An Empirical Study of Vision Transformers for Fine-Grained Visual Classification

Shiyu Chen Jingyun Huang Zijian Lin

School of Computer Science and Engineering, Sun Yat-sen University

{chenshy239, huangjy, linzj}@mail2.sysu.edu.cn

A. Abstract

Fine-grained vision classification (FGVC) is a challenging task due to small inter-class variance, large intra-class variance, multiple object scales, and complex backgrounds. Current methods in FGVC can be categorized into localization-based and attention-based methods. Localization-based methods require manual annotation of discriminative parts, which is labor-intensive. Attention-based methods automatically detect key regions via self-attention mechanism but suffer from computational cost. In this paper, we propose to use the vision transformer as the backbone for FGVC. We conduct comprehensive experiments to evaluate the performance of the vision transformer in fine-grained classification tasks. Our results show that the vision transformer achieves competitive performance and outperforms the baseline models. Furthermore, we investigate the effects of training tricks, such as layer normalization, data augmentation, and learning rate scheduling, and demonstrate their impact on the model's performance. Additionally, we explore the benefits of transfer learning by using pretrained weights, which significantly improves the model's performance compared to training from scratch. Our findings suggest that the vision transformer is a promising approach for fine-grained vision classification tasks.

B. Introduction

Fine-grained vision classification (FGVC) is a challenging classification task whose purpose is to classify some subclasses of a given object category, for example, subcategories of birds, dogs or car. What make this task difficult is that Objects in fine-grained tasks usually share small inter-class variance and large intra-class variance along with multiple object scale and complex background, which leads to complex problem space.

Benefiting from the progress of deep neural networks and large-scale annotated datasets [8, 14], FGVC has gained steady improvements in recent years. Current methods on FGVC can be roughly divided into localization-based methods and attention-based methods. The core for solving FGVC is to learn the discriminative features in images. Compared to attention-based methods, the localization-based methods have the advantage that they explicitly capture the subtle differences among subclasses which is more interpretable and yields better results.

To explicitly utilize this key feature, early localization-based methods directly annotate the discriminative parts in images [5, 11]. However, it needs labor-intensive part annotation to build bounding box annotations, hindering the applicability of these methods on real-world applications. To alleviate this problem, recent localization-based methods normally integrate the region proposal network (RPN) to obtain the potential discriminative bounding boxes. These selected proposals are then fed into the backbone network to gain the local features. However, such RPN-base methods ignore the relationships among selected regions and this mechanism tends to propose large bounding boxes to contain the foreground objects as much as possible [7]. These disadvantages would probably hinder the performance of the trained model, because of the inaccurate bounding boxes which may cover many background rather than objects.

Attention-based methods [21, 22] automatically detect the key regions in images via self-attention mechanism. These methods release the reliance on manually annotation for key regions and have gained encouraging results. Recently, the vision transformer [3] achieved huge success in the classification [3], image retrieval [4] and semantic segmentation task [23] which indicates the fact that transformer architecture can capture the important regions with its innate attention mechanism in images by directly segment an image to a sequence of image patches. In contrast, CNNs mainly exploit the locality property of image and only capture weak long-range relation in very high layers. Besides, the subtle differences between fine-grained classes only exist in certain places thus it is unreasonable to convolve a filter which captures the subtle differences to all places of the image.

In our work, we use vision transformer as our backbone

and conduct comprehensive experiment to research the ability of vision transformer in the task of Fine-grained vision classification.

C. Related Work

C.1. Fine-Grained Vision Classification

Methods on FGVC can be coarsely divided into two groups: localization-based methods and attention-based methods.

localization-based methods focus on detecting the foreground objects and performing classification based on them, which is similar to object detection task. Some works in the early stage take advantage of part annotation to supervise the learning of the detection branch [5, 11]. However, as discussed above, it needs labor-intensive part annotation to build bounding box annotations, hindering the applicability of these methods on real-world applications.

