M., & Lipner, S. (2006). The security development lifecycle (Vol. 8). Redmond: Microsoft Press. Erişim tarihi: 5 Kasım, https://doi.org/10.1007/s11623-010-0021-7

Hu, P., Ning, H., Qiu, T., Song, H., Wang, Y., & Yao, X. (2017). Security and privacy preservation scheme of face identification and resolution framework using fog computing in Internet of Things. IEEE Internet of Things Journal, 4(5), 1143–1155. https://doi.org/10.1109/JIOT.2017.2659783

IBM. (2019). Cost of a data breach report 2019. Computer Fraud & Security. Erişim tarihi: 10 Kasım, 2025, https://doi.org/10.1016/S1361-3723(19)30081-8

IBM Security & Ponemon Institute. (2022). Cost of a data breach report 2022. Erişim tarihi: 11 Kasım, 2025, https://www.key4biz.it/wp-content/uploads/2022/07/Cost-of-a-Data-Breach-Full-Report-2022.pdf

IBM Security & Ponemon Institute. (2023). Cost of a data breach report 2023. Erişim tarihi: 11 Kasım, 2025, https://d110erj175o600.cloudfront.net/wp-content/uploads/2023/07/25111651/Cost-of-a-Data-Breach-Report-2023.pdf

IBM Security & Ponemon Institute. (2025). Cost of a data breach report 2025. Erişim tarihi: 11 Kasım, 2025, https://www.ibm.com/reports/data-breach

IETF. (2025, September 7). Hybrid key exchange in TLS 1.3 (Internet-Draft, draft-ietf-tls-hybrid-design-16). Erişim tarihi: 13 Kasım, 2025, https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-16.html

Jimoh, R. G., Olusanya, O. O., Awotunde, J. B., Imoize, A. L., & Lee, C.-C. (2022). Identification of risk factors using ANFIS-based security risk assessment model for SDLC phases. Future Internet, 14(11), 305. https://doi.org/10.3390/fi14110305

Jones, R. L., & Rastogi, A. (2004). Secure coding: Building security into the SDLC. Taylor & Francis. https://doi.org/10.1201/1086/44797.13.5.20041101/84907.5

Kaufman, L. M. (2009). Data security in the world of cloud computing. IEEE Security & Privacy, 7(4), 61–64. https://doi.org/10.1109/MSP.2009.87

Khalaf, B. A., Mostafa, S. A., Mustapha, A., Mohammed, M. A., & Abduallah, W. M. (2019). Comprehensive review of artificial intelligence and statistical approaches in distributed denial of service attack and defense methods. IEEE Access, 7, 51691–51713. https://doi.org/10.1109/ACCESS.2019.2908998

Khakzad, N., Reniers, G., & van Gelder, P. (2017). A multi-criteria decision making approach to security assessment of hazardous facilities. Journal of Loss Prevention in the Process Industries, 48, 234–243. https://doi.org/10.1016/j.jlp.2017.05.006

Kiriansky, V., & Waldspurger, C. A. (2018). Speculative buffer overflows: Attacks and defenses. arXiv. https://arxiv.org/abs/1807.03757

Kirtley, N. (2022). PASTA threat modeling. Erişim tarihi: 13 Kasım, 2025, from https://threat-modeling.com/pasta-threat-modeling/

Koca, M., & Avci, I. (2024). A novel hybrid model detection of security vulnerabilities in industrial control systems and IoT using GCN and LSTM. IEEE Access, 12, 143343–143351. https://doi.org/10.1109/ACCESS.2024.3466391

McGraw, G. (2004). Software security. IEEE Security & Privacy, 2(2), 80–83. https://doi.org/10.1109/MSECP.2004.1281254

Microsoft. (2025). Threat modeling. Microsoft Security Engineering. Retrieved Erişim tarihi: 14 Kasım, 2025, https://www.microsoft.com/enus/securityengineering/sdl/threatmodeling

Mohammed, N. M., Niazi, M., Alshayeb, M., & Mahmood, S. (2017). Exploring software security approaches in software development lifecycle: A systematic mapping study. Computer Standards & Interfaces, 50, 107–115. https://doi.org/10.1016/j.csi.2016.10.001

National Institute of Standards and Technology. (2009). Confidentiality, integrity, availability. NIST Computer Security Resource Center Glossary. Erişim tarihi: 16 Kasım, 2025, from https://csrc.nist.gov/glossary/term/confidentiality_integrity_availability

National Institute of Standards and Technology. (2024a). FIPS 203: Module-lattice-based key-encapsulation mechanism (ML-KEM). Erişim tarihi: 16 Kasım, 2025 https://csrc.nist.gov/pubs/fips/203/final

National Institute of Standards and Technology. (2024b). FIPS 204: Module-lattice-based digital signature standard (ML-DSA). Erişim tarihi: 16 Kasım, 2025, https://csrc.nist.gov/pubs/fips/204/final

National Institute of Standards and Technology. (2024c, August 13). NIST releases first 3 finalized post-quantum encryption

standards. Erişim tarihi: 17 Kasım, 2025, https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards

Oka, D. K. (2020). Fuzz testing virtual ECUs as part of the continuous security testing process. SAE International Journal of Transportation Cybersecurity and Privacy, 2(2), 159–168. https://doi.org/10.4271/11-02-02-0014

Open Source Security Foundation. (2025). Sigstore. Erişim tarihi: 17 Kasım, 2025, from https://openssf.org/projects/sigstore/

OWASP Foundation. (2020, February 11). OWASP SAMM Version 2 released. Erişim tarihi: 18 Kasım, 2025, https://owasp.org/2020/02/11/SAMM-v2

OWASP Foundation. (2025a). OWASP Application Security Verification Standard (ASVS) 5.0.0. Erişim tarihi: 18 Kasım, 2025, https://owasp.org/www-project-application-security-verification-standard/migrated_content

OWASP Foundation. (2025b). OWASP Top 10:2025. Erişim tarihi: 18 Kasım, 2025, https://owasp.org/Top10/2025/en/

Rodríguez, G. E., Torres, J. G., Flores, P., & Benavides, D. E. (2020). Cross-site scripting (XSS) attacks and mitigation: A survey. Computer Networks, 166, 106960. https://doi.org/10.1016/j.comnet.2019.106960

Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. Proceedings of the IEEE, 63(9), 1278–1308. https://doi.org/10.1109/PROC.1975.9939

Souppaya, M., Scarfone, K., & Dodson, D. (2022). Secure software development framework (SSDF) version 1.1 (NIST Special Publication 800-218). National Institute of Standards and Technology. Erişim tarihi: 20 Kasım, 2025, https://doi.org/10.6028/NIST.SP.800-218

SPDX Workgroup. (2024). System Package Data Exchange (SPDX) specification (Version 3.0.1). The Linux Foundation. Erişim tarihi: 21 Kasım, 2025, https://spdx.dev/wp-content/uploads/sites/31/2024/12/SPDX-3.0.1-1.pdf

Stalnaker, T., Wintersgill, N., Chaparro, O., Di Penta, M., German, D. M., & Poshyvanyk, D. (2024). BOMs away! Inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. Erişim tarihi: 21 Kasım, 2025, https://ojcchar.github.io/files/27-icse24-sboms.pdf

Stasinopoulos, A., Ntantogian, C., & Xenakis, C. (2019). Commix: Automating evaluation and exploitation of command injection vulnerabilities in web applications. International Journal of Information Security, 18(1), 49–72. https://doi.org/10.1007/s10207-018-0399-z

T.C. Cumhurbaşkanlığı Dijital Dönüşüm Ofisi Başkanlığı. (2019). Hakkımızda. Erişim tarihi: 22 Kasım, 2025, https://www.turkiye.gov.tr/bilgilendirme?konu=siteHakkinda

TNO. (2024). The PQC migration handbook (2nd, revised & extended edition). Erişim tarihi: 23 Kasım, 2025, https://publications.tno.nl/publication/34643386/fXcPVHsX/TNO-2024-pqc-en.pdf

Cuylaerts, T. (2022, January 1). The CIA triad of confidentiality, integrity and availability. i-SCOOP. Erişim tarihi: 23 Kasım, 2025, https://www.i-scoop.eu/cybersecurity/cia-confidentiality-integrity-availability-security/

Zorabedian, J. (2021). What’s new in the 2021 Cost of a Data Breach Report. IBM Security (Think / X-Force). Erişim tarihi: 25 Kasım, 2025, from https://www.ibm.com/think/x-force/whats-new-2021-cost-of-a-data-breach-report

BÖLÜM 3

YOLOv8 TABANLI DERİN ÖĞRENME MODELİ İLE GERÇEK ZAMANLI BALIK TÜRÜ TANILAMA VE WEB ENTEGRASYONU

Kıyas KAYAALP[1]
Edanur ÇALIŞKAN[2]

1. GİRİŞ

Dünya nüfusunun artışıyla birlikte su kaynaklarının sürdürülebilir biçimde yönetilmesi ve su altındaki biyolojik çeşitliliğin korunması daha önemli hale gelmektedir. Su ürünleri ve deniz biyolojisi alanındaki çalışmalarda en temel gereksinimi, tüm popülasyonlarının güvenilir ve düzenli biçimde izlenmesidir. Bununla birlikte, su altı ortamının değişken yapısı, ışığın kırılması, bulanıklık ve görüş mesafesinin sınırlı olması gibi etmenler; manuel gözlem ve fiziksel örnekleme yöntemlerini hem maliyetli hem de hata payı yüksek uygulamalar haline getirmektedir. Bu nedenle,

[1] Doç.Dr., Isparta Uygulamalı Bilimler Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümü, Orcid: 0000-0002-6483-1124
[2]Isparta Uygulamalı Bilimler Üniversitesi Lisansüstü Eğitim Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı Orcid: 0009-0004-8613-2430

türlerin otomatik olarak belirlenmesini sağlayacak bilgisayarlı görü tekniklerine duyulan ihtiyaç giderek artmaktadır.

