
# Greening Seattle

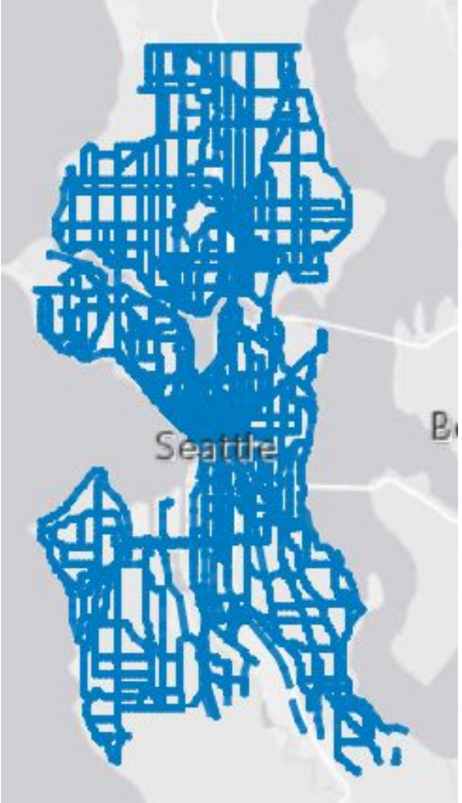
## *Part II: GIS Visualizations*

## Final Presentation

Sarah Pristash, Shaun Gallagher, Wenqi Cui

# Problems

- Transportation accounts for 60% of total core emissions in Seattle, 61% percent of which is attributed to gasoline/diesel sources
- Population increased 25% from 2008 - 2018, projected to continue and intensify city-wide traffic burden
- Citizen survey data indicates current transportation must be more robust and equitable, especially for BIPOC communities
- Large data sets make it difficult to visualize, and consistently report data




From Seattle.gov: 2018 Traffic Flow Counts

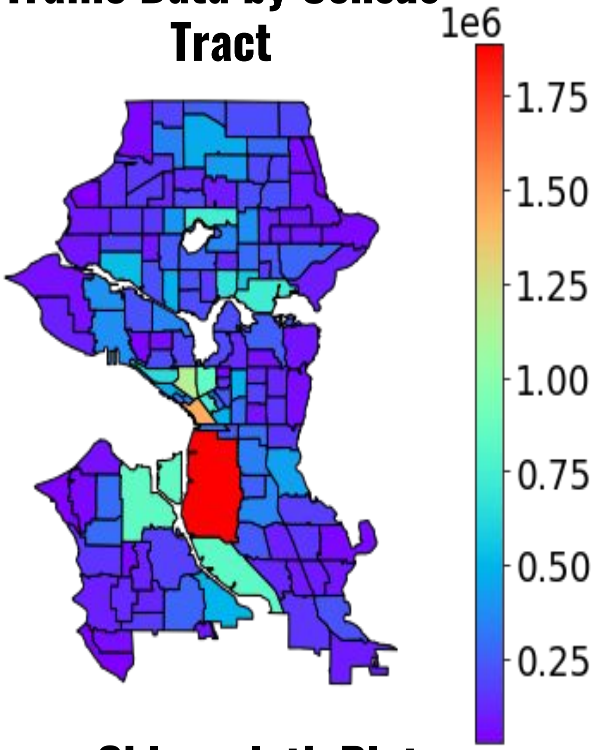
# Background and Scope

- Modeling the impact on traffic can give an idea of which features most affect flow

- Visualizing these changes regionally allow for a regional approach to discover the best solutions for specific areas of Seattle

- **Questions:**
  - How can we use data science to predict traffic volumes based on urban features?
  - How can data from traffic flow be effectively visualized?
  - On what length scale should we examine the data?

