Urban VS

Documentación Técnica

“Sistema de análisis de tráfico aéreo con evidencia verificable en blockchain (BSV)”

NeuralHack — IA + Blockchain

Jaime Borrero Ramírez · Juan Manuel Palencia Osorio

Versión: v1.0.0

08/02/2026

Organizado por:

Student Club Universidad Europea

{C-INT}

ÍNDICE

1. Visión General del Proyecto..... 1
2. Arquitectura del Sistema..... 2
3. Instalación y Configuración..... 3
3.1. Dependencias..... 3
3.2. Configuración .env..... 3
3.3. Datasets..... 3
4. Uso del Sistema..... 4
4.1. Modo Aplicación (Recomendado)..... 4
4.2. Modo Entrenamiento..... 4
4.3. CLI..... 4
5. API REST (Desarrollo Futuro)..... 5
6. Métricas y Blockchain..... 6
6.1. Métricas Calculadas..... 6
6.2. Blockchain BSV..... 6
7. Simulador de Tráfico..... 7
8. Desarrollo Futuro..... 8
9. Conclusión..... 9

1. Visión General del Proyecto

Urban VS es un sistema integral de análisis de tráfico que combina detección de vehículos por visión computacional, cálculo de métricas avanzadas y registro inmutable en blockchain BSV.

Funcionalidades Principales:

El sistema integra un modelo YOLOv8 fine-tuned específicamente en datasets aéreos (UAV + Roundabout), alcanzando una precisión de 87.2% mAP@50 en la detección de cuatro clases vehiculares: cars, motorcycles, trucks y buses. Esta especialización permite un análisis de tráfico mucho más preciso que modelos genéricos.

Para cada imagen procesada, el sistema calcula métricas avanzadas incluyendo un grid de densidad 5x5, porcentaje de ocupación de vía, detección de colisiones mediante IoU, y evaluación automática del nivel de riesgo (LOW/MEDIUM/HIGH/CRITICAL) considerando múltiples factores como densidad vehicular y tipo de vehículos presentes.

Todo análisis genera un hash SHA-256 que se registra en blockchain BSV mediante transacciones OP_RETURN, proporcionando un timestamp inmutable y verificable. Adicionalmente, el simulador permite predecir métricas bajo diferentes escenarios (hora del día, condiciones climáticas, eventos especiales), facilitando la planificación urbana. La interfaz Streamlit proporciona visualizaciones interactivas con heatmaps, grids de densidad y overlays de detecciones.

2. Arquitectura del Sistema

Urban VS implementa una arquitectura modular basada en un pipeline de procesamiento secuencial. El flujo comienza cuando una imagen ingresa al sistema, ya sea mediante la interfaz Streamlit, la API REST o el CLI. Esta imagen es procesada por el VehicleDetector, que ejecuta inferencia con el modelo YOLOv8 fine-tuned y, cuando es necesario, fusiona resultados de un modelo de respaldo para clases difíciles como autobuses.

Las detecciones obtenidas son analizadas por el TrafficAnalyzer, que calcula el conjunto completo de métricas: grid de densidad dividiendo la imagen en 25 celdas, porcentaje de ocupación sumando áreas de bounding boxes, detección de colisiones mediante cálculo de IoU entre vehículos, y clasificación de riesgo considerando factores como número de colisiones y concentración vehicular. Este análisis se serializa en un payload JSON canónico que se hashea con SHA-256 para garantizar integridad criptográfica.

El hash resultante se registra en blockchain BSV mediante el BSVAdapter, que construye una transacción con output OP_RETURN conteniendo el hash y metadatos, la firma con la clave privada del sistema, y la transmite a la red mediante ARC. Paralelamente, el evidence record se guarda en un ledger local (ledger.jsonl) para consultas rápidas sin depender de la blockchain.