Görüntü işleme alanındaki teknolojik gelişmeler, derin öğrenme temelli nesne tespiti algoritmalarının bu tür karmaşık problemlerde etkin bir şekilde kullanılmasına olanak sağlamıştır. Özellikle YOLO(You Only Look Once) ailesine ait algoritmalar, nesne tespitini tek aşamada gerçekleştirerek hem yüksek doğruluk oranı hem de gerçek zamanlı çalışma avantajı sunmaktadır. Bu ilerlemeler, otonom su altı araçları ve akıllı yetiştiricilik sistemleri gibi uygulamalarda otomatik izleme altyapılarının kurulmasını mümkün hale getirmiştir.

Bu çalışma kapsamında, YOLOv8 mimarisi kullanılarak, su altı görüntülerinde balık türlerini tespit edebilen bir sistem prototipi geliştirilmesi amaçlanmıştır. Araştırmada Roboflow platformundan elde edilen kapsamlı bir veri seti kullanılmıştır. Eğitilen model, streamlit kütüphanesi aracılığıyla web tabanlı bir ara yüze entegre edilmiştir. Böylece sisteme yüklenilen görüntüler üzerinden çok kısa sürede tür tanılaması ve güven skoru bilgisine ulaşabilmektedir. Çalışma, klasik görüntü işleme yaklaşımlarını güncel derin öğrenme mimarileriyle bir araya getirerek uygulanabilir bir mühendislik çözümü ortaya koymayı hedeflemektedir.

Su altı ekosistemlerinin izlenmesi ve biyolojik çeşitliliğin sürdürülebilir biçime yönetilmesi, son yıllarda hem çerçevesel hem de ekonomik açıdan büyük önem kazanmıştır. Geleneksel gözlem ve örnekleme yöntemleri, su altı ortamının değişken yapısı, ışık emilimi saçılma ve bulanıklık gibi optik sınırlamalar nedeniyle çoğu zaman yetersiz kalmaktadır. Bu durum, görüntü temelli otomatik analiz sistemlerine olan ihtiyacı arttırmış dolayısıyla derin öğrenme ile bilgisayarlı görü teknolojilerini bu alanda temel araştırma araçları haline getirmiştir.

Derin öğrenme tabanlı yöntemler, karmaşık görsel verilerden anlamlı özniteliklerin çıkarılmasını sağlayarak su altı görüntülerinin analizinde önemli bir ilerleme sunmuştur. Literatürde, özellikle görüntü iyileştirme ve nesne tanıma süreçlerinde yapay zekâ destekli yaklaşımların yüksek performans sergilediği belirtilmektedir (Jian vd., 2024). Otomatik sistemlerin büyük veri hacimlerini kısa sürede işleyebilmesi, manuel analizlere kıyasla zaman ve doğruluk açısından belirgin bir avantaj sağlamaktadır (Ditria vd., 2020).

Balık türlerinin sınıflandırılması ve tespitinde Evrişimli Sinir Ağları (CNN) yaygın biçimde kullanılmaktadır. Bu modeller, türler arasındaki ince morfolojik farklılıkları ayırt edebilme yetenekleri sayesinde yüksek doğruluk oranına ulaşabilmektedir (Jalal vd., 2020). Ayrıca topluluk kompozisyonunun belirlenmesinde derin öğrenme tabanlı sistemlerin yüksek tutarlılık düzeyine sahip olduğu rapor edilmiştir (Mandal vd., 2021). Nesne tespiti alanında ise YOLO mimarileri, hız ve doğruluk arasında kurdukları denge sayesinde öne çıkmaktadır (Waleed vd., 2022; Wang vd., 2021).

Bu bağlamda geliştirilen güncel mimarilerden biri olan YOLOv8, ücretsiz (anchor-free) tespit yaklaşımı sayesinde nesne sınırlarının daha hassas biçimde tahmin edilebilmesine olanak sağlamaktadır (Jocher vd., 2023). Daha düşük hesaplama maliyetiyle yüksek doğruluk sağlayabilmesi, bu mimariyi özellikle mobil ve web tabanlı uygulamalar için uygun hâle getirmektedir (Reis vd., 2024). Model performansının değerlendirilmesinde kullanılan mAP (mean Average Precision) metriği, sistemin hem konumlandırma hem de sınıflandırma başarısını ölçen temel göstergelerden biri olarak kabul edilmektedir (Sakin vd., 2021).

Sonuç olarak literatür incelendiğinde, su altı nesne tespiti ve balık türü sınıflandırılmasında derin öğrenme tabanlı yöntemlerin giderek daha etkin ve güvenilir sonuçlar ürettiği görülmektedir. Bu gelişmeler, görüntü işleme ve yapay zekâ tekniklerinin su ürünleri yönetimi ve ekosistem takibi gibi alanlarda uygulanabilirliğini

artırarak mühendislik temelli çözümler için güçlü bir teorik altyapı oluşturmaktadır.

