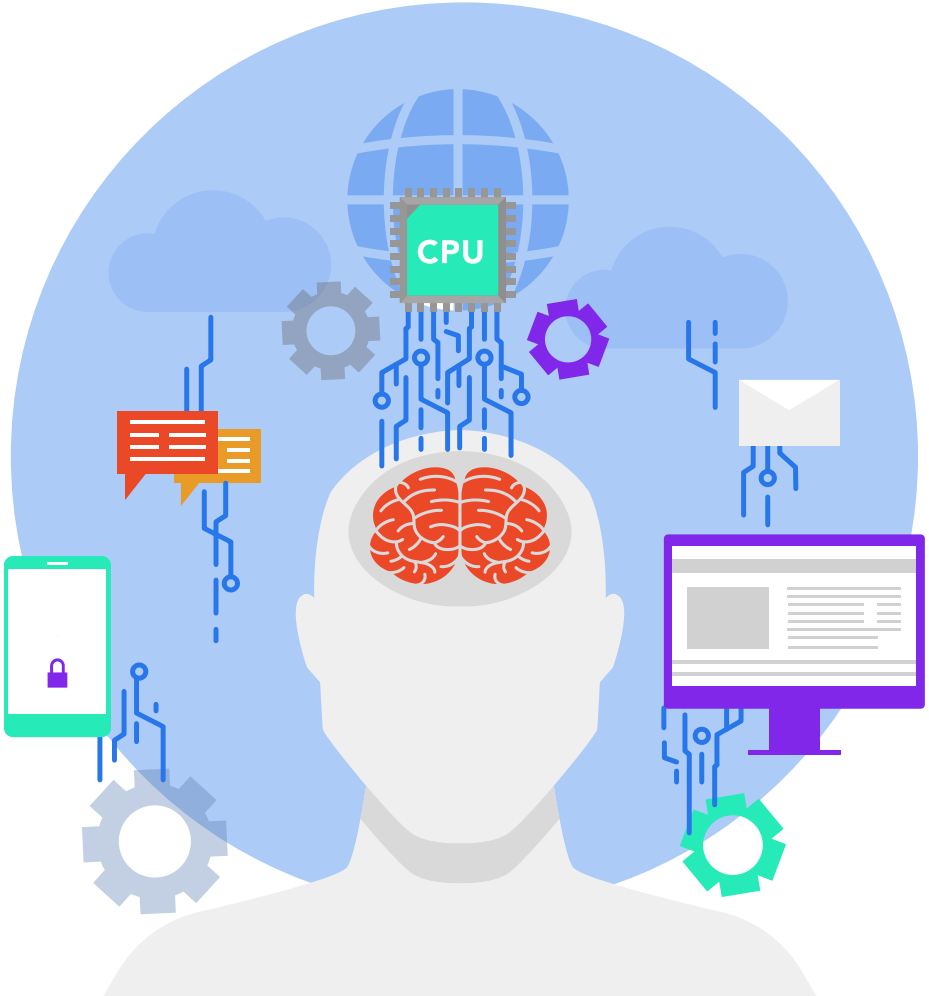


Machine Learning for CV

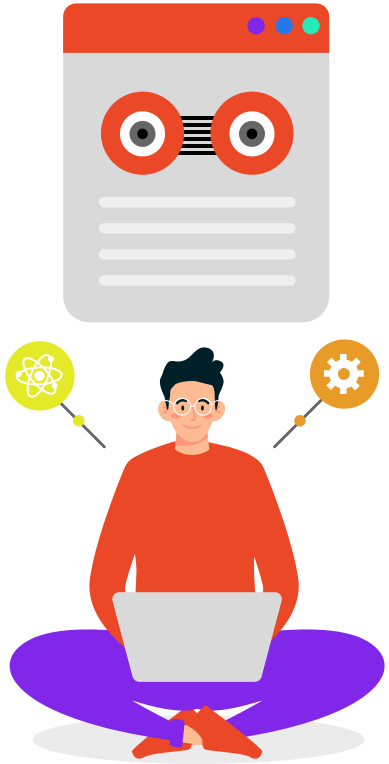
Image Classification

Group 5:
Miruna-Diana Jarda
Gunjan Paul
Diana Tat

Introduction

The goal of this week project is to improve the Bag-of-Visual-Words (BoVW) algorithm for better picture identification. The method entails testing different factors and approaches in a methodical manner. We assess the effects of varying local feature amounts by substituting dense SIFT at different scales for identified keypoints.

