Enhancements in Neural Networks for Fine-grained Classification

Yuhao Fang
Sun Yat-sen University
Guangzhou, P. R. China
fangyh27@mail2.sysu.edu.cn

Yubing Chen
Sun Yat-sen University
Guangzhou, P. R. China

Yao Duan
Sun Yat-sen University
Guangzhou, P. R. China

Abstract

Fine-grained image classification, a challenging task due to high class similarity. We explored using various backbones on the CUB200 and Stanford Dogs datasets. Employing transfer learning, optimization techniques (like image size adjustment, learning rate strategies, and regularization), and advanced models (like ViT-L/16) achieved superior results. Despite attempts with GANs, Diffusion Models and image processing for data augmentation, success was limited. Vision-Language Models and ensemble learning improved outcomes, while parameter quantization lightened models. CAM heatmaps explained model results, and FGSM/PGD attacks revealed robustness issues, with hyperparameter sensitivity, particularly learning rate and fold number, impacting performance.

1. Introduction

Fine-grained image classification is a task of great research value in the field of computer vision. Different from ordinary image classification, the difference between different classes of data in fine-grained image classification is smaller and more similar. For example, classifying different classes of birds/flowers/ores. This similarity between classes, brings more challenges to the training of neural networks. This is difficult for simple, traditional CNN networks to solve.

In our article, we summarize and utilize the effective models and training techniques available today to classify two fine-grained image datasets - CUB200 Bird Dataset and Stanford Dogs Dataset. The content will cover different model backbones, diverse training strategies, and comprehensive analysis of model performance.

Residual neural networks (Resnet) and vision transformer (ViT) are the two models we use as the backbone. We use transfer learning to accelerate and improve the training speed during model parameter initialization. In terms of optimization, adjusting the image size, Momentum, weight decay, learning rate adjustment strategy, loss function selection, regularization, and Dropout technique selection have a significant impact on improving model performance and help us better improve the classification accuracy. After comparison, combining the above techniques, ResNeXt50_32X4d and ViT-L/16 achieve the best classification results.

In terms of data augmentation, GAN and Diffusion Models have been tried, but based on the characteristics of fine-grained image classification and the limited training resources, no good results have been achieved. To enable the utilization of additional data, we use Vision-Language Models (VLM) as a supplement to the task and fine-tune it. Faced with a huge model and a long training process, we adopted parameter quantization to lightweight the model. Finally, we use ensemble learning to integrate several models used so far to obtain the final result.

In terms of model evaluation, we used CAM to generate heatmaps for samples that were correctly classified and incorrectly classified, in order to explain the model's results. To evaluate the robustness of the model, we used the FGSM and PGD methods to generate adversarial samples to attack the model, both of which caused the model's classification accuracy to drop to around 0, indicating that the model's robustness is poor. In terms of the sensitivity of hyperparameters, we adjusted a series of hyperparameters, with the learning rate and fold number having a greater impact on the model.

After our comparative experiments on different datasets, the accuracy of the final model reaches 89.6%.

2. Related Work

Fine-grained image classification has been an active area of research due to its inherent challenges related to high class similarity. Numerous approaches have been explored to improve the performance of models on such tasks. This section provides an overview of significant works and methodologies in this field.

Several studies have focused on developing more sophisticated neural network architectures tailored for fine-grained image classification. ResNet [8] and DenseNet [10]

are among the widely used backbones due to their ability to capture detailed features through deep network layers. More recently, Vision Transformers (ViT) [5] have demonstrated strong performance in capturing global dependencies in images, making them suitable for fine-grained tasks.

Transfer learning has been a pivotal technique in leveraging pre-trained models on large datasets to improve performance on fine-grained classification tasks. Works like Huh et al. [11] highlight the benefits of fine-tuning models pre-trained on large datasets like ImageNet.

Data augmentation methods, such as Generative Adversarial Networks (GANs) [6] and Diffusion Models [9], have been explored to increase the diversity of training samples. However, as noted by Zhu et al. [20], these methods have shown mixed results in the context of fine-grained classification due to the subtle differences between classes.

Incorporating Vision-Language Models (VLMs), such as CLIP [15], has shown promise in providing additional context to image classification tasks. These models leverage textual descriptions to enhance the understanding of visual content, as demonstrated by Jia et al. [12].

