SIFT And CNN in Objects Classcification

COMP61342&41342 Cognitive Robotics And Computer Vision Assignment Report

Yuyi Yang
yuyi.yang@postgrad.manchester.ac.uk
p68393yy

I. Introduction

In recent years, equipping robots with reliable perception capabilities has become a foundation for autonomous manipulation, navigation, and human–robot interaction. Object recognition, identifying and localizing objects from camera data, remains especially challenging in unstructured, dynamic environments where lighting, occlusion, and viewpoint changes can drastically reduce accuracy.

Traditional computer-vision pipelines, based on handcrafted local features (i.e. SIFT) and classifiers (e.g. SVM, K-NN), offer interpretability and low inference latency; however, they may falter when faced with complex environment and large intra-class variability. Conversely, convolutional neural networks (CNNs) have revolutionized vision performance, automatically learning hierarchical features, but at the cost of greater computational demand and complex hyperparameter tuning. Despite extensive research in object recognition, systematic comparisons between traditional handcrafted methods and CNN-based approaches remain limited. Thus, this work aims to fill this gap by systematically benchmarking and analyzing these paradigms, providing practical insights for researchers and practitioners in robotic perception.

This work systematically compares CNN models and Local Feature methods (i.e., SIFT+BoVW) on the CIFAR-10 [1] and iCubWorld 1.0 [2] benchmark, researching how choice of feature extractor, classifier, network architecture, and training strategy affect performance of classification accuracy and inference speed, and model robustness to environmental changes.

The rest of the report is organised as follows: Section 2 makes a literature review on CNNs, Section 3 describes the methodology, detailing the model selection, design, and feature extraction process; Section 4 rpresents comprehensive experiments and results on both datasets; Section 5 offers an in-depth comparative discussion, critically evaluating accuracy, inference speed, interpretability, and robustness to environmental variations; and Section 6 concludes with future directions.

II. Literature Review

Robotics Vision is always an important but challenging task in the field of computer vision. Considering that the vision is an esstential way for both human begings and robots to perceive the world, it is important to make the robot understand the world through vision. In the past decades, scientists and researchers have been trying to seek an effective and stable way to achieve this goal. The mainstream methods can be divided into two categories:

- Traditional Local Feature Extraction Methods
- Deep Learning Methods

III. Methodology

A. Dataset

CIFAR-10 and iCubWorld 1.0 are two datasets used in this project for object classification tasks. CIFAR-10 is a well-known dataset containing 60,000 32x32 color images (10,000 for test) in 10 classes, with 6,000 images per class. While, iCubWorld 1.0 is a dataset collected by the iCub robot, including 10 categries, each categries contains 3 instances in the training set. Both dataset are famous benchmark in the field of computer vision, which is very suitable to compare the performance of local feature and deep learning methods. For each dataset, I applied corresponding data pre-processing methods separately, trying to improve the accuracy and robustness of the model, and compare the performance of different pre-processing methods.

1) Resizing: From the perspective of datasets, I applied different resizing approaches, since the image size variance. For CIFAR-10, the original image size is only 32x32, which is definitely too small for SIFT method, whose basic patch size is 16x16. Therefore, I resized the images to 128x128 to ensure that it can get enough SIFT descriptors from the images. In this process, I choose the *Bicubic Interpolation* as the interpolation method, which can produce smoother results than other methods like bilinear interpolation. For iCubWorld 1.0, the original image size is 640x480, and all of the training images are pre-cropped to 160x160. In this case, I also applied the *Bicubic Interpolation* method to resize the image to 224x224, which better fits the input size of ResNet models.

2) Normalization: In the SIFT+BoVW process, instead of doing a global "minus mean/divide variance" normalization on the whole image, we usually detect the keypoints on the original pixels and compute the SIFT descriptors, which perform L2 normalization and threshold truncation of the gradient histograms within each local patch, which itself confers partial invariance to lighting changes. In contrast, CNNs are more sensitive to the

distribution of the input data. Thus, I calculated the global mean and standard deviation of each channel (R/G/B) on the training set, then applies the z-score normalization to each image. Additionally, the BatchNorm technique is used inside the network to stabilize the training process.