**2015 City of Seattle Traffic Data by Census Tract**




**Chloropleth Plot Generated in geoplot**

# Technology Overview: Geographic Information Systems (GIS)

ArcGIS

GeoPandas

Folium

- \+ Convenient Pandas-based package for geography
- \+ Interactive maps that interface well with data
- \- More difficult cleaning process

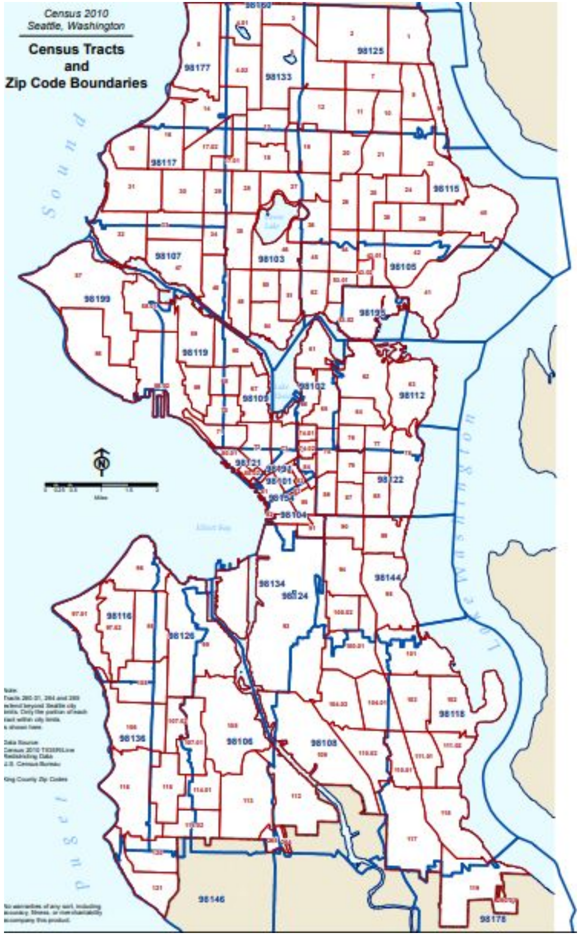
## Useful packages:

**Geopandas:** geographical data can be stored in Pandas-like dataframes. Shapely spatial functions can then be performed while maintaining the link to the data

**Shapely:** allows us to merge data based on geographic location - i.e. point in polygon, line intersecting polygon, etc.

**Folium:** Creates customizable, interactive maps of geospatial data

# Zip Codes as a Geographic Marker




➔ Looking at every individual street could obscure area-level trends, the same is true for a city-wide view

➔ Here several census tracts are contained within zip codes, we can sum up these data to get a representative intermediate-level pictures

➔ This can be accomplished through Shapely sjoin Feature, which can group the data sets into larger pre-defined features

# Data Cleaning: A geographic approach

**Objective:** Use spatial aspects of geographical data to organize input feature data into geographic locations

# Data Cleaning: A geographic approach

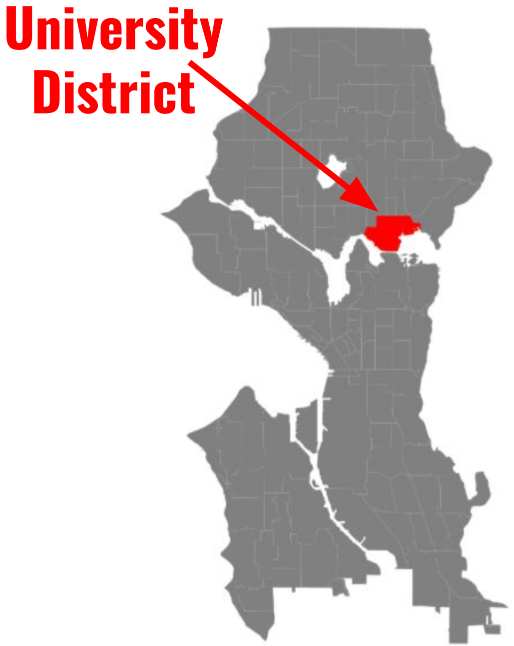
```
city_by_zip = gpd.sjoin(zip_bounds, gdf_15, op='intersects')
```

# Data Cleaning: A geographic approach

+

=

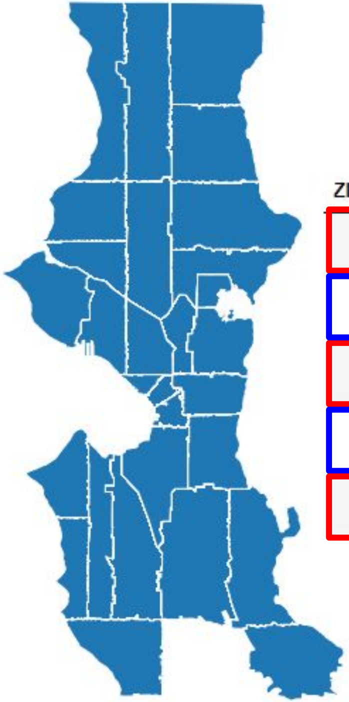
**Combined**




```
city_by_zip = gpd.sjoin(zip_bounds, gdf_15, op='intersects')
```

