Absolutely! Let's enhance your Angular Dijkstra visualizer by integrating TensorFlow.js to dynamically adjust edge weights based on real-time data, such as weather conditions or traffic levels.

🧠 Objective

Modify your existing Angular Dijkstra visualizer to:

- **Integrate TensorFlow.js**: Use a pre-trained machine learning model to predict edge weight adjustments.
- **Adjust Edge Weights**: Dynamically update edge weights based on real-time inputs (e.g., weather, traffic).
- **Visualize Changes**: Reflect these adjustments in your grid visualization.

🔧 Step 1: Install TensorFlow.js

In your Angular project directory, install TensorFlow.js:
npm install @tensorflow/tfjs

🧩 Step 2: Create a Weight Prediction Service

Generate a service to handle weight predictions:
ng generate service weightPrediction
In weight-prediction.service.ts:

```
import * as tf from '@tensorflow/tfjs';

@Injectable({
  providedIn: 'root',
})
export class WeightPredictionService {
  private model: tf.LayersModel;

  constructor() {
    this.loadModel();
  }

  private async loadModel() {
    this.model = await tf.loadLayersModel('assets/model.json');
  }

  public async predictWeight(features: number[]): Promise<number> {
```

```
    const inputTensor = tf.tensor([features]);
    const prediction = this.model.predict(inputTensor) as tf.Tensor;
    return prediction.arraySync()[0][0];
  }
}
```

🌦️ Step 3: Integrate Real-Time Data

Modify your grid component to adjust edge weights based on real-time data:

```
import { WeightPredictionService } from './weight-prediction.service';

@Component({
  selector: 'app-grid',
  templateUrl: './grid.component.html',
  styleUrls: ['./grid.component.css']
})
export class GridComponent {
  constructor(private weightPredictionService: WeightPredictionService) {}

  public async adjustEdgeWeight(from: string, to: string): Promise<void> {
    const features = this.extractFeatures(from, to);
    const adjustmentFactor = await
this.weightPredictionService.predictWeight(features);
    this.updateEdgeWeight(from, to, adjustmentFactor);
  }

  private extractFeatures(from: string, to: string): number[] {
    // Example features: [temperature, isSnowing, trafficLevel]
    return [5, 1, 0.7]; // Replace with actual data
  }

  private updateEdgeWeight(from: string, to: string, adjustmentFactor:
number): void {
    // Update the edge weight in your graph
  }
}
```

🖼️ Step 4: Visualize Adjustments

In your grid component's template, use Angular's binding mechanisms to reflect edge weight adjustments:

```
<div *ngFor="let edge of edges">
  <div [ngStyle]="{'background-color': getEdgeColor(edge.weight)}">
    <!-- Edge visualization -->
```

```
  </div>
</div>
```

In your component:

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

📚 Additional Resources

- Angular & TensorFlow.js Integration Tutorial
- Pathfinding Algorithm Visualizer with Angular

If you need further assistance with integrating real-time data sources or visualizing the adjusted paths, feel free to ask!