3) Data Augmentation: After that, in the following experiments pipeline, data augmentation is a common technique to artificially increase the generation of training data by applying various transformations to the original images. This helps to improve the generalization ability of the model and reduce overfitting. In this project, the data augmentation methods include: *Resizing*, *Cropping*, *Flipping*, *Color Jittering*, *Rotation*, *Affining*, *Perspective Changing*, *Erasing* and *Normalization*. Different data augmentation strategies are evaluated, tested, and the best combinations are applied to the model to participate in subsequent experiments.

B. SIFT+BoVW

In this section, the brief workflow of SIFT+BoVW method will be introduced.

1) Keypoints Detection: The first step is to detect interest points in the image, which are more likely to be stable and distinctive, making them suitable for matching across different images. In order to detect these interest points for the following SIFT descriptor extraction, in this preject, I tried two different methods: Difference of Gaussian (DoG) and Harris-Laplace.

In computer vision, scale invariance is one of the key requirements for feature detection. And DoG is an efficient scale-space keypoint detection method, which approximates the Laplace-Gaussian (LoG) operator by making a subtraction between different Gaussian smoothing scales and octaves, and searches for spatial-scale extreme points in the generated DoG pyramid, obtaining points of interest with invariance to scale, which in turn significantly reduces the amount of computation.

While, for the Harris–Laplace method, its central idea lies in effectively combining the spatially accurate Harris corner detector with the scale-invariant properties of the scale-normalized Laplacian-of-Gaussian (LoG) operator. This ensures that detected interest points not only have high spatial precision but are also stable across scales. Its brief workflow is as follows:

1. The Harris corner detection algorithm is applied independently to the input image at several scales (constructed via Gaussian smoothing). This step provides precise spatial localization of corner-like structures across each individual scale.
2. For each corner candidate detected from the previous step, the LoG response is computed across scales. The scale at which the candidate corner exhibits the strongest (local extrema) response is selected as its characteristic scale, ensuring robust scale invariance.

By this way, the Harris–Laplace method effectively emphasizes corner-type interest points, which typically convey richer structural information compared to edge points, while simultaneously preserving the desirable property of scale invariance inherent in LoG-based approaches. In addition, the traditional Harris algorithm is also tested in this project, and their results are compared in the Experiments section.

2) SIFT Descriptor Extraction: After gaining the interest points in Section III-B1, the project manually implements the various steps of SIFT descriptor extraction. The following will briefly describe the SIFT workflow in the project:

1. Gradient magnitude and direction are calculated at each keypoint neighborhood through a 16*16 sized region. Then, an orientation histogram with 36 bins (each covering 10°) is constructed. The peak direction from this histogram is selected as the keypoint' s main orientation, ensuring rotation invariance.
2. After aligning the neighborhood region with the keypoint' s main orientation (rotating the local patch accordingly), the region is divided into a 4×4 grid of smaller cells. Within each of these 16 cells, gradient orientations are quantized into an 8-bin histogram covering 360°. These histograms are then concatenated, forming a $4 \times 4 \times 8 = 128$-dimensional vector.
3. Applying L_2 normalization to the 128-D vector to obtain a final descriptor that is robust to make them independent of illumination and contrast variations, thus enhancing robustness.

3) Build Bag of Visual Words: After extracting SIFT descriptors (the 128-D vector) for all the training images, all the descriptors are clustered by kMeans method to get the cluster center (i.e. visual word) of each cluster. And the dimensionality of the final BoVW image coding vector is determined by the number of target clusters, presenting the expressive capacity of the BoVW.

For each image, the descriptor calls *kmeans.predict()* to assign each descriptor to the nearest visual word; then use *np.histogram* to count the number of descriptors that fall in each cluster, and obtain a one-dimensional histogram.

Finally, the histogram is normalized to convert the frequency into a probability distribution to form the final BoVW feature vector. And these BoVW vectors are utilized as feature inputs to a classifier (e.g. SVM) to perform the image classification task (target of this project).

4) Classifier: This preject test diffetent classifiers on the BoVW feature vectors, including: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Softmax Regression. The performance of these classifiers will be compared with each other to explore the best classifier for the BoVW feature vectors.

C. CNN

CNN is selected as another approach to achieve object classification, which is more state of the art and widely used in the field of computer vision.

1) Network Architecture - Self-Designed Simple CNN: CNN Network Architecture is a key factor that affects the performance of CNN, which may influence the accuracy, speed, and robustness of the model. In this project, three different network architectures are tested:

- Self-Designed Simple CNN
- ResNet-18
- ResNet-50

All of them will implemented by self-coding, and the performance of them will be compared with each other.