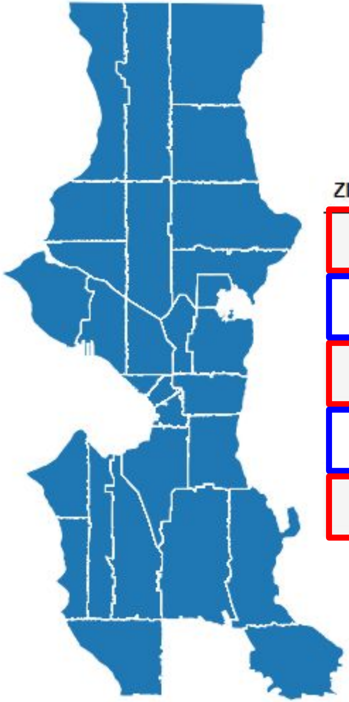
# Data Cleaning: The GeoData Frame

➔ Want to Aggregate data in GeoDataFrame by zip code using built-in geometry column



| ZIPCODE | geometry | NAME10 | SHAPE_Area_left | index_right | YEAR | AAWDT | GEOBASID | STNAME | SHAPE_Length |
|---|---|---|---|---|---|---|---|---|---|
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1163 | 2015 | 10639.335836 | 800.0 | ALASKAN WAY | 1245.144941 |
| 98121 | POLYGON ((-122.36110 47.61854, -122.36095 47.6... | 81.00 | 1.225219e+07 | 1163 | 2015 | 10639.335836 | 800.0 | ALASKAN WAY | 1245.144941 |
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1186 | 2015 | 2500.000000 | 823.0 | VIRGINIA ST | 372.729354 |
| 98121 | POLYGON ((-122.36110 47.61854, -122.36095 47.6... | 81.00 | 1.225219e+07 | 1186 | 2015 | 2500.000000 | 823.0 | VIRGINIA ST | 372.729354 |
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1197 | 2015 | 11000.000000 | 705.0 | ALASKAN WAY | 613.438180 |

# Data Cleaning: The GeoData Frame



| ZIPCODE | geometry | NAME10 | SHAPE_Area_left | index_right | YEAR | AAWDT | GEOBASID | STNAME | SHAPE_Length |
|---|---|---|---|---|---|---|---|---|---|
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1163 | 2015 | 10639.335836 | 800.0 | ALASKAN WAY | 1245.144941 |
| 98121 | POLYGON ((-122.36110 47.61854, -122.36095 47.6... | 81.00 | 1.225219e+07 | 1163 | 2015 | 10639.335836 | 800.0 | ALASKAN WAY | 1245.144941 |
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1186 | 2015 | 2500.000000 | 823.0 | VIRGINIA ST | 372.729354 |
| 98121 | POLYGON ((-122.36110 47.61854, -122.36095 47.6... | 81.00 | 1.225219e+07 | 1186 | 2015 | 2500.000000 | 823.0 | VIRGINIA ST | 372.729354 |
| 98101 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 74.02 | 1.470012e+07 | 1197 | 2015 | 11000.000000 | 705.0 | ALASKAN WAY | 613.438180 |

```
traffic_zones = city_by_zip.dissolve(by='ZIPCODE', aggfunc = sum)
```

# Data Cleaning: The GeoData Frame

| | NAME10 | geometry | AAWDT |
|---|---|---|---|
| ZIPCODE | | | |
| **98101** | 13323.60 | POLYGON ((-122.34598 47.60892, -122.34490 47.6... | 1.877668e+06 |
| **98102** | 2738.74 | POLYGON ((-122.33574 47.64203, -122.33108 47.6... | 4.782904e+05 |
| **98103** | 5670.00 | POLYGON ((-122.35808 47.69966, -122.35741 47.6... | 1.880369e+06 |
| **98104** | 13944.00 | POLYGON ((-122.34105 47.59627, -122.34031 47.5... | 1.847450e+06 |
| **98105** | 6560.00 | MULTIPOLYGON (((-122.32859 47.66646, -122.3285... | 1.762661e+06 |