The k-nearest neighbours (knn) classifier is studied in relation to various k values, appropriate codebook sizes (k), and normalization/scaling. Investigations are conducted into alternative distance metrics and support vector machine (SVM) kernels, such as linear, RBF, and histogram intersection. Dimensionality reduction is taken into consideration, using Fisher Vectors and Spatial Pyramids utilising the Yael library as optional tools. Throughout, the BoVW architecture is optimised for strong image identification by the application of rigorous cross-validation.

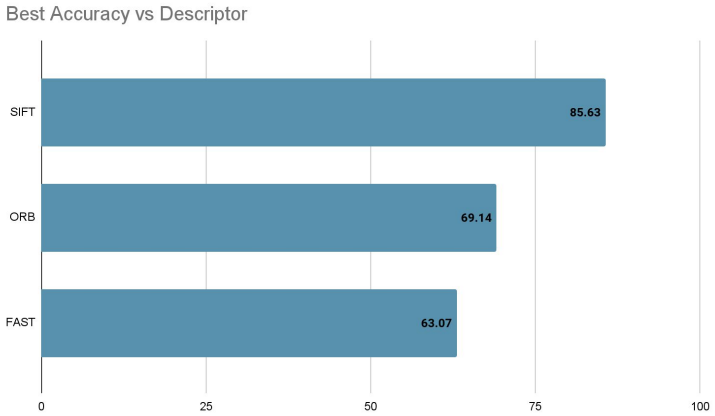


Test local features

What performed the best?

For this task we have used:

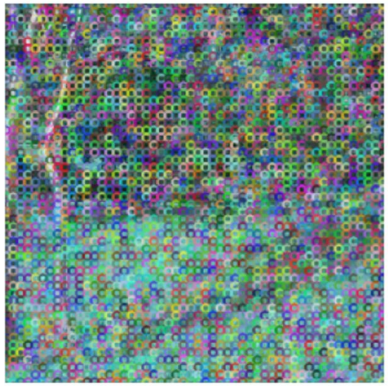
- **SIFT**
  - Detects keypoint locations and computes descriptors based on local image gradients.
  - Scale-invariant and resistant to changes in lighting and viewpoint.
- **ORB**
  - Combines the efficiency of FAST keypoint detection with BRIEF descriptors.
- **FAST**
  - A corner detection algorithm designed for speed.
  - Operates by identifying corner points using pixel intensity comparisons in a circular neighborhood.



Conclusion: SIFT performed the best.

Examples of dense SIFT with different step size

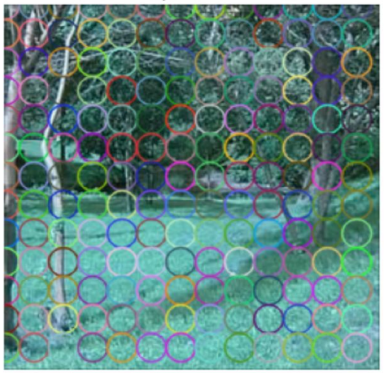
Step Size: 5



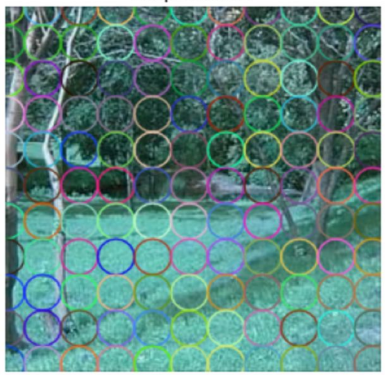
Step Size: 10

Step Size: 15

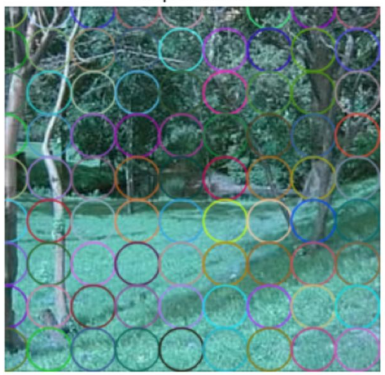
Step Size: 20



Step Size: 25



Step Size: 30



Use dense Representation

Code modification

Dense representation is a method where features are extracted at regular intervals across an image, instead of focusing on specific key points.