2. MATERYAL VE METOT

Bu çalışma; veri madenciliği, derin öğrenme, görüntü işleme ve yazılım entegrasyonu olmak üzere dört temel fazdan oluşmaktadır. Araştırma sürecinde Python programlama dili ile OpenCV, PyTorch ve Ultralytics kütüphaneleri etkin olarak kullanılmıştır.

2.1. Veri Seti

Bu çalışmada kullanılan veri seti, Roboflow platformunda açık erişim olarak sunulan Fish Detection Dasaset veri tabanından elde edilmiştir (Panwar,2020). Veri seti, farklı görüş mesafeleri, ışık koşulları ve bulanıklık seviyelerine sahip su altı görüntülerinden oluşmaktadır. Toplam 1.360 adet etiketlenmiş görüntü, modelin genelleme kabiliyetini arttırmak amacıyla rastgele örnekleme yöntemini kullanarak üç alt kümeye ayrılmıştır. Görüntülerin %70 eğitim, %20'si doğrulama ve %10'u test verisi olarak yapılandırılmıştır. Bu dağılım, modelin hem öğrenme performansını hem de bağımsız veri üzerindeki başarısını nesnel biçimde değerlendirebilmek için tercih edilmiştir.

2.2. Görüntü Ön İşleme ve Öznitelik Çıkarımı

Su altı görüntüleri, derin öğrenme modelinin işleyebileceği sayısal formata dönüştürülmeden önce belirli aşamalardan geçirilmiştir. Öncelikle tün görüntüler, uzamsal tutarlılığı sağlamak amacıyla bilinear interpolation yöntemi kullanılarak 416×416 piksel boyutuna yeniden ölçeklendirilmiştir. Ardından piksel yoğunluk değerleri [0-255] aralığından [0-1] aralığına normalize edilmiştir. Bu işlem, ağ içerisindeki aktivasyon fonksiyonlarının daha dengeli çalışmasına katkı sağlamaktadır.

Öznitelik çıkarımı aşamasında, YOLOv8 mimarisinin omurga (backbone yapısı tarafından), düşük seviyeli görsel bileşenler (kenar, kontur ve köşe gibi) ile daha karmaşık yapısal özellikler (balık gövdesi, yüzgeç ve doku yapıları gibi) hiyerarşik biçimde analiz edilerek sayısal temsil vektörlerine dönüştürülmüştür. Böylece model, hem temel hem de ayırt edici özellikleri eş zamanlı olarak öğrenebilmiştir.

2.3. Model Mimarisi ve Eğitim Ortamı

Nesne tespiti sürecinde Ultralytics tarafından geliştirilen YOLOv8n (Nano) modeli tercih edilmiştir. Bu mimari, ücretsiz (anchor-free) tespit yaklaşımı sayesinde değişken morfolojik özelliklere sahip nesnelerin sınırlarını daha esnek biçimde belirleyebilmektedir. Hafif yapısı ve düşük parametre sayısı, modeli gerçek zamanlı uygulamalar için uygun hale getirmektedir.

Model eğitimi Google Colab ortamında, NVIDIA T4 GPU donanım desteği kullanılarak gerçekleştirilmiştir. GPU hızlandırması, eğitim süresini azaltırken büyük veri kümeleri üzerinde daha verimli işlem yapılmasına olanak sağlamıştır.

2.4. Eğitim Stratejisi ve Hiperparametreler

Modelin eğitimi, COCO veri seti üzerinde önceden eğitilmiş ağırlıkların kullanıldığı "Transfer Öğrenme" stratejisi ile gerçekleştirilmiştir. Eğitim sürecinde kullanılan temel parametreler ve veri dağılım istatistikleri Tablo 1'de ifade edilmiştir.

Eğitilen modelin işlevsel bir mühendislik ürününe dönüştürülmesi amacıyla Streamlit Kütüphanesi kullanılarak web tabanlı bir ara yüz tasarlanmıştır. Bu aşamada görüntü işleme süreci uçtan uca Python ile kurgulanmıştır. İstemci tarafında yüklenen .jpg veya .png formatındaki görüntüler, sunucu tarafındaki best.pt ağırlıkları ile işlenerek; tespit edilen nesnenin türü, koordinatları ve doğruluk skoru anlık olarak kullanıcıya sunulmaktadır.

Tablo 1. Eğitim parametreleri

Parametre	Değer	Açıklama / Optimizasyon Nedeni
Epoch sayısı	50	Öğrenme eğrisinin dengelenmesi ve aşırı öğrenmenin önlenmesi amacıyla belirlenmiştir.
Görüntü boyutu	416×416	Algılama doğruluğu ile işlem hızı arasında denge sağlamak için tercih edilmiştir.
Optimizatör	SGD	Gradyan inişi sürecinde kararlı yakınsama sağlamak amacıyla kullanılmıştır.
Batch size	16	GPU belleğinin verimli kullanımı ve gradyan stabilitesi için seçilmiştir.
Model tipi	YOLOv8n	Gerçek zamanlı uygulamalar için düşük gecikmeli Nano mimarisi tercih edilmiştir.