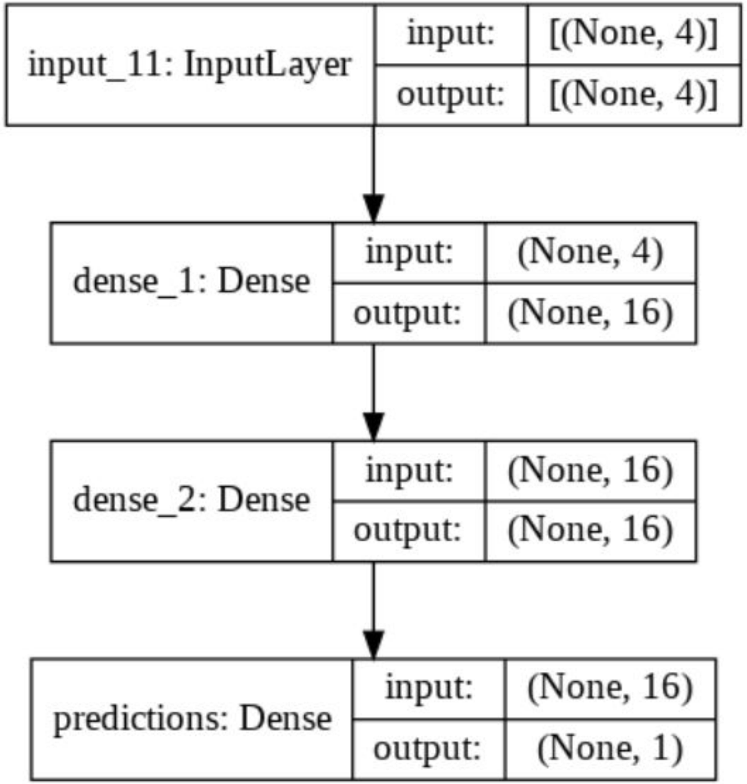
```
traffic_zones = city_by_zip.dissolve(by='ZIPCODE', aggfunc = sum)
```

