Pattern Recognition and Machine Learning, Spring '24
Project Report

Anushk Gupta[1] Khushi Agrawal[1] Yashraj Chaturvedi[1] Ishita Sancheti[1]
Aatif Ahmad[1] Aditi Rawat[1]

[1]IIT Jodhpur
{b22ai007, b22cs005, b22ai059, b22ee093, b22ai002, b22ai004}@iitj.ac.in

Abstract

Image retrieval is important in various applications that allows users to search and retrieve relevant images with ease. In our project, we focused on image retrieval using traditional machine learning techniques on the CIFAR-10 dataset which consist of 60,000 colored images from ten classes. We have tried classical ML methods like classification and clustering based techniques for effective retrieval of the best pictures. At initial stages after data visualization and feature extraction we used Single-layer perceptrons, K-nearest neighbors, Naive Bayes', Logistic Regression, Random forests which gave us a good insights along-with with accuracies ranging between 33% and 55%. In the later stages of this project we implemented Support vector machines and Multi-layer perceptrons(MLP) to improve model performance. Although the scratch version of SVM was working good on other data sets, it failed badly on CIFAR-10 so we did a 'failure-case analysis' for that and went ahead with sklearn SVM. To manage dataset complexity issues we used techniques such as PCA and K-Means were but finally proceeded with PCA. This project helped us in understanding image retrieval techniques and handling complex & large datasets, aiming to identify effective strategies for real-world applications in a concise yet comprehensive manner.

Keywords: Image retrieval, CIFAR-10, Clustering, Classification, Model Performance

Acknowledgements

We would like to express our sincere gratitude to Prof. Anand Mishra for their invaluable guidance and support throughout the duration of this course. Their expertise and encouragement have been instrumental in shaping the direction of this work. We are also thankful to each member of this project group for their tireless efforts during the various stages of this project. Without their input and feedback, this project would not have been a success.

Finally, we would like to express our vote of thanks to Indian Institute of Technology, Jodhpur for providing necessary resources and facilities to carry out this project effectively.

In Knowledge We Trust

Contents

1 Objectives

- Develop and compare machine learning algorithms including those used from scratch to implement image retrieval on CIFAR-10 Dataset.
- Analyze complexity of the Dataset, understanding and implementing feature extraction and pre-processing methods on the CIFAR-10 Dataset.
- Systematically evaluate the performance of the algorithms, Analyze strengths and weaknesses and choose most efficent method for image retrieval.
- Highlighting key factors influencing the performances of the techniques utilized.
- Performing failure-case-analysis to understand the limitations of the selected algorithms as well as identifying potential areas for improvement.

2 Dataset

2.1 Understanding CIFAR-10

The CIFAR-10[1] dataset consists of 60,000 colour images (32x32 pixels) distributed across 10 different classes with 6,000 images per class. Out of these 60,000 images, 50,000 are for training and 10,000 are for testing purpose. The classes include objects such as airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships and trucks. The dataset is often used for tasks such as image classification and object recognition as well as for training and testing in the field of ML and Computer Vision.

The dataset is divided into 6 batches with 10,000 images each. Out of these, five are training batches and one test batch. 1000 randomly selected images from each class are contained in the test batch. In the same way, the training batches contain the remaining images in random order.
Point to be noted is that some training batches may contain more images from one class than another. Although, the training batches contain exactly 5000 images from each class.

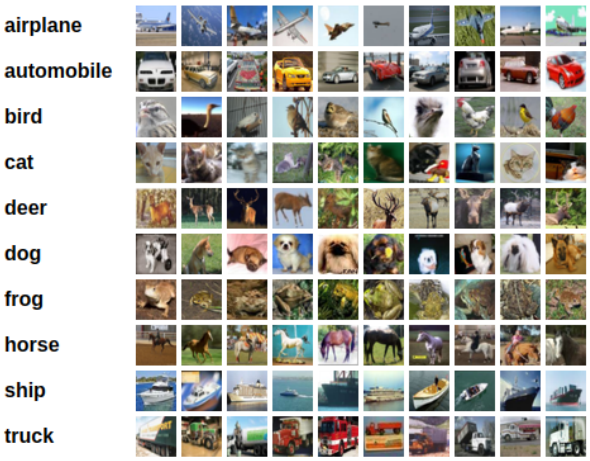


Figure 1: CIFAR-10

2.2 Feature Extraction

The features from the images are extracted for further preprocessing and classification purposes. The method of Histogram of Gradients(HoGs) is used for extracting the features out of the images. But before that, the RGB images are converted to Gray-Scale images as the Gray-Scale is simpler during processing and often sufficient for tasks like object detection.

