Dataset

The Caltech-101 dataset is utilized, which comprises of images categorized into 101 distinct object classes. Each category typically contains between 40 and 800 images, with most classes having approximately 50 images. The image resolution is approximately 300×200 pixels. Figure: 1 shows sample images from the dataset.

Figure 1: Sample images from Caltech-101

Each image in the Caltech-101 dataset is meticulously labeled with its corresponding object category, ensuring precise and dependable ground truth for training and evaluation purposes.

Method 1: Convolution Neural Networks

Data preparation: The dataset undergoes modifications to focus on fine-grained classification, including the removal of the *BACKGROUND_Google* class. This results in a dataset of 8677 images. To address the irregular shapes of the images, they are standardized and resized to a uniform size of (224,224). Subsequently, the dataset is divided into training (60%), testing (20%), and validation (20%) sets.

Basic CNN model:

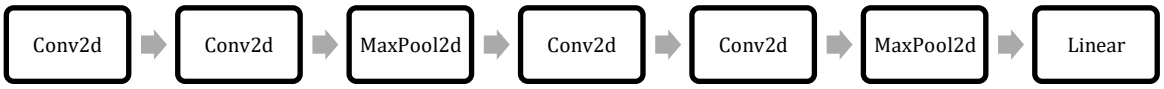

Figure 2: Architecture of the CNN model

Initially, a simple CNN model with 7 layers is considered, as shown in Figure: 2. The model utilizes ReLU activation functions and employs the Max pooling technique without dropout layers. Cross-Entropy is consistently used as the loss function. Initially, the hyperparameters include a batch size of 16, 15 epochs, and a learning rate of 0.0001, and the Adam optimizer.
As depicted in Figure 3, the validation loss curve shows a gradual decrease without flattening, while the gap between train and validation losses widens, indicating the model is experiencing overfitting. To address this, L2 regularization using Stochastic Gradient Descent (SGD) with an initial weight decay of 0.001 is used. Additionally, the batch size is increased to 64.

Hyperparameter exploration: Grid search is conducted over a range of learning rates and weight decay values to identify the optimal parameters. As seen from Figure: 4, the optimal learning rate =0.001 and weight decay=0.0001. Using these parameters, for 20 epochs, the train accuracy achieved 26.78% and the validation accuracy achieved 24.21%

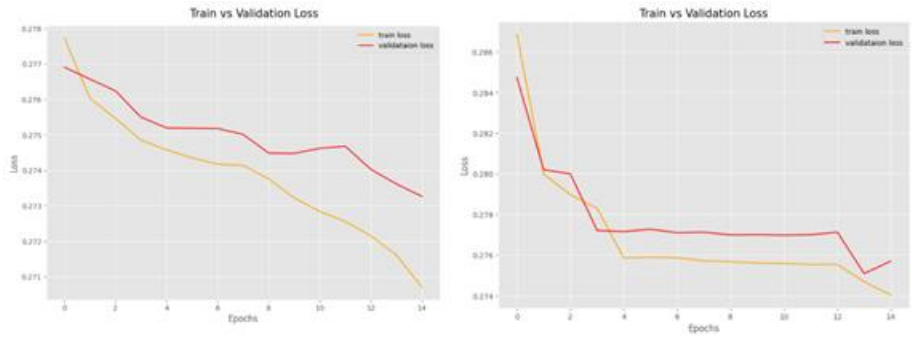

Figure 3: Train/Validation Loss Curve before (left image) & after (right image) regularization.

Fine-tuning pre-trained ResNet34 model:

Given the limited size of the dataset, training a network from scratch was expected to yield poor results. Transfer learning proves beneficial in such scenarios, significantly improving accuracies and enhancing model performance. The approach involves leveraging a pre-existing neural network trained on a large, relevant dataset and fine-tuning its weights using our smaller dataset, which shares similarities. In this case, a pretrained ResNet34 model [2] from ImageNet is utilized, a choice driven by the similarity between ImageNet and Caltech-101 in terms of object categories. Initially, a batch size of 16 for 15 epochs with a learning rate of 0.0001 using the Adam optimizer is adopted.

