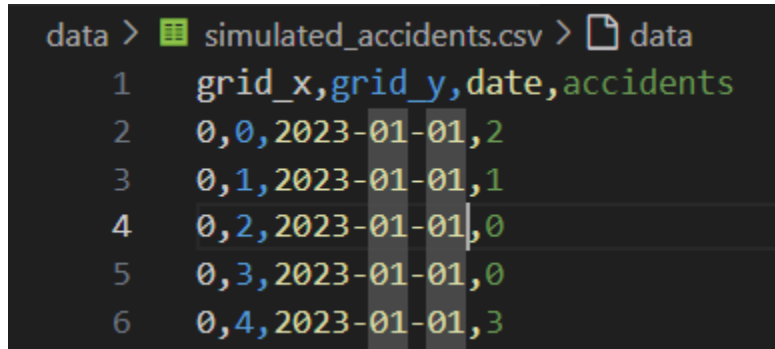


Road Accident Forecasting Sample

Data



```
data > simulated_accidents.csv > data
```

	grid_x	grid_y	date	accidents
1	0	0	2023-01-01	2
2	0	1	2023-01-01	1
3	0	2	2023-01-01	0
4	0	3	2023-01-01	0
5	0	4	2023-01-01	3
6				

Scripts

1. generate_data.py

This Python script simulates road accident data for a 10×10 spatial grid over a period of 60 days and saves the results to a CSV file. Each grid cell represents a specific region (such as 1 km²), and for each day, the script generates a random number of accidents using a Poisson distribution with a randomly selected average between 0.1 and 1.5. The simulation loops through each day and each grid cell, records the accident count, and stores the information (grid coordinates, date, and accident count) in a structured table using pandas. This data is then exported to data/simulated_accidents.csv and can be used for training spatio-temporal machine learning models like Hetero-ConvLSTM. The script is designed to execute only when run directly, making it easy to integrate into larger forecasting projects.

2. model_hetero_conv lstm.py

The scripts/model_hetero_conv lstm.py script defines the Hetero-ConvLSTM model, which is designed to learn spatio-temporal patterns in grid-based accident data. This model combines an LSTM layer and a fully connected (Linear) layer to predict accident risks across a 10×10 spatial grid. The LSTM is used to model the temporal dependencies by processing sequences of past accident data, where each time step represents a flattened version of the grid (i.e., 100 values for a 10×10 area). After the LSTM processes the sequence, only the output from the final time step is passed into a fully connected layer, which maps the hidden state to a vector of 100 values—representing predicted accident

risk across all grid cells. This output is then reshaped back to a 2D grid of shape $[B, 10, 10]$, matching the original spatial layout. The script is not executed directly but is instead imported by the training and visualization scripts that utilize the model for forecasting and generating predictions.

3. `train_grid.py`

This training script is designed to simulate input data, train the `HeteroConvLSTM` model, and save the trained model for later use. It begins by generating random tensors to simulate accident data over a 10×10 grid for 7 consecutive days, representing the input, and a corresponding tensor for the target output (accident risk for the next day). The model is initialized using the custom `HeteroConvLSTM` class, and the Mean Squared Error (MSE) loss function is used to measure how close the predictions are to the actual values. The Adam optimizer is used to update the model's parameters with a learning rate of 0.001. The script runs for 20 training epochs, during which it performs forward passes, computes loss, backpropagates errors, updates weights, and prints the training loss for each epoch. After training, the model's parameters are saved to a file named `hetero_conv_lstm.pth` inside the outputs folder, enabling future evaluation or visualization without retraining.

Mean Squared Error (MSE) Loss is a common loss function used in regression problems. It measures the average of the squares of the errors—that is, the difference between the predicted values and the actual values.

Formula:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- y_i = actual value
- \hat{y}_i = predicted value
- n = number of data points

Key Points:

- **Punishes larger errors more** due to the squaring.
- **Always non-negative**; lower MSE means better model performance.
- Common in **regression tasks**, like predicting temperature, stock prices, etc.

Training epochs refer to the number of times the entire training dataset is passed forward and backward through the model during training.

- **1 epoch** = model sees **all** training data **once**.
 - If you train for 10 epochs, the model sees the **entire dataset 10 times**.
 - A single pass might not be enough to learn patterns.
 - Multiple epochs help the model gradually reduce the loss and improve accuracy.
 - Too few epochs → **Underfitting** (model hasn't learned enough).
 - Too many epochs → **Overfitting** (model memorizes data, fails to generalize).
- You usually tune the number of epochs using **validation loss or early stopping**.

4. visualize_predictions.py

This code is used to **visualize the accident risk forecast** made by a trained HeteroConvLSTM model. It begins by importing the necessary libraries—PyTorch for handling the model and tensors, and Matplotlib for plotting the heatmap. The script then initializes the HeteroConvLSTM model and loads its previously trained weights from the `outputs/hetero_convlstm.pth` file. The model is set to evaluation mode to ensure consistent behavior during inference. A random input tensor representing 7 days of accident data over a 10×10 grid is created and passed through the model to generate a prediction. The output is detached from the computation graph and converted to a NumPy array for plotting. Finally, the prediction is visualized as a heatmap using Matplotlib, with color intensity indicating the predicted accident risk in each grid cell. The heatmap is saved as an image file (`predicted_heatmap.png`) and displayed to the user, providing a spatial overview of areas with higher or lower forecasted accident risk.