1차 모델 비교

1) 예측(링크 속도·도착/헤드웨이) 중심 모델 비교

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **범주** | **모델** | **쓰임새** | **강점** | **약점** | **인퍼런스/운영** | **근거/핵심** |
| 도로 시공간 | **DCRNN** | 링크 **속도/소요시간** 예측(도로 그래프) | 확산합성곱+RNN으로 **공간·시간 의존** 동시학습, 교통데이터 SOTA급 | 구현/학습 상대적으로 무거움 | GPU 권장(학습), 추론은 CPU도 가능 사례 보고 | DCRNN 원논문·ICLR’18, 대규모 배치 사례(캘리포니아 고속도로) ([arXiv](https://arxiv.org/abs/1707.01926" \o "Diffusion Convolutional Recurrent Neural Network)) |
| 도로 시공간 | **STGCN** | 링크 **속도/TT** 예측 | GCN+TCN 블록으로 **빠른 추론**, 파라미터 적음 | 아주 장기 의존은 한계 | GPU 권장 | STGCN(IJCAI’18) ([arXiv](https://arxiv.org/abs/1709.04875" \o "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting)) |
| 도로 시공간 | **Graph WaveNet** | 링크 **속도/TT** 예측 | **적응형 인접행렬**로 숨은 공간의존 학습, **dilated conv**로 장기 시퀀스 | 구현 난도↑ | GPU 권장 | IJCAI’19 Graph WaveNet ([ijcai.org](https://www.ijcai.org/proceedings/2019/264)) |
| 다변량 시계열 | **TFT** | **멀티-호라이즌** 예측(혼잡·도착·수요 등) | **해석가능 어텐션**(변수/시간 중요도), 미래가 알려진 변수 활용(달력·예보) | 모델/튜닝 복잡, 지연↑ | GPU 권장 | TFT arXiv·저널 버전 ([arXiv](https://arxiv.org/abs/1912.09363" \o "Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting)) |
| 장기 시계열 | **Informer** | 장기 입력/출력(예: 24h 이상) | ProbSparse로 **효율 좋은 트랜스포머**, 대규모 장기예측 강함 | 데이터 적으면 과적합·복잡도↑ | GPU 필요 | AAAI’21 논문/공개 구현 ([arXiv](https://arxiv.org/abs/2012.07436" \o "Informer: Beyond Efficient Transformer for Long Sequence ...)) |
| 확률 예측 | **DeepAR** | 버스/지하철 **도착·헤드웨이**의 **불확실성**(분위수/PI) | **다수 시계열 공유학습**, 예측구간으로 보수/공격 경로 의사결정 | RNN 기반(학습 지연), 해석성 낮음 | GPU 권장 | DeepAR(2017/2020) ([arXiv](https://arxiv.org/abs/1704.04110" \o "DeepAR: Probabilistic Forecasting with Autoregressive ...)) |
| 탭형(경량) | **LightGBM/XGBoost/CatBoost** | POI/정류장/링크 단위 **단·중기 예측**(풍부한 피처) | **CPU도 고속**, 결측·이상치 강함, **해석 용이** | 순차·장기 의존은 **피처공학** 필요 | CPU OK(저비용) | (GBDT 계열 개요 — 표준 레퍼런스는 필요 시 별도 제공) |

2) 경로탐색(알고리즘)·데이터 표준

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **범주** | **알고리즘/표준** | **쓰임새** | **강점** | **약점/주의** | **근거/핵심** |
| 차량 | **시간의존 최단경로**(Time-Dependent Dijkstra/A\*) | 예측된 링크 **시간의존 가중치**로 경로 | 실시간 변화 반영, 표준 그래프 탐색 확장 | 가중치 함수 설계/업데이트 필요 | 시간의존 최단경로 고전·응용 연구 ([트래블 리서치 보드 온라인 간행물](https://onlinepubs.trb.org/Onlinepubs/trr/1993/1408/1408-012.pdf)) |
| 대중교통 | **RAPTOR** | **다중기준**(환승/대기시간 등) 대중교통 여정 탐색 | 매우 빠름, 실무 적용 다수 | 그래프가 아닌 **라운드 기반** 사고 필요 | RAPTOR 원논문(ALLEnex’12 등) ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) |
| 대중교통 | **Connection Scan Algorithm(CSA)** | 시간표 기반 **최단 도착시간** 탐색 | **연결 스캔**으로 고속, 구현 단순 | 실시간 업데이트 처리 설계 필요 | CSA 저널/ArXiv ([ACM Digital Library](https://dl.acm.org/doi/10.1145/3274661)) |
| 표준 | **GTFS(Static/Realtime)** | 노선/정차/일정 + 실시간 지연/운행상태 | 산업 표준, 다양한 도구/라이브러리 | 데이터 품질·지자체별 차이 | GTFS 공식/구글 개발자 문서 ([gtfs.org](https://gtfs.org/documentation/overview/)) |

