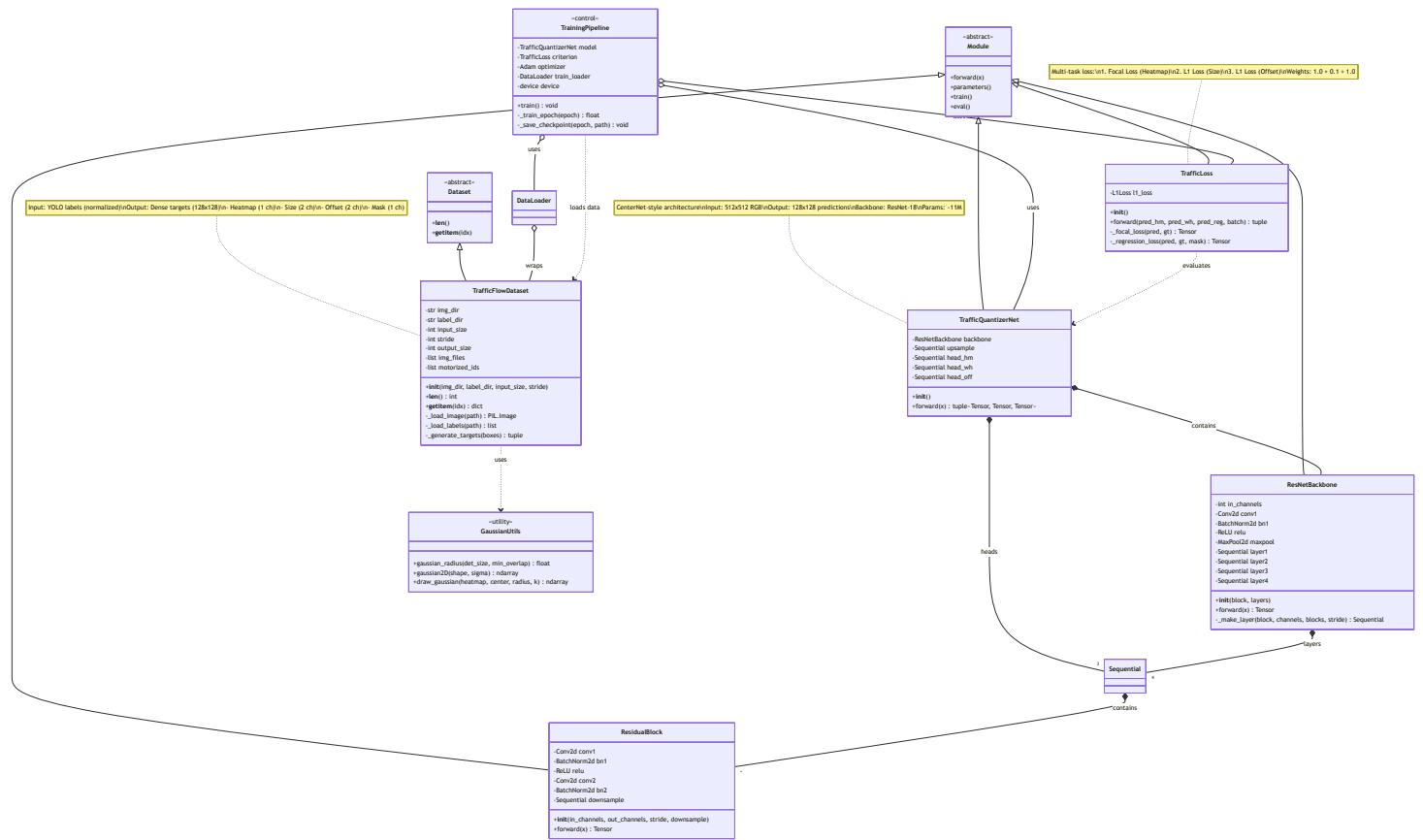
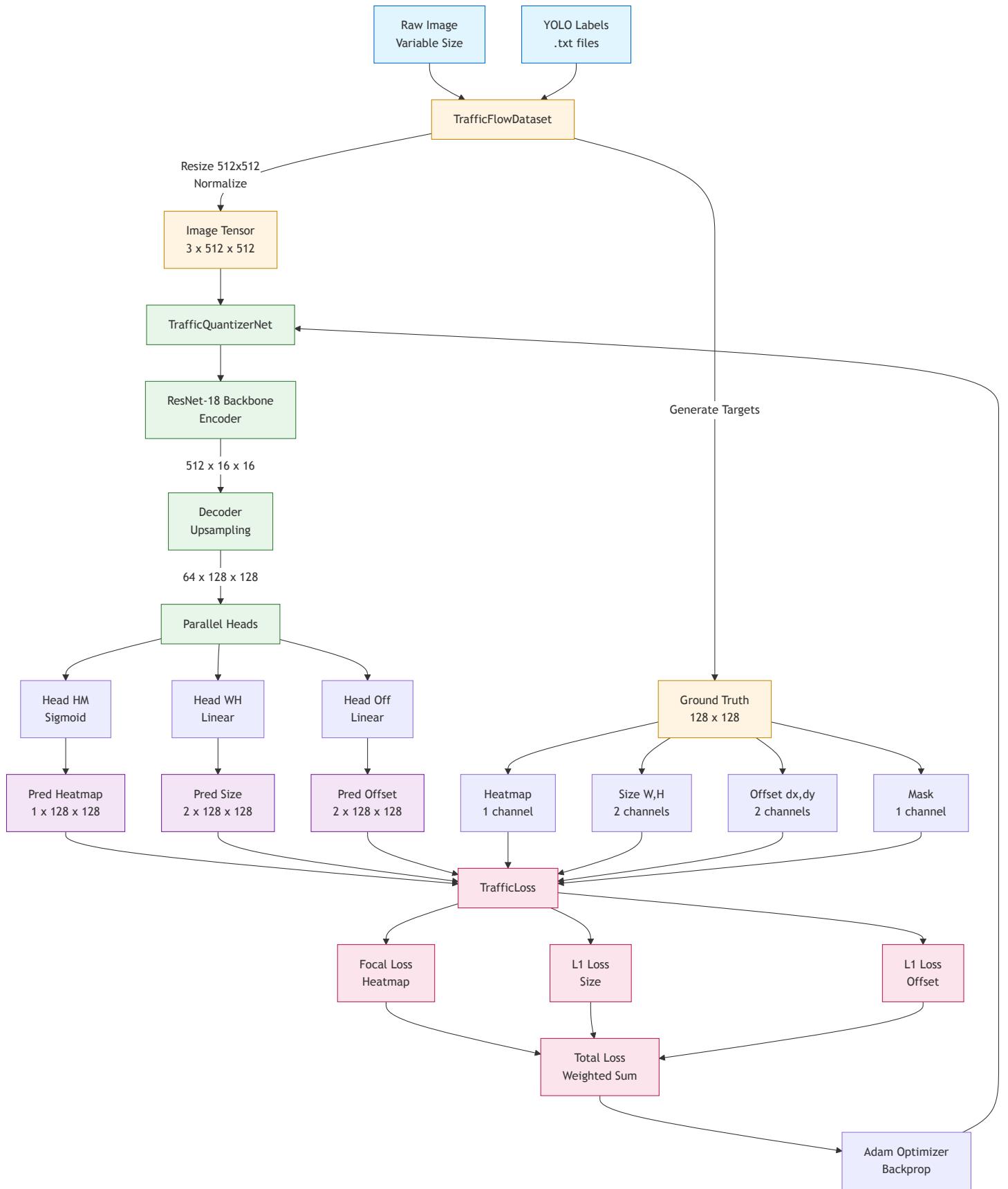


# Diagrama UML - TrafficQuantizerNet Architecture

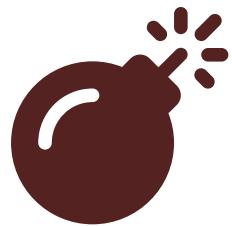
## Diagrama de Clases Completo



# Diagrama de Flujo de Datos

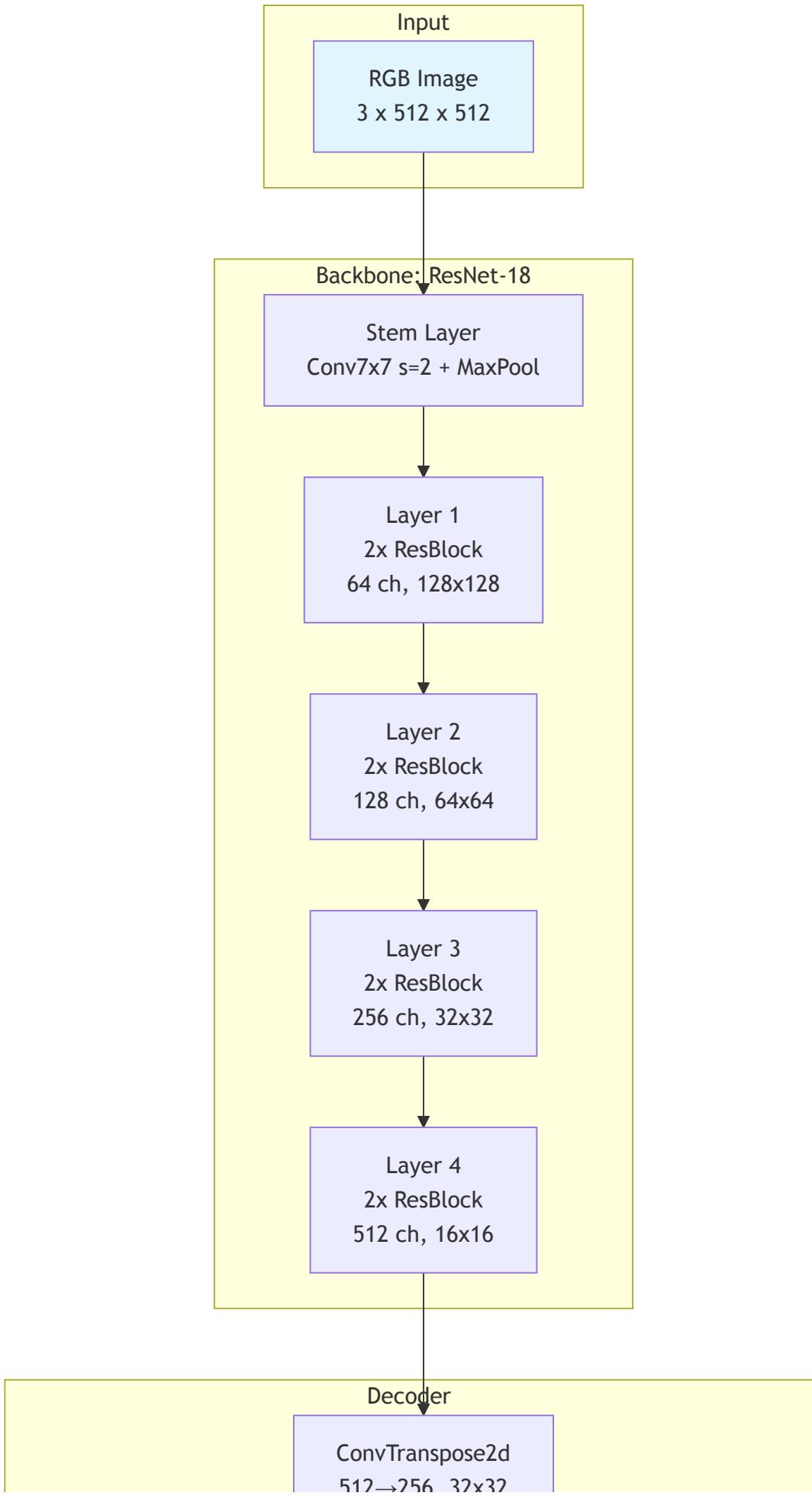


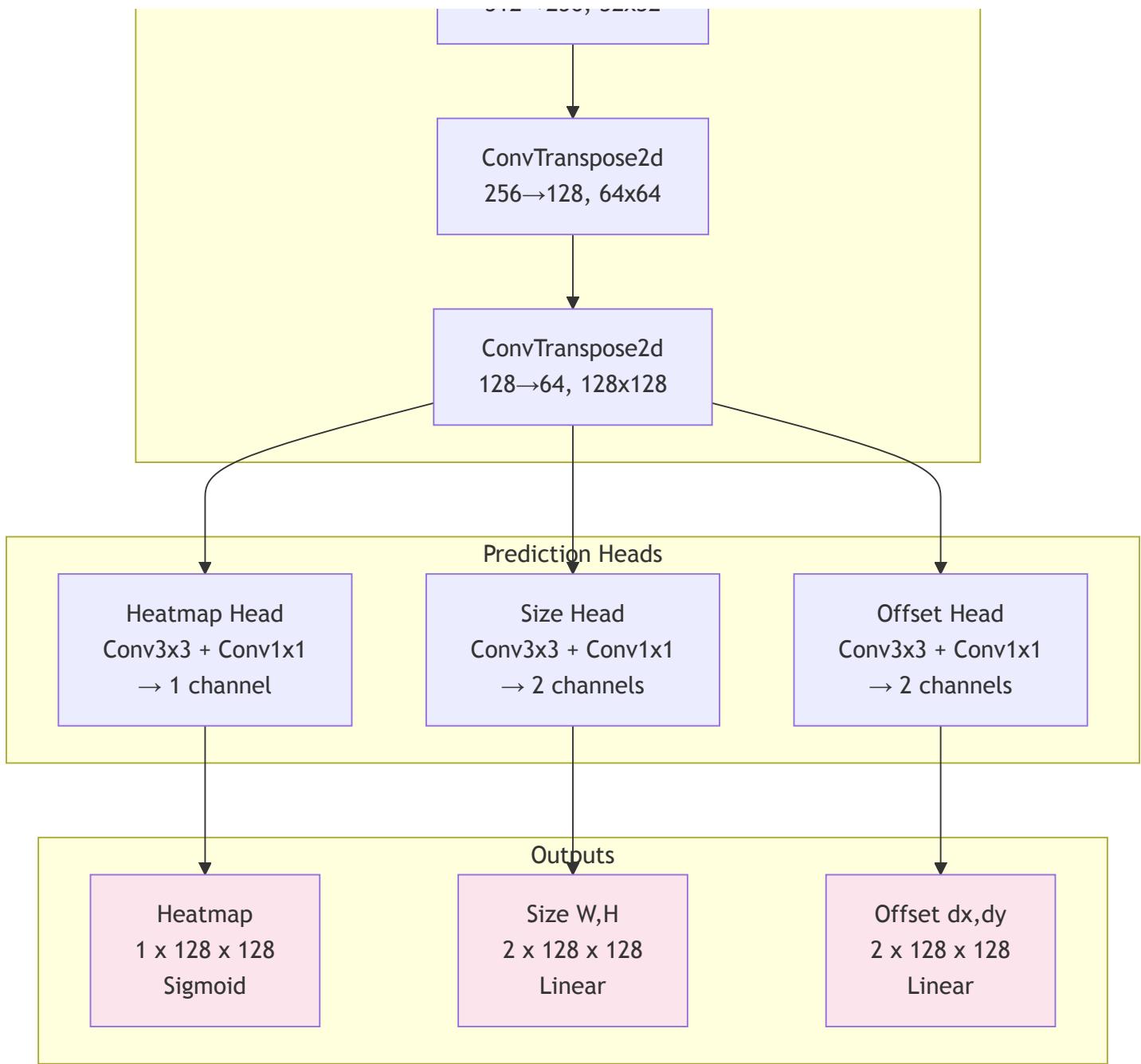
# Diagrama de Secuencia - Training Loop



Syntax error in text  
mermaid version 11.12.1

# Arquitectura del Modelo (Detallada)





## Componentes Clave

### 1. TrafficFlowDataset

- **Propósito:** Cargar imágenes y convertir anotaciones YOLO a targets densos
- **Entrada:** Imágenes + archivos .txt (YOLO format)
- **Salida:** Tensors con heatmaps gaussianos
- **Dependencias:** GaussianUtils para generar distribuciones

## 2. TrafficQuantizerNet

- **Propósito:** Modelo de detección end-to-end
- **Arquitectura:** Encoder-Decoder con múltiples cabezales
- **Backbone:** ResNet-18 (4 capas, 2 bloques c/u)
- **Decoder:** 3 capas de ConvTranspose2d
- **Heads:** 3 ramas paralelas para diferentes predicciones

## 3. TrafficLoss

- **Propósito:** Función de pérdida multi-tarea
- **Componentes:**
  - Modified Focal Loss ( $\alpha=2$ ,  $\beta=4$ )
  - L1 Loss para regresión de tamaño
  - L1 Loss para regresión de offset
- **Ponderación:** 1.0 : 0.1 : 1.0

## 4. GaussianUtils

- **Propósito:** Generar representaciones gaussianas de objetos
- **Funciones:**
  - gaussian\_radius() : Calcula radio adaptativo
  - gaussian2D() : Genera kernel gaussiano
  - draw\_gaussian() : Dibuja en heatmap con max-pooling

## Parámetros del Modelo

Componente	Parámetros	Salida
Stem	~9K	64 x 128 x 128
Layer 1	~148K	64 x 128 x 128
Layer 2	~525K	128 x 64 x 64
Layer 3	~2.1M	256 x 32 x 32
Layer 4	~8.4M	512 x 16 x 16
Decoder	~3.5M	64 x 128 x 128

Componente	Parámetros	Salida
<b>Heads</b>	~40K	5 x 128 x 128
<b>TOTAL</b>	<b>~11.4M</b>	-