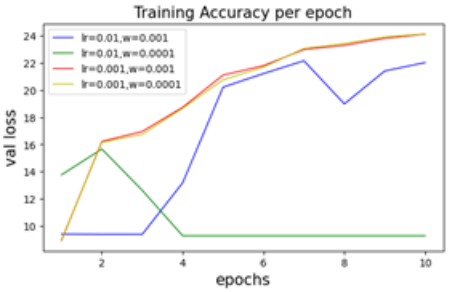

Figure 4: Training Accuracy per epoch

The model is trained and a training accuracy of 99.11% and validation accuracy of 94.58% is achieved. Further, the model is enhanced by incorporating data augmentation, dropout layers, and L2 regularization (using previously determined optimal values (lr=0.001, weight_decay=0.0001), resulting in an improved training accuracy of 98.81% and validation accuracy of 96.89%.

Results: The simple CNN model, having fewer layers and parameters, may struggle to capture intricate dataset patterns effectively, resulting in a testing accuracy of 35.56% after 30 epochs. Conversely, the tuned ResNet model, with its deeper and sophisticated architecture, achieves a much higher testing accuracy of 95.57% in just 5 epochs. This rapid improvement suggests that the ResNet model is adept at learning meaningful representations due to its deeper structure and utilization of techniques like residual connections, which help address issues like vanishing gradients during training. The slower progress of the simple CNN model indicates that it may require additional training time or adjustments to architecture and hyperparameters for enhanced performance.

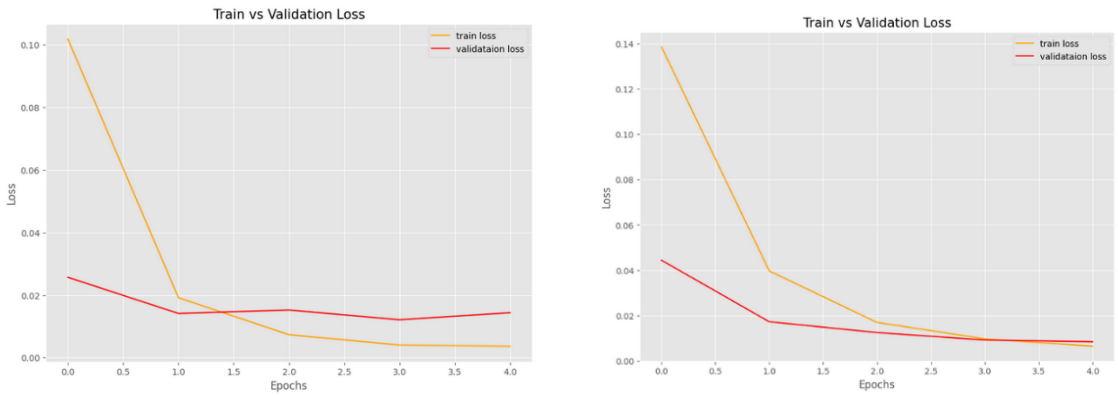

Figure 5: Train/Validation Loss Curve before (left image) & after (right image) fine-tuning.

Future work: One can enhance accuracy and robustness of the model through ensemble learning by combining predictions from multiple models, such as different CNN or ResNet variants, using bagging or boosting methods. Model configurations can be optimised by changing the learning rates, batch sizes, and other parameters systematically using advanced hyperparameter optimization methods like Bayesian optimization or genetic algorithms. To dynamically adjust the learning rate based on predefined rules or schedules, implement a learning rate scheduler that aims to improve training convergence and enhances the model performance.

Method 2: Local Feature Extraction:

Data preparation: As mentioned previously, BACKGROUND_Google class is removed and the dataset is standardized and resized to a uniform size of (224,224). The images are also converted from RGB to grayscale. The entire dataset is split into training (80%) and testing (20%) sets.

Scale-Invariant Feature Transform (SIFT)

SIFT [3] is a popular computer vision algorithm used for detecting and describing distinctive keypoint features in images and is widely used for object recognition due to its robustness against changes in scale, rotation, and illumination. Figure 6 shows the steps involved in object classification using SIFT feature extraction. K-means clustering is used throughout for clustering and for classification, Support Vector Machine (SVM) , K-Nearest Neighbor (K-NN), Random Forest (RF) and Naïve Bayes (NB) classifiers are experimented.