HoGs are extracted from these (refer Fig.2) gray-scale images. HoGs captures the structure of an object which is defined in the distribution of edge directions or gradients.

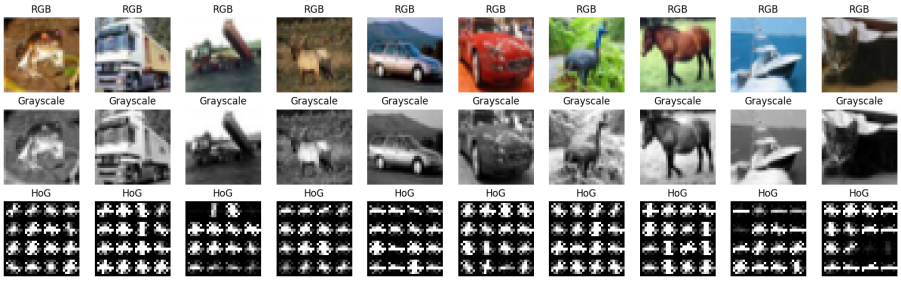


Figure 2: RGB, gray-scale and HoG images displayed for visual comparison.

Point to be noted is that the number of features in the images is 324 after HoG extraction.

HoG features vs CNN features: Although CNNs are generally more powerful and flexible than HoG, as well as they can learn to capture a more complex pattern in the data, yet we have not used them because they require more computational resources and take a long time on a large dataset to train. On the other hand, HoG features are much more simpler and faster for feature extraction than the CNN features. Hence, they are used here.

3 Preprocessing

For the machine learning tasks, the dataset should be prepared via the pre-processing steps like standardization, Principle Component Analysis, K-Means compression. These steps are very crucial for the image retrieval on the dataset as they properly scale the data, reduce its dimensionality [2] such that relevant features are extracted or compressed conserving maximum possible variance (a trade off). These techniques improves the model's performance and efficiency.

Please refer to [2] for more information on the curse of dimensionality.

3.1 PCA

PCA is an unsupervised technique that is used for dimensionality reduction, conserving the most significant variation in the data. Performing PCA on HoG extracted features require a number of steps such as standardization, covariance matrix, decomposition into eigenvectors and eigenvalues, sorting eigenpairs, calculation of explained variance and finally projection into a New Feature Space, where these new features are the linear combinations of original features. These new features are called principal components.

From the Cumulative Explained Variance vs Number of Components plot (Refer Fig.3a), we can clearly see that for 130 number of components, maximum variance was explained.

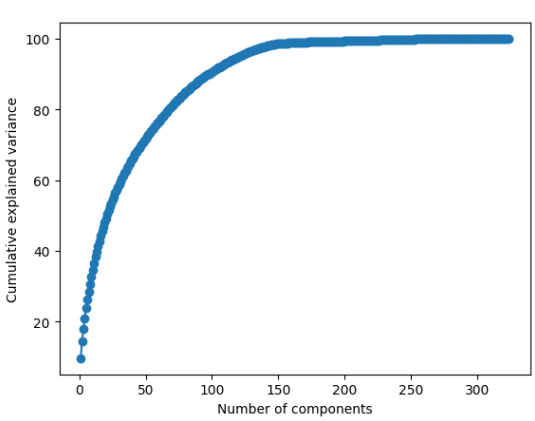
For visualization of data complexity, the data has been projected onto two components which captures 14.48% of total variations (Refer Fig.3b). **Note**: This two component projection is only for visualization purposes. Further, 130 components have been used in the reduced dataset.

A from-scratch implementation is provided in the Project Repository.

