US Traffic Sign Classification

CSE 802: Pattern Recognition & Analysis | Final Project Report

Michigan State University | Aashish Harishchandre

1. Introduction

This project is a traffic sign classification problem in pattern recognition and computer vision. There are safety implications for autonomous vehicle navigation and Advanced Driver Assistance Systems. A good classifier should correctly classify stop signs, speed limits, or warning signs in a real-world setting with significant variability in light intensity, motion blur from dash cam video, partial occlusion by trees or other vehicles, sign surfaces degrading, and similarity between sign types sharing the same shape or color scheme. A fraction of a percentage point of classification could represent a safety problem in the real world.

The goal of this project is to compare and analyze the performance of three existing software package classifiers (***SVM, Random Forest, and k-NN***) against two Bayesian custom built classifiers, a parametric Maximum a Posteriori (MAP) classifier based on multivariate Gaussians and a non-parametric Parzen-window density estimator. Additionally the other objective is to evaluate two dimensionality reduction techniques (PCA and MDA) are used to to test the performance of SVM, Random Forest, k-NN, Parametric and Non-Parametric Bayesian classifiers.

Instead of extracting simple pixels from image, this project takes modern deep learning with classic statistical pattern recognition. This is done via using MobileNetV2, which is a pre-trained deep convolutional neural network. It is used as a frozen feature extractor to transform each image into a small, 1280-dimensional embedding vector. These embeddings are then processed through a dimensionality reduction stage before classification. The steps are: *Raw image → MobileNetV2 feature extraction (1280-Dimensions) → PCA or MDA reduction → Classifier → Prediction*. This project uses the LISA Traffic Sign Dataset which provides the data for 15 classes used thoughtout the project.

2. Description of Dataset

The ***LISA Traffic Sign Dataset*** is a real-world data that was collected from video captured via dash cams in San Diego, California. In the images, US traffic signs were extracted directly from the video, with realistic photo challenges such as motion blur, variable lighting, and perspective distortion. The entire dataset was not used, but selected 15 of the most common sign classes to provide sufficient samples per class for training and statistical analysis, which still beyond satisfies the 4 class project requirement.

Table 1: The raw sample counts for each of the 15 selected classes.

Class Name	Total Samples
stop	929
signalAhead	338
pedestrianCrossing	230
speedLimit35	208
keepRight	174
merge	128
speedLimitUrdbl	113
school	89
speedLimit25	80
stopAhead	76
addedLane	76
speedLimit40	73
speedLimit45	68
yieldAhead	53
schoolSpeedLimit25	51

Table 1: Class distribution in the LISA dataset (15-class subset, N = 2,686 total images).

There is a clear difference in the amount of traffic signs varied by classes. The stop sign has the most images at 929 which is nearly 35% of the entire samples. While the rarest class is schoolSpeedLimit25 at just 51 images, roughly 2%. Due to the 18:1 ratio, it is quite likely that a classifier can still get high accuracy by simply assuming the most frequent class. To take on this challenge, the data is stratified so that each splits creates sets where the distribution of classes in the training, validation, and test sets mirror that distribution overall, and minorities are represented.

2.1 Preprocessing & Data Splitting

Each raw image was rescaled to **224 x 224 pixels** and normalized with the ImageNet mean and standard deviation ($\mu = [0.485, 0.456, 0.406]$, $\sigma = [0.229, 0.224, 0.225]$) per RGB channel. This normalization is necessary because MobileNetV2 was first trained on ImageNet1K dataset, so that the input pixels map correctly. The 2,686 images were then stratified into:

Split	Proportion	Approximate Count	Purpose
Training	70%	~1,880 images	Model fitting
Validation	15%	~402 images	Hyperparameter tuning
Test	15%	~404 images	Final evaluation

Table 2: Dataset split strategy.