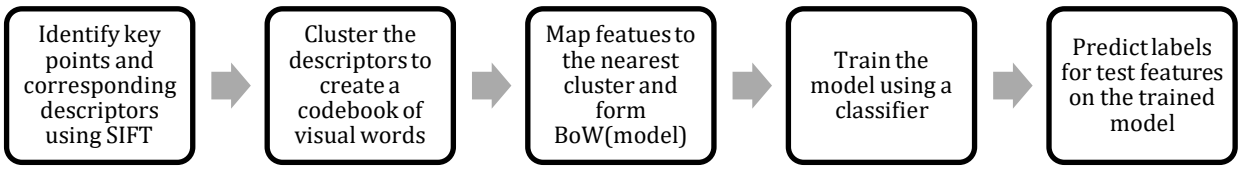

Figure 6: Simplified flowchart of Object Classification using SIFT

Hyperparameter exploration: Initially 5 classes= ['airplanes', 'Motorbikes', 'Faces', 'bonsai', 'Leopards'] are considered for feature extraction. Totally 8,55,440 features are extracted. To find the optimal value of k (number of clusters) during clustering, the Elbow method is used where the elbow point represents the optimal number of clusters.

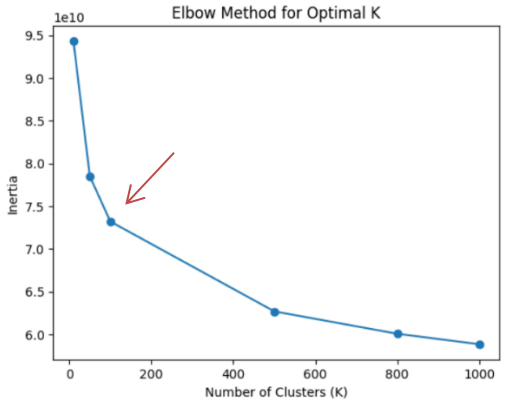

Figure 7: Find optimal k

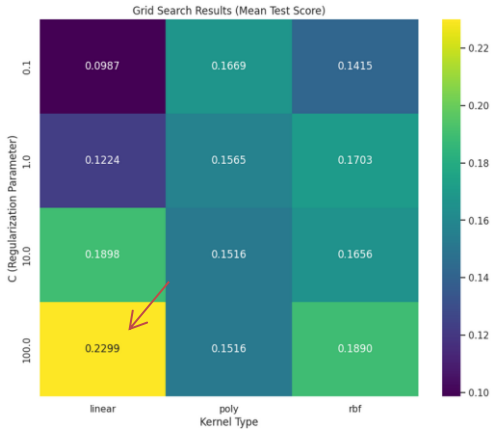

Figure 8: Find optimal parameters for SVM

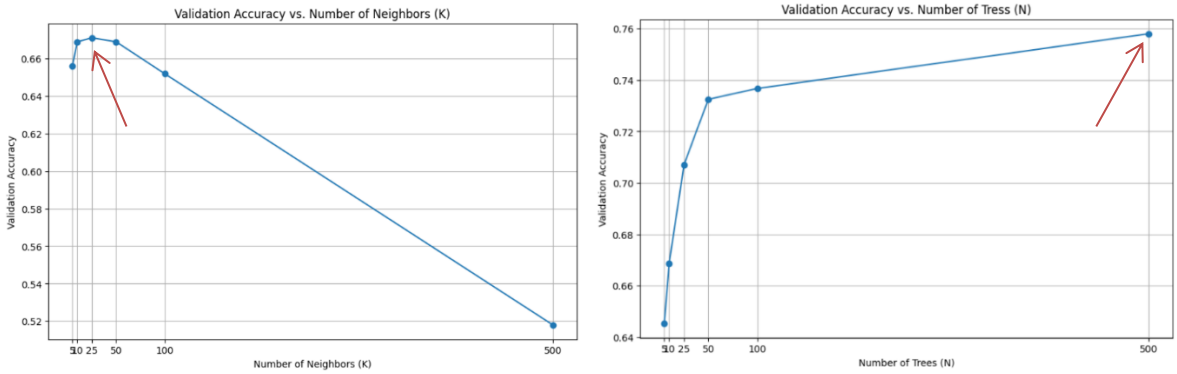

Figure 9: Random search for neighbors in KNN (left) and number of trees in RF (right).

