AADT Prediction Model using PyTorch

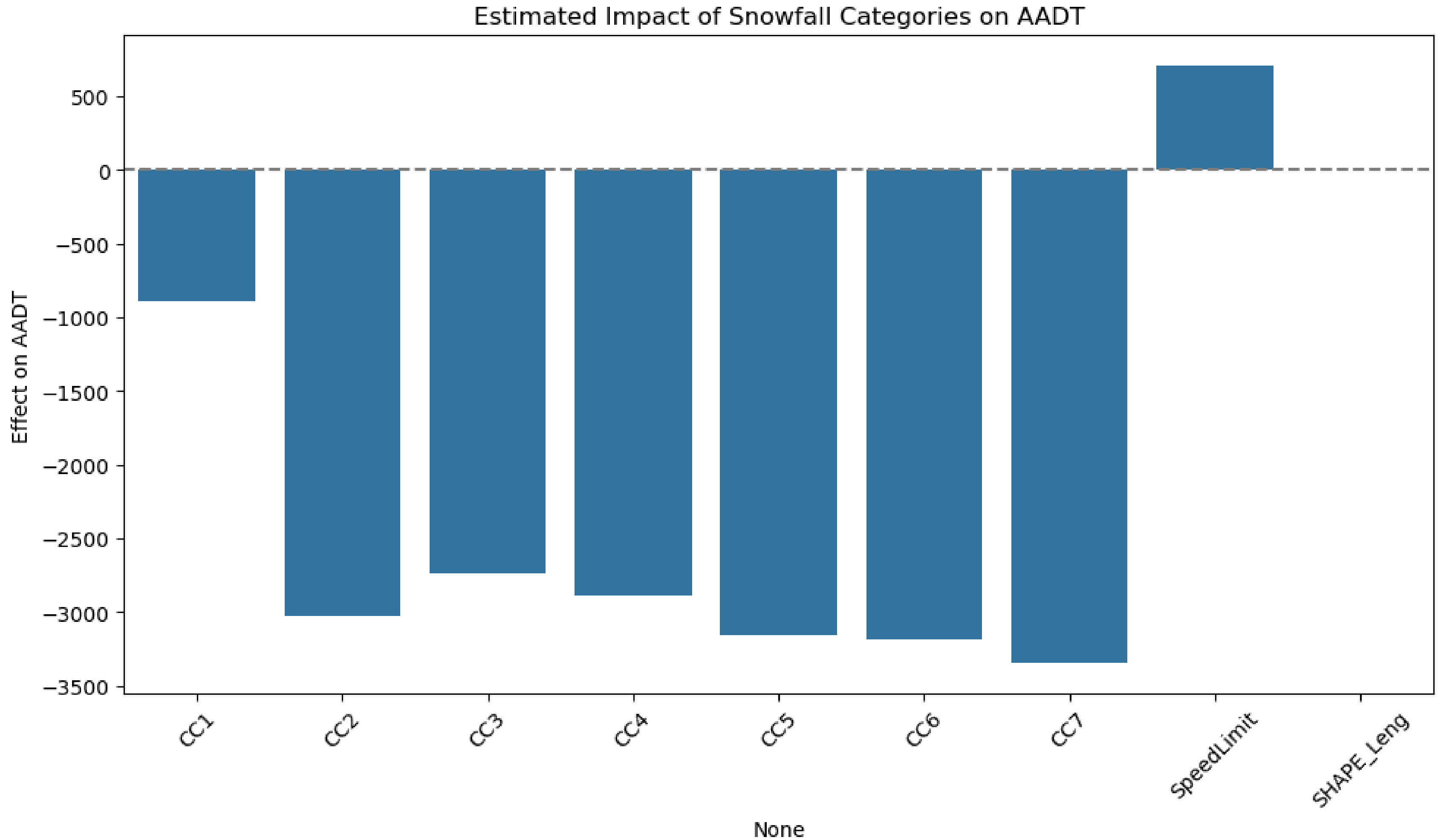
This notebook demonstrates how to use PyTorch to predict Annual Average Daily Traffic (AADT) using winter-related features.

