

Barcode nasıl çalışır, *neden böyle kurduk?*

Barkodsuz ürünler için kasa yazılımına gömülen görsel tanıma servisi: mimari, teknoloji kararları ve alternatifleri, ölçümler, güvenlik ve bilinen sınırlar.

Bu doküman Barcode'nin nasıl çalıştığını, hangi teknolojiyi neden seçtiğimizi ve iddialarımızın arkasındaki ölçümleri anlatır. Her iddianın yanında kaynağı yazıyor:  **Ölçtük** (kendi verimiz ve kodumuz, tekrar üretilebilir) veya  **Gerekçe** (literatür, mühendislik değerlendirmesi — henüz ölçmedik). Ölçmediğimiz şeyleri ayrıca **§9 Bilinen sınırlar** bölümünde listeledik.

1. Özet

Fırın, manav, şarküteri ve kafeterya ürünlerinin barkodu yok. Bugün kasada bu ürünler dokunmatik ekranda onlarca ürünlük bir ızgaradan elle seçiliyor: yavaş, hataya açık, yeni personel için zor.

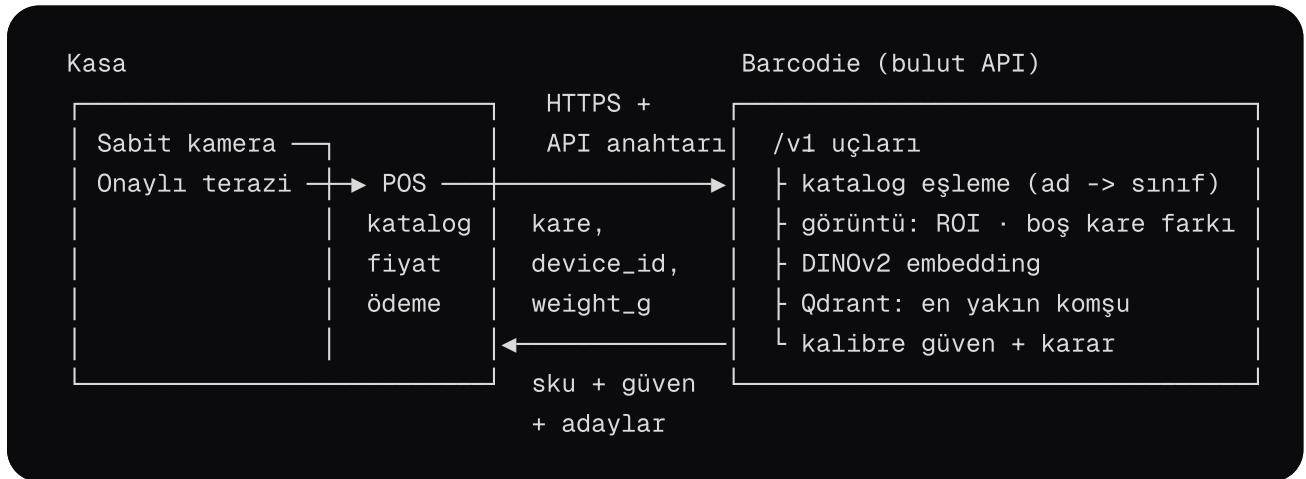
Barcode kasaya bakan bir kameranın karesini alır ve **POS'un kendi ürün kodunu (sku)** döner. Fiyatı POS hesaplar, ağırlığı onaylı terazi verir; biz sadece "bu hangi ürün" sorusunu cevaplarız. Model eğitimi yoktur: yaygın ürünler POS kataloğu senkronlanınca otomatik tanınır, geri kalanı için kasada 3 fotoğraf yeter. Emin olmadığımızda tahmin yazmayız — kasiyere en olası adayları gösteririz.