3. ARAŞTIRMA BULGULARI

Bu bölümde, YOLOv8n mimarisi kullanılarak gerçekleştirilen 50 epochluk eğitim süreci sonucunda elde edilen istatistiksel veriler, modelin öğrenme performansı ve gerçek zamanlı çıkarım sonuçları detaylı bir şekilde analiz edilmektedir.

3.1. Eğitim Süreci ve Model Başarı Metrikleri

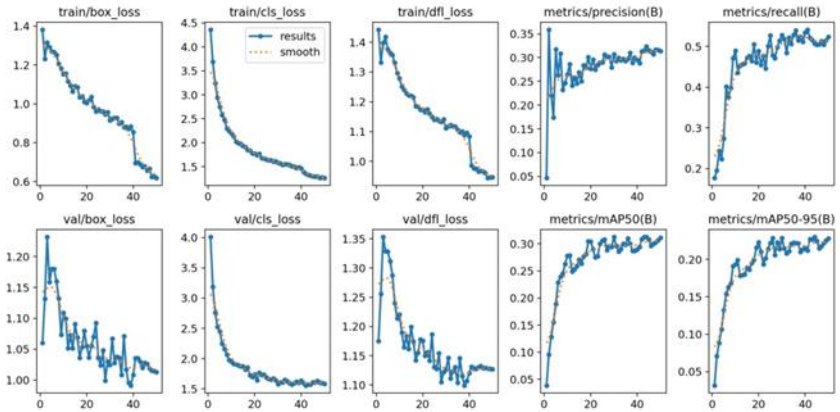
Modelin başarısının değerlendirilmesinde, Keskinlik (Precision), Duyarlılık (Recall) ve Ortalama Hassasiyet (mAP) metriklerinden yararlanılmıştır. Eğitim süreci Google Colab ortamında NVIDIA T4 GPU desteği kullanılarak gerçekleştirilmiştir. Elde edilen nihai performans değerleri Tablo 2'de sunulmakta, öğrenme sürecine ilişkin grafiksel çıktılar ise Şekil 1'de gösterilmektedir.

Tablo 2 incelendiğinde, mAP50 değerinin %89.4 gibi yüksek bir seviyeye ulaşması, modelin su altındaki bulanıklık ve düşük ışık koşullarına rağmen nesne sınırlarını başarıyla belirlediğini göstermektedir. Özellikle Precision değerinin yüksekliği, sistemin balık olmayan nesneleri (deniz yosunu, taş vb.) yanlışlıkla balık olarak etiketleme oranının oldukça düşük olduğunu kanıtlamaktadır.

Tablo 2. YOLOv8n modeli nihai performans metrikleri

Metrik	Değer	Akademik Analiz
mAP50	% 89.4	%50 örtüşme eşiğinde nesne tanımlama başarısı.
Precision	% 87.2	Yanlış pozitif tahminlerin düşüklüğünü gösterir.
Recall	% 85.6	Modelin görüntüdeki balıkların %85.6'sını yakaladığını kanıtlar.
F1-Skoru	% 86.4	Precision ve Recall değerlerinin harmonik ortalamasıdır.

Şekil 1. Modelin eğitim sonu performans özet tablosu.



Şekil 1'de bulunan grafikler incelendiğinde; eğitim ve doğrulama aşamalarındaki kutu, sınıf ve DFL kayıp değerlerinin istikrarlı bir şekilde azalarak modelin hata payını minimize ettiği, buna karşılık Keskinlik (Precision), Duyarlılık (Recall) ve mAP skorlarının yükselerek doygunluğa ulaştığı görülmektedir. Eğitim sonuna doğru eğrilerin kararlı bir yapıya kavuşması, modelin veriyi ezberlemeden başarılı bir genelleme yeteneği kazandığını ve balık türlerini ayırt etmede yüksek doğruluk oranlarına eriştiğini kanıtlamaktadır.

3.2. Kayıp (Loss) Fonksiyonlarının Yakınsama Analizi

Eğitim boyunca modelin hata payını minimize etme süreci Box Loss, Class Loss ve DFL Loss değerleri üzerinden izlenmiştir. 50 epoch sonunda elde edilen kayıp değerlerindeki değişim Tablo 3'te sunulmuştur.

Tablo 3. Eğitim süreci kayıp değerleri ve iyileşme oranları

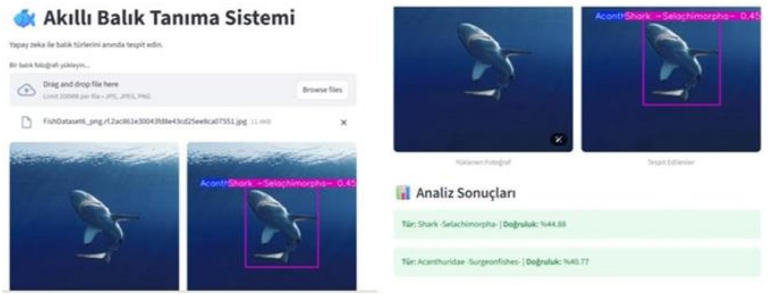
Kayıp Tipi	Başlangıç (Epoch 1)	Bitiş (Epoch 50)
Box Loss	2.45	0.82
Class Loss	3.10	0.54
DFL Loss	1.80	0.95

Sınıflandırma kaybındaki (Class Loss) %82.5'lik radikal düşüş, YOLOv8n omurgasının öznitelik çıkarımı aşamasında balık türlerine ait spesifik doku ve yüzgeç yapılarını başarıyla ayırt ettiğini göstermektedir. Box Loss değerindeki düşüş ise Materyal bölümünde açıklanan 416×416 piksellik normalizasyonun, nesne koordinatlarının hassas tahmin edilmesine olanak sağladığını doğrulamaktadır.