Therefore, recent localization-based methods introduce the weakly supervised object detection (WSOD) technique to predict the potential discriminative regions with only image-level label to alleviate above problem. WSOD and instance segmentation techniques are introduced to obtain the rough object instances, and then the model can select the important instances to perform classification [5]. Moreover, correlations between regions to select distinguished parts is found to be helpful [20]. However, these methods require a well designed WSOD branch to propose potential discriminative regions. Moreover, the selected parts sent to the classification head often cover the whole object instead of the truly discriminative parts.

Recently, attention-based methods automatically localize the discriminative regions via self-attention mechanism without extra annotations. Many diversified visual attention networks are applied to FGCV, and they are demonstrated to be efficient to collect dicriminative information [22]. Zheng et al. [23] proposed a progressive-attention to progressively detect discriminative parts at multiple scales. However, these methods often suffer from huge computational cost.

C.2. Transformer

Transformer has achieved incredible performance in natural language processing [2, 17]. Motivated by this success, many researchers try to exploit the transformers in computer vision. Recent work ViT [3] achieves the state-of-the-art performance on image classification by employing a pure transformer architecture on a number of fix-sized image patches. And later, many usages of the pure transformer in other computer vision tasks were explored [23], and also the study in FGVC [7].

TransFG [7] is the first study to extend the ViT into FGVC. TransFG selects the discriminative tokens and directly send them to the last transformer layer. And FFVT [19] was proposed to aggregate the local and different level information from each layer to enrich the feature representation capability via feature fusion.

D. Method

We will describe our model and some notices in this section.

D.1. Baseline

We choose vision transformer(ViT) as our baseline [3]. As shown in figure 1 in a ViT, the input image is divided into fixed-size patches, which are then flattened and linearly projected to a sequence of tokens, similar to how words are represented in natural language processing. The Transformer architecture is then applied to this sequence of tokens to extract features and make predictions.

To be specific, we first preprocess the input image $x \in R^{H*W*C}$ into a sequence of flattened patches $x_p \in R^{N*(P^2*C)}$, where (H, W) is the resolution of the image, C is the number of channels, N is the number of patches and P is the size of each patch. Then, we map the vectorized patches x_p into a D-dimensional patch embedding through a trainable linear projection and add a learnable position embedding to it:

$$z_0 = [x_p^1 E; x_p^2 E; ...; x_p^N E] + E_{pos}, \quad (1)$$

where N is the number of image patches, $E \in R^{(P^2*C)*D}$ is the patch embedding projection, and $E_{pos} \in R^{N*D}$ denotes the position embedding.

According to the structure of Transformer encoder [18], we input the patch embedding to L layers of multi-head self-attention (MSA) and multi-layer perceptron (MLP) blocks, so the output of the l-th layer can be written as follows:

$$z'_l = MSA(LN(z_{l-1})) + z_{l-1}, l = 1, 2, 3, ..., L \quad (2)$$

$$z_l = MLP(LN(z'_{l-1})) + z'_{l-1}, l = 1, 2, 3, ..., L \quad (3)$$

where $LN(\cdot)$ denotes the layer normalization operation and z_l is the encoded image representation. ViT exploits the first token of the last encoder layer z_L^0 as the representation of the global feature and forward it to a classifier head to obtain the final classification results without considering the potential information stored in the rest of the tokens.

D.2. Our Own Idea

E. Experimental Result

We will describe the detailed implementation of our experiments step by step in the following section. And we will show the results in the corresponding section.

First, we report the overall training loss and the accuracy of our model in Figure 3 and 4 respectively, indicating the stable training process and achieving good performance.