The optimal value is 100 as seen from figure 7. Similarly, for each classification method, optimal values are found and accuracy is calculated. Figure 8 shows grid search technique applied to find optimal parameters for SVM. Figure 9 shows random search technique applied to find optimal parameters for K-NN and RF classifiers. The red arrows in the figures show the optimal value.
Similarly, the above steps are repeated for 10 classes = ['airplanes', 'Motorbikes', 'Faces', 'bonsai', 'Leopards','brain','ketch','chandelier','hawksbill','grand_piano']. Based on Table 1, it is evident

that accuracy decreases with an increase in the number of classes. We attempted to apply the algorithm to 100 classes; however, due to its high computational cost, this was not feasible. To address this challenge, we opted for ORB.

Classifier	Accuracy (5 classes)	Accuracy (10 classes)
K-NN	66.88%	58.50 %
RF	64.54%	56.74%
NB	73.67%	57.97 %
SVM	85.38%	69.42%

Table 1: Classifier vs Test accuracy for 5 classes and 10 classes

Oriented FAST and Rotated BRIEF

ORB [4] is a feature detection and description algorithm developed to address the limitations of existing methods like and SURF (Speeded-Up Robust Features). ORB combines the efficiency of FAST (Features from Accelerated Segment Test) keypoint detection with the robustness of BRIEF (Binary Robust Independent Elementary Features) descriptors, making it suitable for real-time applications requiring fast and reliable feature extraction.

		SIFT(5)	SIFT(10)	ORB(5)	ORB(10)	ORB(100)
KNN	Accuracy	0.669	0.580	0.510	0.4325	0.187
	Precision	0.73	0.28	0.53	0.27	0.07
	Recall	0.62	0.31	0.44	0.23	0.05
	F1-score	0.62	0.26	0.40	0.21	0.04
NB	Accuracy	0.737	0.580	0.510	0.43432	0.139
	Precision	0.67	0.47	0.46	0.27	0.08
	Recall	0.72	0.50	0.49	0.31	0.07
	F1-score	0.69	0.45	0.44	0.26	0.05
RF	Accuracy	0.758	0.646	0.510	0.5499	0.209
	Precision	0.80	0.39	0.46	0.36	0.03
	Recall	0.65	0.36	0.46	0.27	0.04
	F1-score	0.68	0.33	0.45	0.28	0.03
SVM	Accuracy	**0.853**	**0.694**	**0.648**	**0.583**	**0.236**
	Precision	0.84	0.59	0.51	0.29	0.06
	Recall	0.80	0.51	0.53	0.29	0.05
	F1-score	0.82	0.52	0.51	0.28	0.04

Table 2: Comparative analysis of SIFT and ORB performance across various classifiers and classes

The object classification process in this method follows similar steps as before, but with the substitution of ORB for SIFT in local feature extraction. A total of 2,75,202 features were extracted using ORB. The hyperparameter exploration conducted for ORB is similar to that of SIFT. Furthermore, we extended this exploration to 100 classes, demonstrating ORB's efficiency with significantly reduced running time compared to SIFT.

Results: Based on Table 2, SVM demonstrates superior performance in terms of both accuracy and precision compared to other classifiers. Despite ORB's faster runtime, it achieves lower accuracy than SIFT, likely attributed to the fewer number of features extracted by ORB.

Future work: Improving object classification through local feature extraction can benefit from data augmentation techniques to enrich the dataset and capture diverse local features. Additionally, combining multiple types of local features such as SIFT, SURF, and ORB can harness complementary information, enhancing robustness in classification tasks.

Comparison

The ResNet model achieved a significantly higher accuracy of 95.57% compared to ORB, which achieved a much lower accuracy of 23.6%. This stark difference highlights the superior performance of deep learning-based methods like ResNet over traditional feature-based methods like ORB in object recognition tasks. Nonetheless, both methods have their own strengths and limitations as show in Table 3.

	Deep Learning model	Traditional Feature-based model
Strength	-Automatically deduces from raw data complex patterns and hierarchical features. -Able to handle large-scale datasets and adapt well to a variety of image variants.	-Computationally efficient and operates in real-time. -Relies on handcrafted features, making it less dependent on large amounts of training data.
Limitation	-Computationally expensive and time-consuming. -Prone to overfitting with insufficient data or inadequate regularization.	-Limited capability to learn complex and abstract features. -Prone to performance degradation on datasets requiring high-level semantic understanding and those with large fluctuations in scale, rotation, or lighting.

Table 3: Comparison of Deep Learning and Feature-based model.