We have applied this representation with SIFT, ORB and FAST. It offers a thorough and detailed description of visual information. Step size determines the density of features: larger steps reduce complexity but could overlook finer details, whereas smaller steps capture more details but need more processing. Dense representation, which strikes a compromise between computing speed and detail capture, is helpful when a comprehensive study of visual characteristics over the whole image is required.

The primary benefit of dense representation is its capacity to extract information from the whole picture, offering a more thorough and detailed depiction of the visual elements. When crucial features are dispersed across the image rather being focused at a few crucial spots, this might be advantageous.

Optuna

Hyperparameter search

We have automated the search of the hyperparameters using *Optuna*. The Optuna library efficiently explores the hyperparameter space using Sequential Model-based Optimization (SMBO), adapting its search strategy based on past trial outcomes. This adaptability makes Optuna more efficient than traditional methods, leading to faster convergence and better results. The code runs the optimization process, extracts the best trial results, and prints the optimal hyperparameters and corresponding accuracy.

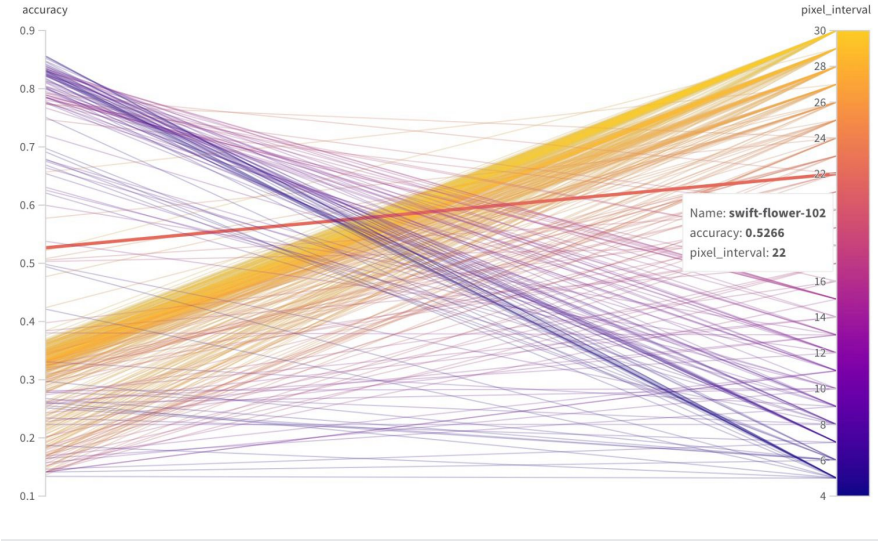
Optuna cleverly samples and assesses various hyperparameter configurations by establishing a search space for hyperparameters, stating an objective function, and outlining a plan to minimise or maximise it. Based on previous assessments, it constantly modifies its search and progressively approaches ideal hyperparameter values.

This framework is very useful for saving time and money by automating the tiresome process of hyperparameter tweaking. Optuna may be integrated by researchers and practitioners into their machine learning processes to improve model performance on a variety of tasks and datasets.

