Advanced AI/ML Computer Vision API Documentation

Overview

The **AI/ML Vision Inference API** delivers high-throughput, low-latency computer vision via a distributed, scalable architecture. It supports advanced model ensembles (Vision Transformer, Cascade R-CNN), robust authentication, and enterprise-grade compliance.

Table of Contents

Architecture Overview

- **Distributed Inference:** Requests routed to nearest edge node for minimal latency.
- **Model Orchestration:** Ensemble of ViT-L/16 and Cascade R-CNN, configurable per request.
- **Scalable Processing:** Supports >50,000 requests/sec globally.

Authentication & Security

JWT Authentication:
All requests require a signed JWT in the `Authorization` header.

Required Headers:

Header	Description
`Authorization`	Bearer token (JWT)
`X-API-Key`	RSA-encrypted client ID
`X-Timestamp`	Unix epoch in nanoseconds
`X-Content-Signature`	SHA3-512 hash of request body

Example JWT Payload:

```
{
  "iss": "your-client-id",
  "sub": "api-access",
  "exp": 1718246400,
  "scope": "vision:inference"
}
```

API Endpoints

POST `/v2/vision/inference`

Parameters:

Name	Type	Required	Description
`image`	binary	Yes	Raw image bytes (max 20MB)
`operations`	array	Yes	List of processing steps
`model_ensemble`	string	No	Ensemble profile (A, B, C, D)
`quality_tradeoff`	float	No	Speed/accuracy balance (0.0-1.0)
`region_hints`	array	No	Geographical prioritization codes

Sample Request (JSON):

```
{
  "operations": [
    {
      "type": "object_detection",
      "params": {
        "threshold": 0.85,
        "max_objects": 50
      }
    },
    {
      "type": "optical_character_recognition",
      "params": {
        "languages": ["eng", "spa"],
        "handwriting": false
      }
    }
  ],
  "model_ensemble": "A",
  "quality_tradeoff": 0.7
}
```

Request and Response Schema

Sample Response:

```
{
  "request_id": "01HXYZABCDEF1234567890",
  "results": [
    {
      "type": "object_detection",
      "data": {
        "objects": [
          {
            "label": "vehicle:car",
            "confidence": 0.978,
            "bbox": [0.12, 0.45, 0.34, 0.67],
            "attributes": {
              "color": "red",
              "orientation": 45.2
            }
          }
        ]
      }
    }
  ],
  "model_versions": {
    "object_detection": "cascade-rcnn-2025.1.3",
    "ocr": "tesseract-5.3.1"
  },
  "performance_metrics": {
    "total_inference_time": 142.3,
    "breakdown": {
      "preprocessing": 12.1,
      "model_execution": 123.4,
      "postprocessing": 6.8
    }
  }
}
```


Response Fields:

Field	Type	Description
`request_id`	string	Unique request identifier
`results`	array	List of operation results
`model_versions`	object	Model names and versions used
`performance_metrics`	object	Timing breakdowns for each stage

Advanced Parameters

Model Ensemble Profiles:

Profile	Models Used	Use Case
A	ViT-L/16 (80%), Cascade R-CNN (20%)	General purpose

B	Cascade R-CNN (100%)	High-precision detection
C	ViT-L/16 (100%)	Contextual understanding
D	YOLOv9 + ViT-L/16	Real-time video analysis

Quality Tradeoff Matrix:

Value	Resolution	Model Quantization	Batch Size
0.0	1024px	FP32	1
0.5	512px	FP16	8
1.0	256px	INT8	32

Error Handling & Retry Strategies

Common Error Codes:

Code	Error Type	Resolution
400	Invalid Request	Check image encoding and parameters
401	Unauthorized	Verify JWT and API key
413	Payload Too Large	Reduce image size
429	Rate Limit Exceeded	Implement exponential backoff
500	Internal Error	Retry with idempotency key

Retry Example (Python):

```python
import time, random

def retry_request(func, max_retries=3):
    for attempt in range(max_retries):
        try:
            return func()
        except RateLimitError:
            time.sleep((2 ** attempt) + random.random())
    raise Exception("Max retries exceeded")
```

Rate Limits & Quotas

Tier	RPS	Burst	Monthly
Starter	10	50	100K
Professional	100	500	10M
Enterprise	Custom SLA	Custom	

Model Cards & Versioning

cascade-rcnn-2025.1.3

Property	Value
Training Data	COCO 2024 + proprietary (2M images)
mAP@0.5	0.892
mAP@0.95	0.745
Hardware	NVIDIA A100 80GB (FP16)
Bias Mitigation	Oversampling, fairness metrics
Deprecation Policy	6-month support window

Performance Optimization

- Use **gRPC** for batch streaming when processing >100 images/second.
- Set `quality_tradeoff` to 0.5 for faster, lower-precision inference.
- Pin model versions in request headers for reproducibility.

Compliance & Ethics

- **Bias Mitigation:** Dataset balancing, fairness metrics (Equal Opportunity Difference < 0.05).
- **Anonymization:** Automatic blurring of faces/license plates.
- **GDPR:** Data retention policies enforced (raw images: 24h, metadata: 30d).

Quickstart Guide

Python Async Example:

```
import aiohttp

async def analyze_image(image_bytes):
    headers = {
        'Authorization': f'Bearer {JWT_TOKEN}',
        'X-API-Key': API_KEY
    }
    async with aiohttp.ClientSession() as session:
        async with session.post(
            'https://api.vision.ai/v2/vision/inference',
            headers=headers,
            data=image_bytes
        ) as resp:
            return await resp.json()
```

Troubleshooting

Symptom	Resolution
403 Forbidden	Rotate API keys, check permissions
429 Too Many Requests	Implement retry/backoff
Model Version Conflict	Pin versions in request
Bounding Box Drift	Calibrate camera intrinsics