	SONUÇ	KAYNAK
Tek ürün, işletme 3 foto ekledikten sonra	%92,5–100 doğru (ilk tahmin)	▬ Tepeden kasa açısı, 40 kare, 3 model
Tek ürün, kurulum günü (hiç foto yok)	Doğru ürün ilk 3 adayda %95–100 ; ilk tahmin %73–90	▬ Aynı set, katalogla sınırlı arama
Otomatik onay (kasiyere sormadan)	Tahminlerin %38,8'i , bunların %99,2'si doğru	▬ 340 görsel, çapraz doğrulamalı ¹
Tanıma süresi (embedding)	47 ms CPU · 22 ms GPU, tek kare	▬ Apple M5; sunucu donanımında ölçülmedi
Karışık tepside ürün sayma	Adet doğruluğu %93 (yol haritası, pilotta kapalı)	▬ 30 tepsi, 88 ürün

¹ Model karşılaştırmasındaki (Benchmark 1) çapraz doğrulamalı sonuç. Aynı veriyle servisin kendisi (Qdrant + DINOv2-base, güncel kalibrasyon) %42,6 otomatik onay ve %99,3 doğruluk veriyor; bkz. §5.1.

2. Ürün ve entegrasyon modeli

Barcodeie kendi başına bir POS değil, POS'un içine giren bir servistir. Müşterimiz POS sağlayıcısı (entegratör), onun müşterileri işletmelerdir.



Entegrasyon akışı (POS tarafının yapacakları):

NE ZAMAN	ÇAĞRI	NE OLUR
Kurulum, bir kez	PUT /v1/merchants/{id}	İşletme ve KVKK izni kaydı
Kurulum + değişikçe	PUT /v1/merchants/{id}/catalog	POS kataloğu gönderilir; ürün adları global sınıflara otomatik eşlenir . Yanıt: hazır / onay bekleyen / foto gereken ürünler
Sadece gerekenler	POST .../catalog/{sku}/images	Eşleşmeyen ürüne 3–5 foto
Cihaz başına, bir kez	PUT .../devices/{device}/calibration	Boş terazi karesi + ilgi alanı
Her satış	POST .../recognitions	Kare (+ weight_g) → sku , adet/ağırlık, aday listesi
Kasiyer düzeltince	POST .../recognitions/{id}/feedback	Doğru cevap kaydedilir (izin varsa eğitim verisi)

Sözleşmenin tamamı: `api-contract/plugin-api.v1.yaml` (OpenAPI 3). Kasada canlı tetik için isteğe bağlı bir edge ajanı var: kamerayı izler, ürün yerleşip el çekildiğinde (veya terazi sabitlendiğinde) API'yi kendisi çağırır, sonucu yerel WebSocket ile POS'a iletir. Model çalıştırmaz. Sözleşmesi: `api-contract/edge-events.v1.asyncapi.yaml`.

3. Tanıma nasıl çalışır

- Görüntü hazırlığı** — kalibrasyonda verilen ilgi alanı (ROI) kırılır. Boş terazi karesiyle fark alınır: değişim yoksa "terazi boş" denir ve model hiç çalıştırılmaz.
- Embedding** — kare, DINOv2 ile 768 boyutlu bir vektöre çevrilir.
- Arama** — vektör, Qdrant'ta **sadece o işletmenin kataloğunda olan** ürünler arasında aranır: işletmenin kendi fotoğrafları + kataloğa eşlenmiş global sınıflar.
- Kalibre güven** — ham benzerlik skoru olasılık değildir (ölçümde yanlış tahminlerin medyan skoru 0,83, doğruların 0,85 — neredeyse aynı). Bu yüzden skoru ve ilk iki aday arasındaki farkı, ölçülmüş doğruluğa göre öğrenilmiş bir lojistik modelden geçiririz: $confidence = \text{sigmoid}(a \cdot \text{benzerlik} + b \cdot \text{fark} + c)$. Sonuç: "0,95 güven" gerçekten "bu düzeydeki 100 tahminin ~95'i doğru" demek. 
- Karar** — üç durum:

DURUM	KOŞUL	POS NE YAPAR
confirmed	Güven $\geq$ eşik (varsayılan 0,95, işletme değiştirebilir) ve ikinci adaydan açık fark ve sık karışan bir benzer ürün yok	Doğrudan sepete
needs_confirmation	Tanıdık ama emin değil	Kasiyere en olası 3 adayı gösterir
unrecognized	Hiçbir ürüne yeterince benzemiyor (el, poşet, katalog dışı)	Hiçbir şey

"Sık karışan benzer ürün" listesi de ölçümden gelir (golden $\leftrightarrow$ yeşil elma, şeftali $\leftrightarrow$ nektarin, limon $\leftrightarrow$ misket limonu...): bu çiftlerde güven yüksek olsa bile varsayılan kural otomatik onay vermez. **status** bir öneridir; POS isterse kendi kuralını **confidence** üzerine kurar.

