COMP 61342

Student number: 11356590

I. Introduction

In the rapidly evolving field of machine learning, the importance of robust datasets is paramount for both training and evaluating algorithms. This study utilizes the CIFAR-10 dataset, a standard benchmark that comprises 60,000 32x32 color images divided into ten distinct classes. Each class represents a different category, including airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks, with the dataset split into 50,000 training and 10,000 testing images.

This rich diversity and complexity of the CIFAR-10 dataset introduce unique challenges and opportunities for assessing the capabilities of traditional computer vision techniques and modern deep learning approaches. The modest resolution of the images combined with their variable orientations and scales makes CIFAR-10 an ideal testbed for evaluating the robustness and efficiency of different image recognition methods. This introductory analysis sets the stage for a detailed exploration of how these algorithms perform against this dataset and highlights the advantages of using advanced deep learning models, particularly convolutional neural networks (CNNs), in complex visual data interpretation tasks.

II. Traditional Computer Vision Algorithms

A. Choice of Data Extraction Methods:

Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Histogram of Oriented Gradients (HOG), and Local Binary Patterns (LBP) are prominently recognized feature extraction techniques. However, the effectiveness of each method can vary significantly depending on the characteristics of the dataset in use.

The CIFAR-10 dataset comprises images that exhibit various object sizes and orientations, which presents challenges. Methods like SIFT and SURF, tend to perform better on images where key points are distinct and prominently featured(Hambunan, Fernandez and Mendoza, 2018). Given the small size of images and the subtle key point distinctions within the CIFAR-10 dataset, these methods may not always yield optimal results.

Consequently, I selected HOG and LBP to facilitate data extraction.

- **HOG** is particularly adept at capturing edge and gradient structures that are vital for recognizing and differentiating objects in images. This method efficiently outlines the shapes and forms within an image, making it easier to identify objects despite their reduced scale**.**
- **LBP** chosen for its ability to capture image textures through patterns of local pixel intensity comparisons, proves highly effective when images exhibit varying and intricate surface patterns. This method can distinguish between similar shapes with different textural features, offering a significant advantage in complex visual scenes like those found in the CIFAR-10 dataset.

B. Analysis of Color Spaces:

In our approach to feature extraction, I initially utilize both RGB and HSV color spaces to leverage their distinct advantages in preserving and separating color information. Here is how I process these color spaces:

- **RGB:** : Directly derived from image data, useful for processing color characteristics.
- **HSV:** Unlike RGB, the HSV (Hue, Saturation, Value) color space separates the color information (hue and saturation) from the brightness (value). The separation of color characteristics from brightness is beneficial for

applications that require distinct analysis of these attributes.

Despite the initial use of these color spaces, we convert images to grayscale before feature extraction. This conversion is because:

- **Simplification for Algorithmic Processing**: Many traditional computers vision (CV) algorithms, including HOG LBP, are optimized for grayscale images. These algorithms focus primarily on detecting texture and structural patterns which are more discernible when the image is stripped of color.
- **Enhanced Feature Detection**: Grayscale images emphasize intensity variations rather than color differences, which helps in more accurately detecting edges, shapes, and textures. This is beneficial for CIFAR-10, as it contains a wide variety of objects and scenes where such features are pivotal for effective classification.
- **Computational Efficiency**: Processing images in grayscale reduces computational load by eliminating the need to handle three color channels, thereby speeding up the feature extraction process without sacrificing the quality of the features extracted.

By using RGB and HSV for initial processing and then converting to grayscale for feature extraction, I could ensure that models are not only fed with high-quality data but are also operating under conditions optimized for speed and accuracy. This methodical approach to color space utilization and transformation significantly enhances the performance of our object recognition tasks.

C. Parameter Tuning:

Effective parameter tuning is essential for optimizing the performance of feature extraction methods used on CIFAR-10's diverse image set.

For **HOG**, I adjusted the parameters to better capture the dataset's details:

- **Orientations:** Increased to twenty from a coarser setting of eight to enhance angular resolution, which helps in capturing edge directions more precisely.
- **Pixels Per Cell:** Reduced to (8, 8) from (16, 16), allowing for finer spatial information capture within the images.
- **Cells Per Block:** Maintained at (1, 1), focusing on maximizing the granularity of the gradient and edge detection.