Use dense Representation

Conclusions

We have obtained:
Accuracy: 84.26 %
Best hyperparameters: {'step_size': 14}
bovw_num_cluster(Kmeans)=242



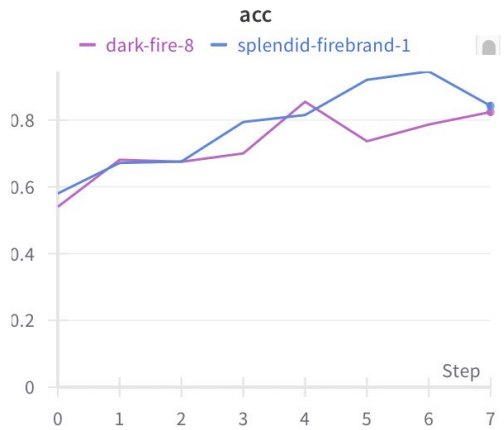
<u>How does the step_size influence the results?</u>

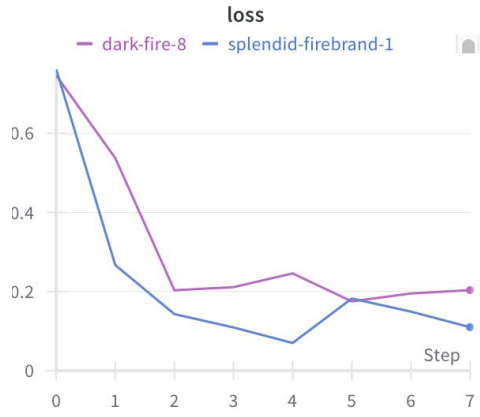
A lower step_size leads to a higher keypoint density, and this in turn allows the descriptors to capture more local information. It makes it possible to capture more precise spatial information in the picture, which is useful for differentiating between various objects or visual patterns. More distinguishing and discriminating descriptors result, which might be important for precise categorization.

Using WANDB

Test accuracy and loss

For the next tasks we have used Weights&Biases and dense SIFT as our local feature, since it has the best results. W&B offers experiment tracking, which makes it possible to log measurements, artifacts, and hyperparameters effectively. A W&B run is started with the *wandb.init* function, and accuracy metrics are recorded for every combination of hyperparameters using the *wandb.log* function.





Use norm or scaler

Does it help?

The next step was to step if a norm or scaler helps. We did that by adding a boolean parameter, *use_norm_scaler*, allowing the inclusion of normalization or scaling to the Dense SIFT feature extraction process. The accuracy function now checks this parameter and applies preprocessing if set to True. The normalize_or_scale function uses the StandardScaler for feature transformation. We have also modified the objective function to include this parameter in the accuracy computation for Optuna optimization. This addition provides flexibility to explore the impact of normalization or scaling on Dense SIFT features during hyperparameter tuning.

Best trial:
Accuracy: 82%
Best hyperparameters: {'step_size': 15, 'use_norm_scaler': False}

We have obtained a lower result using the norm or scaler.

Use norm or scaler | Comment the results

Best trial:
Accuracy: 82%
Best hyperparameters: {'step_size': 15, 'use_norm_scaler': False}

We have obtained a lower result using the norm or scaler.

Addressing the prior scenario, a lower *step_size* resulted in the maximum accuracy of 76.70%. Additionally, better results are obtained when 'use_norm_scaler' is set to False for all trials, indicating that feature normalisation or scaling may not be helpful in this particular situation.Overall, the results highlight the importance of the'step_size' option in Dense SIFT technique optimisation, as well as the consistent selection of 'use_norm_scaler': False indicates that further normalisation or scaling in this case does not increase performance.

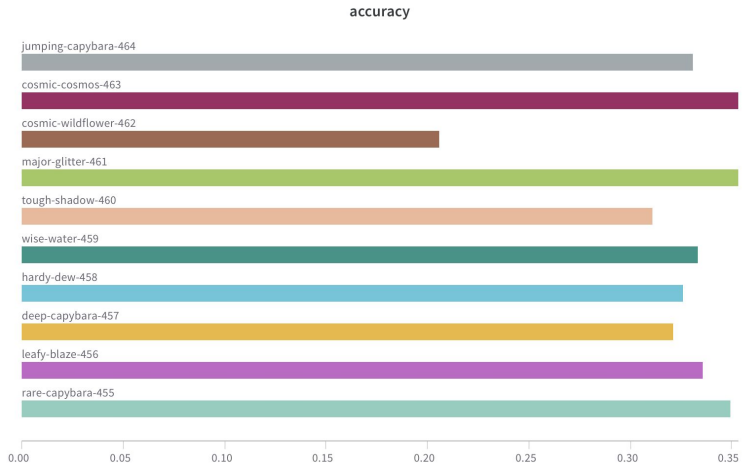
Codebook of size k

What performed best?