4. Teknoloji kararları

Her karar için: ne seçtik, alternatifler neydi, neden, kanıt ve production'da ne olacak.

4.1 Model eğitmiyoruz: embedding + en yakın komşu

Karar	Ürünleri sınıflandıran bir model eğitmek yerine, hazır bir görsel temsil modeliyle her görseli vektöre çevirip en yakın referansı buluyoruz.
Alternatif	Ürün sınıflarıyla ince ayar (fine-tune) edilmiş bir sınıflandırıcı.
Neden	Menü haftalık değişiyor. Sınıflandırıcıda her yeni ürün = veri toplama + yeniden eğitim + dağıtım. Bizde yeni ürün = 3 fotoğrafın vektörü veritabanına yazılır, saniyeler içinde aktif. Ayrıca her işletmenin poğaçası farklı; tek bir global model bunu öğrenemez.
Kanıt	 Tepeden kasa setinde işletme 3 foto ekledikten sonra ilk tahmin doğruluğu %92,5–100 (3 farklı modelde, detector'süz).
Bedeli	Benzer ürünleri (şeftali/nektarin) ayırmak sınıflandırıcıdan zor olabilir. Buna karşı: kalibre güven + benzer ürün kuralı + kasiyer onayı.

4.2 Görsel temsil: DINOv2

Karar	facebook/dinov2-base (Meta, Apache-2.0). Hafif seçenek: dinov2-small .
Alternatifler	CLIP (OpenAI), SigLIP2 (Google), DINOv2-large.
Neden	İşimiz görselden görsele eşleştirme; metin yok. DINOv2 bu iş için eğitilmiş bir öz-denetimli model. CLIP/SigLIP metin-görsel hizalaması için eğitildi.
Kanıt	 Aynı 340 görsellik test setinde (Benchmark 1):

MODEL	İLK TAHMİN	İLK 3	OTOMATİK ONAY (DOĞRULUĞU)	CPU SÜRESİ	LİSANS
DINOv2-base	%85,9	%97,4	%38,8 (%99,2)	47 ms	Apache-2.0
DINOv2-small	%84,7	%97,7	%33,2 (%99,1)	20 ms	Apache-2.0
DINOv2-large	%85,9	%97,1	%46,2 (%99,4)	162 ms	Apache-2.0
SigLIP2-base	%85,6	%99,7	%42,4 (%100)	96 ms	Apache-2.0
CLIP B/16	%69,7	%95,3	%15,3 (%100)	78 ms	MIT
CLIP B/32	%69,4	%92,3	%12,9 (%97,7)	38 ms	MIT

- CLIP ilk tahminde ~16 puan geride: elendi.
- İlk dört model ilk tahminde istatistiksel olarak berabere (340 görselde ± 4 puan belirsizlik). DINOv2-base'i **en iyi doğruluk / en düşük gecikme dengesi** için seçtik; **small** neredeyse aynı doğrulukta 2,3× hızlı — GPU'suz ucuz sunucu için güçlü aday.
- **SigLIP2 gerçek bir rakip**: ilk 3'te ve otomatik onayda daha iyi, CPU'da 2× yavaş. Pilot verisiyle yeniden karşılaştıracğız. Model değişikliği tek ayardır (`EMBEDDING_MODEL`) + galeriyi yeniden yükleme.

4.3 Ürün kutulama (detector): pilota yok, yol haritasında var

Kutulama, bir karede **birden fazla ürünü** ayırıp saymak için gerekir. Terazide tek ürün senaryosunda gerekmez.