Ensemble learning techniques, which combine multiple models to improve classification performance, have been effectively utilized in fine-grained image classification. Dietterich [4] provide comprehensive surveys of ensemble methods, highlighting their benefits in reducing overfitting and improving accuracy.

To address model interpretability and robustness, Class Activation Mapping (CAM) [19] and adversarial attacks like FGSM [7] have been employed. These techniques help in understanding model decisions and evaluating their resilience to adversarial perturbations.

Lastly, hyperparameter tuning remains a critical aspect of optimizing model performance. Studies by Bergstra and Bengio [1] and Loshchilov discuss the impact of various hyperparameters, such as learning rate and regularization, on model efficacy.

3. Model Refinements and Improvements

This section elucidates the methodologies employed in fine-grained classification. We delve into a range of methodologies aimed at refining and augmenting our model's efficacy. From Momentum and Weight Decay to K-fold Cross-Validation, and from exploring the nuances between ResNeXt and ResNet architectures to implementing Dropout, each technique plays a pivotal role in optimizing our model's performance. The results are listed in table 1.

3.1. Image Size Optimization

The image size, as a fundamental parameter in model training inputs, can significantly impact the classification performance by influencing the detailed information crucial for fine-grained image classification tasks. In bird image classification tasks, the key features for classification might be extremely subtle, and larger image sizes could potentially provide more detailed information, reducing the loss of such details and prompting the model to focus on finer features.

3.2. Momentum in Optimization

Momentum enhances convergence speed and stability by incorporating a 'momentum' term into the update rule [18]. This term accelerates optimization and reduces parameter space oscillations by considering previous gradient directions when computing the current update direction.

The update equations for momentum can be summarized as follows:

$$v_{t+1} = \beta \cdot v_t + (1 - \beta) \cdot \nabla J(\theta_t)$$

$$\theta_{t+1} = \theta_t - \alpha \cdot v_{t+1}$$

Where v_t is the momentum term at iteration t, incorporating previous gradients' direction; β is the momentum parameter, usually between 0 and 1; $\nabla J(\theta_t)$ is the gradient of the loss function with respect to the parameters; α is the learning rate; θ_t and θ_{t+1} are the parameter values before and after the update.

3.3. Weight Decay in Optimization

Weight decay prevents overfitting by penalizing large parameter values [13]. It adds a term to the loss function proportional to the square of the parameters, discouraging the algorithm from overfitting and enhancing generalization.

The update equation is:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla J(\theta_t) - \lambda \cdot \theta_t$$

Where θ_t and θ_{t+1} are the parameter values before and after the update; α is the learning rate; $\nabla J(\theta_t)$ is the gradient of the loss function; λ is the weight decay coefficient.

3.4. K-fold Cross-Validation in Optimization

K-fold cross-validation entails dividing the dataset into K subsets, known as folds, and iteratively training the model on K-1 folds while using the remaining fold for validation. This process is repeated K times, with each fold being used once for validation. The final performance metric is typically computed as the average across all K folds.The choice of K depends on various factors, including dataset size, computational resources, and the desired trade-off between computational cost and performance estimation accuracy.

3.5. ResNeXt and ResNet in Optimization

Residual Neural Networks (ResNet) address gradient vanishing by introducing Shortcut Connections in blocks. Traditional deep neural networks improve with more layers, but deeper networks may suffer from gradient vanishing, hindering performance. ResNet tackles this by using Shortcut Connections to facilitate easier gradient updates, maintaining a gradient of 1.

In contrast, ResNeXt50_32X4d partitions ResNet into groups [8], allowing parallel convolutional branches to extract different features. More groups enable ResNeXt to better learn complex data features, enhancing performance. While ResNet50 follows a traditional approach, ResNeXt's grouped structure may offer superior performance.

3.6. Dropout in Optimization

To address model overfitting, we experimented with Dropout [17], a technique that randomly deactivates a proportion of neuron outputs during neural network training [17]. This prevents specific neurons from dominating the training process, encouraging the network to learn more robust features.

3.7. Learning Rate Adjustment Strategy

Learning rate is an important hyperparameter in neural network training. When we use a fixed learning rate, it is difficult to fit the current model training situation, and thus may lead to a suboptimal result or slow convergence. Therefore, we adopt the strategy of dynamically adjusting the learning rate, so that the value of the learning rate can be changed with the current model training situation, helping us to obtain faster convergence speed and better results. In this experiment, we take 5 common strategies including different aspects such as decay, adaptive, performance based, one cycle and warming up.

3.8. Vision Transformers (ViT)

The Vision Transformer (ViT) divides the input image into fixed-size patches, flattens these patches, and maps them into a high-dimensional vector space with positional information added. These vectors are then processed by a multi-layered Transformer encoder, which includes self-attention mechanisms and feed-forward neural networks, capturing global information from the image. Finally, a classification head outputs the result. ViT effectively handles image classification tasks by leveraging the Transformer's self-attention mechanism.

Unlike CNNs, which primarily focus on local information, ViT can globally capture long-range dependencies in images through its self-attention mechanism. This allows ViT to integrate global information across the entire image, providing a better understanding of complex images' details and overall structure.

3.9. Vision-Language Models(VLM)

Vision-Language Models(VLM) have learned rich visual-language associations from a large amount of image-text data.

CLIP [15] is a text-image pre-trained model launched by OpenAI. In this model, different feature extraction modules are created to extract information from images and text, mapping them to the same vector space. The similarity between them is calculated through contrastive learning.

In our experiments, the introduction of CLIP as part of ensemble learning enables our task to leverage information from additional datasets and achieve zero-shot transfer.

3.10. Transfer Learning

Transfer learning refers to fine-tuning and retraining the parameters of a pre-trained model published by someone else on new data for this task. This can not only improve the performance of the model on small data sets, but also significantly reduce the training time and resource consumption.

In this experiment, we have three fine-tuning strategies based on the range of freezing parameters/trainable parameter degrees of freedom: 1. Maximum degrees of freedom for unfreezing all layers: end-to-end fine-tuning; 2. Freeze part of the parameters, the degree of freedom is centered: Modular Fine-tuning; 3. Modify only the fully connected layer with the smallest degree of freedom: Extract Feature Vector.

3.11. Loss Function Optimization

The choice of loss function may have some impact on model performance in fine-grained classification tasks. In multi-class tasks, the cross-entropy loss function often performs better than other loss functions. However, we still explore and discuss other loss functions.

3.11.1 Cross-Entropy Loss Function

The cross-entropy loss function is one of the commonly used loss functions in classification tasks. It guides the model optimization by measuring the difference between the predicted class and the true label. For fine-grained multi-class tasks, the sensitivity of the cross-entropy loss function to detailed features can be advantageous. The formula for the cross-entropy loss function is as follows:

$$L(\boldsymbol{w}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log f\left(\boldsymbol{x}_i\right) + \left(1-y_i\right)\log\left(1-f\left(\boldsymbol{x}_i\right)\right)\right] \tag{1}$$

Where $\boldsymbol{w}$ represents the model weights, and $f\left(\boldsymbol{x}_i\right)$ represents the output of the model when the input is x_i.

3.11.2 Elastic Net Regression

We first introduce the Elastic Net [2] regression algorithm, which combines L1 and L2 regularization to address feature selection and model stability issues in high-dimensional data. The Elastic Net loss function is defined as:

$$L(w) = \sum_{i=1}^{N} \left(y_i - w^T x_i\right)^2 + \lambda\rho\|w\|_1 + \frac{\lambda(1-\rho)}{2}\|w\|_2^2 \tag{2}$$

Where ρ controls the proportion of Lasso regression and Ridge regression, and λ controls the strength of the regularization term. It's noteworthy that, similar to Lasso regression, the loss function contains absolute values and is not everywhere differentiable, requiring the use of coordinate descent to find the analytical solution for w.

3.12. Lightweight model

During training, we inevitably run out of memory. Parameter quantization becomes a means for us to improve training speed and reduce memory footprint. In order to ensure the stability and final accuracy of the model values, the FP16 format is used in the model forward propagation and loss calculation, and the FP32 format is used in the back propagation through gradient scaling. This makes the training process of the model significantly faster and the hyperparameter adjustable range becomes larger.

4. Data Augmentation

In the training process of deep learning models, overfitting is a common issue, where the model performs well on the training set but poorly on unseen test data. This often occurs because the model learns specific noise and details in the data without capturing more general patterns. Data augmentation artificially increases the diversity of data by applying various transformations such as rotation, scaling, and color adjustments to force the model to learn more abstract and robust feature representations, thereby reducing its reliance on specific training samples.

4.1. Traditional Image Processing

Traditional digital image processing methods for data augmentation include geometric transformations, color transformations, noise addition, random cropping, and style transfer. These methods can further enhance the important detailed features of bird images while diminishing the model's dependence on unimportant features such as background.

4.2. Generative Adversarial Networks

Generative Adversarial Networks (GANs) [6] are powerful deep learning models composed of a generator and a discriminator. The generator aims to produce realistic data, while the discriminator aims to distinguish between generated and real data. During training, the generator and discriminator compete with each other, with the generator continuously learning to produce more realistic data, and the discriminator continuously learning to accurately identify the authenticity of data. This adversarial training ultimately enables the generator to produce high-quality data that closely matches the statistics of real data.

4.2.1 DCGAN

Deep Convolutional Generative Adversarial Networks (DCGAN) [16] is a variant of GANs that uses deep convolutional neural networks in the generator and discriminator. It pioneered the combination of CNNs with GANs for image processing, laying the groundwork for subsequent GAN models.

4.2.2 ACGAN

Auxiliary Classifier Generative Adversarial Networks (ACGAN) is the main generative model we used because in fine-grained bird image classification tasks, the previous GAN models do not provide the class of the generated images, while ACGAN can generate high-quality images and produce the class of the generated images simultaneously.

4.3. Diffusion Models

Diffusion Models [9], originating from the denoising probabilistic diffusion model, are an emerging class of powerful generative models in deep learning. They can generate high-fidelity images, audio, and other types of data. The core idea of diffusion models is to model the data generation process as a progressive, multi-step process, starting from a simple distribution (such as Gaussian noise), gradually adding details, and eventually generating target data. The foundational work of diffusion models is DDPM.

In our task, we need to use diffusion models to generate augmented images with class information. Therefore, we attempted to use DMBGIS [3].

5. Study on Model Robustness

As deep learning models are widely used in image recognition today, the issues of security and robustness of models are increasingly receiving attention. In particular, the existence of adversarial attacks makes it possible for models to fail when facing carefully crafted input disturbances. This section will briefly introduce two common adversarial attack methods: the Fast Gradient Sign Method (FGSM) and the Projected Gradient Descent Method (PGD), and discuss their impact on model robustness after.

5.1. Principle of FGSM

The Fast Gradient Sign Method (FGSM) is a simple yet effective adversarial attack method first proposed by Goodfellow et al. [7]. The core idea of FGSM is to construct adversarial samples using the gradient information of the model. Specifically, given a classification model $f(\cdot)$ and an input sample x, FGSM calculates the gradient of the loss function L with respect to the input sample, and perturbs the sample in the direction of the gradient to generate an adversarial sample. The formula is as follows:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x L(f(x), y)) \tag{3}$$

where y is the true label of the sample, ϵ is the magnitude of perturbation, and $\nabla_x L(f(x), y)$ is the gradient of the loss function with respect to the input sample.

5.2. Principle of PGD

The Projected Gradient Descent Method (PGD) is an iterative adversarial attack method proposed by Madry et al. [14]. Unlike FGSM, PGD updates the adversarial sample through multiple iterations, with each update in the direction of the gradient of the model's loss function with respect to the input sample. The goal of PGD is to maximize the model's classification error on the adversarial sample while maintaining similarity to the original sample. The iterative update formula of PGD is as follows:

$$x_{t+1} = \text{clip}(x_t + \alpha \cdot \text{sign}(\nabla_x L(f(x_t), y)), x, x_{\text{max}}) \tag{4}$$

where x_t is the adversarial sample at step t, α is the step size, clip$(\cdot)$ function is used to constrain the sample within a legal input range, and x_{max} is the maximum value of the input sample.

6. Experimental Results and Analysis

6.1. Image Size Optimization

In our experiments, we tested image sizes of 224 and 448 under $epoch = 20$ and $epoch = 100$, respectively. The validation set's classification performance peaked at $epoch = 100$ with an image size of 448.

According to the experimental results, training did not fully converge at $epoch = 20$, but at $epoch = 100$, the training results converged. The accuracy achieved with an image size of 448($Acc = 0.293$) surpassed that with an image size of 224($Acc = 0.281$). This could be attributed to the pre-training interpolation, which magnifies the images before training, making it easier for the model to focus on finer details, thereby achieving better fine-grained classification performance.

6.2. Addition of Momentum

The introduction of momentum in our experiments resulted in notable improvements in both convergence rate and accuracy. Momentum considers information from past gradient updates, mitigating the risk of incorrect gradient descent directions induced by adverse data in specific mini-batches. Moreover, the incorporation of global information during gradient updates has accelerated the rate of gradient updates, thereby enhancing both accuracy and convergence rate.

6.3. K-Fold Cross-Validation Results

Our experiments revealed that setting $K = 10$ yielded the most favorable performance. This finding suggests that a larger value of K may be advantageous, particularly for smaller datasets. Maximizing the size of the training set in each fold ensures the model learns from an extensive dataset, thereby facilitating the development of more robust and generalizable models.

6.4. ResNet / ResNeXt Results

We conducted experiments comparing the performance of ResNet50 and ResNeXt50_32X4d models. Our findings indicate that ResNet50 exhibits relatively slow convergence, failing to converge even after 70 epochs, with an accuracy plateauing around 0.28. In contrast, ResNeXt50_32X4d demonstrates significantly faster convergence, typically achieving convergence around 40 epochs, accompanied by a substantial improvement in accuracy. This difference in performance may be attributed to dataset characteristics, particularly the presence of multiple groups in the dataset, which may impact model performance.

6.5. Learning Rate

In the experiments, in terms of accuracy, MultiStepLR is the highest and ReduceLROnPlateau is the lowest. In terms of convergence speed, convergence of MultiStep and ExponentialLR fast, when convergence of two kinds of periodic strategy CosineAnnealingLR and CosineAnnealingWarmRestarts slower. In terms of randomness, two kinds of periodic strategy CosineAnnealingLR and the trend of CosineAnnealingWarmRestarts exists jump out of local optimum. The results are listed in table 3.

6.6. Transfer Learning

In this experiment, the use of transfer learning [11] greatly improved the accuracy and training speed of the model under the same conditions. For different fine-tuning strategies, in terms of convergence speed, it is end-to-end $>$ fine-tune $>$ extract-feature-vector, which is closely related to the degree of freedom of model parameters that

Table 1. Model Accuracy

Model Variation	Accuracy
Resnet18	0
Resnet18 + Shuffle	0.1
Resnet18 + Momentum + Weight Decay	0.23
Resnet18 + Momentum + Weight Decay + 10-fold	0.26
ResNet50 + Momentum + Weight Decay + 10-fold	0.28
ResNeXt50_32X4d + Momentum + Weight Decay + 10-fold	0.33
ResNeXt50_32X4d + Momentum + Weight Decay + 10-fold + MultiStep + Transfer Learning	0.71
ViT-B/32 + Momentum + Weight Decay + 10-fold + MultiStep + Transfer Learning	0.61
ViT-L/16 + Momentum + Weight Decay + 10-fold + MultiStep + Transfer Learning	0.72

Table 2. Experimental Results of Image Size

Epoch	Image Size	Accuracy
20	224	0.25209
20	448	0.24130
100	224	0.26461
100	448	0.29229

Table 3. Experimental Results of Learning Rate Adjustment Strategies

Strategy	Accuracy
MultiStep	0.3489
ExponentialLR	0.33828
CosineAnnealingLR	0.3208
CosineAnnealingWarmRestarts	0.3456
ReduceLROnPlateau	0.31481

can be modified by model training. In terms of accuracy, the accuracy of extract feature vector with low trainable degrees of freedom is much lower than the other two, which also shows that the fitting effect of the model is difficult to achieve the expectation in the case of low degrees of freedom, and the risk of overfitting may also exist in the case of high degrees of freedom. This was not the case in this experiment, though.

6.7. Elastic Net Regression

In practical testing, we attempted to set the parameters to $\lambda = \rho = 0.5$, but the final effect decreased instead. The analysis suggests two possible reasons:

Firstly, we did not explore more parameters, thus restricting the model's normal fitting effect. Elastic Net regression introduces additional regularization terms, which may reduce the model's complexity and mitigate overfitting risks. However, excessive regularization can lead to underfitting or the model's failure to learn data patterns fully, resulting in overly sparse models, loss of critical feature information, and thus affecting model performance.

Secondly, incorporating stronger regularization terms typically requires more training iterations to ensure the model converges to optimal performance. We did not attempt to increase the number of iterations, so the model may not have converged to the best solution.

6.8. Traditional Image Processing for Data Augmentation

In our experiments, based on the custom data augmentation methods in the original project, we added random flipping, image color adjustment, and random rotation operations for image data. Traditional data augmentation methods were found to improve performance to some extent but did not have as significant an effect as anticipated. The main reasons may be twofold:

Firstly, it could be due to computational limitations. We did not conduct experiments with large batch sizes, and under such conditions, data augmentation may disrupt model convergence because each mini-batch is small, and the model may not learn generalizable features. Additionally, excessive data augmentation may disturb the original model's convergence.

Secondly, improper parameter settings and mismatch between data augmentation operations and task types may lead to over-operation, resulting in generated new sample data that differs significantly from the original sample data, even masking the detailed features used for classification.

6.9. GAN for Data Augmentation

In our experiments, we explored the use of DCGAN for generating new augmented data. However, due to computational constraints and the relative simplicity of the model, we faced significant challenges in producing high-quality bird images. It was only when the training epochs approached 1000 that the generated images began to bear any resemblance to real bird images. Even so, the quality of these images was insufficient for them to be effectively used as augmented data for training purposes.

During the training process, we also attempted to use label smoothing techniques to improve the stability and performance of the GAN. Despite these efforts, the generated image quality remained suboptimal.

The primary reasons for the poor generation quality are believed to be the structural limitations of the model and the phenomenon of mode collapse. The DCGAN model used in this project is relatively simple and may not be capable of capturing the complex features and textures inherent in bird images. This inadequacy in feature representation directly impacts the quality of the images produced, resulting in outputs that are not sufficiently realistic or detailed. Additionally, mode collapse, where the generator produces limited varieties of images, further exacerbates the issue, reducing the diversity and overall quality of the generated data.

Moreover, we also attempted to use the Auxiliary Classifier GAN (ACGAN) for image generation. The ACGAN incorporates additional class information into the generation process, potentially allowing for more fine-grained and class-specific image generation. However, due to limitations in computational resources and the need for further optimization of the code, we have not yet been able to obtain effective results using this approach.

While GANs, particularly DCGAN and ACGAN, hold promise for data augmentation, the current results highlight the need for more sophisticated models and greater computational power to achieve the desired quality in generated images. Future work will focus on enhancing the model architecture, introducing more comprehensive data augmentation techniques, and optimizing the training process. These improvements could potentially enable the generation of higher-quality and more realistic bird images, making GANs a more viable option for data augmentation in bird image datasets.

6.10. VIT and Dropout Experiment Results

We conducted experiments using ViT-B/32 and ViT-L/16 [5]. The primary advantage of ViT-B/32 is its fast training speed, with each batch taking significantly less time compared to ViT-L/16. This is due to its simpler architecture and fewer parameters. However, this simplicity results in lower accuracy for ViT-B/32; even after 200 epochs of training, the accuracy was only 61.3%.

In contrast, ViT-L/16 performed significantly better than ViT-B/32. Its final accuracy upon convergence is 71.9%, much higher than that of ViT-B/32. The reasons for this include:

Firstly, the ViT-L model has far more parameters than the ViT-B model, enabling it to learn more complex patterns and features. Secondly, ViT-L/16 uses smaller patches (16x16 pixels), whereas ViT-B/32 uses larger patches (32x32 pixels). Smaller patches can capture more detailed local information and finer features. More patches also mean that, for the same input image, ViT-L/16 has a higher resolution input representation, which helps the model capture more detailed information and subtle patterns. The self-attention mechanism in ViT-L/16 can focus on the image's details, which is particularly important for tasks involving recognition and classification where details are crucial. However, the aforementioned reasons also lead to the training speed of ViT-L/16 being significantly slower than that of ViT-B/32.

Additionally, we experimented with dropout methods. However, after multiple trials, we found that the smaller the dropout parameter, the better the model performed. In other words, the model performed best without using dropout. This is likely because the current model does not suffer from overfitting issues.

6.11. VLM Experiment Result

In our experiment, we utilized clip-vit-base-patch16 and employed the category names as textual information. The specific training results are shown in the table 4.

During the experiment, it was found that increasing the number of fine-tuning layers and adjusting the learning rate, etc., could not provide significant help for the model performance. After multiple attempts, the accuracy rate was basically stable between 70% and 77%. It is suspected that due to the limited training samples and resources, the model has entered a local optimum, but it is sufficient to supplement our original model.

Table 4. Performance Metrics for CUB_200_2011 and Stanford_Dog Datasets

State	CUB_200_2011	Stanford_Dog
Accuracy without fine-tuning	0.4940	0.4877
Validation set accuracy after fine-tuning	0.7763	0.7700
Test set accuracy after fine-tuning	0.7523	0.7784

6.12. Lightweight Model Experimental Results and Analysis

In our experiments, the training time and accuracy with or without parameter quantization are compared, and the results are recorded in table 5.

In terms of time, after only 10 epochs, using parameter quantization is almost 100s faster than without it. In terms of accuracy, there is no significant difference in the improvement effect of the two methods on the model performance.Further proof that parameter quantization helps us train models faster and better.

Table 5. Comparison of Quantized and Non-Quantized Parameter Training

Whether to Quantize Parameters	Times	Accuracy (Training set /Validation set)
Yes	619.2071s	0.76870 / 0.68167
No	713.5996s	0.76398 / 0.67083

6.13. CAM Experimental Results and Analysis

In our experiments on fine-grained bird species recognition, we utilized CAM to generate heatmaps for both correctly and incorrectly classified images. These heatmaps not only help us understand the model's decision-making process but also reveal the model's focus areas when processing different images.

Figure 1. CAM Heatmap for Correct Classification. The figure shows correctly predicted bird images, with nine images randomly selected.

CAM Heatmap for Correct Classification: The figure 1 shows an image that was correctly classified. For correctly classified images, the CAM heatmap indicates that the model accurately focuses on key feature regions of the bird. For instance, during the recognition of various bird species, the model focuses on prominent features such as the bird's head, wings, and feathers. These regions are often critical features that human observers also consider important. Therefore, these heatmaps for correctly classified images validate the model's effectiveness in capturing the target object's features in fine-grained classification tasks.

CAM Heatmap for Incorrect Classification: For incorrectly classified images, we identified two main scenarios:

1. Highly Similar Bird Species: As shown in the first row of figure 2, the model incorrectly identifies one bird species as another very similar species. The CAM heatmap indicates that the model does focus on the bird's key feature regions, such as the head and wings. However, due to the high visual similarity between the two species, the model fails to accurately distinguish them. For example, birds with very close color and shape are easily confused, highlighting the model's limitations in handling highly similar species.

Figure 2. CAM Heatmap for Incorrect Classification. The figure shows incorrectly predicted bird images, with three representative images selected. Each row represents one prediction, with the first column showing the predicted result, the second column showing the ground truth, and the third column showing the CAM heatmap.

2. Model Focuses on Non-Bird Regions: As shown in the second and third rows of figure 2, in these cases, the model's attention is drawn to the background or other non-bird regions, leading to incorrect classification. The CAM heatmap shows that the model might focus on elements like grass, water ripples, or other environmental features, while neglecting important features of the bird itself. This indicates that when dealing with complex backgrounds or distracting information, the model may fail to effectively focus on the target object. Compared to correct classifications, it is likely that our model's criteria for identifying birds may be flawed, potentially relying on the complexity of the texture in the region, which is evidently an incorrect criterion.

6.14. Robustness Experiment and Analysis

In the experiment, we applied both FGSM (figure 3) and PGD (figure 4) methods to generate adversarial samples for attacking the model.

Visualization Analysis: Visually, the adversarial samples generated by the FGSM method exhibit significant noise, particularly in the red and blue channels. On the other hand, the adversarial samples generated by the PGD method have relatively less noise but still show noticeable changes in the color channels. This indicates that although the two methods differ in the degree and form of perturbations, both significantly affect the color distribution of the original images.

Prediction Probability Analysis: After generating the adversarial samples, we performed a statistical analysis of the model's prediction probabilities. For the FGSM method, the prediction probability for the originally correct categories dropped to less than 0.01%, while the prediction probability

Figure 3. Adversarial attack results using the FGSM method. The first column shows the original images, the second column shows the generated adversarial samples, and the third column shows the images corresponding to the incorrect categories predicted by the model.

Figure 4. Adversarial attack results using the PGD method. Each set of three columns shows examples: the first column is the original image, the second column is the generated adversarial sample, and the third column is the image corresponding to the incorrect category predicted by the model.

for the incorrect categories rose to 20% to 40%. In contrast, the PGD method had a more thorough impact on the model, reducing the prediction probability for the correct categories to nearly 0, and increasing the prediction probability for the incorrect categories to 100%. These results indicate that the PGD method is more effective in generating adversarial samples, significantly decreasing the model's confidence in the correct categories and increasing its misclassification rate. This demonstrates the advantage of the PGD method in adversarial attacks, as it can produce more deceptive adversarial samples, making it difficult for the model to classify them correctly.