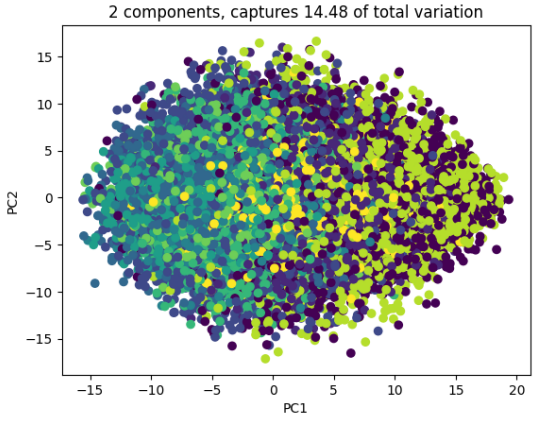
3.2 K-Means

K-means clustering, an unsupervised learning algorithm, is used for compressing the dataset by replacing similar similar data points with centroids of each clusters containing those . It does not reduce the dimensionality of data rather it groups similar data-points together. This compression reduces the storage space which is required for image representation while preserving essential features doing so.

In this particular case PCA is better than K-means clustering (compression) because we want to reduce the dimensionality of data and not to cluster the similar points. PCA reduces the dimensionality of data and in the mean time also conserves most of its original variance, this makes it more suitable for this task. Also, PCA helps us to remove correlations between the features and in the simplification of the dataset.



(a) Cumulative Explained Variance v/s No. Components



(b) Dataset projected onto two principle components

Figure 3: Visualization of PCA

4 Classifiers

4.1 Multi-Class Single Layer Perceptron

A perceptron[3] (Refer Fig.4) is a simplified model of a biological neuron which is suitable for binary classification purposes. In this approach, this model has been extended to handle multiple classes using a concept called One-vs-All. Each class is assigned to a particular perceptron and it tells whether a data-point belongs to that class not. The classification is done by taking a vote from each perceptron and then classifying the data based on the majority votes.

Strength of this approach: Simple yet effective and works well on linearly separable data.

Weaknesses of this approach:

- It can only find a solution if the data-points are linearly separable otherwise the algorithm will not converge to a solution.
- Its performance is highly dependent on the learning rate and the number of iterations, thus these parameters need to be tuned carefully.

Key factors influencing the performance:

- The performance of the classifier depends on the representation of the dataset, which includes the choice of features and the transformation methods used (in this case PCA and HoGs).
- The performance is also significantly impacted by the choice of hyperparameters (learning rate and number of iterations). Therefore, these are needed to be tuned carefully.

In, summary, the Multi-class Single Layer Perceptron offers simple yet effective approach for CIFAR-10 classification. Nevertheless, it is limited in its capacity and is sensitive to the large noise of the CIFAR-10 dataset which hinders its performance. **Accuracy Obtained: 35.68%**

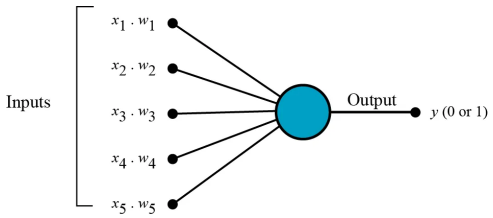


Figure 4: Perceptron

4.2 Logistic Regression

Logistic Regression is a simple yet powerful algorithm for binary classification tasks. It models the relationship between the input features and the output using a linear combination of features. This linear combination z is then transformed using a logistic sigmoid function $\sigma(z)$ which gives us the predicted probability $\hat{y}$ that the input belongs to the positive class.
Logistic Regression generally uses the binary cross-entropy loss function that measures the difference between the predicted and true labels. The goal of training this model is to know the optimal values of weights w and bias b that minimizes the loss function $J(w, b)$.

Model Representation:

$$z = b + \sum_{i=1}^{n} w_i \cdot x_i$$
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Loss Function:

$$J(w, b) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Prediction:

$$\text{Predicted Label} = \begin{cases} 1 & \text{if } \hat{y} \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Strengths:

- This works well on linearly separable data, it can be quite useful for decision making tasks as it uses a sigmoid function which allows the output to be interpreted as probability.
- The use of binary cross-entropy loss makes this method quite useful for binary and multi-class classification tasks.

Weaknesses:

- This model assumes that the input features are linearly related to log odds of the output which may not be the case in complex datasets like CIFAR-10.
- Its performance is highly dependent on selection of proper hyperparameters (learning rate and number of iterations).
- It is vulnerable to the outliers and noisy features which can significantly impact the coefficients and predictions of the model.