Karar	Pilot tek ürün (terazi) → detector kapalı. Karışık tepsi (fırın, kafeterya) yol haritasında.
Alternatifler	YOLO-World (Ultralytics), OWLv2 (Google), modelsiz boş kare farkı.
Kanıt	 <u>Benchmark 2</u> , 30 karışık tepsi / 88 ürün, işletme 3 foto ekledikten sonra:

YÖNTEM	ADET DOĞRU	ÜRÜN YAKALAMA	ÜRÜN KESİNLİĞİ	SEPET BİREBİR DOĞRU	LİSANS
YOLO-World + ürün adları sözlüğü	%93	%94	%91	%77	AGPL-3.0
YOLO-World, genel sözlük	%47	%73	%98	%43	AGPL-3.0
OWLv2 + ürün adları sözlüğü	%53	%93	%64	%43	Apache-2.0
OWLv2, genel sözlük	%30	%69	%62	%7	Apache-2.0
Boş kare farkı, bağlı bileşenler (modelsiz)	%17	%48	%65	%0	—

- **Teknik olarak en iyisi YOLO-World**; ayrıca 50× hızlı (CPU'da 115 ms, OWLv2 1,4 sn).
- **Ama lisansı AGPL-3.0**: ağ üzerinden hizmet olarak sunulursa servisin kaynak kodu kullanıcılara açılmalı. Ticari kullanımı yasaklamaz; POS'un kodunu etkilemez (HTTP ile konuşuyoruz). Yine de kapalı kaynak bir ticari serviste kullanmıyoruz.
- **Yol**: (1) pilot detector'süz → lisans sorunu yok; (2) karışık tepsi için kendi tepeden verimizle Apache lisanslı bir detector (RT-DETR / YOLOX / D-FINE) eğitmek; (3) ara çözüm olarak Ultralytics ticari lisansı. YOLO-World'ün %93'ü hedef çıtımız.
-  Raf fotoğrafı galeride ise detector **zarar veriyor** (ilk tahmin %85,9 → %76,5): yığından tek meyve kırıyor. Bu yüzden galeri her zaman kırpımsız yüklenir; detector yalnızca kasadaki sorgu karesine uygulanır.

4.4 Vektör veritabanı: Qdrant

Karar	Qdrant (Apache-2.0), tek düğüm, HNSW + kosinüs.
Alternatifler	FAISS, pgvector, Milvus, Pinecone.
Neden	 Üç ihtiyacımız var: (1) canlı ekleme/silme — işletme ürün ekler, anında aranabilir olmalı; (2) filtreli arama — her arama sadece o işletmenin kataloğunda (çok kiracılı izolasyon, §6); (3) kendi başına çalışan, kalıcı, API'li bir servis. FAISS bir kütüphanedir (kalıcılık, filtre, CRUD yok). Pinecone yönetilen ve bulut dışına çıkamaz — veri yerelliği/KVKK açısından istemedik. Milvus bu ölçek için ağır.
Dürüst not: pgvector	Ölçeğimiz küçük (aşağıda). pgvector da yeterli olur ve "tek veritabanı" avantajı getirir. Production'da Postgres'e geçerken pgvector'a taşınmak gerçek bir seçenek; iki yolu pilot verisiyle karşılaştırıp karar vereceğiz. Arama katmanı tek bir sınıfta (<code>QdrantRepo</code>), değişiklik yerel kalır.

Ölçek hesabı  (varsayım: 10.000 işletme × 20 kendi ürünü × 5 foto):

	VEKTÖR	HAM BOYUT (768 × FLOAT32)	İNT8 NİCEMLEMEYLE
Global galeri (59 sınıf × ~40 görsel)	~2.400	~7 MB	~2 MB
İşletme ürünleri	~1.000.000	~3,1 GB	~0,8 GB

Tek bir Qdrant düğümüne (veya pgvector'lu bir Postgres'e) rahatça sığar. Arama işletme filtresiyle yapıldığı için her sorgu fiilen birkaç yüz vektöre bakar; vektör DB darboğaz değildir — darboğaz embedding'dir.

4.5 İlişkisel veri: SQLite → Postgres

Bugün	SQLite (<code>plugin.db</code>): işletme, katalog + eşleme durumu, cihaz kalibrasyonu, tanıma kayıtları, geri bildirim. Atomik SQL güncellemeleri, idempotency anahtarı çıkarımdan önce talep edilir.
Neden şimdilik	Tek dosya, sıfır işletim yükü, pilot ölçeği için fazlasıyla yeterli.
Production	Postgres. Tetikleyici: birden fazla worker/replika. SQLite, oran sınırı ve galeri önbellegi süreç içi olduğu için bugün tek worker varsayımı var; ölçeklemek için Postgres + gateway seviyesinde oran sınırı + şema migrasyonu (Alembic) gerekir.