6.15. Experimental Evaluation on Multiple Datasets

To evaluate the performance of our model, we conducted comparative experiments across different datasets, specifically the bird classification dataset CUB-200 and the dog breed classification dataset Stanford Dogs. On Stanford Dog dataset, the accuracy can reach up to 89.6%, while on CUB dataset, the accuracy is around 77.6%.

The experimental results indicate a significantly lower accuracy on the CUB-200 dataset compared to the Stanford Dogs dataset. Upon analysis, we attribute this discrepancy to the more pronounced differences between dog breeds. These distinct differences allow the model to more easily extract features that distinguish between various dog breeds.

In contrast, bird classification presents a greater challenge due to the generally smaller size of birds and the less pronounced differences between bird species compared to dog breeds. Consequently, the fine-grained classification task for birds is inherently more difficult than for dogs. Therefore, the lower performance of the model on the CUB-200 dataset is understandable.

Overall, the model demonstrates strong performance in fine-grained classification tasks, excelling particularly in scenarios where the inter-class variability is more pronounced.

6.16. Hyperparameter Sensitivity Experiments

We also conducted hyperparameter sensitivity experiments, adjusting a range of parameters including the learning rate and the number of folds. Here, we highlight the hyperparameters that had a significant impact on the model's performance:

The first key hyperparameter is the learning rate. Despite using a dynamic learning rate adjustment strategy, the initial learning rate setting is crucial. We experimented with various learning rates and found that when the learning rate was set relatively high (on the order of 10^{-3}), the final accuracy was not only low, but the convergence speed was also slow, and the training process was unstable. Conversely, when the learning rate was set very low (on the order of 10^{-6}), the convergence speed was extremely slow, the model was prone to getting stuck in local minima, and the generalization performance was poor, resulting in low final accuracy. After extensive testing, we determined that a learning rate of 2×10^{-5} achieved the best results, offering a fast convergence speed, stable training process, high accuracy, and good generalization performance.

The second key hyperparameter is the number of folds (fold num). We experimented with fold numbers of 5 and 10 and found that the accuracy was significantly higher with 10 folds. This is because with 10 folds, each training iteration uses more data, which is particularly important given our small dataset. Additionally, more folds mean more training and validation cycles, which allows for a more detailed evaluation of the model's performance, reducing random errors due to data partitioning. Multiple validations provide a more comprehensive assessment of the model's stability and generalization ability, leading to more accurate performance evaluations. A smaller dataset can also lead to overfitting; fewer folds mean less data per training cy-

cle, increasing the risk of overfitting to noise in the training data. Increasing the fold number means more data is used in each training cycle, reducing the risk of overfitting and improving the model's generalization ability. Although using 5 folds results in faster computation and lower overhead, the small size of our dataset and the detailed and complex nature of our fine-grained classification tasks make 10 folds more suitable. Therefore, we selected a fold number of 10 for our final experiments.

7. Conclusion

In this paper, we explored various model architectures including ResNet, ViT, and VLM for fine-grained classification tasks. We employed a range of techniques such as transfer learning, dropout, k-fold cross-validation, dynamic learning rate adjustment strategies, and data augmentation including GANs. Our evaluation spanned datasets like Stanford Dogs and CUB, where we rigorously tested hyperparameters, employed CAM for interpretability analysis, assessed model robustness through adversarial examples, and pursued lightweight model designs. Our best-performing model achieved up to 89.6% accuracy on the Stanford Dogs test set, demonstrating robust classification capabilities across different datasets.

8. Contribution

Yuhao Fang: 37%, Yubing Chen: 33%, Yao Duan: 30%.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012. 2

[2] Christine De Mol, Ernesto De Vito, and Lorenzo Rosasco. Elastic-net regularization in learning theory. *Journal of Complexity*, 25(2):201–230, 2009. 4

[3] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 4

[4] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, Berlin, Heidelberg, 2000. 2

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 7

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020. 2, 4

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 2, 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3

[9] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4

[10] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[11] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 2, 5

[12] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, et al. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916, 2021. 2

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[14] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 5

[15] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763, 2021. 2, 3

[16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 4

[17] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3

[18] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013. 2

[19] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 2

[20] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 2