By refining the parameters, the accuracy of object recognition has significantly improved, rising from near 40% to over 50%. This enhancement handles the dataset's challenges and complexities.

For Local Binary Patterns (LBP), I configured the feature extraction with:

- **Radius:** Set at 3 pixels to achieve an optimal balance between texture detection granularity and noise sensitivity.
- **Sampling Points:** twenty-four points used to capture detailed local textural patterns without being overwhelmed by the dataset's variability.

This LBP setup enhances texture analysis capability, significantly improving object recognition by highlighting subtle textural differences that are crucial for differentiating between the varied categories within CIFAR-10. This approach ensures robustness and maintains high performance across the dataset’s assorted content.

D. Model Selection:

For this study, two models were evaluated:

- **Support Vector Machine(SVM)** with an RBF kernel was selected for its effectiveness in high-dimensional spaces, typical of features from HOG and LBP. Using cross-validation, the optimal parameters were found to be C=10 and gamma=0.1. A higher C value enforces a

stricter margin that reduces misclassification. The lower gamma helps create a smoother decision boundary, avoiding overfitting and enhancing generalization across the dataset.

- **Decision Tree Classifier** was chosen for its interpretability and straightforward approach to classification using feature attributes from the dataset.
  I set max_features=None, allowing the classifier to consider all features at each split to make the best possible decisions. This approach aims to prevent overfitting while ensuring that the model captures enough detail to accurately classify the images.

E. Preliminary Results: Model Performance and Computational Efficiency

SVM Model Accuracies by Feature Extraction Method and Color Space

	RGB	HSV
HOG	0.56	0.44
LBP	0.17	0.13

Table 1: SVM Model Performance

Decision Tree Classifier Accuracies by Feature Extraction Method and Color Space

	RGB	HSV
HOG	0.22	0.17
LBP	0.17	0.14

Table 2: Decision Tree Classifier Performance

SVM with HOG features in the RGB space achieved the best result of 0.56. This suggests that the SVM approach to handling complex feature interactions is particularly beneficial for detailed image analysis. However, its computational demand is significant, requiring approximately fifteen minutes to train due to the complex calculations involved in managing the large feature set and optimizing the margin between classes.

In contrast, the Decision Tree Classifier, while less accurate offers computational efficiency. With a training time of about one minute, it presents quicker execution times make it suitable for applications requiring faster decision-making with acceptable accuracy.

III. CNN for Vision

A. Network Design:

There are four different configurations of convolutional neural networks, which are the architectures range from three-layer to six-layer networks, have utilized in this coursework.

- **Three-layer CNN**: These Configuration employ Rectified Linear Unit (ReLU) or Exponential Linear Unit (ELU) activation functions across all convolutional layers to introduce non-linearity while maintaining computational efficiency. The architecture is as follows:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_1 (Conv2D)	(None, 16, 16, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_1 (Dropout)	(None, 8, 8, 64)	0
conv2d_2 (Conv2D)	(None, 8, 8, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_2 (Dropout)	(None, 4, 4, 128)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 128)	262,272
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

Total params: 1,070,431 (4.08 MB)
Trainable params: 356,810 (1.36 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 713,621 (2.72 MB)

Table 3: The architecture of Three-Layer CNNs

- **Six-layer CNN:** To evaluate deeper networks, a six-layer CNN was also developed, using ReLU and ELU activation functions to compare their effectiveness in deeper network contexts.