4.6 Fiyatı ve kiloyu biz hesaplamıyoruz

- **Fiyat dönmüyoruz, sadece `sku`** . Fiyatlar günlük değişir, şube/kampanya fiyatı vardır; fiyatı senkron tutmak bizi hata kaynağı yapar. Fiyat otoritesi POS'tur.
- **Kiloyu fotoğraftan tahmin etmiyoruz.** 📷 Tek fotoğraftan ağırlık tahmini laboratuvar da bile %11–20 hata verir; tartıyla satışta onaylı terazi yasal zorunluluk (Otomatik Olmayan Tartı Aletleri Yönetmeliği, 2014/31/AB). `weight_g` istekten aynen yansıtılır.

4.7 Servis katmanı

BİLEŞEN	SEÇİM	NEDEN
API	Python 3.11 + FastAPI	ML ekosistemi Python'da; async, OpenAPI'den şema
Sözleşme	OpenAPI 3 (<code>plugin-api.v1.yaml</code>)	Tek gerçek kaynak; değişiklik önce sözleşmede
Hata modeli	RFC 9457 problem+json	Her hata (500 dahil) tek biçimde; iç detay sızmaz
Dağıtım	Docker; CPU imajı mevcut	GPU zorunlu değil (§7)
Edge ajanı	Python, OpenCV, WebSocket	Kasada model yok; hafif bilgisayar yeter

5. Ölçümler

İki benchmark, iki farklı soru. Her ikisi de repoda tek komutla tekrar üretilir; Docker veya vektör DB gerekmez (arama, Qdrant'ın bu ölçekteki sonucuyla aynı olan kesin kosinüs ile yapılır — servis yoluyla çapraz kontrol edildi, sonuç birebir aynı çıktı).

5.1 Benchmark 1 — Model seçimi (raf fotoğrafı)

- **Veri:** Grocery Store Dataset (Klasson vd., WACV 2019, MIT) + Fruits-360 (MIT). Galeri 419 görsel / 42 sınıf; test 340 görsel / 34 sınıf, galeriyle çakışmaz.
- **Karar kuralı:** servisle aynı; kalibrasyon çapraz doğrulamalı (yarısında öğren, diğer yarısında ölç).
- **Sonuçlar:** §4.2 ve §4.3. Ayrıntı: [benchmarks/2026-09-30/summary.md](#) .

- **Servisin kendisiyle (Qdrant + DINOv2-base, kırpımsız):** ilk tahmin %85,9, ilk 3 %97,4; kalibre güven 0,95 eşliğinde tahminlerin %42,6'sı otomatik onaylanıyor, bunların %99,3'ü doğru (çapraz doğrulamada %99,2). Güven ölçeği tutarlı: 0,99 üstü güvenle verilen tahminlerin gerçek doğruluğu %99,2.
- **Tekrar:** `cd ai && python -m scripts.benchmark_models --latency-devices cpu mps`

5.2 Benchmark 2 — Kasa açısı (tepeden, sabit kamera)

- **Veri:** sabit tepeden telefon kamerası, tek tezgâh. 40 tekli kare (11 tür), 30 karışık tepsi (88 ürün), 1 boş tezgâh. Galeri yine raf fotoğrafı — yani "kurulum günü" durumunu ölçüyor.
- **"3 foto sonra":** her ürüne kendi kasasından en fazla 3 foto eklenir; birini-dışarıda-bırak — test edilen kare hiçbir zaman galeride değildir.