# Map Demo: Interactive plots using folium

http://localhost:8888/view/map_html/choropleth_map_v3.html

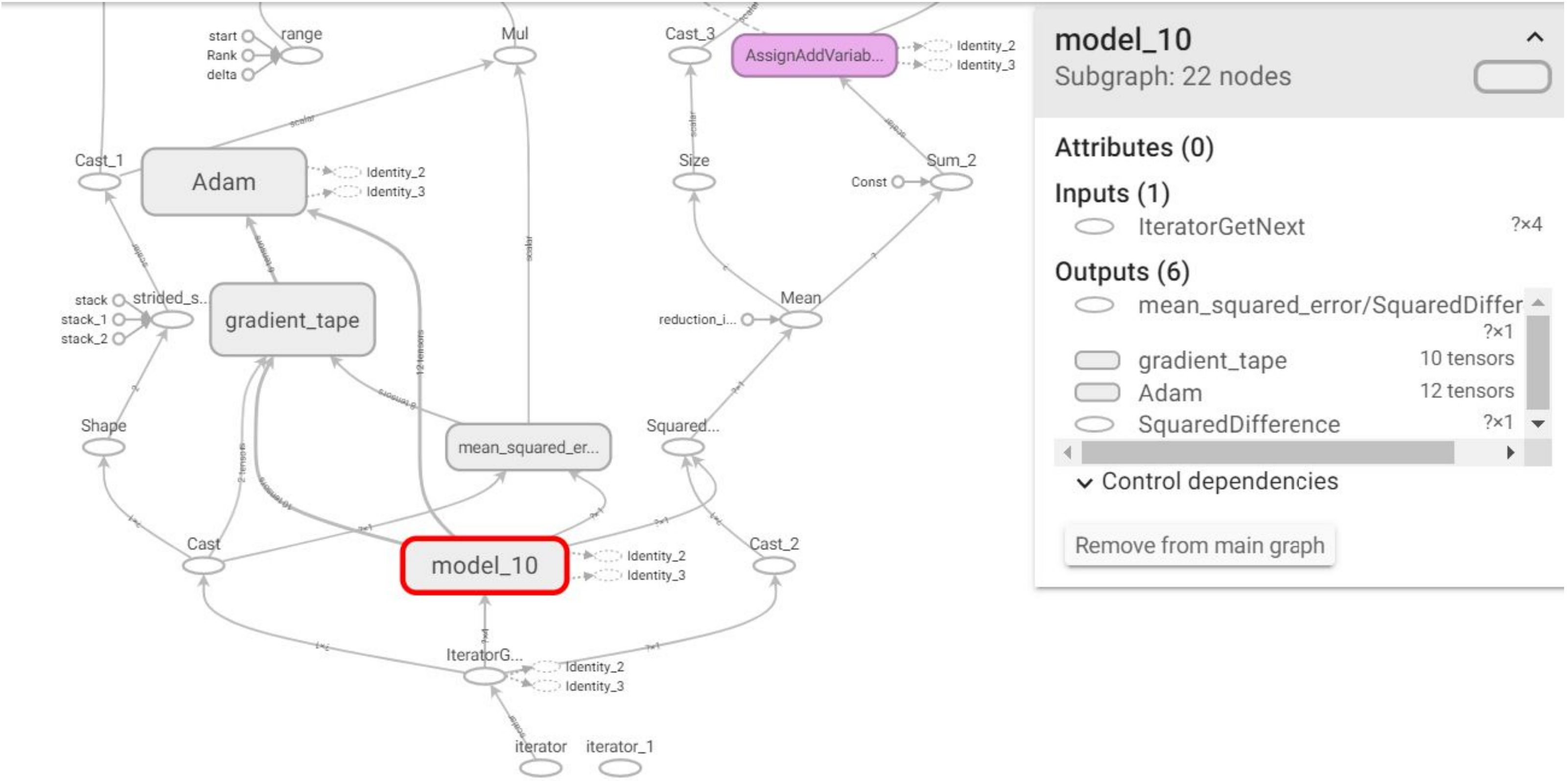
# Visualize the model: Green Seattle

- ➔ Feedback Loop: Clean data, send to prediction team, visualize data...
- ➔ Make use of the ‘keras.utils.plot_model’ function in Tensorflow
- ➔ The structure is dense neural network with two hidden layers, each has 16 neurons, with tanh activation