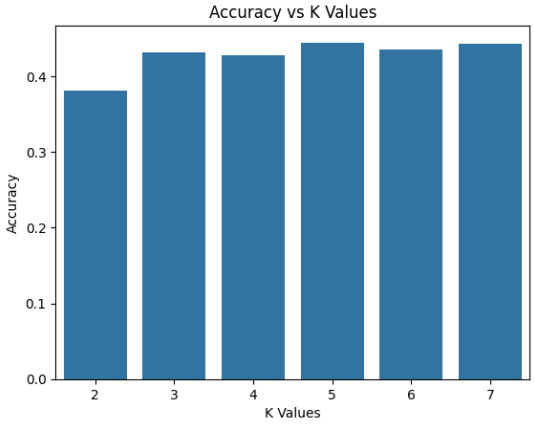
Key Factors Influencing Performance:

- The quality of input features obtained by reducing HoG features via PCA highly influence model's performance.
- Hyperparameters such as learning rate and number of iterations are also important factors influencing the performance of model.
- Poor initialization of weights and biases may lead to a slow convergence or to a sub-optimal solution.

In summary, Logistic Regression offers efficiency and interpret-ability but it may face some problem while capturing complex patterns in CIFAR-10 Dataset. Optimizing hyperparameters may be helpful.
Accuracy Obtained: 40.26%

4.3 K-Nearest Neighbour

A KNN classifier is implemented from-scratch and is applied to the CIFAR-10 Dataset. It operates by calculating the euclidean distance between the test-point and all training points, selecting the K-Nearest Neighbours and determining the majority class among them. Post training the model, the algorithm was tested for various values of k ranging from 2 to 7. The **Highest Accuracy** obtained was for **k=5**. For classification reports, Refer 5a. Some of the **Optimization** techniques used are **Vectorization** and **Batch Processing**.



(a) Accuracy v/s k values

Model accuracy (Scratch) = 46.92
Score:

	precision	recall	f1-score	support
0	0.57	0.57	0.57	1000
1	0.58	0.69	0.63	1000
2	0.33	0.43	0.37	1000
3	0.32	0.23	0.27	1000
4	0.28	0.48	0.35	1000
5	0.49	0.25	0.33	1000
6	0.42	0.69	0.52	1000
7	0.73	0.40	0.52	1000
8	0.69	0.49	0.58	1000
9	0.71	0.45	0.55	1000
accuracy			0.47	10000
macro avg	0.51	0.47	0.47	10000
weighted avg	0.51	0.47	0.47	10000

(b) Classification Report

Figure 5: KNN Stats.

Strengths:

- This approach is quite simple to implement and is intuitive. It works well on dataset with complex decision boundaries.
- It doesn't make any assumptions about underlying data distribution.
- It has high interpret-ability because KNN relies on the similarity of instances. Hence, its prediction can be interpreted.

Weaknesses:

- High computational cost.
- Performance is highly dependent on the choice of k, which needs to be carefully tuned.
- This algorithm strongly requires the data to be normalized.
- This algorithm is inefficient with very high dimensional data and is also sensitive to noise.[4]

Key Factors Influencing the Performance:

- KNN may struggle to make accurate predictions if the classes are not well separated.
- The choice of k and the distance metric.
- The quality of input features after PCA reduction.

The KNN classifier gives promising results while classifying images from CIFAR-10 dataset. However, it is essential to consider its limitations, such as computational inefficiency and sensitivity to noise. **Accuracy Obtained for k=5 is 46.92%**

4.4 Naive Bayes'

Naive Bayes' algorithm[5] is a supervised learning algorithm, which is based on Bayes' theorem and used for solving classification problems.It is based on the assumption that features are independent of each other. This algorithm has multiple versions in the form of Gaussian Naive Bayes, Multinomial Naive Bayes' and Bernoulli Naive Bayes'. Real-time prediction, multi-class prediction, text-classification and recommendation system are some of the real world applications of this algorithm.

Strengths of this approach: It is easy and fast to predict class of test data. It also performs well in multi-class prediction. When assumption of independence holds, it performs better compared to other machine learning models.

Weaknesses of this approach:

- Suppose some category is present in test data but is not observed in training data, this will result into a problem called *"Zero Frequency"*. Techniques like *Laplace Smoothing* can be used to deal with it.
- This model can be inaccurate because in real life, most of the features are not actually independent unlike this model's assumption.
- For the data with negative values, the multinomial naive bayes' classifier is not suitable.