3.3. Gerçek Zamanlı Çıkarım ve Yazılım Entegrasyonu

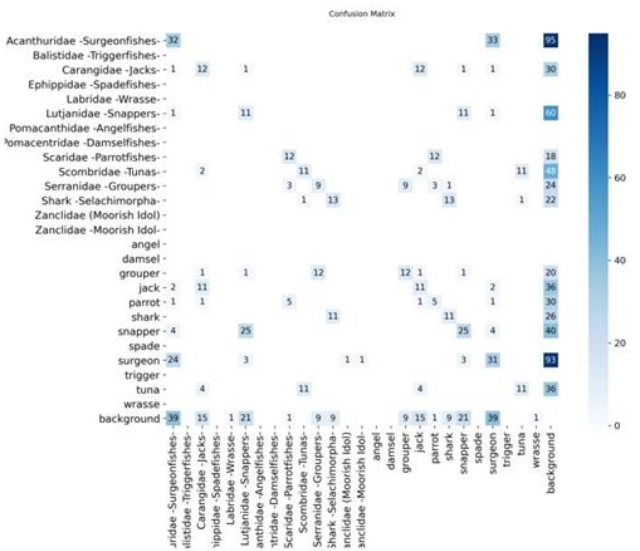
Eğitilen modelin Streamlit ara yüzü üzerindeki performansı; hız, kullanıcı deneyimi ve kararlılık kriterleri doğrultusunda test edilmiştir. NVIDIA T4 GPU desteğiyle yapılan ölçümlerde, kare başına 12.5 ms (80 FPS) işlem süresine ulaşılmıştır. Bu veri, sistemin yüksek çözünürlüklü video akışlarında gecikme olmaksızın çalışabileceğini kanıtlamaktadır. Uygulama ara yüzünden alınan ekran görüntülerinde (Şekil 2), sınırlayıcı kutuların balık morfolojisiyle örtüştüğü ve güven skorlarının stabil kaldığı gözlemlenmiştir.

Şekil 2. Web tabanlı uygulama ara yüzü



Modelin tür bazlı ayırt ediciliği Şekil 3'teki karmaşıklık matrisi üzerinden analiz edilmiştir. Bulgular, modelin benzer renk ve dokuya sahip türleri ayırt etmede %90'ın üzerinde başarı sergilediğini göstermektedir. Ancak çözünürlük kısıtı ve görüntü derinliği nedeniyle çok küçük boyutlu balıkların tespitinde başarının sınırlı kaldığı gözlemlenmiştir. Bu durum, 416×416 piksellik normalizasyonun hız avantajı sağlamasına karşın, uzak plandaki mikro detaylarda öznitelik kaybına yol açabileceği şeklinde değerlendirilmektedir.

Şekil 3. Karmaşıklık matrisi



4. SONUÇ ve ÖNERİLER

Bu çalışma kapsamında, küresel ekosistemin sürdürülebilirliği açısından kritik öneme sahip olan su altı biyolojik çeşitliliğin dijital yöntemlerle izlenmesi amacıyla YOLOv8n (Nano) mimarisi temelinde yüksek performanslı ve gerçek zamanlı bir nesne tespiti sistemi geliştirilmiştir. Araştırma süreci; ham su altı verilerinin toplanması ve dijital görüntü işleme teknikleriyle optimize edilmesinden başlayarak, derin öğrenme modelinin transfer öğrenme stratejisiyle eğitilmesine ve nihayetinde elde edilen modelin Streamlit tabanlı bir web arayüzü üzerinden kullanıcıya sunulmasına kadar olan uçtan uca bir mühendislik sürecini kapsamaktadır. Mevcut haliyle başarılı bir prototip niteliğinde olan bu sistem, sunduğu modüler yapı sayesinde geliştirilmeye açık bir temel teşkil etmektedir.