3) **엔드투엔드 조합**(서울시 데이터에 딱 맞춘 실전 파이프라인)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **조합(파이프라인)** | **입력 데이터(예)** | **경로 산출 방식** | **장점** | **단점/주의** | **언제 쓰나** |
| **[차량] GNN(도로속도 예측) → 시간의존 최단경로** | 도로 링크별 평균속도·상태, 과거 속도 시계열, 기상/이벤트, 인근 인구혼잡(패널티) | 예측 **링크 TT(t)** → **Time-Dependent Dijkstra/A**\* | 실시간 정체 반영, **소요시간 최소 경로** 직접 산출 | 그래프/센서 매핑·학습 인프라 필요 | 차량 경로가 핵심일 때 **기본 선택** ([arXiv](https://arxiv.org/abs/1707.01926" \o "Diffusion Convolutional Recurrent Neural Network)) |
| **[대중교통] DeepAR/TFT(도착·헤드웨이) + GTFS → RAPTOR/CSA** | 정류장/역별 도착·헤드웨이 시계열, **GTFS-Static/Realtime**, 실시간 인구(혼잡/쾌적도), 환승거리 | 예측된 **머무름/대기·환승 시간**을 반영해 **RAPTOR/CSA**로 여정 | 빠른 라우팅, **멀티기준(환승/대기/쾌적도)** 가능 | GTFS 품질, 실시간 피드 안정성 의존 | 대중교통 경로 추천의 **표준 해법** ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) |
| **[혼합] 차량·대중교통 후보 경로 → GBDT 재순위(개인화)** | 사용자의 선호(환승회피/보행한도), 실시간 인구혼잡, 날씨, 과거 선택 로그 | 후보 경로들의 \*\*특징(시간·환승·혼잡·신뢰도)\*\*로 **LightGBM/CatBoost** 재랭크 | 개인화·설명 용이, **경량·저지연** | 피처 설계/로그 축적 필요 | **맞춤형 추천**(퍼스널라이즈) 마무리 단계 |
| **[강건] 확률예측(DeepAR 분위수) → 위험회피 경로** | 도착/속도 **예측구간(PI)**, 사고/지연 리스크 | **기대시간 + 리스크(상위 분위수)** 최소 경로 | 지연 리스크 관리, **신뢰성↑** | 정책/튜닝 복잡 | **정시성**이 중요한 출퇴근/환승 시 |

2차 차량/ 버스/ 지하철 별 모델 비교

차량용(도로) — 엔드투엔드 파이프라인 후보

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **파이프라인** | **입력(필수/옵션)** | **예측(링크 속도/TT)** | **경로탐색** | **개인화/재랭킹** | **모델 서빙/배포** | **캐시·SLO** | **장점** | **주의/언제** |
| **A1. GNN 기반(표준형)** | 도로 링크별 실시간·과거 속도/상태, 이벤트/날씨, 인구혼잡(패널티) | **DCRNN/STGCN/Graph WaveNet** 중 1~2개 비교 적합 (시공간 의존 포착) ([OpenReview](https://openreview.net/pdf?id=SJiHXGWAZ" \o "DIFFUSION CONVOLUTIONAL RECURRENT NEURAL ...)) | **시간의존 최단경로**(TD Dijkstra/A\*) — 링크 가중치=예측 TT(t) ([arXiv](https://arxiv.org/pdf/1705.02044" \o "A Survey of Shortest-Path Algorithms)) | 후보 경로 특징(예상시간, 변동성, 혼잡)을 **LightGBM/CatBoost**로 재랭킹 | **Triton**(ONNX/TF/PyTorch 혼용), **모델 저장소=S3**/로컬 디렉토리, gRPC/HTTP 서빙 ([NVIDIA Docs](https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/architecture.html)) | 링크 TT 30–60초 캐시, SLA P95<300ms(서버 내 배치/동시성 조정) | 실제 도로망에 강함, 확장성 | 그래프 구축/학습 무게 ↑. **정확도·지연 균형** 필요 |
| **A2. 경량 하이브리드** | 위와 동일(+ 달력/행사) | **GBDT(LightGBM/CatBoost)** + 풍부한 시계열 피처(shift/rolling) | TD Dijkstra/A\* | 동일 | **TensorFlow Serving**(TF 계열), 또는 Triton의 Python backend로 GBDT도 통합 ([TensorFlow](https://www.tensorflow.org/tfx/guide/serving)) | 10–30초 캐시 | **저지연/저비용**, CPU만으로도 충분 | 장기 의존/상호작용은 한계 — 베이스라인/초기 운영에 |
| **A3. 장기예측 특화** | 장주기 패턴(주간/계절) 강조 | **TFT/Informer**로 멀티-호라이즌(6–24h) 보정 ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) | TD Dijkstra/A\* | 동일 | Triton(TensorRT/ONNX) | 분 단위 캐시 + 배치 재계산 | 장기 혼잡·이벤트 대응 | 모델 무거움, 데이터 충분할 때 |