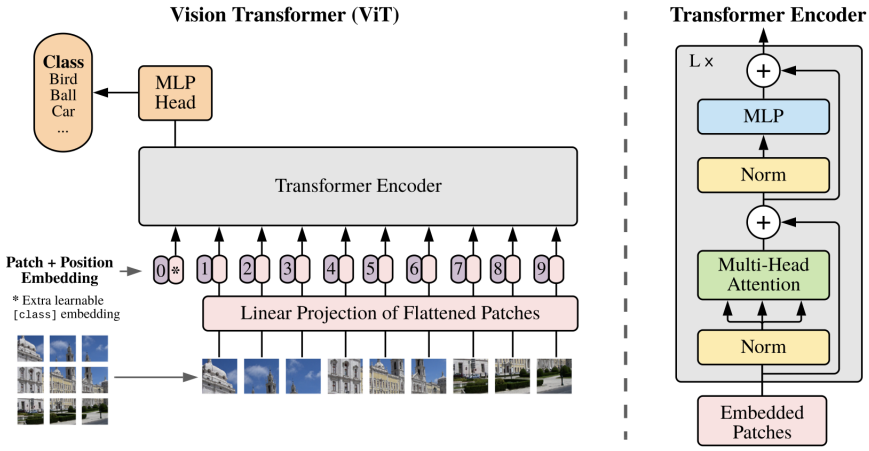

Figure 1. ViT overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence.

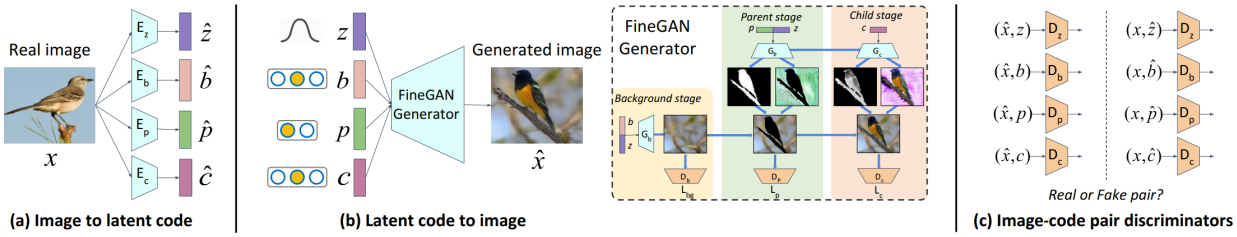

Figure 2. MixNMatch architecture. (a) Four different encoders, one for each factor, take a real image as input to predict the codes. (b) Four different latent codes are sampled and fed into the FineGAN generator to hierarchically generate images. (c) Four image-code pair discriminators optimize the encoders and generator, to match their joint image-code distributions.

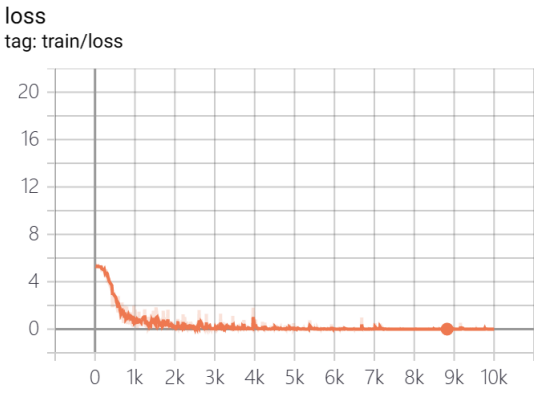

Figure 3. Training loss variation curve in our experiment.

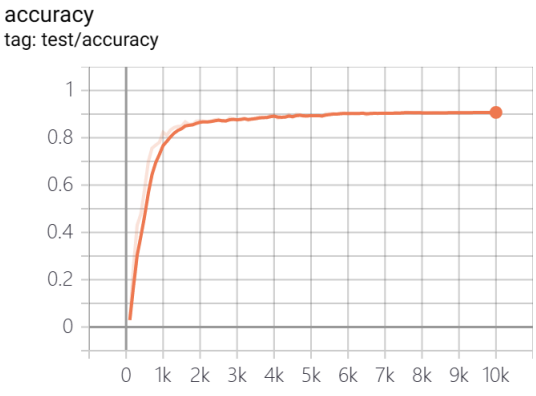

Figure 4. Accuracy variation curve in our experiment.

E.1. Training Tricks

In our experiment, we use multiple training tricks to improve the training stability and performance of the model, including layer normalization, learning rate scheduler, and simple data augmentation.

- **Layer normalization** [1] computes the mean and vari-

ance used for normalization from all of the summed inputs to the neurons in a layer on a single training case. it also give each neuron its own adaptive bias and gain which are applied after the normalization but before the non-linearity. And unlike batch normalization, layer normalization performs exactly the same computation at training and test times.