Tek ürün, pilot ayarı (DINOv2-base, detector'süz). Servis, boş kare kalibrasyonu varsa kareyi değişen bölgeye kırpar ("boş kare farkı"), yoksa tüm kareyi kullanır; ikisini de ölçtük:

DURUM	TÜM KARE: İLK TAHMİN / İLK 3	BOŞ KARE FARKI: İLK TAHMİN / İLK 3
Kurulum günü, tüm 42 sınıfta arama	%45 / %85	%55 / %82,5
Kurulum günü, işletme kataloğuyla sınırlı (servisin gerçek davranışı)	%80 / %100	%77,5 / %95
İşletme 3 foto ekledikten sonra	%100 / %100	%92,5 / %100

Boş kare farkı bu sette dezavantajlı: kareler farklı videolardan alındığı için kamera hafifçe kaydı, sabit montajlı gerçek kurulumda bu olmaz.

Kurulum günü hataları büyük ölçüde makul karışmalar: yeşil mandalina → misket limonu (galerideki mandalinalar turuncu), kuru soğan → kırmızı soğan, Türk salatalığı → kabak. Diğer modellerde (detector'süz) kurulum günü ilk tahmin %73–90 (SigLIP2 %90), 3 foto sonrası %92,5–100. Ayrıntı:

[benchmarks/2026-10-02-overhead/summary.md](#).

Satış cümlesi buradan çıkıyor: *"Kurulum günü yaygın ürünlerde doğru ürünü kasiyere ilk 3 aday içinde sunar; her ürün için kasada 3 foto çekildiğinde neredeyse kusursuz tanır."*

- **Tekrar:** `cd ai && python -m scripts.benchmark_overhead (veri ai/data/overhead/ + labels.json)`

5.3 Gecikme

🔧 Tek kare, ısınmadan sonra medyan (Apple M5 dizüstü; **sunucu CPU/GPU'sunda henüz ölçülmedi**):

ADIM	CPU	GPU (APPLE MPS)
DINOv2-small embedding	20 ms	10 ms
DINOv2-base embedding	47 ms	22 ms
YOLO-World kutulama (yol haritası)	115 ms	13 ms
Boş kare farkı (ROI, model yok)	< 5 ms	—

Ağ gidiş-dönüşü ve görüntü yükleme bunlara eklenir. Edge ajanının hedefi tetikten sonuca p50 ≤ 700 ms. 📖

6. Güvenlik, çok kiracılık ve KVKK

Çok kiracılık. Kiracı = (API anahtarının sahibi entegratör, `merchant_id`). Her SQL sorgusu ve her vektör araması kiracıyla filtrelenir; bir işletmenin ürünleri başka bir işletmenin tanımasını etkileyemez.

Kimlik ve sınırlar. Bearer API anahtarı (sabit zamanlı karşılaştırma), entegratör başı dakikada 600 istek (429 + `Retry-After`), dosya 4 MB, çözünürlük 40 MP (dekompresyon bombasına karşı çözmeden önce kontrol), istek gövdesi sınırı kimlik doğrulamadan önce. `ENVIRONMENT=prod` iken varsayılan anahtar, bellek içi vektör DB veya korumasız eski uçlar varsa servis **açılmaz**.

KVKK.

VERİ	SAKLAMA
Kasa karesi — işletme izni yoksa	Diske hiç yazılmaz
Kasa karesi — izin var, geri bildirim gelmedi	24 saat, sonra otomatik silinir
Kasa karesi — izin var, kasiyer düzeltti	Eğitim verisi; izin geri çekilince işletmenin tüm kareleri silinir
Tanıma kaydı (yanıt, kare yok)	30 gün

Yüz tanıma yapmayız; kamera ROI ile sadece tartı/tezgâh alanına bakar. Çıkarımın bulutta mı kasada mı yapılacağı entegratörle birlikte netleşecek açık bir sorudur (§9).

Lisanslar.

BİLEŞEN	KULLANIM	LİSANS
DINOv2 (Meta)	Görsel temsil	Apache-2.0
Qdrant	Vektör DB	Apache-2.0
FastAPI, Pydantic	Servis	MIT
SQLite	İlişkisel veri	Kamu malı
Grocery Store Dataset, Fruits-360 (eski sürüm)	Global galeri	MIT
YOLO-World (Ultralytics)	Sadece geliştirme/demo — yayında yok	AGPL-3.0

Ticari kullanıma kapalı veri setleri (RPC, SKU-110K, D2S) bilinçli olarak kullanılmadı.