2.2 Feature Extraction via MobileNetV2

Raw pixels are not good bases for a Bayesian classifier. For example, a 224 x 224 RGB image is a 150,528-dimensional vector, an impossible high-dimensional space where estimating covariance matrices is nontrivial (the “curse of dimensionality”). Pixels in the surroundings are correlated and encode low-level texture rather than semantics.

To address this, each image was passed through a **MobileNetV2** which has been pre-trained on the **1.2 million** image of ImageNet dataset. The softmax classification head was stripped to expose the final global average pooling layer. This yielded a **1280-dimensional feature vector** per image capturing high-level visual concepts such as shapes, colors, and textures which are all relevant to identifying traffic signs. Importantly, the weights of the MobileNetV2 network were frozen, so that it was considered a deterministic, pretrained model rather than a trainable model.

2.3 Embedding Normalization

The requirement of the project to test feature normalization was addressed through three different schemes on the 1280-dimensional MobileNetV2 embeddings. SVM (C=10, RBF kernel) was picked as the classifier and was trained and tested directly on the embeddings without dimensionality reduction to test the stand-alone effect of normalization.

Scheme	Test Accuracy
None (raw embeddings)	91.61%
Z-score (zero mean, unit variance per feature)	93.43%
Min-Max scaling [0, 1]	86.86%

The accuracy of raw embeddings was increased by 1.82 percentage points using Z-score normalization, but the accuracy of min-max was reduced by 4.75 percentage points using Min-Max scaling. Shows that the 1280 embedding dimensions is not uniformly scaled and the SVM RBF kernel is sensitive to this variation. Z-score normalization is clearly an improvement for future work.

3. Description of Analysis Conducted

3.1 Dimensionality Reduction

Although 1280-dimensional MobileNetV2 embeddings are much smaller than raw pixels, they are still high-dimensional to provide a reliable Bayesian density estimate. If there are limited training samples for each class (some classes have fewer than 60 training examples), the

covariance values per class in 1280-dimensional space would be rank-deficient. To also satisfy the requirement of testing two dimensionality reduction schemes two methods were used to compress the features into a tractable subspace. The mean vector μ is computed per feature for all training samples (i.e., X_train.mean(axis=0)), so each 1280 dimensions is centered.

Principal Component Analysis (PCA)

PCA is an *unsupervised* technique. We zero-center the 1280-dimensional training embeddings and compute eigenvectors of the sample covariance matrix. Then project the data onto the top 50 principal components, which capture the directions of maximum variance in our feature space. This projection is:

$$z = W^T(x - \mu), W \in \mathbb{R}^{1280 \times 50}$$

where W is the top 50 eigenvectors of the covariance matrix. 50 dimensions was selected for the project requirement to capture the large majority of variance but also computationally tractable. PCA finds structure in the overall data distribution without regard to class labels, hence unsupervised learning.

Multiple Discriminant Analysis (MDA)

MDA is a *supervised* technique. Rather than maximizing the total variance, MDA seeks to maximize the ratio of between-class scatter S_B to within-class scatter S_W:

$$J(w) = \frac{w^T S_B w}{w^T S_w W} \rightarrow \text{ solved via generalized eigenvalue problem: } S_w^{-1} S_B w = \lambda w$$

S_B has rank at most c -1 (c is the number of classes), so MDA can get at most c - 1 = **14 discriminant dimensions** for our 15-class problem. This is a higher compression than PCA's 50 dimensions, but the dimensions are tailored to separate classes.

3.2 Classifiers Evaluated

Existing Software Package Classifiers (Scikit-learn): Three classifiers used from Scikit-learn existing software packages are the following:

Support Vector Machine (SVM): An SVM with an RBF kernel finds the maximum-margin hyperplane separating classes in the feature space. Parameter C determines the trade-off between margin width and training error. SVMs perform well in high-dimensional spaces with clear margins between classes.