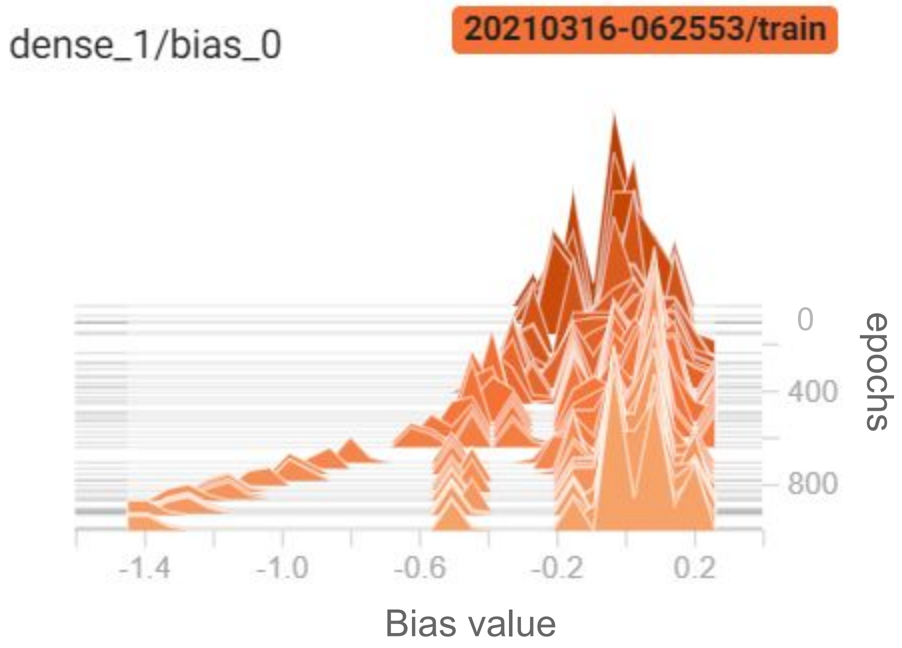


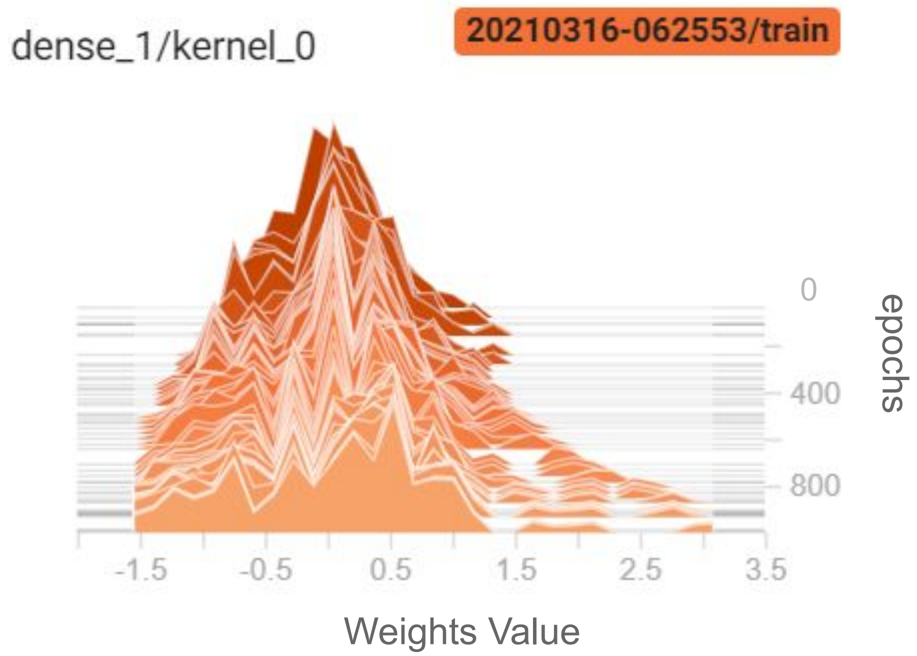
# Visualize the Training Structure

➔ Model Flow: input → Model → Loss → Gradient → Optimizer



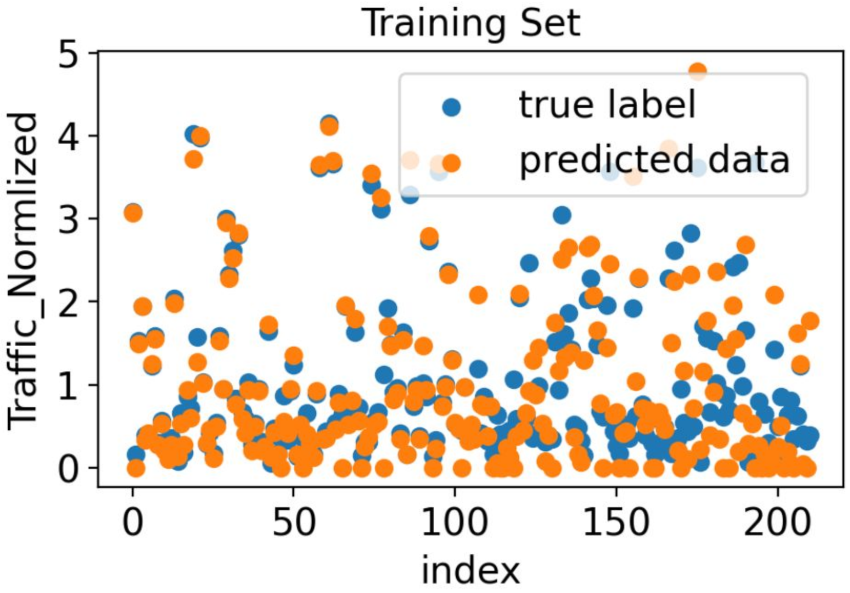
# Visualize the Weight and Bias in Training