Key Factors Influencing the Performance:

- Choice of classifier (Gaussian, Multinomial).
- Quality of the input features after PCA reduction.
- Distribution of classes also affect the accuracy prediction of classifier.

[6]**Bayes Theorem:**

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

Decomposition of Likelihood using Chain Rule:

$$P(X|y) = P(x_1, x_2, \ldots, x_n|y) = P(x_1|y) \cdot P(x_2|x_1, y) \cdot \ldots \cdot P(x_n|x_1, x_2, \ldots, x_{n-1}, y)$$

Conditional Independence Assumption (Naive Bayes):

$$P(X|y) = P(x_1|y) \cdot P(x_2|y) \cdot \ldots \cdot P(x_n|y)$$

Posterior Probability with Conditional Independence:

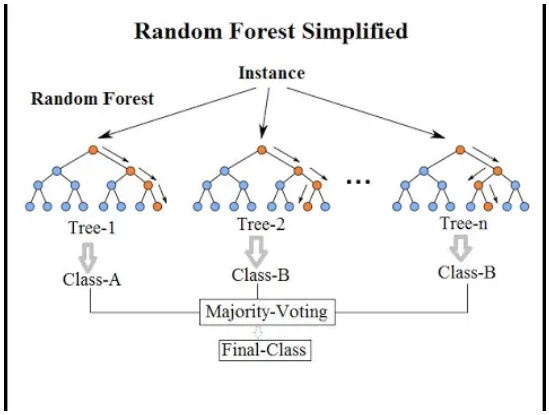
$$P(y|X) \propto P(y) \cdot \prod_{i=1}^{n} P(x_i|y)$$

In summary, Naive Bayes is not an ideal machine learning algorithm for image classification problems because the dataset does not stand with the assumption of independent features. **Accuracy Obtained: 39.85%**

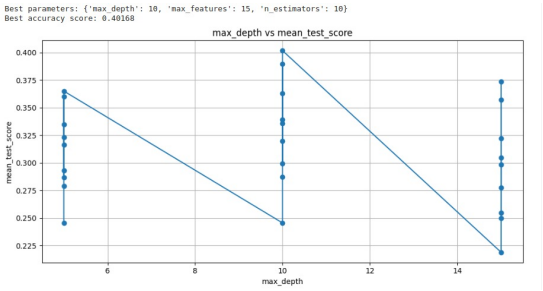
4.5 Ensemble Learning - Random Forest

A scratch implementation of Random Forest classifier, which is an ensemble learning method is provided. It operates by constructing a specific number of decision trees at training time and trains each one on different bootstrap sample of the data. Each decision tree makes a prediction for each sample and the final prediction is the majority prediction across all trees. The implemented code is robust against over-fitting as a pruning function is also implemented, which removes branches from the decision tree that do not improve the accuracy on the validation set.

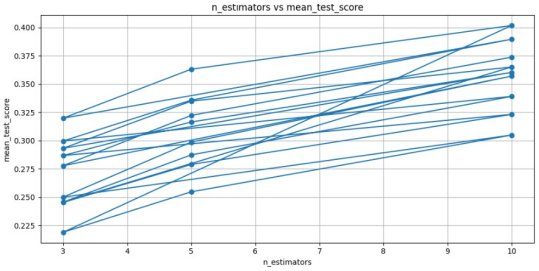
The implementation uses helper functions to calculate the entropy of a set to create a bootstrap of the sample, also to find the most common label in the set and also to find the best split. Multiprocessing module is used to train multiple decision trees in parallel on multi core machines. This reduces the training time. This implementation also includes a Grid-Search to find the best parameters (*max_depth, number of trees and max_features*)



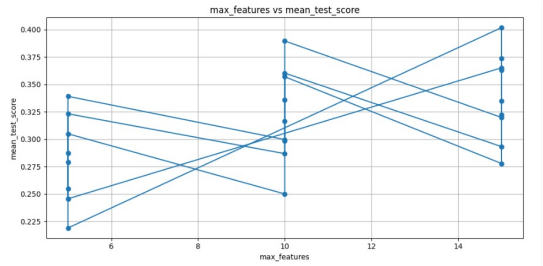
(a) Schematic Diagram[7]



(b) Max Depth Variation



(c) No. Trees Variation



(d) No. Features Variation

Figure 6: Parameters v/s Accuracy Variations

Strengths of this Approach:

- Random Forest is a powerful method to achieve high accuracy (Depends upon the number of trees).
- This approach can handle a large number of features and doesn't require feature selection.
- It can handle both continuous and categorical data.
- It is lees likely to over-fit than a single decision tree because it is the average of the results of many trees.

