Peak Hours Dispatch — BMTC 365J

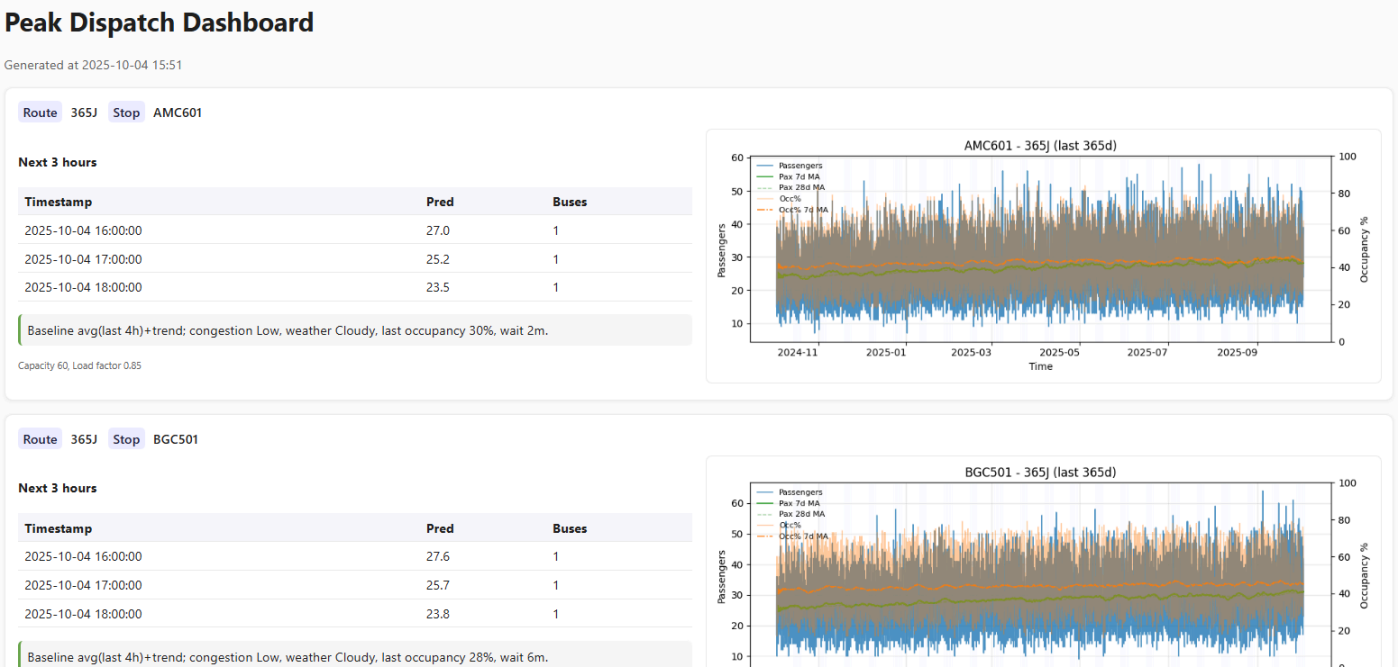
A fast, explainable peak-dispatch system for bus operations. It generates next 3-hour passenger forecasts, recommended bus counts, reasons, and a dashboard with full-year trend graphs. Trained GRU/LSTM model artifact is provided for judges, while the live demo uses a lightweight baseline for instant results.

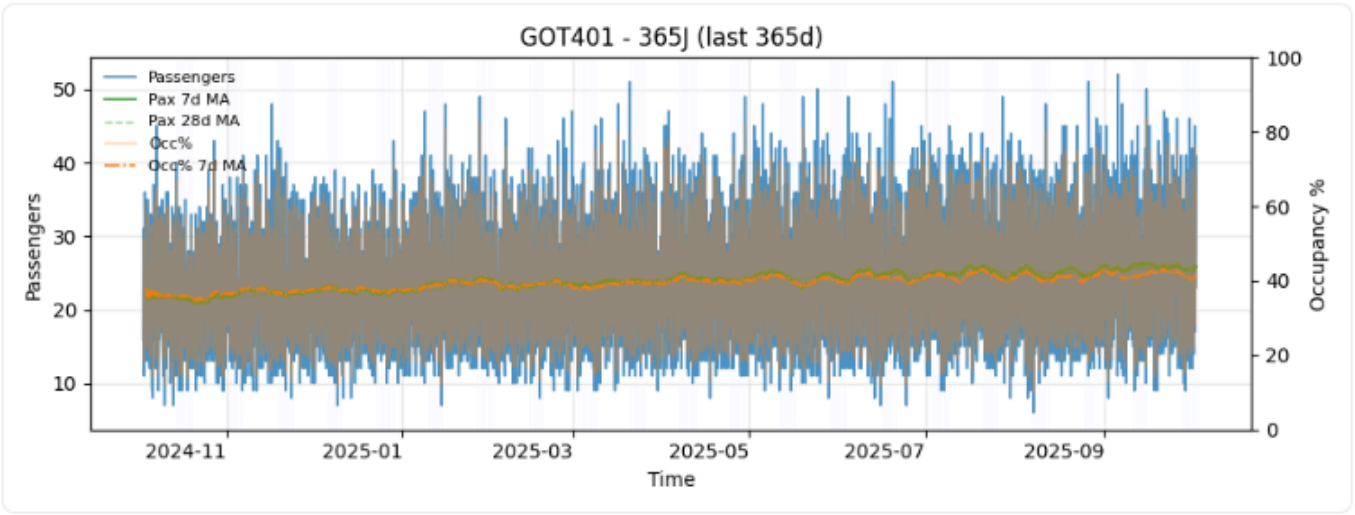
Quick links

LST Peak Hour Model :
https://drive.google.com/file/d/1PLmF2YjWLbb5e9CzRtfgR2yu4PnFrTIC/view?usp=sharing

Demo video : 🌐 LSTM-Based Peak Hour Prediction for BMTC – Next 3 Hours

Demo screenshots :





Route 365J Stop GOT401

Next 3 hours

Timestamp	Pred	Buses
2025-10-04 16:00:00	24.5	1
2025-10-04 17:00:00	22.2	1
2025-10-04 18:00:00	20.0	1

Baseline avg(last 4h)+trend; congestion Low, weather Cloudy, last occupancy 24%, wait 4m.

Capacity 60, Load factor 0.85

Folder structure

.venv/ — Python virtual environment

peak_hours_1y.csv — one-year dataset (or peak_hours.csv)

run_peak_dispatch_fast.py — fast inference (no training)

make_history_plots.py — export per-stop full-year charts

build_peak_dashboard.py — renders dashboard_peak.html

dispatch_next_3h_explained.csv — next 3-hour results with reasons

plots/ — generated charts (PNG per route/stop)

dispatch_seq.keras — GRU model artifact (optional, for judges)

quick_train_and_save.py — small GRU trainer to reproduce the model

requirements.txt — minimal dependencies (pandas, matplotlib, tensorflow-lite acceptable if used)

Prerequisites

Windows 10/11

Python 3.11

PowerShell

Optional GPU not required; CPU is fine

Create and activate venv (if not already)

From the peak-hours folder:

python -m venv .venv

..venv\Scripts\Activate.ps1

pip install -r requirements.txt

How to run :

From C:\Users\HP\Downloads\peak-hours with venv active:

Fast dispatch (creates CSV with reasons)

python .\run_peak_dispatch_fast.py

Charts (full 365 days with smoothing and downsampling)

python .\make_history_plots.py

Build dashboard

python .\build_peak_dashboard.py

Open dashboard

Start-Process .\dashboard_peak.html

Tip: PowerShell requires running scripts via python or .\script.py. If a command “is not recognized,” prefix with python .\script.py.

What the outputs mean

dispatch_next_3h_explained.csv

Pred_t+1/2/3: forecasted passengers next 1/2/3 hours

Buses_t+1/2/3: recommended bus count using capacity × load factor

Dispatch_Reason: textual explanation including congestion, weather, occupancy, wait, and peak-hour override

plots/history_{STOP}_{ROUTE}.png

Full-year passengers line (downsampled), 7-day and 28-day moving averages, occupancy trend, and light peak-hour shading

dashboard_peak.html

Per-stop cards with the next 3-hour table, reasons, and the corresponding chart

Live demo script choices

Fast pipeline (recommended for judging/demo):

run_peak_dispatch_fast.py → make_history_plots.py → build_peak_dashboard.py

Runs in seconds on CPU

Neural model artifact (for judges):

quick_train_and_save.py saves dispatch_seq.keras (GRU on a 90-day slice for a busy stop)

Train and export a small GRU model:

python .\quick_train_and_save.py

Output: dispatch_seq.keras

Notes: Uses engineered time/categorical features and a 48-step window to predict 3 steps ahead; small epoch count for speed.

Configuration notes

Dataset filenames:

Keep either peak_hours_1y.csv or peak_hours.csv in the folder

Paths:

Scripts use absolute paths or local relative paths to avoid current-directory issues

Charts clarity:

Current plots use downsampling and 7d/28d moving averages over 365 days with peak-hour shading for readability

To switch to 90 or 30 days, adjust the cutoff in make_history_plots.py

Troubleshooting

“No dataset found” but file exists:

Ensure the working folder is peak-hours: pwd should show C:\Users\HP\Downloads\peak-hours

Run via python .\script.py, not just script name

If needed, set absolute CSV paths inside scripts (already provided in final versions)

“Command not recognized”:

Use python .\run_peak_dispatch_fast.py

Dashboard shows “No plot”:

Ensure plots/ contains history_{STOP}_{ROUTE}.png

Re-run make_history_plots.py, then build_peak_dashboard.py

Performance profile

Fast path: seconds on CPU for 7 stops × 1 route

GRU quick-train: under a minute for one busy stop (small epochs)

Tech stack

Data processing: pandas

Visualization: matplotlib

Model (optional): TensorFlow GRU

Frontend: static HTML + embedded PNG charts

Demo assets (replace with Drive links)