Course Project

TRAFFIC SIGN DETECTION

Statistical Machine Learning

Annu Kumari | Anand Vimal

About The Project

Problem statement

Human error in recognizing or reacting to traffic signs is a significant contributing factor to traffic accidents which has become a major global concern. In this project, we aim to address this issue by exploring different ML models for traffic sign detection (TSD). By developing an accurate TSD system, we can improve road safety by:

- **Alerting drivers:** The system can provide real-time notifications about upcoming traffic signs, particularly when a driver might be distracted.
- **Enhancing autonomous vehicles:** TSD is a crucial component for self-driving cars to navigate roads safely by recognizing and interpreting traffic signals.

Literature review

Traditional methods for traffic sign detection struggled with real-world variations. Deep learning, particularly Convolutional Neural Networks (CNNs), have revolutionized the field. CNNs excel at recognizing signs despite variations in pose, lighting, and background clutter.

Pioneering works by Ji et al. (2012) and LeCun et al. (1998) paved the way for CNN-based traffic sign detection. Recent advancements include YOLO (Redmon et al., 2016) for real-time applications and Mask R-CNN (He et al., 2017) for potential future uses.

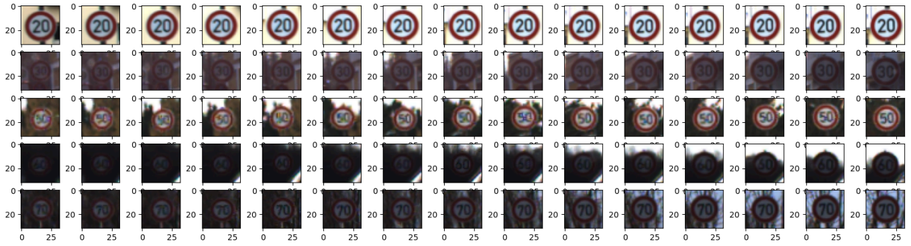
Challenges remain, including occlusions and real-time performance. Research is ongoing to address these issues for robust autonomous vehicle applications.

Dataset

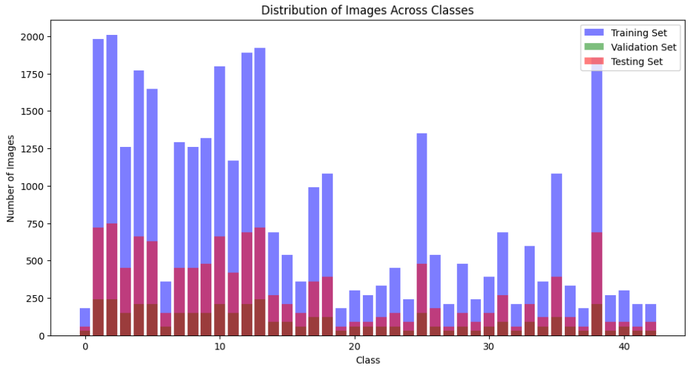
We utilized the widely used GTSRB (German Traffic Sign Recognition Benchmark) dataset.

- It contains over 50,000 images of 43 traffic signs with lighting, pose, and occlusion variations.
- The dataset is split into training, validation, and testing sets for model development and evaluation.
- We also used the Traffic sign dataset.

One image from each class



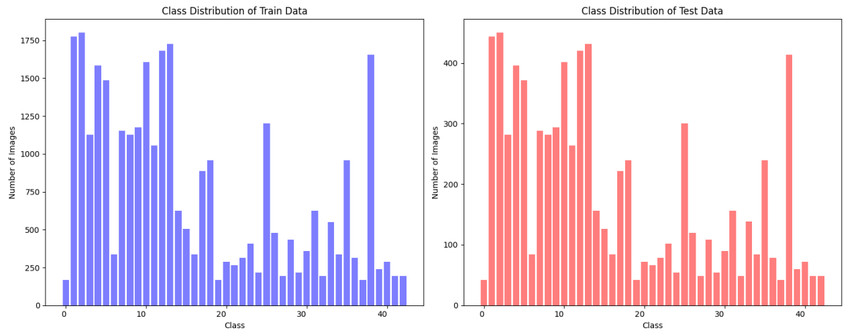
Multiple Image from same class



Dataset split

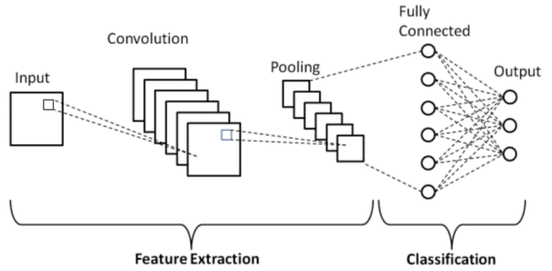
Data Preprocessing

- **Visualization:** Analyzed data to understand content.
- Background Removal: Removed background color for focus.
- **Grayscale Conversion & Resizing:** Converted to grayscale and resized to 16x16 pixels (reduced complexity).
- **Data Split:** Divided data into training (80%) and validation (20%) sets.
- **Dimensionality Reduction:** Tested autoencoder but accuracy suffered (potential overfitting or architecture issue).
- Opted for PCA as it captured key structures for classification while non-linear info from autoencoder might not be relevant.