7. Production'a geiř

KONU	BUGÜN	PRODUCTION	NE ZAMAN
Hesaplama	CPU yeterli (47 ms)	CPU ile bařla; canlı tetikte gecikme yetmezse GPU	Pilot ölçümüne göre
İliřkisel veri	SQLite, tek worker	Postgres + Alembic migrasyon + çoklu worker	Pilot sonrası
Vektör DB	Qdrant tek düğüm, iç ağı	API anahtarı + TLS + yedek; gerekirse int8 nicemleme. Alternatif: pgvector'a birleřtirme	Bulut kurulumunda
Oran sınırı	Süreç içi	API gateway	Çoklu worker ile
Gözlemlenebilirlik	Yapılandırılmış log	Metrikler: gecikme p50/p95, otomatik onay oranı, kasiyer düzeltme oranı	Pilottan önce
Yedek	Yok	<code>plugin_data</code> + Qdrant snapshot	Pilottan önce
Kalibrasyon	Raf fotoğrafıyla öğrenildi	Pilot kasasının kareleriyle yeniden öğrenilir (<code>eval_holdout --fit</code>)	Pilotun ilk haftası

Veri döngüsü. Kasiyerin her düzeltilmesi (`/feedback`), izin varsa etiketli bir kasa karesidir. Asıl rekabet avantajımız hazır veri setleri değıl, sahadan gelen bu veridir: kalibrasyon, benzer ürün listesi ve ileride kendi detector'ımız bununla iyileřir.

8. Yol haritası

- Pilot — tek ürün, terazili kasa.** Detector'süz, lisans temiz. Ölçülecekler: kasa süresi, otomatik onay oranı, kasiyer düzeltme oranı, gecikme. Kalibrasyon pilot verisiyle yeniden öğrenilir.
- Karışık tepsi.** Pilot karelerinden etiketli veri → Apache lisanslı kendi detector'ımız. Çıta: YOLO-World + ürün adları ile ölçtüğümüz %93 adet doğruluğı.
- Model güncellemesi.** SigLIP2 / DINOv2-small, pilot verisiyle yeniden karşılaştırılır.
- Tek fotoğrafla ürün tanıma.** Üretken görsel modelle tek fotodan çeřitilme + "benziyor ama değıl" negatifleri (`plans/AI_AUTO_ENROLL_PLAN.md`).

9. Bilinen sınırlar ve açık sorular

Ölçüm sınırları.

- Benchmark 2 küçük: 70 kare, tek tezgâh, tek ışık; ürün bazında oranlar güvenilir değil. "Yön gösteren ilk sonuç" olarak okunmalı. "3 foto sonra" testinde kurulum ve test kareleri aynı oturumdan — iyimser.
- "Katalogla sınırlı" sonuçlarda katalog 12 ürünlüktü; gerçek bir manav kataloğu daha büyük olacak ve benzer ürünleri (kırmızı soğan, misket limonu) birlikte içerecek — oranlar düşebilir.
- Benchmark 1'in verisi market rafı fotoğrafı, kasa açısı değil.
- Gecikme sunucu donanımında ölçülmedi.
- Pastane ürünleri (simit, poğaç, açma) için global galeri yok; bunlar işletme fotoğrafıyla tanıtılır ve bu senaryo henüz ayrıca ölçülmedi.

Bilinen zayıflıklar.

- Görsel olarak çok benzeyen ürünler (şeftali/nektarin, golden/yeşil elma, limon/misket limonu) — sistem bunlarda otomatik onay vermez, kasiyere sorar.
- Aynı global sınıfa eşlenmiş iki ürün ("Domates" ve "Organik Domates") ayırt edilemez; biri için işletme fotoğrafı gerekir.
- Pilotta bir karede tek ürün.

Entegratörle netleşecekler.

- Çıkarım bulutta mı, kasada mı?
- Terazı bağlantısı (POS üzerinden mi, seri port mu)?
- Kamera donanımı ve montaj standardı.
- KVKK aydınlatma metni ve veri işleme sözleşmesi.

Barcodie, Moka United AI Ideathon & Hackathon'da 400 başvuru arasından ikincilik aldı. Kod ve tüm ölçüm script'leri talep halinde paylaşılabilir.