- **Data augmentation** in computer vision refers to the process of generating new training data by applying various transformations or modifications to existing images. It is a common technique used to expand the size and diversity of the training dataset, which helps improve the performance and robustness of computer vision models. Data augmentation techniques can be broadly categorized into geometric transformations, color transformations, and other specialized techniques. For geometric transformations, we can rotate the image by a certain angle, shift or mirror the image horizontally or vertically and resize the image to a larger or smaller size.

- **Learning rate scheduler** is a strategy which helps us train the model more stable and better. In our experiment, We adopt cosine annealing as the scheduler of our optimizer. Specifically, linearly increases learning rate from 0 to 1 over s_1 training steps. Decreases learning rate from 1. to 0. over remaining s_2 steps following a cosine curve. s_1 and s_2 are hyperparameter in our experiment. We plot the learning rate of our experiment in Figure 7.

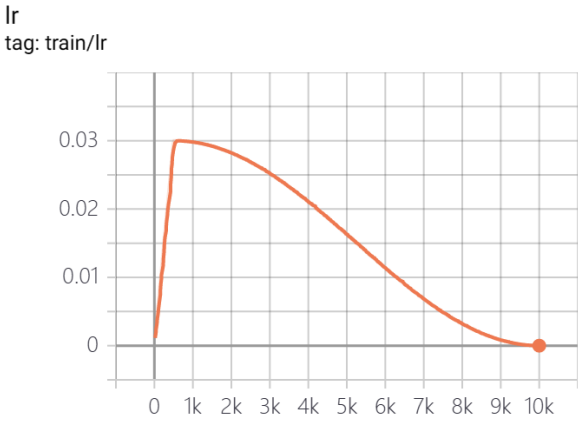

Figure 5. Learning rate variation curve in our experiment.

E.1.1 Experimental Result

In our experiment, we use ViT as our backbone. And layer normalization has been implemented into ViT, so we do not empirically show the effect of layer normalization. As shown in Table 1, we report the result of some ablation study. Data augmentation is a useful and powerful trick which can improve the performance by 1.6%, because it can make the model robust to more different data. For Learning rate scheduler, it seems make little effect on our model. One potential reason is that the use of pretrained model makes the start point train process better due to the large amount of knowledge stored in the model parameter.

model	CUB200 Bird	Stanford Dogs
Ours (ViT)	90.7%	87.2%
w/o DA	89.1%	85.1%
w/o LRS	90.5%	86.8%

Table 1. The performance of some tricks. *w/o DA* means our model trains without data augmentation and *w/o LRS* means our model trains without learning rate scheduler.

E.2. Transfer Learning

Transfer learning, used in machine learning, is the reuse of a pretrained model on a new problem. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another. For example, in training a classifier to predict whether an image contains food, you could use the knowledge it gained during training to recognize drinks.

For pretrained model, we want to utilize the general knowlegde in the model to help finish the downstream task. We can use a finetune-base method to improve the performance of the model, which has been shown to be a powerful approch.

For this part, we investigate the impact of using the pretrained weight to initialize the model. As shown in Table 2, when using the pretrained weight, the model have a superior performance compared to train the model from scratch.

model	CUB200 Bird	Stanford Dogs
w/o pretrain	75.3%	71.1%
w/ pretrain	90.7%	87.2%

Table 2. The performance of different initialization. *w/o pretrain* means training the model from scratch, and *w/ pretrain* means using the pretrained weight to initialize the model.

E.3. Attend to local regions: object localization or segmentation

E.3.1 Object localization

Object localization is a task that involves detecting the location of one or more objects within an image or video frame. The goal is to identify where the object is located in the

image, typically by drawing a bounding box around it, and possibly assigning a label to the object.