For this task we focused on optimizing hyperparameters, specifically the step size and codebook size. We adapted the accuracy function to log results to WandB, enabling real-time monitoring of the optimization process. The objective function evaluates the model's performance based on the suggested hyperparameters, and the optimization loop aims to find the combination that maximizes accuracy.

This method looks for an ideal setup that improves the accuracy of the SVM classifier by simultaneously optimising step size and codebook size. This is because step size is important in figuring out the appropriate K value for generating the BOVW and ultimately predicting the outcome.

Best trial:
Accuracy: 85.63%
Best hyperparameters: {'step_size': 15, 'bovw_num_cluster(Kmeans)': 340}



Codebook of size k

What performed best?

Trial 1 was the most successful of the ten trials, with the maximum accuracy of 85.63% at **step size** of 5 and larger **codebook size** of 340. This emphasises how important it is to select hyperparameters in a balanced manner for optimal feature representation. Significant differences in codebook_size have been found to have a significant effect on accuracy, highlighting the importance of having a wide visual vocabulary. Lower accuracy was produced by trials with step size and codebook size values that were either too small or big, indicating the necessity for careful parameter balancing. All things considered, our findings offer insightful information on how hyperparameters interact to optimise Dense SIFT for image classification tasks.

Best Trails	Step Size	Codeblock Size	Accuracy(%)
0	5	340	85.63
1	7	378	85.5
2	9	351	84.76
3	5	113	84.63
4	7	368	84.26
5	7	387	84.14
6	8	343	83.89

KNN classifier k value

What performed best?

The k-value used in the k-Nearest Neighbors (abbrv. KNN) determines the number of neighbors taken into account in the process of classification.

While choosing a small value might lead to an increased sensitivity to noise, a large k might smooth out local patterns so finding the optimal k is essential.

Best performing k values: 5 (accuracy 0.834), 7 (accuracy 0.8302), 18 (accuracy 0.8253)

KNN classifier distances

Does it make a difference?

Distance metrics are the deciding factor into how near two neighbors actually are as they determine how the distance between two data points is calculated in the feature space.

- **Euclidean Distance** - Measures the straight-line distance between two points in Euclidean space (highest accuracy achieved 0.834).
- **Minkowski Distance** - Generalizes both Euclidean (when p = 2) and Manhattan distances (when p = 1) (highest accuracy achieved 0.7856).
- **Chebyshev Distance** - Measures the maximum absolute difference between coordinates (highest accuracy achieved 0.5527).
- **Cosine Similarity** - Measures the cosine of the angle between two vectors determined by the data points (highest accuracy achieved 0.6778).
- **Hamming Distance** - Measures the number of positions at which the corresponding symbols differ (highest accuracy achieved 0.6568).
- **Jaccard Similarity (index)** - Measures the similarity by dividing the size of the intersection of two sets by the size of the union of the two sets (highest accuracy achieved 0.1437).

SVM classifier

Linear

The Linear Kernel is a specific type of SVM kernel function used when data can be efficiently separated by a line (hyperplane). The linearly separated data represent the different classes of the data set.

Highest accuracy result achieved: 0.8463

SVM classifier

RBF

Radial Basis Function (RBF) kernel is a more complex kernel compared to the linear one as it manages to tackle the issue of non-linearly separable data by first projecting it into a higher-dimensional space and then aiming to find a boundary. This is achieved using the radial basis function:

$$K(X_1, X_2) = \exp(-\frac{\|X_1 - X_2\|^2}{2\sigma^2})$$

Highest accuracy result achieved: 0.8525

SVM classifier

Histogram Intersection

The Histogram Intersection Kernel is a specific type of kernel function used SVMs, primarily in the context of image and video recognition tasks. The Histogram Intersection Kernel specifically measures the similarity between two histograms. It does this by summing the minimum values of corresponding bins in these histograms.