The following Figure 1 is the brief architecture of the self-designed simple CNN model. It contains 2 stage, each stage contains 1 convolutional layers, 1 max pooling layer, and 1 ReLU activation layer. Then, the output of the last stage is flattened and passed to a fully connected layer. Its main character is to play as the baseline model, and

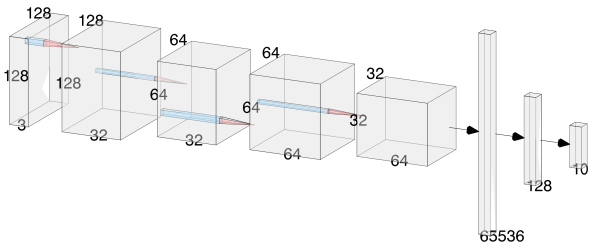


Fig. 1: Architecture of the Self-Designed Simple CNN

compare with the other two morden architectures (i.e. ResNet-18 and ResNet-50) to explore the performance of different network architectures.

layer name	output size	18-layer	50-layer
conv1	112×112	7×7, 64, stride 2	
conv2_x	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 3\times3,\ 64 \\ 3\times3,\ 64 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 64 \\ 3\times3,\ 64 \\ 1\times1,\ 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3,\ 128 \\ 3\times3,\ 128 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 128 \\ 3\times3,\ 128 \\ 1\times1,\ 512 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 3\times3,\ 256 \\ 3\times3,\ 256 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 256 \\ 3\times3,\ 256 \\ 1\times1,\ 1024 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 3\times3,\ 512 \\ 3\times3,\ 512 \end{bmatrix}\times2$	$\begin{bmatrix} 1\times1,\ 512 \\ 3\times3,\ 512 \\ 1\times1,\ 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax	
FLOPs		1.8×10^9	3.8×10^9

TABLE I: Architectures for ResNet (reproduced from [3])

2) Network Architecture - ResNet: The Table I shows the architecture of ResNet-18 and ResNet-50, which are two popular architectures in the field of computer vision [3]. Compared with the self-designed simple CNNs, ResNets are more complex and deeper, but their most notable feature is the introduction of a residual learning mechanism to alleviate the degradation problem that occurs in the training process of deep networks. For traditional deep CNN models (e.g., my self-designed CNN model), although theoretically I can enhance the model's representation ability by adding more layers and building a deeper network, they are prone to suffer from the gradient disappearance or model performance degradation in practice.

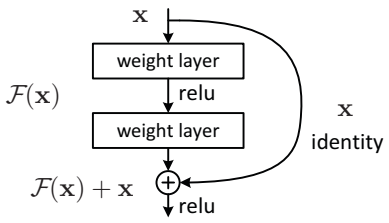
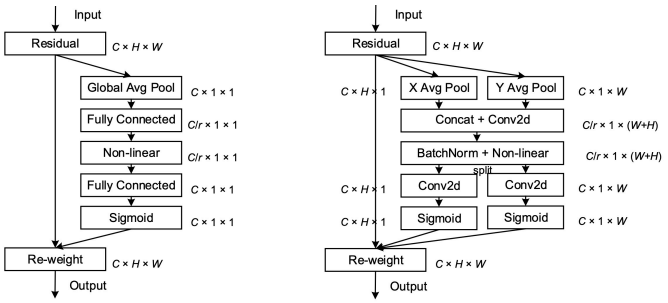


Fig. 2: Residual Block Architecture (reproduced from [3])

In order to solve this bottleneck, ResNet introduces skip connections in the network structure, as shown in Figure 2, which allows the network to learn a residual mapping, thus transforming the final objective function into the form of $H(x) = F(x) + x$. In addition, this structure also promotes the reuse of features, allowing shallow features to directly affect deeper outputs, effectively mitigating the problem of gradient attenuation in deeper networks during backpropagation.