In our work, the input image is divided into a grid of fixed-size patches, which are fed into the ViT encoder to extract features. Then a region proposal network (RPN) is applied to the ViT features to generate a set of candidate object regions which are pooled into fixed-size feature vectors and fed into a bounding box regression network, which predicts the object's bounding box coordinates relative to the region's center. The candidate object regions are also classified using a linear classifier, which predicts the probability of each region containing an object of a particular class. Each selected region is cropped from the input image and resized to a fixed size, and then fed back to the extracted feature as output to make predictions. The above steps are shown clearly as follows:

$$z'' = RPN(z_L^0) \tag{4}$$

where z_L^0 is the feature extracted from ViT and z'' is the output of RPN network. Then we attain ROIs r from z'' and put it through a linear layer.

$$z''' = FC(r) \tag{5}$$

Finally, we simply add the output of object localization module to the ViT feature as final output.

$$z = z''' + z_L^0 \tag{6}$$

E.3.2 Experimental Result

model	CUB200 Bird	Stanford Dogs
Baseline (Ours)	90.7%	87.2%
w/ ObLocal	90.8%	87.5%

Table 3. The performance of model with object localization. *w/ ObLocal* means that we attend object localization to our network

The above results show that object localization improves the network mildly mainly for two reasons: (1) Object localization extract finer local features than simply using ViT; (2) Object localization fails to utilize those finer local features well in prediction.

E.4. Synthetic image generation as part of data augmentation

E.4.1 MixNMatch

MixNMatch [10], a conditional generative model encoding and disentangling background, object pose, shape and texture from real images, is utilized to generate synthesized images based on CUB200 Bird dataset: 11,788 bird images from 200 classes.

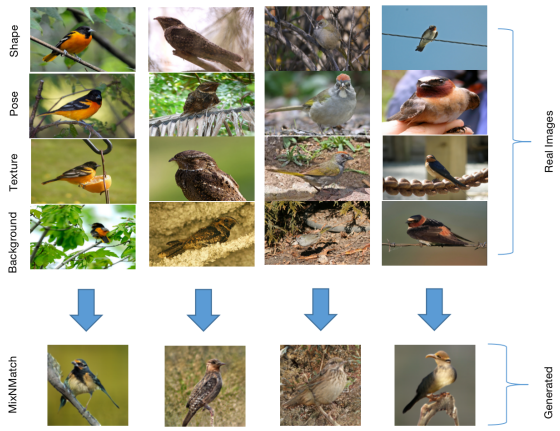

Figure 6. Conditional mix-and-match image generation sample. Real images from CUB200 Bird dataset formulating four factors——shape, object pose, texture, and background are sampled arbitrarily in one class to synthesize images belonging to the same class.

Figure 2 (a-c) shows the encoders, generator, and discriminators of MixNMatch. Figure 2 (a) shows four encoders simultaneously learn to encode background, object pose, shape, and texture factors associated with images in $\mathcal{I} = \{x_1, ..., x_N\}$ which includes an unlabeled image collection of a single object category (e.g., birds). Figure 2 (b) indicates FineGAN [15] takes as input four randomly sampled latent codes (z, b, c, p) to hierarchically generate an image in three stages : (1) a background stage where the model only generates the background, conditioned on latent one-hot background code b; (2) a parent stage where the model generates the object's shape and pose, conditioned on latent one hot parent code p as well as continuous code z, and stitches it to the existing background image; and (3) a child stage where the model fills in the object's texture, conditioned on latent one-hot child code c. In both the parent and child stages, FineGAN automatically generates masks (without any mask supervision) to capture the appropriate shape and texture details. The goal is to generates high quality images matching the true data distribution $P_{data}(x)$ by combining latent factors from the disentangled code space.

FineGAN is trained with three losses, one for each stage, the full loss can be simply denoted as:

$$\mathcal{L}_{finegan} = \mathcal{L}_b + \mathcal{L}_p + \mathcal{L}_c \tag{7}$$

where $\mathcal{L}_b$, $\mathcal{L}_p$, and $\mathcal{L}_c$ denote the losses in the background,parent, and child stages.