Módulos Principales:

El módulo de detección (detection/detector.py) encapsula toda la lógica de inferencia YOLOv8, mientras que el analizador de métricas (metrics/analyzer.py) implementa los algoritmos de cálculo incluyendo soporte especializado para rotondas. El sistema de hashing (hashing/integrity.py) garantiza la construcción determinística de payloads JSON, y el adaptador blockchain (blockchain/adapter.py) maneja la construcción, firma y broadcast de transacciones BSV. Adicionalmente, el módulo de visualización (visualization/overlays.py) genera heatmaps y overlays, y el simulador (simulator/engine.py) implementa el modelo estocástico para predicciones de escenarios.

3. Instalación y Configuración

3.1. Dependencias

pip install -r requirements.txt

Este comando instalará todas las dependencias necesarias, incluyendo el framework Ultralytics para YOLOv8, bibliotecas de procesamiento de imágenes (OpenCV, NumPy), frameworks web (Streamlit, FastAPI), la biblioteca bsvlib para interacción con blockchain, y herramientas adicionales para análisis científico y gestión de configuración.

3.2. Configuración .env

El proyecto incluye .env preconfigurado con clave BSV fondeada. No requiere configuración adicional.

BSV_NETWORK=main

BSV_PRIVATE_KEY=<incluida>

YOLO_MODEL=runs/detect/.../best.pt

DEVICE=cpu

3.3. Datasets

La carpeta data/ contiene muestras para demo. Para entrenamiento completo, descargar datasets desde Kaggle:

- traffic_aerial_images_for_vehicle_detection/ (~5000 imágenes)
- roundabout_aerial_images_for_vehicle_detection/ (~800 imágenes)

4. Uso del Sistema

4.1. Modo Aplicación (Recomendado)

streamlit run app.py

Abre http://localhost:8501. Incluye modelo pre-entrenado, listo para usar.

Páginas disponibles:

- Análisis: Sube imagen -> detecciones + métricas + registro blockchain
- Verificación: Ingresa hash -> consulta ledger + blockchain
- Simulador: Configura escenarios -> métricas simuladas
- Registros: Explora histórico, filtra, exporta JSON

4.2. Modo Entrenamiento

1. Descargar datasets completos -> extraer en data/
2. python setup_train.py -> combina datasets en formato YOLO
3. python train.py --epochs 50 --batch 16 -> entrena modelo
4. Entrenamiento: 2-8h CPU, 30-90min GPU. Resultado en runs/detect/.../best.pt

4.3. CLI

python cli.py analyze image.jpg --register

python cli.py verify <hash>

5. API REST (Desarrollo Futuro)

Nota: La API está implementada en api.py pero se presenta como funcionalidad futura.

Endpoints Planeados:

- POST /analyze: Analiza imagen multipart -> detecciones + métricas + hash + tx_id
- GET /verify?hash=: Busca análisis por hash -> evidence_record o null
- GET /records?limit=: Lista registros recientes -> array JSON
- POST /simulate: Ejecuta simulación -> métricas what-if
- GET /health: Health check -> status sistema

Ejecución:

uvicorn api:app --host 0.0.0.0 --port 8000 --workers 4

Docs interactivos: http://localhost:8000/docs

6. Métricas y Blockchain

6.1. Métricas Calculadas

- Density Grid: Imagen dividida en 5x5, cuenta vehículos por celda. >6 vehículos/celda = hot spot.
- Occupancy: suma(áreas_bbox) / área_imagen × 100. <5% ligero, 5-12% moderado, >20% saturado.
- Colisiones: IoU entre pares de vehículos. >0.4 = HIGH, 0.3-0.4 = MEDIUM, 0.2-0.3 = WARNING.
- Riesgo: Función de colisiones + densidad máxima + % vehículos pesados. Clasificación: LOW/MEDIUM/HIGH/CRITICAL.

6.2. Blockchain BSV

El sistema utiliza BSV blockchain para proporcionar un registro inmutable y públicamente verificable de cada análisis de tráfico. El proceso comienza construyendo un payload JSON canónico mediante serialización determinística (claves ordenadas alfabéticamente, sin espacios en blanco), del cual se calcula un hash SHA-256. Este hash de 64 caracteres hexadecimales actúa como huella digital única del análisis.