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 32, 32, 32)	896
activation (Activation)	(None, 32, 32, 32)	0
batch_normalization (BatchNormalization)	(None, 32, 32, 32)	128
conv2d_7 (Conv2D)	(None, 32, 32, 32)	9,248
activation_1 (Activation)	(None, 32, 32, 32)	0
batch_normalization_1 (BatchNormalization)	(None, 32, 32, 32)	128
max_pooling2d_6 (MaxPooling2D)	(None, 16, 16, 32)	0
dropout_8 (Dropout)	(None, 16, 16, 32)	0
conv2d_8 (Conv2D)	(None, 16, 16, 64)	18,496
activation_2 (Activation)	(None, 16, 16, 64)	0
batch_normalization_2 (BatchNormalization)	(None, 16, 16, 64)	256
conv2d_9 (Conv2D)	(None, 16, 16, 64)	36,928
activation_3 (Activation)	(None, 16, 16, 64)	0
batch_normalization_3 (BatchNormalization)	(None, 16, 16, 64)	256
max_pooling2d_7 (MaxPooling2D)	(None, 8, 8, 64)	0
dropout_9 (Dropout)	(None, 8, 8, 64)	0
conv2d_10 (Conv2D)	(None, 8, 8, 128)	73,856
activation_4 (Activation)	(None, 8, 8, 128)	0
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 128)	512
conv2d_11 (Conv2D)	(None, 8, 8, 128)	147,584
activation_5 (Activation)	(None, 8, 8, 128)	0
batch_normalization_5 (BatchNormalization)	(None, 8, 8, 128)	512
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 128)	0
dropout_10 (Dropout)	(None, 4, 4, 128)	0
flatten_2 (Flatten)	(None, 2048)	0
dense_4 (Dense)	(None, 10)	20,490

Total params: 309,290 (1.18 MB)
Trainable params: 308,394 (1.18 MB)
Non-trainable params: 896 (3.50 KB)

Table 4: The architecture of Six-Layer CNNs

The choice of ReLU and ELU across different models was aimed at assessing their performance in handling non-linearities in deep learning tasks.

B. Activation Functions and Hyperparameter

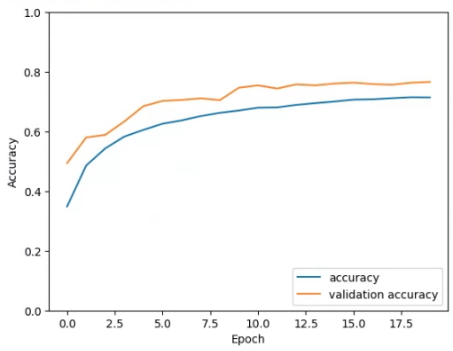
- **Settings and Activation function:** The network leverages ReLU and ELU activation functions, chosen for their benefits in neural network training. ReLU is favored for its computational efficiency and ability to speed up the convergence of stochastic gradient descent, significantly more than sigmoid or tanh functions. In contrast, ELU is employed to mitigate the dead neuron problem inherent in ReLU, allowing small negative outputs when the input is less than zero, thereby potentially increasing the model's robustness.
- **Hyperparameter Tuning and Learning Rate Strategy:** Hyperparameters were carefully tuned, as recommended by Nestorojeda (2024), focusing particularly on the learning rate, to optimize training outcomes. An exponential decay schedule for the learning rate was implemented to moderate the learning speed across epochs, ensuring high initial learning rates that taper for finer tuning as training progresses.
- **Batch Size Considerations:** The impact of batch size on model training was thoroughly examined. While smaller batch sizes tend to provide a regularizing effect and reduce generalization error, they may prolong training time and add volatility in convergence. Conversely, larger batch sizes enhance computational efficiency through better use of parallel processing, although they might result in less optimal training paths.
- **Systematic Experimentation and Comparative Analysis:** Through systematic experimentation that varied these parameters, the study undertook a comparative analysis of their effects. Initial tests varied batch sizes (e.g., 32, 64, 128) and dynamically adjusted the learning rate to gauge their impact on the model's training dynamics. The findings from these tests are supported by accuracy and loss metrics over training epochs, illustrated in detailed plots to provide clear visual evidence of the optimal configurations.

This approach to hyperparameter adjustment not only clarifies their influence but also demonstrates how precise changes can significantly enhance model performance, particularly for complex vision tasks such as those presented by the CIFAR-10 dataset.

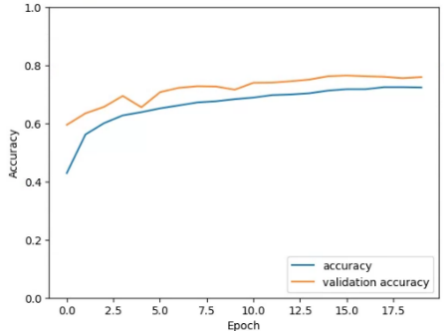
C. Performance Metrics

The training process highlighted the strengths of each configuration, evidenced by the following training and validation accuracy plots for each model, showcasing how accuracy evolved over epochs:

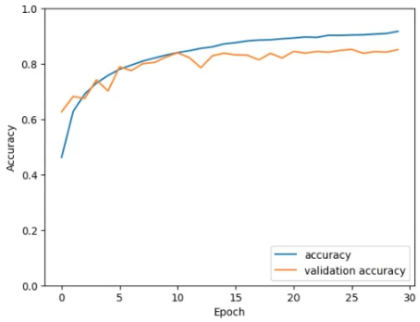
- **Training and Validation Accuracy for Three-layer CNN with ReLU**:



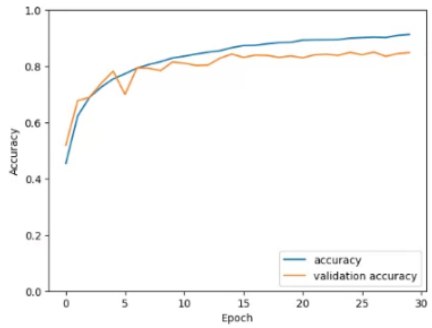
- **Training and Validation Accuracy for Three-layer CNN with ELU**:



- **Training and Validation Accuracy for Six-layer CNN with RELU**:



- **Training and Validation Accuracy for Six-layer CNN with ELU**:



Final test accuracies after training completion were:

- **Three-layer CNN with ReLU:** 0.76
- **Three-layer CNN with ELU:** 0.75
- **Six-layer CNN with ReLU:** 0.84
- **Six-layer CNN with ELU:** 0.84

The final test accuracies demonstrate that the six-layer CNN models, both with ReLU and ELU activation functions, achieved an accuracy of 0.84, showing superior performance compared to the three-layer models, which recorded accuracies of 0.76 for ReLU and 0.75 for ELU. This indicates that for the CIFAR-10 dataset, increasing the network depth can indeed enhance performance when combined with effective activation functions. This outcome suggests that the selection of network depth and activation functions should be strategically aligned with the dataset's characteristics and computational capabilities.

IV. Comparative Analysis of Recognition Results

A. Performance Comparison: Direct comparison of traditional computer vision algorithms and deep learning approaches such as convolutional neural networks (CNNs) reveal significant differences in performance and applicability. While the SVM with HOG features achieved the highest accuracy among traditional methods at 0.56, it still falls short compared to the CNN outcomes. Six-layer CNN models, regardless of using ReLU or ELU activation functions, reached accuracies up to 0.84 on the CIFAR-10 dataset. This underlines the enhanced capability of CNNs in managing more complex image data and extracting relevant features for accurate classification.

B. Advantages and Limitations:

- **Traditional CV Algorithms** enhanced interpretability and faster training times, which are valuable in scenarios where transparency and operational efficiency are key considerations. However, traditional CV algorithms face limitations when applied to more complex datasets like CIFAR-10, as they typically achieve lower accuracy and extensive feature engineering to perform effectively. This trade-off highlights the need to balance between model simplicity and its ability to handle complicated tasks.

- **CNN** models automate feature extraction and significantly improve accuracy on complex datasets, and scalable and adaptable to a wide range of image-related applications. However, the higher computational cost and the opaque nature of their decision-making processes can be problematic in resource-limited or transparency-demanding scenarios.

V. Conclusion

In this study, I observed that the accuracy of CNN models is significantly higher than traditional models. This is because CNNs are more effective at learning and processing the spatial hierarchies and features in image data. The use of convolutional layers in CNNs, which preserve the spatial relationships in images, makes these networks particularly well-suited for image recognition tasks (Ruiz-del-Solar, Loncomilla and Soto, 2018). Furthermore, CNNs reduce the number of parameters in the model significantly by pooling layers, which condense the data by reducing its spatial dimensions, and dropout, which randomly deactivates certain neurons during training. This combination not only enhances computational efficiency but also effectively reduces the risk of overfitting (Ruiz-del-Solar, Loncomilla and Soto, 2018).