Weaknesses of this Approach:

- This approach can be slow to train if the number of trees is large (trade-off with accuracy).
- The interpret-ability is not as good as single decision trees.
- If there are highly imbalanced classes, it may not perform well.

Key Factors Influencing the Performance (Refer Fig.6):

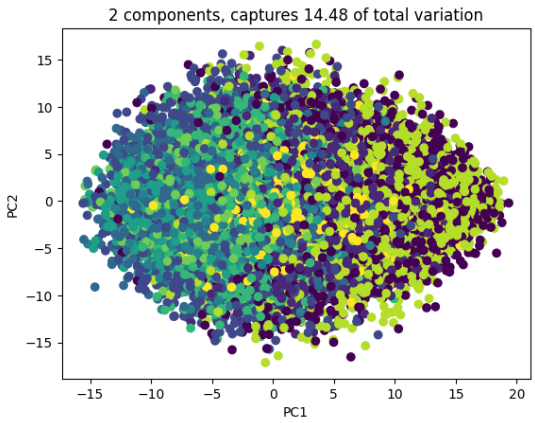
- Increasing the number of trees generally improves the performance, but also increases the training time.
- The more is the max_depth, the more complex patterns are captured by the trees but such trees are more likely to over-fit.
- On taking more number of features, we get better splits but this also increases the training time (trade-off).

Random Forest offers high accuracy and can handle large feature sets without feature selection. However, they may be slow to train on Datasets like CIFAR-10. So, this can be managed by adjusting the parameters such as *number of trees and tree depth.*
From-Scratch Accuracy = 33% (Number of Trees = 5, Max_depth = 5, Max_feature = 10)
On analyzing accuracy with best parameters, using scikit-learn library (*parameter tuning was time consuming using from-scratch implementation*).
Sklearn Accuracy = 40.168% using best parameters(Max_depth = 10, Max_feature = 15, Number of Trees = 10)

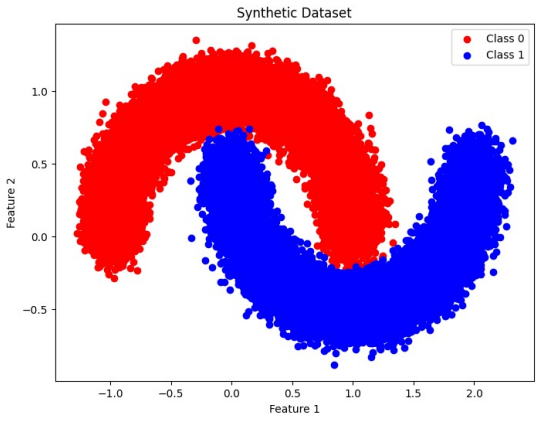
4.6 Support Vector Machines

SVM[8] is a powerful and flexible kind of supervised learning algorithm for both classification and regression purposes. It best segregates the two classes a hyperplane/line. A from-scratch implementation of SVM classifier is applied on CIFAR-10 Dataset. However, the accuracy of the from-scratch model on CIFAR-10 Dataset is only 10% which is quite low. There could be several reasons for this such as complexity, high dimensionality as well as too much noise in the CIFAR-10 dataset as well as using inappropriate hyperparameters (trial & error).

The Hyperparameter tuning part is significantly increasing the training time of the from-scratch model. Therefore, trial & error is used for hyperparameter tuning. This is one of the reasons for low accuracy along with high complexity and dimensionality and too much noise in the data. For **failure case analysis**, another synthetic dataset is generated from scikit-learn **(made using make_moons with n_samples = 50000, noise = 10%)**. The accuracy of our from scratch model was significantly higher on this and the training time was also less compared to the CIFAR-10 Datastet. This suggests that from-scratch implementation's of SVM might be for simpler and lower dimensional datasets with less noise.



(a) $c = 1$, $\gamma = 0.01$, rbf kernel
From-scratch accuracy on CIFAR-10 = 10%



(b) $c = 10$, $\gamma = 1$, rbf kernel
From-scratch accuracy on Synthetic = 99.96%

Figure 7: From-Scratch SVM Accuracy Comparisons

Accuracy of scikit-learn's SVM on CIFAR-10 Dataset along with *rbf* kernel is *61.89%*.