Random Forest (RF): RF is a set of decision trees, trained on bootstrap samples of the training data, split into random feature subsets at each split. The final prediction is made by majority vote. RF is good at handling overfitting and natural nonlinear boundaries.

k-Nearest Neighbors (k-NN): k-NN, a non-parametric lazy learner, classifies a new sample according to the majority vote of its k nearest neighbors in the training set by Euclidean distance. k-NN does not make any assumptions about the model or decision boundaries, but it has a low computational complexity.

Custom Bayesian Parametric Classifier (MAP with Multivariate Gaussian)

The parametric classifier is based on the decision rule of Maximum a Posteriori (MAP): give the class ω_i having the highest posterior probability:

$$\omega = argmax_i \left\{ \log p\left(x \mid \omega_i\right) + \log P\left(\omega_i\right) \right\}$$

The class-conditional density $p\left(x \mid \omega_i\right)$ is a multivariate Gaussian with MLE estimate mean μ_i and covariance Σ_i. The per-class covariance matrices can be ill-conditioned in 50 dimensions with limited training samples, so a regularization term is added (shrinkage) to the diagonal: $\Sigma_{reg} = \Sigma + \lambda I$, where λ is chosen via validation set search. The prior $P\left(\omega_i\right)$ is the empirical class frequency. This classifier was developed from first principles using the MAP principles presented in homework assignments.

Custom Bayesian Non-Parametric Classifier (Parzen Window)

Rather than using a Gaussian shape for the class conditional density, we derive $p\left(x \mid \omega_i\right)$ directly from the training samples using Parzen window (kernel density) estimation with a Gaussian kernel:

$$p\left(x \mid \omega_i\right) = \left(\frac{1}{n_i}\right) \Sigma_j \left(\frac{1}{h^d}\right) \varphi\left(\frac{x - x_i}{h}\right)$$

Where φ is the standard Gaussian kernel, h is the bandwidth hyperparameter, and d is the dimensionality. The best bandwidth h has been chosen from the candidate values in the validation set. The MAP posterior is then calculated as $\log p\left(x \mid \omega_i\right) + \log P\left(\omega_i\right)$, and the class with the highest score is predicted.

3.3 Experimental Design

Classifier	Hyperparameter(s)	Optimal (PCA)	Optimal (MDA)
SVM	$C \in 0.1, 1, 10, 100$, kernel=RBF, $\gamma \in scale, auto$	$C = 10$, $\gamma = auto$	$C = 0.1$, $\gamma = scale$
Random Forest	$n_{estimators} \in 50, 100, 200$, $max_{depth} \in N/A, 10, 20, 30$	$N = 100$, depth=20	$N = 50$, depth=None
k-NN	$K \in 1, 5, 10, 15, 20, 25, 30, \ldots, 50$, metric = Euclidean	$K = 1$	$K = 10$
Bayesian Parametric	$shrinkage\ \lambda \in 1e\text{-}6, 1e\text{-}4, \ldots, 100$	$\lambda = 0.1$	$\lambda = 0.0001$
Bayesian Non-Parametric	$bandwidth\ h \in 0.1, 0.5, 1, 5, 10$	$h = 0.1$	$h = 0.1$

Table 3: Hyperparameter and results for PCA and MDA reduction spaces.

Repeated Experiments for Stability Assessment

In order to test for classifier stability and the impact of training data variability, the test repeated 5 consecutive splits of training and validation data at 80% train and 20% validation. Then (1) re-fit PCA and MDA using the new training data; (2) retrain all classifiers using the previous optimized hyperparameters; (3) evaluate the accuracy on the held-out test set (which was constant throughout). Finally report the mean accuracy, variance, and standard deviation over 5 splits of each classifier reduction combination.