버스용 — 엔드투엔드 파이프라인 후보

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **파이프라인** | **입력(필수/옵션)** | **예측(도착/헤드웨이)** | **경로탐색** | **개인화/재랭킹** | **모델 서빙/배포** | **캐시·신선도** | **장점** | **주의/언제** |
| **B1. 확률예측 + 표준 라우터(권장)** | **GTFS-Static**(노선/정차/시간표) + **GTFS-RT**(TripUpdates/VehiclePositions/Alerts) + 인구혼잡/날씨 | **DeepAR**(분위수/PI 제공) 또는 **TFT**(멀티-호라이즌)로 도착/헤드웨이 예측 ([Google for Developers](https://developers.google.com/transit/gtfs-realtime)) | **RAPTOR**(라운드 기반) 또는 **CSA**(연결 스캔)로 Earliest Arrival/최소환승 탐색 ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) | 경로 후보(소요/환승/혼잡/신뢰도)를 **GBDT**로 재랭킹 | Triton/TFS. 모델 저장소에서 버전관리, gRPC로 서빙. | **GTFS-RT는 90초 이내 신선도 권장**(TripUpdates/VehiclePositions) ([gtfs.org](https://gtfs.org/documentation/realtime/realtime-best-practices/)) | 표준 데이터·알고리즘 조합, 빠르고 강건 | GTFS 품질 의존. 실시간 feed 끊김 대비 캐시·폴백 설계 필요 ([gtfs.org](https://gtfs.org/documentation/realtime/reference/)) |
| **B2. 경량 베이스라인** | GTFS-Static/RT | **GBDT**로 도착시간 회귀(풍부한 피처) | RAPTOR/CSA | 동일 | TFS/경량 Flask+Uvicorn(사내 핸들러) | 30–60초 캐시 | 구축 쉬움, 저비용 | 피크/변동성 대응은 확률예측보다 약함 |
| **B3. 장기 패턴 반영** | 위 + 행사/학사일정 | **Informer**로 장기 예측(평균 지연, 헤드웨이 안정성) → 실시간 예측과 앙상블 | RAPTOR/CSA | 동일 | Triton | 분 단위 캐시 | 행사·기상 영향 큰 노선에 유리 | 리소스↑, 데이터 축적 후 단계적 투입 |

지하철용 — 엔드투엔드 파이프라인 후보

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **파이프라인** | **입력(필수/옵션)** | **예측(도착/혼잡)** | **경로탐색** | **개인화/재랭킹** | **모델 서빙/배포** | **캐시·신선도** | **장점** | **주의/언제** |
| **S1. 표준형(장애/지연 대응)** | **GTFS-Static/RT**(특히 TripUpdates/Alerts), 실시간 인구혼잡/승강장 혼잡 | **DeepAR/TFT**로 도착·헤드웨이 예측 + 혼잡 지표 회귀 | **RAPTOR/CSA**로 최단도착/최소환승 + **혼잡패널티** 반영 ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) | **GBDT 재랭킹**(대기/환승/혼잡 불편도, 엘리베이터 여부) | Triton/TFS | GTFS-RT 30–60초 캐시(Alerts는 10분 이내), TripUpdates 신선도 준수 ([gtfs.org](https://gtfs.org/documentation/realtime/realtime-best-practices/)) | 실무 적용 용이, 장애 대응 | 실시간 품질 의존, 역내 IoT 연계 시 추가 공수 |
| **S2. 장기 운행패턴+실시간 앙상블** | 위 + 장기 이력(월/학기) | **Informer/TFT**로 장주기 편차 학습 → 실시간 예측과 앙상블 | RAPTOR/CSA | 동일 | Triton | 분 단위 캐시 | 행사·계절 영향 반영 | 모델 무거움, 점진 도입 |
| **S3. 경량 베이스라인** | GTFS-Static/RT | **GBDT** 단일 모델 | RAPTOR/CSA | 동일 | 경량 서빙 | 30–60초 | 구현·운영 간단 | 예측구간/리스크 미지원(필요시 B1/S1로 승격) |