Para registrar el hash en blockchain, el sistema construye una transacción Bitcoin con dos outputs: el primero es un OP_RETURN con valor cero que contiene el hash y metadatos adicionales, y el segundo devuelve el cambio a la dirección controlada por el sistema. La transacción se firma con la clave privada WIF configurada y se transmite a la red BSV mediante ARC (gorillapool.io).

El tx_id resultante se almacena en el ledger local junto con toda la información del análisis. Para verificar evidencia, el sistema primero consulta el ledger local para búsquedas rápidas, y opcionalmente puede verificar on-chain mediante WhatsOnChain API. El timestamp del bloque en que se incluyó la transacción certifica de manera inmutable el momento exacto del análisis, protegido por el proof-of-work de la red distribuida.

7. Simulador de Tráfico

El simulador implementa un modelo estocástico para predicción de tráfico en escenarios hipotéticos, permitiendo evaluar el impacto de diferentes condiciones sin necesidad de datos reales. Este modelo combina factores determinísticos basados en estudios de tráfico urbano con variabilidad aleatoria para generar distribuciones realistas.

Factores:

- Temporales: Multiplicadores por hora del día (0.05 a 1.00), reducción 40% fines de semana
- Climáticos: Soleado 1.0, nublado 0.95, lluvia 0.80, niebla 0.70
- Eventos: None 1.0, Low 1.1, Medium 1.25, High 1.45

Algoritmo:

1. demand_factor = time × weather × event
2. total_vehicles = BASE_CAPACITY × demand_factor × random(0.85, 1.15)
3. Distribuir por clases según VEHICLE_MIX[scene_type]
4. Generar density_grid simulado con coordenadas aleatorias
5. Calcular métricas derivadas (occupancy, risk)

Casos de Uso:

- Planificación de infraestructura: impacto de cierre de carriles
- Evaluación de eventos: tráfico adicional en conciertos
- Optimización de semáforos: efecto de mejor sincronización
- Estudios de sensibilidad: percentiles de riesgo (P95, P99)

8. Desarrollo Futuro

El proyecto se encuentra en constante evolución. Las próximas iteraciones se enfocarán en el despliegue de la API REST en infraestructura cloud con autenticación robusta, implementación de tracking multi-frame mediante algoritmos como DeepSORT para analizar flujos de video en lugar de imágenes estáticas, y optimización de performance mediante batch processing y cuantización de modelos.

A medio plazo se planea expandir las capacidades de detección incluyendo nuevas clases vehiculares (bicicletas, scooters, vehículos de emergencia), integración con sistemas municipales de control de semáforos para optimización adaptativa en tiempo real, y desarrollo de modelos predictivos que aprovechen históricos de tráfico combinados con datos climáticos y eventos programados.

La visión a largo plazo incluye capacidades avanzadas como detección de comportamientos anómalos mediante autoencoders, predicción de tráfico futuro con modelos de series temporales, y expansión del sistema de verificación blockchain a múltiples cadenas. También se planea el desarrollo de una aplicación móvil para participación ciudadana y la publicación del dataset combinado como recurso público para la comunidad académica.

9. Conclusión

Urban VS combina visión computacional (YOLOv8) y blockchain (BSV) para gestión inteligente de tráfico con evidencia verificable. El sistema automatiza detección, análisis y registro, garantizando transparencia e inmutabilidad.

La arquitectura modular permite extensibilidad: tracking multi-frame, predicción, control de semáforos, sin reescribir código existente. La separación Streamlit/FastAPI/módulos facilita mantenimiento y testing.

Visión futura: integración con sistemas municipales, expansión a otras modalidades de transporte, análisis predictivo, participación ciudadana. Urban VS es un paso hacia ciudades más inteligentes, seguras y transparentes.