4. Presentation of Results

4.1 Repeated Experiment Stability Metrics

Tables 4 and 5 report the mean accuracy, variance, and standard deviation across 5 randomized training splits for PCA-reduced and MDA-reduced feature spaces, respectively

Classifier	Mean Accuracy	Variance	Std Dev	Mean Time (s)
k-NN	92.34%	0.000027	0.0052	0.2569
Bayesian Non-Parametric	92.34%	0.000027	0.0052	0.0754
SVM	91.61%	0.000027	0.0052	0.1172
Bayesian Parametric	90.66%	0.000009	0.0029	0.0055
Random Forest	86.57%	0.000167	0.0129	0.5972

Table 4: PCA: Stability metrics across 5 repeated experiments.

Classifier	Mean Accuracy	Variance	Std Dev	Mean Time (s)
k-NN	91.02%	0.000030	0.0055	0.0068
Bayesian Non-Parametric	90.58%	0.000061	0.0078	0.0224
SVM	90.36%	0.000056	0.0075	0.0221
Bayesian Parametric	89.12%	0.000061	0.0078	0.0014
Random Forest	82.55%	0.000556	0.0236	0.0990

Table 5: MDA: Stability metrics across 5 repeated experiments.

4.2 Single-Run Test Set Performance

Table 6 shows the single run accuracy using the optimal hyperparameters found during validation, for direct comparison.

Classifier	Test Accuracy (PCA)	Test Accuracy (MDA)
SVM	93.43%	91.97%
k-NN	92.70%	91.61%
Bayesian Non-Parametric	92.70%	93.07%
Bayesian Parametric	90.51%	91.61%
Random Forest	90.15%	87.59%

Table 6: Single run accuracy for all classifiers and reduction schemes.

4.3 Confusion Matrix Analysis (SVM + PCA)

Figure 1 is the confusion matrix PCA reduced SVM (93.43%). The matrix has abbrivated classes. Green entries are correct classifications; red entries are errors; dashes are zero.

Pred	AL	KR	MG	PC	SC	SL	SA	S25	S35	S40	S45	SU	ST	StA	YA
AL	7	—	—	—	—	—	1	—	—	—	—	—	—	—	—
KR	—	18	—	—	—	—	—	—	—	—	—	—	—	—	—
MG	—	—	13	—	—	—	—	—	—	—	—	—	—	—	—
PC	—	—	—	21	—	—	2	—	—	—	—	—	—	—	—
SC	—	—	—	—	9	—	—	—	—	—	—	—	—	—	—
SL	—	—	—	—	—	4	—	—	1	—	—	1	—	—	—
SA	—	—	—	—	—	—	34	—	—	—	—	—	—	—	—
S25	—	—	—	—	—	—	—	7	1	—	—	—	—	—	—
S35	—	—	—	—	—	—	—	—	18	—	—	3	—	—	—
S40	—	—	—	—	—	—	—	—	—	8	—	—	—	—	—
S45	—	—	—	—	—	—	—	2	1	—	4	—	—	—	—
SU	—	—	—	—	—	—	—	1	3	—	—	8	—	—	—
ST	—	—	—	—	—	—	—	—	—	—	—	—	93	—	—
StA	—	—	—	—	—	—	2	—	—	—	—	—	—	6	—
YA	—	—	—	—	—	—	—	—	—	—	—	—	—	—	6

Figure 1: Confusion matrix: Class abbreviations: AL=addedLane, KR=keepRight, MG=merge, PC=pedestrianCrossing, SC=school, SL=schoolSpeedLimit25, SA=signalAhead, S25/S35/S40/S45=speedLimit25/35/40/45, SU=speedLimitUrdbl, ST=stop, StA=stopAhead, YA=yieldAhead.

4.4 Most Confused Class Pairs

Table 7 shows the most confused pairs, essentially a condensed list of off-diagonal and easier to read error count by classes and count:

True Class	Predicted Class	Error Count
speedLimitUrdbl	speedLimit35	3
speedLimit35	speedLimitUrdbl	3
speedLimit45	speedLimit25	2
stopAhead	signalAhead	2
pedestrianCrossing	signalAhead	2

Table 7: Top 5 most confused class pairs for SVM + PCA.