Mathematically, if H_i and H_j are two histograms, the Histogram Intersection Kernel $\boldsymbol{K}$ is given by:

$$K(H_i, H_j) = \sum_k min(H_i(k), H_j(k))$$ Where k indexes the histogram bins.

small K implies H_i and H_j are different

Large k implies H_i and H_j are similar

It's effective in handling variations in lighting, background distractions, and viewpoint changes and captures the essence of images by comparing histograms of visual keywords, making it suitable for visual data. This approach ensures that spatial information within the image is better captured, enhancing the performance of the classification model.

The process of dimensionality reduction involves converting data with a high number of dimensions into a lower-dimensional space. Various techniques are used for this purpose, all with the common goal of trying to retain the characteristics of the original data while effectively reducing its dimensionality. Two such techniques are discussed below:

- **Principal Component Analysis (PCA)** is an unsupervised method that transforms data such that it reaches maximum variance in the given dataset.

- **Linear Discriminant Analysis (LDA)** is a supervised dimensionality reduction method aiming to find the linear combination of data points that maximize class separation.

Best performing: LDA with number of components (reduced dimensions) equal to 7.

Spatial Pyramid

What performed best?

Spatial Pyramid Local Feature Extractor is an extension of the Bag of Visual Words (BoVW) model which enhances the traditional BoVW by introducing a multi-level, spatial partitioning of the image, allowing for a more detailed feature extraction.

Spatial Pyramid Structure: The image is divided into increasingly finer spatial sub-regions at different levels of a pyramid. In this implementation, we have three levels:

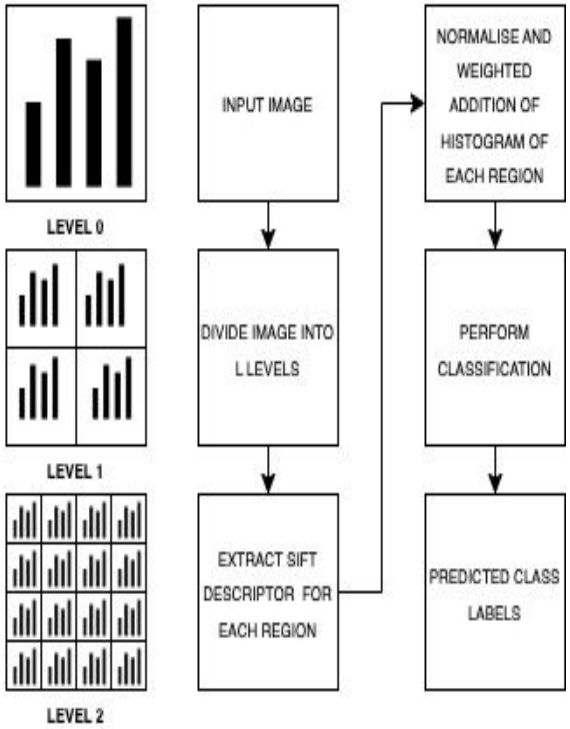
- Level 0: The whole image.
- Level 1: A 2x2 grid, dividing the image into four equal parts.
- Level 2: A 4x4 grid, further dividing each part of the Level 1 grid.

Feature Extraction:

- At each level, the image or sub-region is processed to extract local features.
- Scale-Invariant Feature Transform (SIFT) is used for this purpose. SIFT is effective in detecting and describing local features in images.

Histogram Computation:

- For each set of local features, a histogram of visual words is computed using the provided visual vocabulary. This step translates raw features into a quantifiable and comparable format.
- Then produces a histogram representation.



Fisher Vector

What performed best?

Fisher Vector is a powerful technique used in image classification, which extends the traditional bag-of-visual-words (BoVW) model. It captures not only the frequency of visual words (local image descriptors), like BoVW, but also their statistical properties such as means and covariances. This extra information helps improve the discriminative power of image representations.