3) Coordinate Attention: In the field of deep learning, attention mechanisms [4] have been widely used to enhance the model's performance. In this project, I also applied the Coordinate Attention (CA) mechanism [5] to the ResNets. CA is a lightweight attention mechanism designed to enhance the ability of CNNs to model spatial information while maintaining high efficiency. Unlike traditional channel attention mechanisms (e.g., SE, CBAM) that focus on "channel dimensions", Coordinate Attention encodes both "spatial location information (coordinates)" and "channel dependencies" . "channel dependencies" . The specific schematic comparison in shown in Figure 3



(a) The Raw Output (b) The Processed Output

Fig. 3: Schematic comparison of the proposed Coordinate Attention block (b) to the classic SE channel attention block [6] (a). (Adapted from [5])

IV. Experiments and Results

In the Experiment section, the focus of this project is to design a final model that meets the requirements of the project task through extensive comparison of different hyperparameters and modeling mechanisms.

A. Grid Search

First of all, this experiment was conducted on two datasets to perform a grid search for some of the basic hyperparameters of the two models. For the SIFT method, the main hyperparameters to be adjusted were the BoVW size and the maximum number of keypoints in each image. For the CNN, the main hyperparameters tuned were the batch size, the learning rate, and the epoch number. These basic hyperparameters will be used as control variables in subsequent experiments to support the experiments for more in-depth exploration (e.g., patch size in SIFT, or optimizer weight delay in CNN).

In the grid search process, due to the large amount of data (especially for CIFAR-10), it is inefficient to use the entire dataset. Thus, I applied the downsampling method to randomly select 30% of CIFAR-10 and 70% of iCubWorld 1.0 for the grid searching. In this case, there are 15000 images in CIFAR-10 and 4200 images in iCubWorld 1.0 being used for the grid search, which is representative. In addition, 80% of the data will be used as train set, while the rest of the data will be used as validation set which is used as the evaluation metric for the grid search.

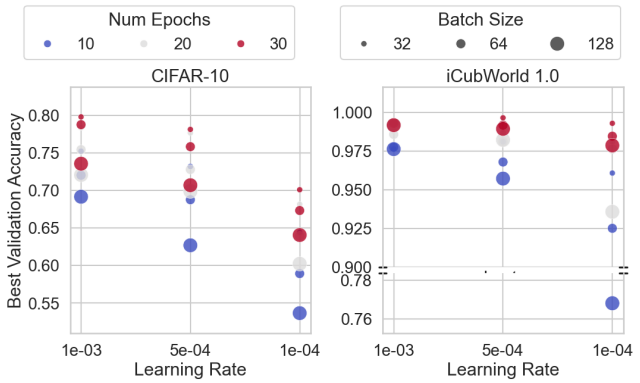


Fig. 4: Grid Search Results of CNN on CIFAR-10 and iCubWorld 1.0 (performed on Ranger optimizer)

Figure 4 shows the results of the lattice search of the CNN. It is clear that in the CIFAR-10 dataset, the distribution of the results is more regular basically following the law that the smaller the batch size, the larger the learning rate, the larger the epoch number, and ultimately the larger the accuracy. Therefore, I finally chose [batch size=32, learning rate=1e-3, epoch number=30] as the parameter combination to get the best fitting performance.

And on the iCubWorld 1.0 dataset, all parameter combinations achieve a high level of fit, except for the case of epoch=10 where underfitting may occur. At a learning rate of 1e-3, all combinations achieve 97.5% accuracy. Also, it can be noticed from the figure that batch size does not have a significant effect on this dataset (about 1%). Therefore, due to the concern of overfitting and the requirement of comprehensive performance, we finally chose, [batch size=128, learning rate=5e-4, epoch number=30] as the parameter combination on the iCubWorld 1.0 dataset.

B. SIFT

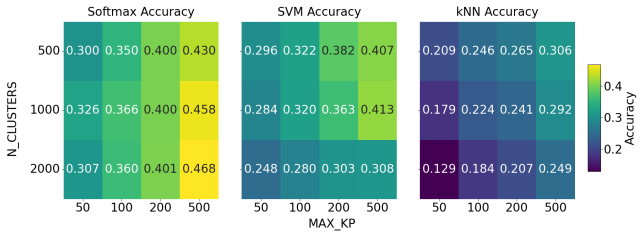


Fig. 5: Grid Search Results of SIFT on Different Classifiers

1) Grid Search: From Figure 5, we can see the grid search results of SIFT on CIFAR-10 dataset with different classifiers. It can be seen that three classifiers have different tendency with the hyperparameters.