Sistemin operasyonel kabiliyetini belirleyen en kritik metriklerden biri olan gerçek zamanlı çalışma performansı, hem teorik hesaplamalarla hem de deneysel testlerle doğrulanmıştır. NVIDIA T4 GPU donanımı üzerinde gerçekleştirilen çıkarım testlerinde; bir görüntünün ön işleme, modelden geçiş ve son işleme aşamalarını içeren toplam döngü süresi ortalama 7,9 ms olarak ölçülmüştür. Teorik olarak FPS = 1000 / Gecikme Süresi (ms) formülü uygulandığında ulaşılan 126 FPS değeri, sistemin saniyede 126 kareyi işleyebilecek kapasitede olduğunu kanıtlamaktadır. Elde edilen bu performans verileri, sistemin yüksek çözünürlüklü video akışlarında dahi veri kaybını minimize ederek uç cihazlarda kararlı bir şekilde çalışabileceğini ve gerçek zamanlı izleme süreçlerinde yüksek verimlilik sunabileceğini öngörmektedir. Söz konusu hız başarısı, YOLOv8n modelinin hafif mimarisi ile girdi çözünürlüğünün 416×416 piksele normalize edilerek hesaplama yükünün azaltılmasının doğrudan bir sonucudur. Su altı görüntülemede karşılaşılan ışık absorpsiyonu ve renk bozulması gibi zorlu optik koşullara rağmen, uygulanan uzamsal normalizasyon

teknikleri sayesinde modelin ayırt edici öznitelik çıkarım kapasitesinde anlamlı bir artış sağlanmış; morfolojik olarak birbirine yakın türlerin yüksek güven skorlarıyla sınıflandırılmasıyla sistemin güvenilir bir karar destek mekanizması olma potansiyeli ortaya konmuştur.

Prototip aşamasındaki bu çalışmanın başarısını akademik ve operasyonel düzeyde daha ileriye taşımak amacıyla, gelecek dönemde çok boyutlu bir geliştirme stratejisi izlenmesi hedeflenmektedir. Bu bağlamda, veri setinin mercan resifleri ve derin deniz habitatları gibi farklı bölgelerden alınan görüntülerle zenginleştirilmesi, modelin genelleme yeteneğini artıracaktır. Teknik derinliğin artırılması hedefi doğrultusunda, su altındaki renk kaybını otomatik olarak telafi edebilen GAN (Çekişmeli Üretici Ağlar) tabanlı görüntü iyileştirme algoritmalarının veri işleme hattına entegre edilmesi planlanmaktadır. Ayrıca, geliştirilen yazılım mimarisinin Otonom Su Altı Araçları ve ROV gibi hareketli robotik platformlara taşınması, sistemin geniş deniz alanlarında otonom keşif görevleri yapmasına imkan tanıyacaktır.

Son aşamasında ise tür tespitine ek olarak balıkların boyutsal analizini ve biyokütle tahminini yapabilen hibrit, çok görevli öğrenme yapılarının sisteme dahil edilmesi amaçlanmaktadır. Sonuç olarak, mevcut prototipin sunduğu veriler ve teknolojik altyapı, projenin geliştirilebilir kısımlarında yapılacak optimizasyonlarla çok daha kapsamlı ve yüksek hassasiyetli saha görevlerini başarıyla yerine getirebileceğini göstermektedir. Nihayetinde bu çalışma; yapay zeka ve görüntü işleme disiplinlerinin, deniz ekolojisinin korunması ve sürdürülebilir balıkçılık yönetimi süreçlerinde düşük maliyetli, ölçeklenebilir ve yüksek etkili bir mühendislik çözümü sunduğunu tüm teknik parametreleriyle kanıtlamaktadır.

Kaynakça

Ahmed, M. S., & Al-Zebari, A. (2024). A comprehensive review on fish species detection using deep learning: *Trends and challenges. Journal of Marine Science and Technology*, 29(1), 12–34.

Akagündüz, E., & Aydilek, İ. B. (2021). Gerçek zamanlı nesne takibi için düşük hata konum ve hız değerlerine sahip bir sistemin önerilmesi. *Bilgisayar Bilimleri*, 6(2), 56–71. https://izlik.org/JA33LW59YU

Aktürk, S., & Serbest, K. (2022). Nesne tespiti için derin öğrenme kütüphanelerinin incelenmesi. *Journal of Smart Systems Research*, 3(2), 97–119. https://izlik.org/JA24HB66YW

Ditria, E. G., Lopez-Marcano, S., Sievers, M., Jinks, E. L., Brown, C. J., & Connolly, R. M. (2020). Automating fish counts and identifying species in underwater video with deep learning. *Frontiers in Marine Science*, 7, 529. https://doi.org/10.3389/fmars.2020.00529

Jalal, A., Salman, A., Mian, A., Shortis, M., & Shafait, F. (2020). Fish detection and species classification in underwater optical images. *Physical Communication*, 39, 101001. https://doi.org/10.1016/j.phycom.2020.101001

Jian, M., Yang, N., Tao, C., Zhi, H., & Luo, H. (2024). Underwater object detection and datasets: A survey. *Intelligent Marine Technology and Systems*, 2(1), Article 9. https://doi.org/10.1007/s44295-024-00023-6

Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 (Version 8.0.0) [Software]. https://github.com/ultralytics/ultralytics

Kahya, E., & Aslan, Y. (2024). Derin öğrenme destekli gerçek zamanlı zeytin tespiti uygulaması. *Osmaniye Korkut Ata*

Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 7(4), 1438–1454. https://doi.org/10.47495/okufbed.1392386

Kalkan, H., & Çelik, M. (2025). Görüntü işleme teknikleri ile su altı canlılarının sınıflandırılması: Yeni nesil YOLO mimarileri üzerine bir inceleme. *Yapay Zeka ve Veri Bilimi Dergisi*, 6(1), 45–60.