In [22]:
```
import pandas as pd
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns



# 1. Load dataset
use_cols = ["AADT", "snow_depth", "SpeedLimit", "SHAPE_Leng"]
df = pd.read_excel("AADT_with_snow.xlsx", usecols=use_cols)
df = df.dropna()
```

In [24]:
```
# Step 1: Define snowfall category (0–5, 5–10, ..., 30–35 inches)
def snow_category(inches):
    if inches < 0: return "baseline"
    elif inches <= 5: return "CC1"
    elif inches <= 10: return "CC2"
    elif inches <= 15: return "CC3"
    elif inches <= 20: return "CC4"
    elif inches <= 25: return "CC5"
    elif inches <= 30: return "CC6"
    else: return "CC7"

# Assign snow categories
df["SnowCat"] = df["snow_depth"].apply(snow_category)

# Create dummy variables (CC1–CC7)
dummies = pd.get_dummies(df["SnowCat"])
```

```
# Replace 'baseline' with Speed_Limit in the design matrix
X = pd.concat([dummies.drop(columns=["baseline"], errors='ignore').astype(float), df[["SpeedLimit", "SHAPE_
X = sm.add_constant(X)
y = df["AADT"].astype(float)

# Fit OLS regression
model = sm.OLS(y, X).fit()
print(model.summary())

# 📊 Visualize coefficients
plt.figure(figsize=(10, 6))
sns.barplot(x=model.params.index[1:], y=model.params.values[1:])
plt.axhline(0, color='gray', linestyle='--')
plt.ylabel("Effect on AADT")
plt.title("Estimated Impact of Snowfall Categories on AADT")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   AADT   R-squared:                       0.104
Model:                            OLS   Adj. R-squared:                  0.104
Method:                 Least Squares   F-statistic:                     255.3
Date:                Mon, 05 May 2025   Prob (F-statistic):               0.00
Time:                        01:24:32   Log-Likelihood:            -1.9746e+05
No. Observations:               19740   AIC:                         3.949e+05
Df Residuals:                   19730   BIC:                         3.950e+05
Df Model:                           9
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -3.497e+04    995.210    -35.137      0.000   -3.69e+04    -3.3e+04
CC1         -889.4267    492.127     -1.807      0.071   -1854.037      75.183
CC2        -3024.0298    505.265     -5.985      0.000   -4014.392   -2033.668
CC3        -2740.9609    537.720     -5.097      0.000   -3794.938   -1686.984
CC4        -2884.1054    623.050     -4.629      0.000   -4105.336   -1662.875
CC5        -3153.8856   1052.245     -2.997      0.003   -5216.375   -1091.396
CC6        -3186.9728   1265.871     -2.518      0.012   -5668.186    -705.759
CC7        -3348.2282   3126.452     -1.071      0.284   -9476.337    2779.881
SpeedLimit   709.8023     15.756     45.049      0.000     678.919     740.686
SHAPE_Leng    -0.1078      0.011     -9.555      0.000      -0.130      -0.086
==============================================================================
Omnibus:                    21553.277   Durbin-Watson:                   1.187
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          2280616.626
Skew:                           5.518   Prob(JB):                         0.00
Kurtosis:                      54.488   Cond. No.                     4.20e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.2e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

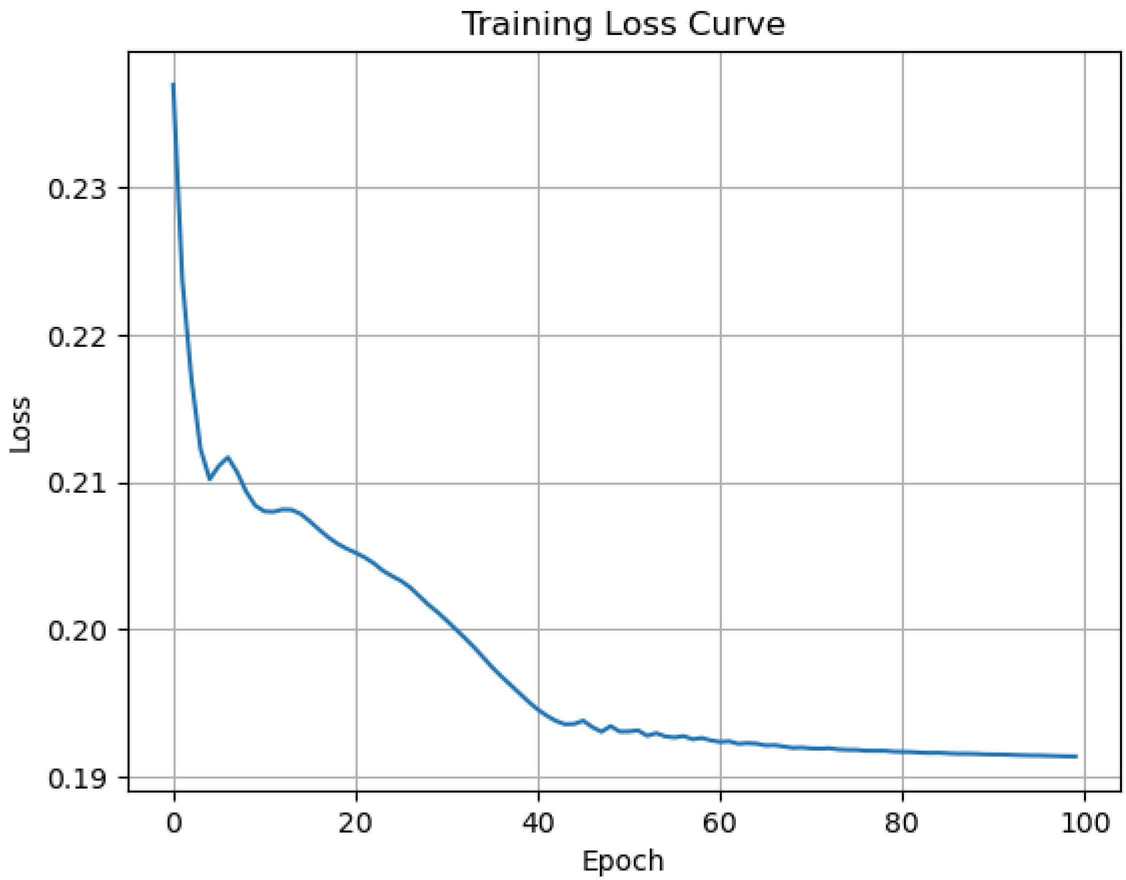


In [26]:
```
# Select feature matrix and label
torch_X = X.drop(columns="const").values  # drop intercept
scaler_X = StandardScaler()
scaler_y = StandardScaler()
X_scaled = scaler_X.fit_transform(torch_X)
y_scaled = scaler_y.fit_transform(y.values.reshape(-1, 1))