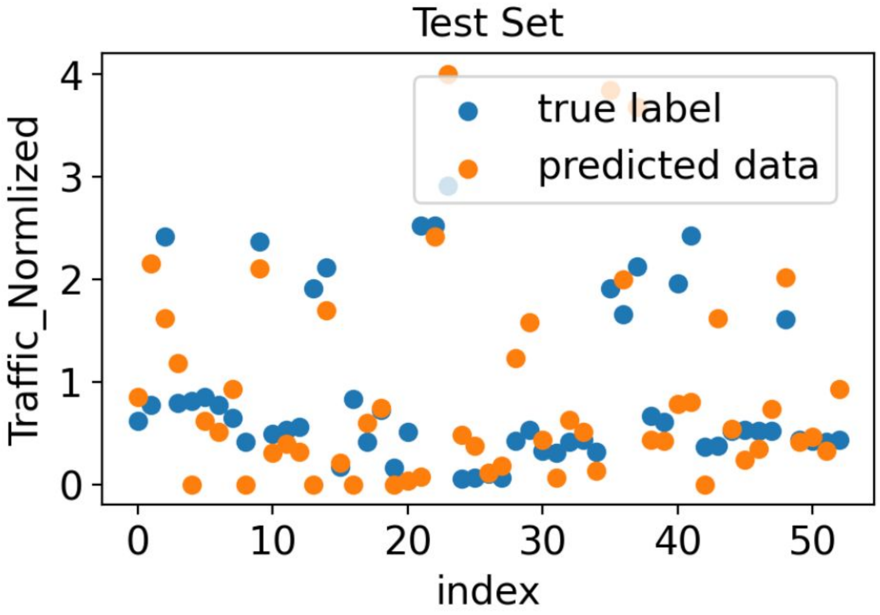



➔ Visualize the convergence of weights and bias of first hidden layer through training epochs

# Visualize the Accuracy of the Prediction Model

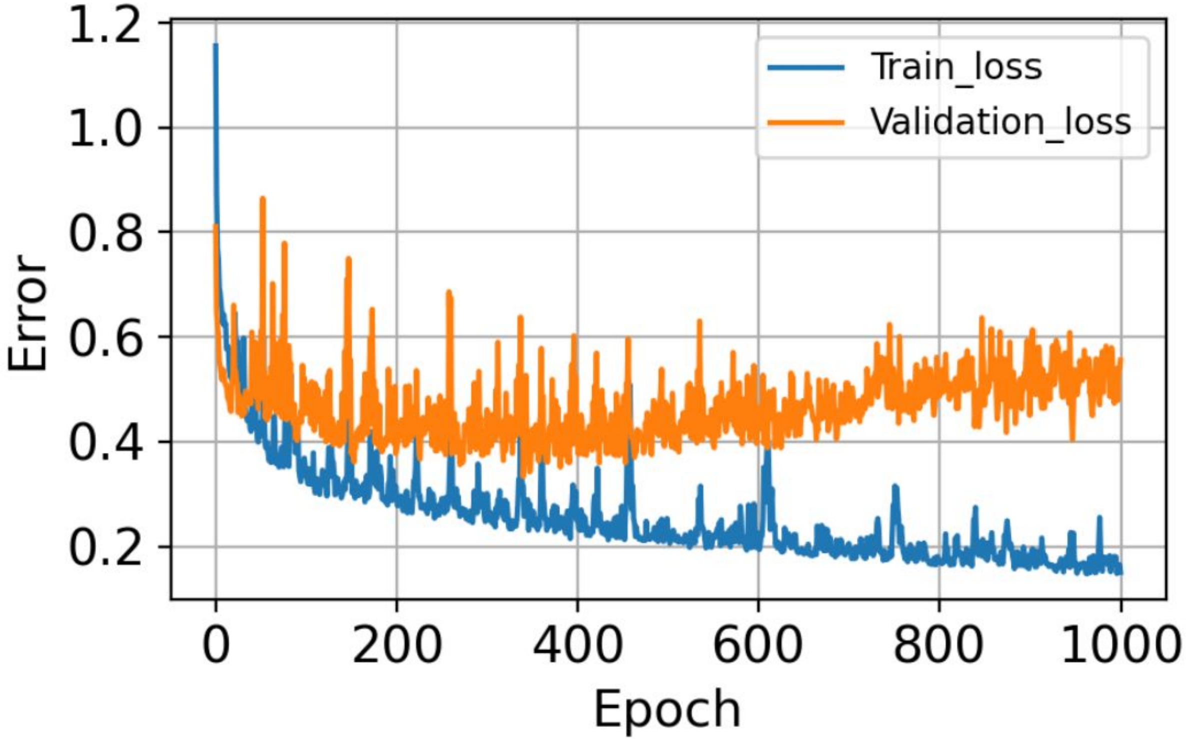



➔ Compare the true label and the predicted label in each sample

# Visualize the Loss Verses Epochs

- Identify the convergence of the training and the bias-variance trade-off



# Compare the error on the training set and the test set