4.5 Per-Class Metrics (SVM + PCA)

Table 8 lists Precision, Recall, and F1-score for the SVM + PCA classifier which reveals the difference between the majority and minority classes; which shows that classes with fewer training samples tend to have lower F-1 scores.

Class	Support	Precision	Recall	F1-Score
stop	93	1.000	1.000	1.000
signalAhead	34	0.872	1.000	0.932
pedestrianCrossing	23	1.000	0.913	0.955
keepRight	18	1.000	1.000	1.000
merge	13	1.000	1.000	1.000
speedLimit35	21	0.750	0.857	0.800
speedLimit45	7	1.000	0.571	0.727
speedLimitUrdbl	12	0.667	0.667	0.667
schoolSpeedLimit25	6	1.000	0.667	0.800

Table 8: Per-class metrics — SVM + PCA. Full test set.

5. Analysis of Results

5.1 PCA vs. MDA

PCA consistently outperformed MDA in the mean accuracy (Tables 4 and 5). The difference was from 1.32 percentage points for k-NN (92.34% vs. 91.02%) to 4.02 percentage points for Random Forest (86.57% vs. 82.55%). The results were surprising considering MDA is a supervised dimensionality reduction algorithm., which can make use of the labeled dataset, whereas PCA is an unsupervised dimensionality reduction algorithm.

The reason is MDA's compression. With 15 classes, MDA is only 14 dimensions compared to PCA's 50. Although MDA maximizes the separability between classes in its 14 dimensions, it also removes most of the within-class variance which may have information that could’ve helped differentiate for each classifier. The MobileNetV2 embeddings are already highly separable, so PCA's additional 36 dimensions gives helpful data rather than adding noise. For Random Forest, the tightly clustered 14-dimensional MDA space might be underfitting.

The SVM classifier had the *lowest* drop between PCA to MDA (91.61% → 90.36%) only dropping 1.25%. Which indicates that the SVM is highly effective even at lower dimensionality.

5.2 Custom Bayesian vs. Off-the-Shelf Classifiers

Most significant result of this work was the comparison of the custom Bayesian classifier with software packages. In PCA, the Bayesian Non-Parametric classifier tied with the k-NN in mean accuracy (92.34%) and outperforming SVM (91.61%) while being a custom built classifier.

The nonparametric approach outperformed the parametric one (90.66%) because the true distribution of MobileNetV2 embeddings is likely *not* perfectly Gaussian. The Parzen-window estimator makes no distribution assumptions and immediately captures the true shape of the data, such that it does not suffer from the Gaussian assumption. The parametric classifier also is sensitive to covariance regularization (shrinkage λ) and even at optimally tuned will not fully

compensate for model misspecification.

The Bayesian Parametric classifier has the lowest run time at just 0.0055s which is the quickest among all classifiers and both PCA and MDA. The Bayesian Parametric scores nearly 90% in both PCA and MDA accuracies which might make it a good fit for time sensitive applications. On the flip side Random Forest has the lowest accuracy and also the highest run time which means this classifier is not useful in any situations.

The normalization experiment in §2.3 further explains the topic of feature sensitivity. The Z-score across the raw 1280-dimensional embeddings improved the SVM test accuracy by 1.82 percentage points over unnormalized features, which indicates that the RBF kernel's distance computations are more accurate if the feature is scaled uniformly even though the inputs are from a trained network. This indicates that each classifier reported in Tables 4–6 has a relative performance ceiling that could be raised by normalization before PCA reduction in a future pipeline revision.

5.3 Classifier Stability

Tables 4 and 5 show variance. Important thing to note is that all classifiers have very low variance across the 5 replicated splits. For example, for the SVM of PCA, variance is **0.000027**, and the standard deviation is **0.0052**. This means that on any random split, the SVM will be within ±1% of its mean with 95% confidence.