Linear Kernel:

$$K(x_1, x_2) = x_1^T x_2$$

x_1, x_2 : Feature vectors
$K(x_1, x_2)$: Kernel function output

Polynomial Kernel:

$$K(x, y) = (x^T y + 1)^d$$

x, y : Feature vectors
d : Degree of the polynomial
$K(x, y)$: Kernel function output

Gaussian (RBF) Kernel:

$$K(x, y) = e^{-\gamma \|x-y\|^2}$$

x, y : Feature vectors
γ : Kernel coefficient
$K(x, y)$: Kernel function output

Strengths of this Approach:

- This approach is found to be more effective in Higher Dimensional Spaces.
- SVM works quite well compared to the other approaches where there is clear margin of separation between the classes.
- SVM's perform well in those cases also where number of features exceeds the number of samples (although in CIFAR-10, this is not the case).
- It can handle both linear and non-linear classification problems via different kernel functions.

Weaknesses of this Approach:

- SVM's algorithm can take high training time for large datasets like CIFAR-10.
- SVM does not perform very well when the dataset has more noise i.e. target classes are overlapped.
- SVM underperforms if number of features exceeds number of training samples.

Key Factors Influencing Performance:

- Choice of kernel (*linear, poly, rbf*) function makes a large difference in the performance.
- $c, \gamma, degree$ parameters also influence the performance significantly.
- The dimensionality and complexity of data significantly affect the performance and training time of SVM model.

Failure Case Analysis
The failure of *from-scratch SVM* on the CIFAR-10 Dataset maybe due to several reasons such as our model may not be able to capture such complex data effectively due to its simplicity. CIFAR-10 Dataset is a complex, high dimensional dataset with images as its data-points. The SVM might not be able to find a hyperplane to separate the classes in such a high dimensional space. Furthermore, the hyperparameters might not be optimal for CIFAR-10 datset by default. On inclusion of the hyperparameter tuning, the training time shoots up due to high dimensionality of data.

4.7 Multi-Layer Perceptron

Implementation of a MLP[9] is done using *PyTorch*. MLP is composed of one or more input layer, one or more hidden layer but only one outer layer. Each layer is fully connected to the next one just as in human nervous system a neuron in a layer is connected to all of the neurons in the next layer. For the classification of PCA reduced CIFAR-10 dataset, a hidden-layer (size = 128) and outer-size (size = 10 = number of classes) has been formed. The model uses Rectified Linear Unit (ReLU) and Cross-Entropy as the activation and loss function respectively. For training the model, Adam Optimizer is used.

MLP represents significant extension of Single Layer Perceptron (SLP) model while SLP can only learn linearly separable patterns, MLP overcomes this by introducing one or more hidden layers with non-linear activation function.

Strengths of this Approach:

- Modelling of complex patterns can be done via MLPs due to their multiple layers and non-linear activation function(s).
- Training can be Parallelized across multiple GPUs, speeding up training for large datasets.
- Training neural networks and implementing their framework becomes quite efficient and flexible using *PyTorch*.
- For faster convergence and better performance, the *Adam* optimizer is used which adapts learning rate during training.

Weaknesses of this Approach:

- MLPs are very complex models (it has too many layers), therefore they can easily overfit.
- Training large MLPs can be computationally expensive and require significant resources.
- The spatial structure of images are not taken into account by the MLPs which may lead to sub-optimal solutions. (lack of spatial coherence)

Key Factors Influencing Performance:

- Thanks to the architecture of MLP (presence of hidden layers), their model can capture more complex patterns but there is a risk of overfitting (tradeoff).
- ReLU as an activation function is a nice choice but there may exists other functions which may be better for some tasks.
- Better performance can be achieved by increasing the number of epochs, but too many epochs may lead to overfitting (tradeoff).
- If the batches are small, this may lead to faster training as well as faster convergence. On the other hand a more accurate estimate of the gradient is provided by larger batches (tradeoff).

```
Epoch [1/20], Loss: 1.5379
Epoch [2/20], Loss: 1.3610
Epoch [3/20], Loss: 1.0601
Epoch [4/20], Loss: 1.3355
Epoch [5/20], Loss: 1.3748
Epoch [6/20], Loss: 1.3779
Epoch [7/20], Loss: 1.5043
Epoch [8/20], Loss: 0.9277
Epoch [9/20], Loss: 1.2497
Epoch [10/20], Loss: 1.0766
Epoch [11/20], Loss: 0.7170
Epoch [12/20], Loss: 1.0398
Epoch [13/20], Loss: 1.3389
Epoch [14/20], Loss: 1.6056
Epoch [15/20], Loss: 1.1935
Epoch [16/20], Loss: 0.8228
Epoch [17/20], Loss: 1.4386
Epoch [18/20], Loss: 0.6620
Epoch [19/20], Loss: 0.9773
Epoch [20/20], Loss: 0.8076
Test Accuracy: 0.5855
```

A more complex and non-linear dataset can be handled using MLPs in a flexible manner. Although, they have their own limitations in capturing spatial dependencies but still they make up a better approach compared to many other approaches for image classification tasks on CIFAR-like datasets. **Accuracy Obtained = 58.55%**

Figure 8: Losses on each epoch

5 Summary

In this project, Image Retrieval was explored using traditional machine learning techniques on CIFAR-10 Dataset. The given dataset consists of 60,000 coloured images distributed among 10 classes. We started with data visualization followed by feature extraction and dimensionality reduction. Ultimately, PCA was used over k-means for reducing the number of features. After the pre-processing, multiple classifiers were employed like SLPs, KNNs, Naive Bayes', Ensemble Learning, which helped in achieving 33% to 55% accuracy. Further, in the project, SVMs and MLP were implemented for improved performance, although the from-scratch SVM resulted in a low accuracy of 10%, yet scikit-learn's SVM achieved an accuracy of 61.89%. On the other hand, MLP were able to capture intricate patterns, achieving an accuracy of 58.55%.

As far as the best classifier is concerned with image retrieval on CIFAR-10, the scikit-learn's SVM with rbf kernel is the best classifier with a impressive accuracy of 61.89%. Despite challenges faced by the from-scratch implementation, the scikit-learn's SVM is the most effective classifier for this task according to our analysis.

A Contributions

1. **Anushk Gupta, B22AI007**: Implemented Pre-Processing before PCA, Single Layer Perceptron. And managed the Project Repository.
2. **Khushi Agrawal, B22CS005**: Implemented KNN and Multi Layer Perceptron classifiers and helped in Project Demo Code.
3. **Yashraj Chaturvedi, B22AI059**: Implemented Ensemble Learning using Random Forest and SVM classifiers.
4. **Ishita Sancheti, B22EE093**: Implemented PCA for dimensional reduction and K-means clustering for compression.
5. **Aatif Ahmad, B22AI002**: Implemented Naive Bayes, SVM classifiers and prepared the demo code and project page.
6. **Aditi Rawat, B22AI004**: Implemented Logistic Regression and helped in Multi Layer Perceptron.

*The content for the Report on the Various topics were provided by the concerned members and the final report was compiled by Yashraj and Anushk.

B Bibliography

References

[1] Alex Kriszhevsky, CIFAR-10 *University of Toronto*, `https://www.cs.toronto.edu/~kriz/cifar.html`

[2] The Curse of Dimensionality, *Medium*, `https://medium.com/vlgiitr/the-curse-of-dimensionality-15f950e519d2`.

[3] What the Hell is Perceptron?, Towards Data Science, *Medium*, `https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53`

[4] Prof. Anand Mishra, Spring 2024, PRML Lecture 2, *IIT Jodhpur*.

[5] Prof. Anand Mishra, Spring 2024, PRML Lecture 22-26, *IIT Jodhpur*.

[6] Naive Bayes Algorithm, *Medium*, https://medium.com/analytics-vidhya/na%C3%AFve-bayes-algorithm-5bf31e9032a2.

[7] Ensemble Learning and Random forest, *Medium*, https://medium.com/@hrishavkmr/ensemble-learning-and-random-forest-aa26a2f1acf1

[8] Patrick Loeber, AssemblyAI- Support Vector Machine, *Youtube*, https://www.youtube.com/watch?v=T9UcK-TxQGw

[9] Kirill Shmilovich, University of Chicago, Multi-Layer-Perceptron *GitHub*, https://github.com/KirillShmilovich/MLP-Neural-Network-From-Scratch