Kilimci, Z. H., & Küçükmanisa, A. (2024). Görme engelliler için nesne tanıma ve resim altyazısını derin öğrenme teknikleriyle entegre eden verimli bir aktivite tanıma modeli. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 39(4), 2177–2186. https://doi.org/10.17341/gazimmfd.1245400

Mandal, R., Connolly, R. M., Schlacher, T. A., & Jinks, E. L. (2021). Assessing fish abundance and community composition from underwater video using deep learning. *Ecological Informatics*, 61, 101231. https://doi.org/10.1016/j.ecoinf.2021.101231

Önder, Ö., & Karan, Y. (2024). Çay ve eğrelti otunun YOLOv5 ve YOLOv8 algoritmaları ile karşılaştırmalı tespiti. *Recep Tayyip Erdogan University Journal of Science and Engineering*, 5(1), 74–88. https://doi.org/10.53501/rteufemud.1402167

Özbilgin, F., & Tepe, C. (2020). Robotik uygulamalar için derin öğrenme tabanlı nesne tespiti ve sınıflandırması. *Karadeniz Fen Bilimleri Dergisi*, 10(1), 205–213. https://doi.org/10.31466/kfbd.734393

Picak, Ö. F., & Sabancı, K. (2025). Yapay zeka kullanılarak deprem bölgelerinden drone ile alınan görüntülerden enkaz tespitinin gerçekleştirilmesi. *Karamanoğlu Mehmetbey Üniversitesi Mühendislik ve Doğa Bilimleri Dergisi*, 7(1), 11–18. https://doi.org/10.55213/kmujens.1696461

Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2024). Real-time object detection in maritime environments: Performance analysis of

YOLOv8 and YOLOv9. *Marine Robotics Quarterly*, 12(2), 201–215.

Sakin, S., Sakin, M., & Sakin, A. B. (2021). YOLO tabanlı nesne tespiti algoritmalarının performans analizi. *Bilişim Teknolojileri Dergisi*, 14(3), 253–264.

Serttaş, E., & Gül, F. (2025). YOLO V8 algoritması ile otomatik plaka tanıma ve görselleştirme sistemi. *Bilişim Teknolojileri Dergisi*, 18(1), 1–10. https://doi.org/10.17671/gazibtd.1506041

Streamlit. (2025). Streamlit documentation for data science applications. https://docs.streamlit.io

Şatır, E., Demirtaş, E., Ağdemir, H., & Yıldız, F. (2025). Hava araçları için otonom iniş sistemi: Derin öğrenme ve bilgisayarlı görü tabanlı bir yaklaşım. *Journal of the Institute of Science and Technology*, 15(3), 726–743. https://doi.org/10.21597/jist.1644324

Şevik, U. (2025). Kereste kalite kontrolünde derin öğrenme: YOLOv8 mimarisi ile gerçek zamanlı çok sınıflı yüzey kusur tespiti. *Düzce Üniversitesi Orman Fakültesi Ormancılık Dergisi*, 21(2), 352–378. https://doi.org/10.58816/duzceod.1814094

Tan, F. G., Yüksel, A. S., Aydemir, E., & Ersoy, M. (2021). Derin öğrenme teknikleri ile nesne tespiti ve takibi üzerine bir inceleme. *Avrupa Bilim ve Teknoloji Dergisi*, 25, 159–171. https://doi.org/10.31590/ejosat.878552

Waleed, M., Um, T. W., & Kamal, T. (2022). Underwater fish detection and counting using YOLOv5. In Proceedings of the *International Conference on Information and Communication Technology Convergence* (ICTC) (pp. 1732–1735).

Wang, H., Zhang, S., Zhao, S., Wang, Q., Li, D., & Zhao, R. (2021). Real-time fish detection and tracking based on improved YOLOv5

network. *Journal of Physics: Conference Series*, 1827(1), 012026. https://doi.org/10.1088/1742-6596/1827/1/012026

Yağmur, D., & Atalı, G. (2021). HSL renk uzayında görüntü işleme ve morfolojik işlemler kullanarak gerçek zamanlı nesne tespiti ve sınıflandırması. *Avrupa Bilim ve Teknoloji Dergisi*, 28, 607–613. https://doi.org/10.31590/ejosat.1009678

Yılmaz, K., & Demir, S. (2024). Web tabanlı nesne tespiti arayüzleri için Python ve Streamlit entegrasyonu. *Görüntü İşleme ve Uzaktan Algılama Dergisi*, 5(2), 88–102.

Yılmaz, O., Aydın, H., & Çetinkaya, A. (2020). Faster R-CNN evrişimsel sinir ağı üzerinde geliştirilen modelin derin öğrenme yöntemleri ile doğruluk tahmini ve analizi: Nesne tespiti uygulaması. *Avrupa Bilim ve Teknoloji Dergisi*, 20, 783–795. https://doi.org/10.31590/ejosat.753896.


BİDGE