This stability is due to the quality of the embeddings in MobileNetV2. The only exception is **Random Forest on MDA**, where the variance is much larger (0.000556, $\sigma = 2.36\,\%$), indicating that RF does not fit well in the constraint of the 14 dimensions of MDA, making its performance more responsive to the training samples used in the training set.

5.4 Error Analysis and Class Imbalance

Table 8 shows that difficulty in classifying is closely related to the number of training samples per class. For example, the *stop* class of 929 training images has perfect Precision, Recall, and F1. But, the *speedLimitUrdbl* class (113 total images, 12 in test) has F1 = 0.667, the lowest value in the dataset. This class often is confused with speed limit signs where the number was illegible.

The biggest confusion is between speed limit signs (*35, 45, 25, and Urdbl*) with identical surface shapes. These circular white signs differ only in the number printed on the sign; which makes it hard to distinguish at low resolutions. Likewise *StopAhead* and *signalAhead* have the same yellow diamond shape, and distinguishing between them is harder due to the small text. These pattern of confusion closely resembles what we would observe with degraded visual quality.

6. Summary and Conclusions

This project is a complete, end-to-end traffic sign classification pipeline for 15 classes from the LISA data set. By combing MobileNetV2 features with PCA and MDA dimensionality reduction and 5 classification techniques, all classifiers performed at least 82% in test accuracies, with the best classifiers performing over 92%. The main results are:

First and foremost, deep network embeddings are an excellent means of integrating raw pixels into statistical classifiers. The 1280-dimensional MobileNetV2 features are rich, stable, and class-separable, enabling any classifier to perform well on challenging real-world data.

Moreover, PCA (50D) was generally more effective than MDA (14D) in this dataset, because MDA's rank specification removes too much information in a case where the features that are already trained and are already well-structured. In this case, more dimensions was beneficial.

To conclude, the custom Bayesian Non-Parametric classifier outperformed most classifiers only to tie or slightly worse than k-NN similarly to SVM on PCA and MDA features, which is an indication that first principles Bayesian classifiers are still competitive when given good input features. The non-parametric classifier outperformed parametric because deep embeddings are not simple Gaussian distributions and Parzen windows more accurately captured the true density.

There is still much more work that can be done in the future. Firstly, applying z-score normalization of MobileNetV2 embeddings prior to PCA reduction can improve the accuracy (§2.3). Second, testing the full normalization comparison within the PCA pipeline might provide an indication whether normalization and PCA interact positively or not. Lastly, expanding the LISA subset beyond 15 classes and including additional data could make it stronger for smaller classes such as *schoolSpeedLimit25*.

7. References

[1] Duda, R. O., Hart, P. E., & Stork, D. G. (2000). Pattern Classification (2nd ed.). John Wiley & Sons. (Referenced for Maximum a Posteriori decision rules, Multivariate Gaussians, Parzen-window density estimation, PCA, and MDA).

[2] Møgelmose, Andreas & Trivedi, Mohan & Moeslund, Thomas. (2012). Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey. Intelligent Transportation Systems, IEEE Transactions on. 13. 1484-1497. 10.1109/TITS.2012.2209421.

[3] Sandler, Mark, et al. “MobileNetV2: Inverted Residuals and Linear Bottlenecks.” *ArXiv.org*, 2018, arxiv.org/abs/1801.04381.

[4] Pedregosa, Fabian & Varoquaux, Gael & Gramfort, Alexandre & Michel, Vincent & Thirion, Bertrand & Grisel, Olivier & Blondel, Mathieu & Prettenhofer, Peter & Weiss, Ron & Dubourg, Vincent & Vanderplas, Jake & Passos, Alexandre & Cournapeau, David & Brucher, Matthieu & Perrot, Matthieu & Duchesnay, Edouard & Louppe, Gilles. (2012). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 12.

[5] Paszke, Adam, et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library.” *ArXiv.org*, 2019, arxiv.org/abs/1912.01703.