Figure 2 (c) shows the discriminator D takes image-code pair as input. When training D, which is set the paired real image x and code $\hat{y}$ extracted from the encoder E to be real, and the paired sampled input code y and generated image $\hat{x}$ from the generator G to be fake. Conversely, when train-

ing G and E, we try to fool D so that the paired distributions $P(data; E(x))$ and $P(G(y); code)$ are indistinguishable, via a paired adversarial loss:

$$\mathcal{L}_{bi\_adv} = \min_{G,\mathbb{E}} \max_{D} \mathbb{E}_x \sim P_{data} \mathbb{E}_{\hat{y} \sim E(x)}[logD(x, \hat{y})] + \mathbb{E}_{y \sim P_{code}} \mathbb{E}_{\hat{x} \sim G(y)}[log(1 - D(\hat{x}, y))]. \quad (8)$$

This loss will simultaneously enforce the (1) generated images $\hat{x} \sim G(y)$ to look real, and (2) extracted real image codes $\hat{y} \sim E(x)$ to capture the desired factors (i.e., pose, background, shape, appearance).

E.4.2 Experimental Result

We generated 50 images for each class in CUB200 Bird dataset obtaining 10000 synthesized images totally, each of which is composed of aforementioned arbitrarily sampled four factor images from the same class. The generated image examples are displayed in Figure 6. We use the generated images by MixNMatch as part of data augmentation. Our result (Table 5) indicates that although MixNMatch can generate visually impressive images, they may not always capture the semantic meaning or high-level concepts that a classification model relies on. MixNMatch focus on encoding and disentangling background, object pose, shape and texture from real images based on the low-level pixel patterns learned from the training data, without explicitly modeling the semantic relationships between different objects or classes. Consequently, the generated images may lack meaningful features or contextual information required for accurate classification.

model	CUB200 Bird	Stanford Dogs
Baseline (Our model)	90.7%	87.2%
w/ MixNMatch	86.8%	84.6%

Table 4. The performance of synthetic image generation as part of data augmentation. *w/ MixNMatch* means that we use the generated images by MixNMatch as part of data augmentation.

E.5. ViT model backbone vs. CNN backbone

E.5.1 Introduction of CNN backbone

CNN (Convolutional Neural Network) has been widely used as a backbone for fine-grained visual classification (FGVC) tasks, which are designed to exploit the spatial structure of images through the use of convolutional layers, which extract local features from different regions of an image.

One of the most well-known CNN architectures for FGVC is the Inception architecture [16], which was introduced by Google in 2014. The Inception architecture is designed to balance the trade-off between the computational cost and the accuracy of the model. It achieves this by using a combination of convolutional layers with different kernel sizes and pooling operations.

Another popular CNN architecture for FGVC is the VGG(Visual Geometry Group) architecture [13], which is known for its simplicity and effectiveness. The VGG architecture consists of a series of convolutional layers followed by fully connected layers. The convolutional layers are designed to extract local features from different regions of an image, while the fully connected layers are responsible for the final classification.

In recent years, there has been a growing trend towards using deeper and more complex CNN architectures for FGVC tasks. For example, the ResNet(Residual Network) architecture [9] has been shown to achieve state-of-the-art performance on several FGVC benchmark datasets. ResNet is designed to address the problem of vanishing gradients in deep CNNs by using residual connections between layers. Here, we take ResNet as our backbone and also our structure on FGVC tasks.

E.5.2 Methodology

The basic building block of ResNet is the residual block, which allows the network to learn residual mappings. A residual block consists of two convolutional layers with a shortcut connection that bypasses the convolutional layers. The shortcut connection allows the network to learn the residual mapping between the input and the output of the block, which makes it easier for the network to learn deeper representations.

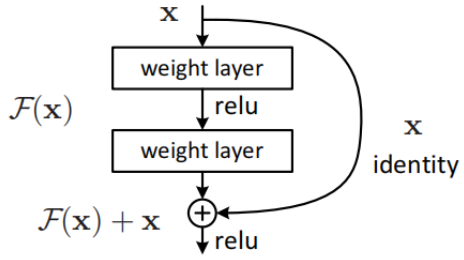

Figure 7. Structure of a residual block

In FGVC, the pre-trained ResNet model(ResNet50) is used as a feature extractor, where the output of the last convolutional layer is used as input to a classifier. This approach allows the network to leverage the pre-trained features to learn the specific features that are relevant for the FGVC task.