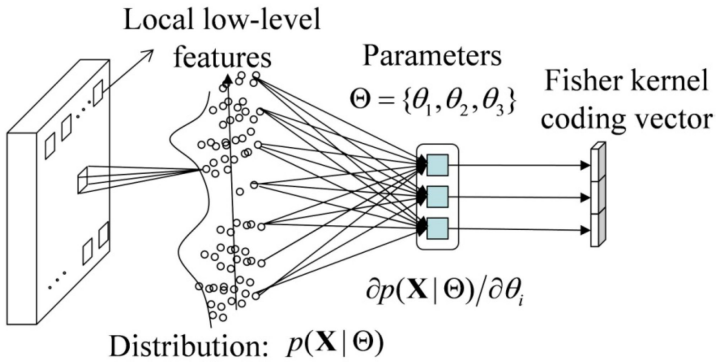
The Fisher Vector representation for image classification consists of the following key steps:

Feature Extraction: Local image descriptors are extracted from images using methods like SIFT, SURF, or CNN features.

GMM Training: A Gaussian Mixture Model (GMM) is trained on these descriptors, where each GMM component represents a cluster of similar descriptors.

Computing Fisher Vectors:
- GMM assigns responsibilities to each descriptor, indicating its likelihood of belonging to each component.
- Fisher Vectors are computed by calculating gradients with respect to GMM parameters (means and covariances) for each descriptor.
- Fisher Vectors can be normalized to balance their magnitudes.

Image Representation: Fisher Vectors from all descriptors in an image are concatenated into a high-dimensional vector, serving as the image representation for classification tasks.



Method comparison

What performed best?

Our hyperparameter optimization analysis has yielded insightful results regarding the influence of different hyperparameters on the accuracy of our model. The bar chart demonstrates the relative importance of each parameter.

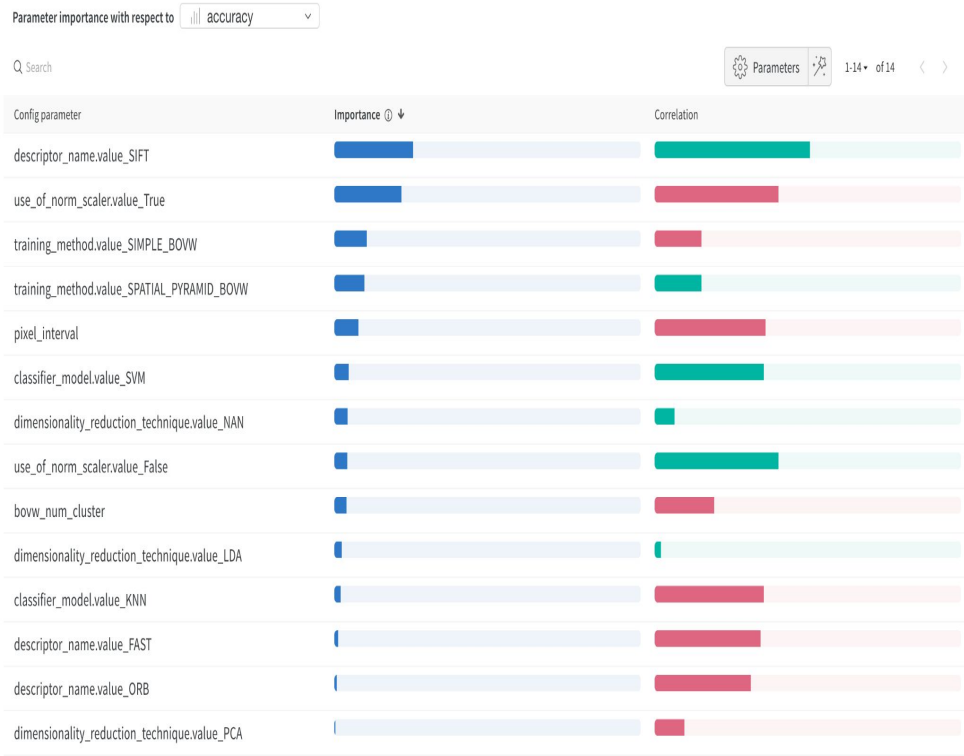
Key Influencers: SIFT descriptor and norm scaling are paramount, strongly driving accuracy.

Training Methods: Simple BOVW and Spatial Pyramid BOVW significantly affect outcomes.

Pixel Interval: Modest but actionable impact, indicating potential gains through optimization.

Classifier Impact: SVM outperforms KNN, suggesting a better fit for our model.