For the SVM and kNN classifiers, the accuracy of the model grows as the maximum number of keypoints allowed (MAX_KP) grows, but this is not the case for the BoVW size (n_CLUSTERS), with the best performance of both classifiers occurring at n_CLUSTERS=1000, and larger or smaller n_CLUSTERS both lead to a decrease in model performance. As for the softmax classifier, when MAX_KP is small, it also conforms to the above law; and when MAX_KP is large, the larger the n_CLUSTERS is, the more accurate the model is.

From the theoretical perspective, the sensitivity of the three classifiers to the number of keypoints and BoVW size differs. First, as MAX_KP increases, the number of SIFT descriptors increases significantly, and the statistics of each visual word in the BoVW histogram is more stable, which enables both kNN based on the distance metric and SVM based on the maximal spacing to obtain more accurate and robust classification boundaries. Second, for the BoVW dictionary size n_CLUSTERS, both SVM and kNN degrade in performance when the dictionary is too small (underfitting) or too large (high-dimensional sparse/overfitting), but they reach a performance balance at a medium size (1000 words). Finally, Softmax linear models (with L2 regularization) show tolerance to high-dimensional BoVW when MAX_KP is very large: on the one hand, the rich descriptors populate the large-dimensional histograms, avoiding extreme sparsity; on the other hand, the regularization helps the model to suppress noisy words and to take full advantage of the finer visual word segmentation to improve the discriminative power. As a result, Softmax favors larger n_CLUSTERS under large MAX_KP conditions and still suffers from the same

sparsity/underfitting problem when descriptors are insufficient. Moreover, these phenomena are also confirmed on the iCubWorld 1.0 dataset. Therefore, I finally selected the basic hyperparameters as follows: [MAX_KP=500, n_CLUSTERS=1000] for the SIFT method on both CIFAR-10 and iCubWorld 1.0 dataset.

2) Classifiers: Figure 5 as well as shows the performance difference between the three classifiers. Overall, the softmax classifier achieves the best performance, while the kNN performs the worst. And the SVM classifier is in the middle. In addition, the three classifiers differed significantly in terms of training time, with kNN being the most efficient on the same device (taking only 0.4 seconds), softmax next (taking 1.8 seconds to complete 10 rounds of epochs), and SVM (using LinearSVC) being the slowest (taking 11 minutes for a full training session). Therefore, for the combined consideration of model performance and training efficiency, SoftMax was used as the default classifier in all subsequent tests in this project.

3) Keypoints Detection: In the SIFT pipeline, I tested four different keypoint detection methods: *Difference of Gaussian (DoG)*, *Harris-Laplace*, *Harris*, and *LoG*. The results of these methods will be compared with each other to explore the best keypoint detection method for the SIFT pipeline.

Dataset	**Harris**	**DoG**	**LoG**	**Harris-Laplace**
CIFAR-10	85.7	39.0	93.1	88.1
Categorization	76.5	21.6	80.0	23.5
Demonstrator	70.4	18.9	74.1	24.9

TABLE II: Accuracy (%) under different keypoint detection methods

By comparing the data in the table, we can find that the following phenomenon occurs on all data sets: Harris has the lowest accuracy, while DoG is slightly inferior to LoG but is generally similar, and Harris-Laplace has the highest accuracy. In addition, there are differences in the calculation time of the four. Harris' fastest processing speed is usually 30it/s, while DoG is about 27it/s, while LoG is only 21it/s, and Harris-Laplace requires a maximum computing speed of about 17it/s.

C. CNN

1) Data Pre-Processing: Data augmentation is an important technical means to enhance model performance and generalization ability in the field of computer vision. The basic idea is to apply various transformations (e.g., rotation, cropping, flipping, color perturbation, affine transformation, etc.) to the original training images to synthesize training samples with higher diversity, thus expanding the data distribution and reducing the risk of model overfitting. Especially in the context of deep learning, data augmentation can effectively alleviate the performance bottleneck caused by the insufficient number of training samples.

The figure shows the effect of different data augmentation strategies on the accuracy of the model on both datasets. It can be found that rotating and cropping the images appropriately on both datasets is positive for the final results.