Traditional models, such as those based on feature extraction techniques, typically rely on manually designed features and complex preprocessing steps. These methods may not be flexible or powerful enough to handle high-dimensional data and non-linear problems. In contrast, CNNs can automatically learn useful features directly from raw image data, eliminating the need for specialized feature extraction processes. This enables the models to perform better across a variety of complex visual tasks, showing greater adaptability and accuracy.

As the field of deep learning continues to advance alongside improvements in hardware technologies, its applications within robotic vision have become increasingly pivotal. By employing CNNs and other sophisticated deep learning frameworks, researchers have developed robotic vision systems that maintain stable operation across varied environmental conditions. Additionally, innovative methods like Generative Adversarial Networks with autoencoders are being utilized to generate training data, particularly beneficial in scenarios where data is scarce (Ger, Jambunath, Yegna Subramanian, and Klabjan, 2019).

In practical applications, autonomous vehicles exemplify one of the most extensive implementations of deep learning within robotics. These vehicles depend on deep learning algorithms to process and interpret complex data streams from multiple sensors, achieving an accurate perception of their surroundings. Advanced object detection and tracking methods, vital for tasks such as environmental monitoring, precision agriculture, and traffic management, have been successfully integrated into unmanned aerial vehicles (UAVs), further demonstrating the critical role of deep learning in enhancing robotic adaptability (Wu et al., 2022).

Moreover, deep learning techniques have shown promising results in the domain of multi-robot communication and coordination. This area involves strategic methodologies and technologies that facilitate collaborative efforts among robots towards a shared objective. Key aspects include effective communication—the exchange of information such as status updates and task assignments crucial for coordinated decision-making—and the management of joint actions, which requires the synchronization of activities and sharing of resources to optimize operational efficiency and effectiveness.

A notable advancement in this field is the development of a cloud-based framework by Doriya et al. (2015), which employs Particle Swarm Optimization (PSO) to enhance cooperation among robots in clustered environments. This framework leverages Cluster Head Gateway Switch Routing (CGSR) to refine communication strategies, thereby enabling robots to share information seamlessly and collaborate on intricate tasks like area exploration and target tracking (Doriya et al., 2015).

In conclusion, this study highlights the superiority of deep learning, especially convolutional neural networks (CNNs), over traditional computer vision methods in handling complex image recognition tasks when it presented by the CIFAR-10 dataset. CNNs excel at processing high-dimensional data and automating feature extraction, offering more accurate and adaptable solutions for applications such as autonomous driving and robotic vision systems.

I. References

nestorojeda (2019). GitHub - nestorojeda/CIFAR-10-CNN: Convolutional neural network to classify the CIFAR10 dataset. [online] GitHub. Available at: https://github.com/nestorojeda/CIFAR-10-CNN

Ruiz-del-Solar, J., Loncomilla, P. and Soto, N. (2018). *A Survey on Deep Learning Methods for Robot Vision*. [online] arXiv.org. Available at: https://arxiv.org/abs/1803.10862

Ger, S., Jambunath, Yegna Subramanian and Klabjan, D. (2019). *Autoencoders and Generative Adversarial Networks for Imbalanced Sequence Classification*. [online] arXiv.org. Available at: https://arxiv.org/abs/1901.02514

Hambunan, G., Fernandez, G. and Mendoza, E. (2018). *Performance Comparison on Various Image Deformations based on Match Ratings using ORB, BRIEF, SURF and SIFT*. [online] Available at:http://www.ejikei.org/Conf/ICTCBW2018/proceedings/materials/proc_files/GS_papers/GS_A010/Camera_ready_ICTCBW2018_GS_A010.pdf

Wu, X., Li, W., Hong, D., Tao, R. and Du, Q. (2022). Deep Learning for Unmanned Aerial Vehicle-Based Object Detection and Tracking: A survey. *IEEE geoscience and remote sensing magazine*, [online] 10(1), pp.91–124. doi:https://doi.org/10.1109/mgrs.2021.3115137.

Rajesh Doriya, Mishra, S. and Gupta, S. (2015). A brief survey and analysis of multi-robot communication and coordination. [online] doi:https://doi.org/10.1109/ccaa.2015.7148524.

IX. Appendices

- Code