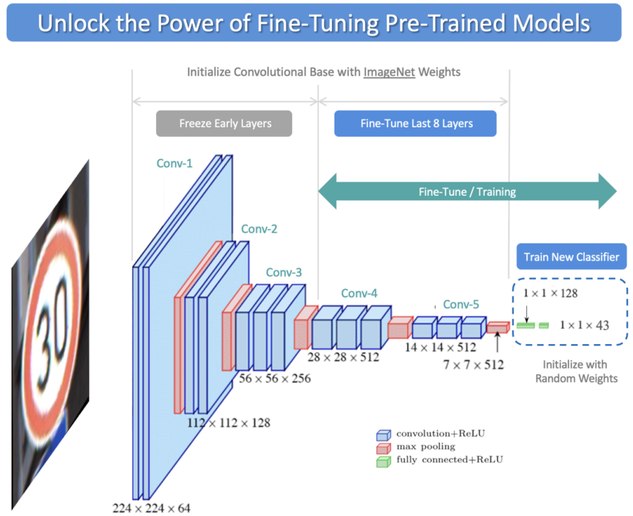


Model Classification

- **Started with LDA & QDA:** Easy to interpret, work well with lower dimensional data (GTSRB might have been pre-processed). But, their linear decision boundaries limited accuracy (71% & 64%).
- **Switched to Random Forest, KNN, Logistic Regression:** These models can handle complex data patterns, achieving higher accuracy (96%, 90%, 88%).
- **CNNs excel at images:** They learn features automatically, understand how pixels connect, and handle non-linear patterns (unlike KNN & Random Forest).
- **VGG16** - A Winning CNN Architecture:
  - Easier to train due to many small filters.
  - Captures complex features with its deep layers.
  - Efficiently reduces data size for better processing.



CNN Architecture



VGG16

Results and Analysis:

Key Findings:

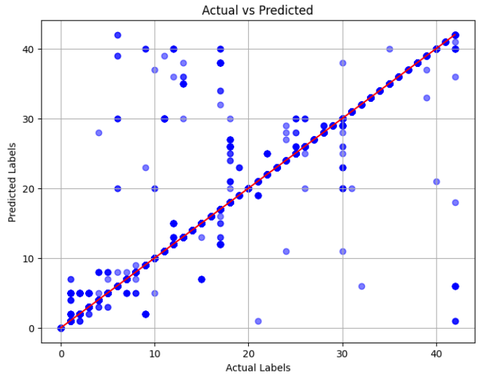
- VGG16 Achieves Top Accuracy (96.9%) - Effectively learned data patterns and generalizes well.
- Random Forest Overfits (97.9% Test, 62% CV) - Memorizes training data, hindering real-world performance.
- Logistic Regression Trade-off - Good accuracy (96.4%) but struggles with complex, high-dimensional data.
- CNNs Excel (95.46%) - Strong performance due to automatic feature learning and handling non-linear patterns.
- SVM, MLP, XGBoost Well-Balanced - High F1-scores indicate a good balance between precision and recall.

Model	Test Accuracy
LDA	71.33%
QDA	64.06%
Random Forest Classifier	97.89%
SVM classifier	60.83%
MLP classifier	85.13%
XG Boost classifier	83.42%
Logistic Regression	96.42%
KNN	90.76%
CNN	95.46%
VGG16	96.9%

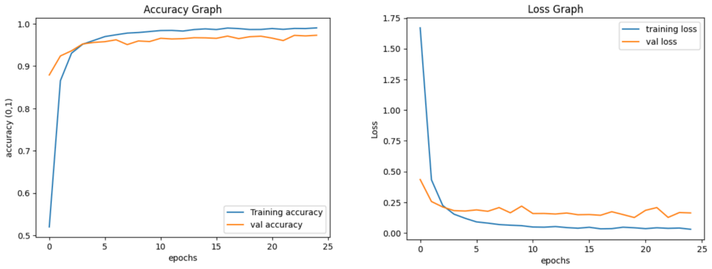
Results and Analysis:

Model Breakdown:

- **VGG16, CNN, SVM:** Perfect precision (1.0) but miss some positive cases (recall 0.824). Balanced F1-score (0.904).
- **MLP, XGBoost:** Excellent F1-scores (0.975 & 0.966) - accurate positive predictions and capture most true positives.
- **Random Forest:** High precision (0.997) but lower recall (0.917) - misses some true positives.
- AdaBoost, Logistic Regression, KNN: Lower F1-scores - require further tuning or alternative algorithms.



VGG16 Model Prediction



CNN Model Prediction

Results and Analysis:

mAP Score Reinforces Findings:

- CNN & VGG16 Top Performers (0.92 & 0.90 mAP): Deep learning excels at complex classification tasks.
- Random Forest, MLP, XGBoost Strong (0.91+ mAP): Good overall performance.
- KNN, AdaBoost Lower mAP: Less suitable for this task.

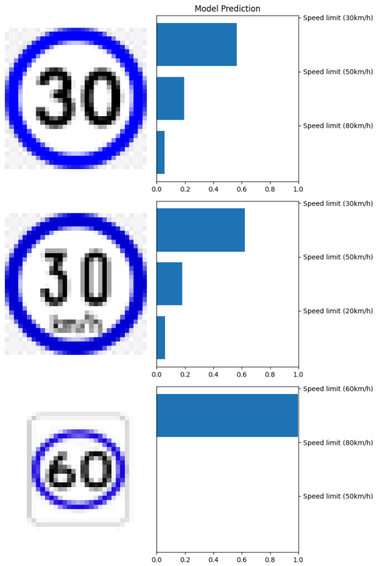
Conclusion: VGG16 achieved the best accuracy, but CNNs are a strong alternative. Deep learning models were most effective, highlighting their importance for complex classification problems.

Predictions

VGG16 model prediction.
Also, we made real-time prediction—an image to be taken on that prediction.

Fig: Input sign given

Fig: Predicted sign



Strategy 01

Upload single or multiple images.

Strategy 02

Turn on the webcam to take pictures and make predictions on them.

THANK YOU