X_tensor = torch.tensor(X_scaled, dtype=torch.float32)
y_tensor = torch.tensor(y_scaled, dtype=torch.float32)

# 5. Train/test split
```

```python
X_train, X_test, y_train, y_test = train_test_split(X_tensor, y_tensor, test_size=0.2, random_state=42)

# 6. Define model
class AADTModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(X_tensor.shape[1], 64),
            nn.ReLU(),
            nn.Linear(64, 32),
            nn.ReLU(),
            nn.Linear(32, 1)
        )
    def forward(self, x):
        return self.net(x)

model = AADTModel()
criterion = nn.SmoothL1Loss()  # Huber Loss
optimizer = optim.Adam(model.parameters(), lr=0.01)

# === Training ===
loss_values = []
for epoch in range(100):
    model.train()
    optimizer.zero_grad()
    outputs = model(X_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    optimizer.step()
    loss_values.append(loss.item())
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch+1}/100, Loss: {loss.item():.4f}")

# Plot loss curve
plt.figure()
plt.plot(loss_values)
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training Loss Curve")
plt.grid(True)
plt.show()
```

```python
# Final test evaluation
model.eval()
with torch.no_grad():
    y_pred_test = model(X_test)
    test_loss = criterion(y_pred_test, y_test)
    print(f"Final Test Loss: {test_loss.item():.4f}")
```

```
Epoch 10/100, Loss: 0.2084
Epoch 20/100, Loss: 0.2055
Epoch 30/100, Loss: 0.2012
Epoch 40/100, Loss: 0.1951
Epoch 50/100, Loss: 0.1931
Epoch 60/100, Loss: 0.1925
Epoch 70/100, Loss: 0.1920
Epoch 80/100, Loss: 0.1917
Epoch 90/100, Loss: 0.1915
Epoch 100/100, Loss: 0.1914
```




```
Final Test Loss: 0.2116
```

In [ ]: