ARCHITECTURE DIAGRAM FOR ROAD SIGN DETECTION FOR AUTONOMOUS VEHICLES BY USING CNN AND KERAS

**1. Data Ingestion**

* **Data Source**: Images of road signs from datasets like GTSRB.
* **Data Loader**: A module that reads and preprocesses the image data.

**2. Preprocessing**

* **Image Resizing**: Adjust images to a uniform size, typically 32x32 pixels.
* **Normalization**: Normalize pixel values to a range between 0 and 1.
* **Data Augmentation** (Optional): Apply transformations like rotation, zoom, and horizontal flips to increase data variety.
* **Data Splitter**: Split data into training, validation, and test sets.

**3. Modeling**

* **Model Architecture**:
  + **Input Layer**: Accepts the input image with a shape of (32, 32, 3).
  + **Convolutional Layers**: Extracts features from the images.
  + **MaxPooling Layers**: Reduces spatial dimensions.
  + **Dropout Layers**: Prevents overfitting by randomly setting some activations to zero during training.
  + **Fully Connected (Dense) Layers**: Interprets the features extracted by the convolutional layers.
  + **Output Layer**: Provides the final classification into road sign categories.
* **Training Module**:
  + **Loss Function**: Categorical cross-entropy, guiding the model to minimize classification error.
  + **Optimizer**: Adam, which adjusts model weights based on gradients.
  + **Hyperparameter Tuning**: Adjust learning rates, batch sizes, and the number of filters in convolutional layers.

**4. Evaluation**

* **Metrics Calculation**:
  + **Accuracy, Precision, Recall, F1-score**: Evaluate the model's classification performance.
  + **Confusion Matrix**: Visualizes classification errors for each road sign category.
  + **ROC-AUC Curve**: Assesses binary classification performance for each road sign.

**5. Model Interpretation**

* **Feature Maps Visualization**: Visualize the features learned by different layers of the CNN.
* **Layer Activation**: Understand how different layers activate for various road signs.

**6. Deployment**

* **Model Serving**: Deploy the trained model as an API or integrate it directly into the vehicle's perception system for real-time predictions.
* **User Interface (UI)**: (Optional) A dashboard for developers to input images and visualize predictions.
* **Real-time Processing**: Enable the model to process video streams or images captured in real time by the vehicle's cameras.

**7. Monitoring & Retraining**

* **Performance Monitoring**: Continuously monitor the model's accuracy on new, unseen data.
* **Data Drift Detection**: Identify changes in input image distributions over time.
* **Model Retraining Pipeline**: Automatically retrain the model if performance degrades significantly.

