Absolutely! Let me walk you through how to **integrate TensorFlow.js into your Angular Dijkstra visualizer** to dynamically adjust edge weights—such as simulating weather impacts like snow slowing down certain routes.

🧩 Goal

Enhance your Angular Dijkstra visualizer by:

- Using a **TensorFlow.js model** to predict weight adjustments on each edge based on real-world features (e.g. temperature, snow, traffic).
- Feeding those adjusted weights into your Dijkstra algorithm before recalculating the path.

🛠️ 1. Install TensorFlow.js

In your Angular project folder:

```
npm install @tensorflow/tfjs
```

🧠 2. Example TensorFlow Model (Mock)

Let's assume you have a small model that accepts inputs like:

- temperature
- isSnowing (0 or 1)
- trafficLevel

And outputs a weight multiplier (e.g., 1.5 means 50% longer travel time).
If you don't have one trained, we'll mock this model for now using a simple function, but leave space for TensorFlow integration.

📦 3. Create a WeightService

Generate a service:

```
ng generate service weight
```

In weight.service.ts:

```
import { Injectable } from '@angular/core';
import * as tf from '@tensorflow/tfjs';

@Injectable({
  providedIn: 'root',
})
export class WeightService {
  // Load a real model in a real case
  async loadModel(): Promise<void> {
    // Example if you had a real model:
    // this.model = await tf.loadLayersModel('assets/model.json');
  }

  // This function mocks prediction
  predictWeightMultiplier(temp: number, snow: boolean, traffic: number):
number {
    if (snow) return 2.0;
    if (traffic > 0.7) return 1.5;
    if (temp < 0) return 1.3;
    return 1.0;
  }
}
```

🧮 4. Use TensorFlow Predictions in Your Grid Component

Let’s assume your grid is a 2D array and edges are just between neighboring cells. Wherever you calculate the weights for Dijkstra, modify like this:

In grid.component.ts:

```
import { WeightService } from '../weight.service';

constructor(private weightService: WeightService) {}

getEdgeWeight(from: string, to: string): number {
  // Extract features (you'd get this from a data source in real use)
  const temp = -5;           // Example: freezing
  const isSnowing = true;    // Example: snowing
  const trafficLevel = 0.3;  // Light traffic

  const multiplier = this.weightService.predictWeightMultiplier(temp,
isSnowing, trafficLevel);
  const baseWeight = 1;

  return baseWeight * multiplier;
}
```

Then plug this into your graph's edge setup (e.g., graph.addEdge(from, to, weight)).

🖼️ 5. Reflect on Visualization

You can color or label affected edges in your grid using the weight multipliers:

```
getColorForWeight(weight: number): string {
  if (weight > 1.5) return 'darkred';
  if (weight > 1.2) return 'orange';
  return 'green';
}
```

Then apply ngStyle or ngClass in your template.

✅ Optional: Load a Real Model

If you have a .json TensorFlow.js model file:

```
this.model = await tf.loadLayersModel('assets/model.json');
const prediction = this.model.predict(tf.tensor2d([[temp, snow ? 1 : 0,
traffic]])) as tf.Tensor;
const multiplier = (await prediction.array())?.[0]?.[0];
```

Would you like me to generate a complete working Angular app (with mock model logic) or help you create a real TensorFlow model for this?