3차 프로젝트 구조 개편 이후 모델 비교

엔드투엔드 파이프라인 후보(Cloudflare + API Gateway/Lambda)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **파이프라인** | **엣지(Cloudflare) 역할** | **AWS 역할** | **데이터 소스 / 신선도** | **지연 & 한계** | **강점** | **주의/리스크** | **적합한 용도** |
| **H1. 엣지 캐시 주도 + 온디맨드 추론(권장 기본형)** | **Workers**가 실시간 외부 API(도시데이터 120장소·버스·지하철·도로)를 **Cron Triggers**로 30–60초 간격 프리패치 → **Cache API/KV**에 저장·TTL 부여. 요청 시 캐시 히트 우선, 필요하면 API Gateway 호출. ([Cloudflare Docs](https://developers.cloudflare.com/workers/configuration/cron-triggers/)) | Lambda(HTTP API/REST API)가 **경량 추론**(LightGBM/소형 PyTorch/ONNX). 컨테이너 이미지는 최대 10GB로 모델/의존성 패키징 가능. ([AWS Docs](https://docs.aws.amazon.com/lambda/latest/dg/images-create.html)) | 실시간 인구/교통(서울 **120장소**, 버스·지하철 도착, 도로 소통). **GTFS-RT는 30초~90초 내 업데이트** 준수. ([서울 열린데이터광장](https://data.seoul.go.kr/dataList/OA-21285/F/1/datasetView.do)) | API Gateway 통합 타임아웃(REST는 29초 초과도 가능, 설정 필요). **Lambda 동기 페이로드 6MB**, API Gateway **10MB**. P95 100~300ms 목표(엣지 캐시 히트 시 수십 ms). ([Amazon Web Services, Inc.](https://aws.amazon.com/about-aws/whats-new/2024/06/amazon-api-gateway-integration-timeout-limit-29-seconds)) | **AWS 비용 최소화**(엣지 캐시로 호출 감축), **짧은 지연**, 실시간 다중 소스 융합 쉬움 | 캐시 일관성 관리, 원본 API rate-limit/장애 시 폴백 필요 | **초기 MVP/상시 운영**에 가장 균형적. 고QPS·낮은 지연 요구 |
| **H2. 사전예측(배치) + 엣지 정적 서빙(초저지연형)** | Workers가 **R2/KV**에서 미리 계산한 예측/링크-코스트를 읽어 **Cache API**로 수십 초 TTL 서빙(필요 시 사용자 특성만 실시간 반영). ([Cloudflare Docs](https://developers.cloudflare.com/r2/api/workers/workers-api-reference/)) | **EventBridge + Lambda**가 1–5분 간격으로 서울시 실시간 데이터를 수집→**예측을 선계산**해 S3/R2 저장. 무거운 모델은 **Lambda 컨테이너(≤10GB) + /tmp≤10GB**로 처리. ([AWS Docs](https://docs.aws.amazon.com/lambda/latest/dg/images-create.html)) | 같은 소스. **GTFS-RT 90초 이내 최신성** 기준으로 배치 주기 설정(TripUpdates/VehiclePositions). ([gtfs.org](https://gtfs.org/documentation/realtime/realtime-best-practices/)) | API Gateway 호출이 **거의 불필요**(캐시/정적 응답). P95 수십 ms. | **응답 시간 최저**, **콜드스타트 영향 최소**. 비용/안정성 우수 | 최신성-지연 **트레이드오프**(분단위 배치), 사전계산 실패 시 공백 | **예측 안정적·재사용성 높을 때**, 대규모 트래픽/초저지연 |
| **H3. 모드 분리: (차량=온디맨드) + (대중교통=사전예측) + 엣지 라우팅** | Workers가 **차량**은 링크-코스트 조회 후 \**시간의존 최단경로(A*/Dijkstra)\*\*를 호출·중계, **대중교통**은 사전예측된 도착/헤드웨이와 **RAPTOR/CSA** 라우팅을 실행(엣지 JS 라이브러리 활용 가능). ([arXiv](https://arxiv.org/pdf/1705.02044" \o "A Survey of Shortest-Path Algorithms)) | Lambda는 차량용 \*\*경량 추론(GBDT/ONNX)\*\*을 온디맨드 수행, 대중교통은 **DeepAR/TFT**로 도착·헤드웨이 **사전예측**만 담당(라우팅은 엣지). ([Microsoft](https://www.microsoft.com/en-us/research/wp-content/uploads/2012/01/raptor_alenex.pdf)) | 같은 소스 + **GTFS-RT 모범 사례** 준수(TripUpdates/VehiclePositions 신선도 90초, Alerts 10분). ([gtfs.org](https://gtfs.org/documentation/realtime/realtime-best-practices/)) | 차량: 수백 ms, 대중교통: 엣지 라우팅으로 수십~수백 ms | **모드별 최적화**(경로탐색까지 엣지 처리→지연 최소) | 엣지 메모리/복잡도↑, 라우팅 정확도 검증 필요 | **사용자 체감 지연 최우선**, 도심 고QPS/피크 대응 |