Dimensionality Reduction: LDA and PCA show smaller, yet meaningful contributions to accuracy.



Cross Validation

Everything

After tuning hyperparameters, the next step is often cross-validation. The purpose of cross-validation is to assess how well your model, with its selected hyperparameters, will generalize to an independent dataset. It helps in understanding the model's performance under different subsets of your data.

Our best parameters are

1. bovw_num_cluster = 340
2. descriptor_name = "SIFT"
3. training_method = "SIMPLE_BOVW"
4. pixel_interval = 5
5. apply_Scaling = False
6. classifier_model_name = "SVM"
7. SVM_classifier_c = 1
8. SVM_classifier_kernel = "sigmoid"
9. dimensionality_reduction_technique = "LDA"

Best accuracy was 85.63%

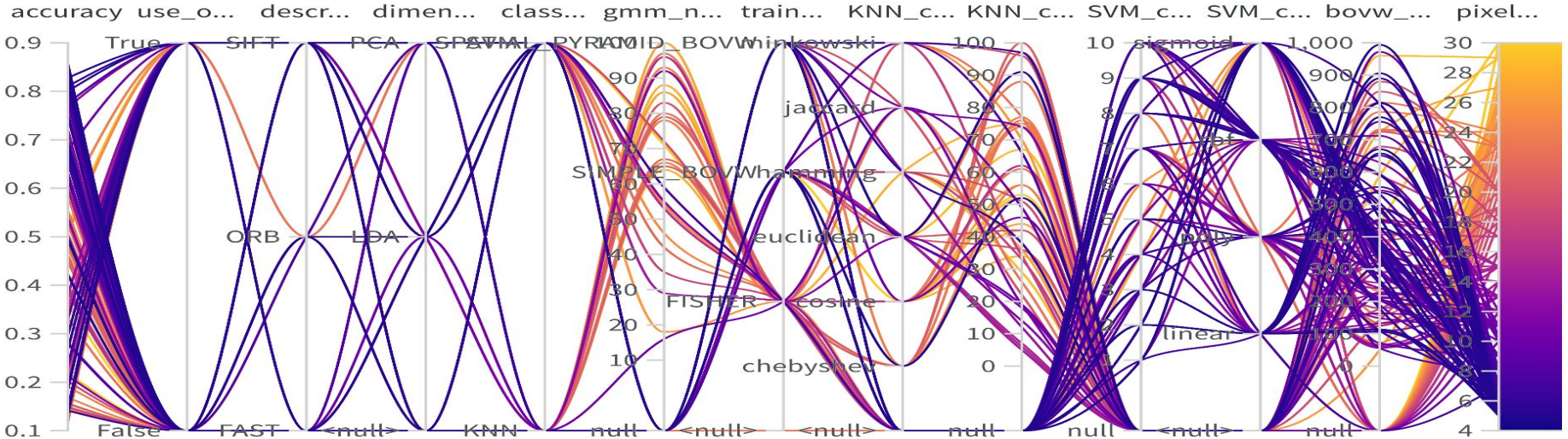
5 Fold Cross validation Result

K=1	Acc = 81.16%
k=2	Acc = 84.57%
k=3	Acc = 79.52
k=4	Acc = 83.24%
k=5	Acc = 85.10%

Average score $= \frac{1}{n}\sum_{i=1}^{n} acc_i$ = 82.72%

Conclusion

Bag of Visual Words for Image Classification



In this study, we systematically optimized the Bag-of-Visual-Words (BoVW) model for image classification. Our findings highlight the Scale-Invariant Feature Transform (SIFT) descriptor as the most significant hyperparameter, critically influencing accuracy.

We employed the Optuna framework for rigorous hyperparameter tuning, achieving a notable accuracy of 85.63%. This optimisation procedure highlighted the best balance required in the BoVW approach's selection of step size and cluster amount.

In summary, the optimization of our image classification model has been driven by a nuanced understanding of how different hyperparameters interact. Even though for certain hyperparameters we have achieved a slightly higher accuracy than the average performance highlighted in the notebook, when performing cross validation the chosen hyperparameters performed the best.