On the iCubWorld 1.0 dataset, it can also be seen that the data augmentation strategy can significantly improve the model's performance on the background and robot test sets (by 20% and 100%, respectively) without drastically affecting the accuracy of the categorization and demonstrator test sets (the variation of the accuracy is about 2%); and the comprehensive performance of the model on the CIFAR-10 dataset is also improved by nearly 5%. This confirms that the data enhancement strategy brings positive results for the CNN model. However, if we manipulate the color of the image (ColorJitter) and the viewing angle (RandomPerspective) it may lead to underfitting. The final accuracy may drop directly from 85% to 65%, and the improvement cannot be realized directly by hyperparameter tuning. It is speculated that the possible reasons for this are that iCubWorld 1.0 is an image dataset acquired on a robotic platform with a clean background, stable lighting, and centered objects, and that the color features in the original dataset are very stable and discriminative (e.g., red apples vs. yellow bananas), and that the data images do not have much variation in the shooting angles during training and testing. Therefore, using the wrong data enhancement strategy in this case would artificially add label noise, destroying the direct correspondence between color and semantics; resulting in the model being forced to learn unnecessary "geometrical invariance" , and failing to build a stable representation of objects at standard angles.

2) Model Architecture: As mentioned before, this project tests three different network architectures: *self-designed simple CNN*, *ResNet-18*, and *ResNet-50*. The following Table III shows their performance on the both datasets.

Dataset	**Self-CNN**	**ResNet-18**	**ResNet-50**
CIFAR-10	83.3	85.4	88.1
iCubWorld 1.0	68.2	75.4	75.7

TABLE III: Classification accuracy (%) of models on CIFAR-10 and iCubWorld 1.0 (categorization) test sets

It can be seen that on CIAFR-10, all of the models have relatively similar performance performance, basically centered around 83% to 88%. However, it is still observed that there is a positive correlation between model complexity and test set accuracy under the selected hyperparameter conditions. This phenomenon is even more obvious on iCubWorld 1.0. Comparing self-CNN and ResNet-18, ResNet-18 shows a significant improvement of 10% on the test set alone, but ResNet-50 shows no significant improvement compared to ResNet-18. The speculative

reasoning is that the data volume limitation of iCubWorld 1.0 resulted in not enough data to support ResNet-50 for continuous learning, causing it to present similar results to the simpler ResNet-18.

3) Attention Mechanism: Attention mechanisms are used as an important component of present-band machine learning, and this implementation conducts comparative experiments with several of the previously mentioned mechanisms. In Table IV, I present part of the experiment results about the attention mechanism on ResNet-50. The results show that the attention mechanism can significantly improve the performance of the model, and by comparing the training curves, it can be seen that the model with the attention mechanism tends to have a better fitting efficiency in the pre-training period. Especially when the training with smaller learning rate, the model can achieve a significant improvement of 7% on the CIFAR-10 dataset by CA. And compared with the SE and CEAM attention mechanisms, CA can achieve a better performance on both datasets. For example, on CIFAR-10, the accuracy of the model with CA is 88.1%, while the accuracy of the model with SE and CEAM are 85.4% and 86.2%, respectively. And on iCubWorld 1.0, the accuracy of the model with CA is 75.7%, while the accuracy of the model with SE and CEAM are 75.4% and 75.5%, respectively.

Dataset	**Original**	**SE**	**CA**
CIFAR-10	85.7	88.0	88.1
Categorization	64.4	70.9	78.0
Demonstrator	81.0	84.7	89.1

TABLE IV: Accuracy (%) under different attention mechanisms

4) Optimizer: In the training process of CNN, the choice of optimizer is also a key factor that affects the performance of the model. This project mainly tests two different optimizers: *AdamW* [7], and *Ranger* [8].

By comparing the performance of the two optimizers on individual models and training sets, it was found that the difference in final accuracy between the two was not large (typically between 1 and 2%), but the Ranger optimizer had a better smoother fit, while the AdamW optimizer's fitting process had larger fluctuations. This may be caused by the weight decay mechanism of AdamW, which is used to help the model get out of the local optimum, while Ranger achieves better results by combining the RAdam and Lookahead mechanisms. Finally, the Ranger optimizer was chosen as the default option to ensure a smoother training process and facilitate the tuning of other parameters.

V. Discussion

VI. Conclusion

References

[1] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images.(2009)," 2009.

[2] S. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone, "icub world: Friendly robots help building good vision data-sets," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2013, pp. 700–705.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] Q. Hou, D. Zhou, and J. Feng, "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 713–13 722.

[6] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[7] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[8] L. Wright, "Ranger - a synergistic optimizer." https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2019.