E.5.3 Experimental Result

As the result is shown, ViT backbone has a better performance than ResNet and the reason is revealed in the next

Backbone	CUB200 Bird	Stanford Dogs
ViT	90.7%	87.2%
ResNet	86.7%	85.8%

Table 5. The performance of model with *ResNet* backbone.

part.

E.5.4 Comparison

The main difference between ViTs and CNNs is that ViT use self-attention mechanisms to capture global context information, while CNN relies on local feature maps to capture spatial information. In FGVC tasks, where subtle differences between categories are important, global context information can be helpful in capturing these differences. This might be the main reason why ViTs performs better than CNNs.

Also, ViTs do not inherently have this property and need positional embeddings to maintain location information, while CNNs are translation invariant by design due to the convolutional filters. And ViTs generally require input images of a fixed size, while CNNs can handle a range of input sizes. This natural property makes CNNs more flexible.

Last but not least, ViTs tend to have much more parameters than comparable CNNs since they use fully connected self-attention layers. This can make ViTs require more computational resources. And CNNs tend to be more data efficient and require less training data to achieve good performance compared to ViTs.

In summary, ViTs perform better due to their self-attention mechanism, but CNNs still work well in practice due to their strengths in properties like parameter efficiency and data efficiency. It depends on the specific task and dataset, as well as available computational resources, whether a ViT or CNN backbone is preferable.

E.6. Interpretation of the model

E.6.1 Method

Gradient-weighted Class Activation Mapping (GradCAM) [12], a class-discriminative localization technique that generate visual explanations for CNNs, Vision Transformers and more has the advantage of not requiring architectural changes or re-training. We leverage GradCAM to uncover the veil of our model by providing the activation map of sample images that correctly classified or misclassified by our model.

E.6.2 Experimental Result

The activation map is utilized to visualize and highlight the regions where the model focuses, the corresponding results on CUB200 bird dataset are displayed in Figure 8 and Figure 9 demonstrating correct and mistaken classification results respectively.

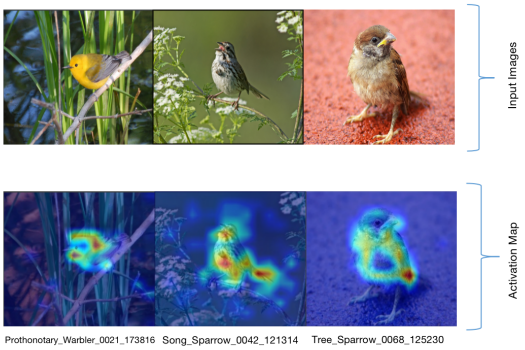

Figure 8. Activation map of correctly classified images

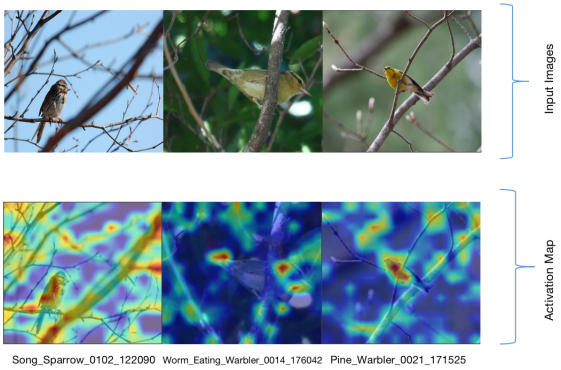

Figure 9. Activation map of misclassified images

According to the activation map of misclassified samples shown in Figure 9, bird in these images are unable to be recognized and the model tends to focus on the background, the commonality of these images is that the bird is covered by a branch partly or the main features of bird are blurred due to the posture of the bird or its background. To verify this point of views, we zoom in the image by cropping the main region of these images to capture the bird and keep it in the middle of the image.

After centring the bird of misclassified samples, the result in Figure 10 indicates that the model is able to capture the body of the bird, this verifies our conjecture that objects covering the bird and the posture of the bird might affect the model to determine the right decision.

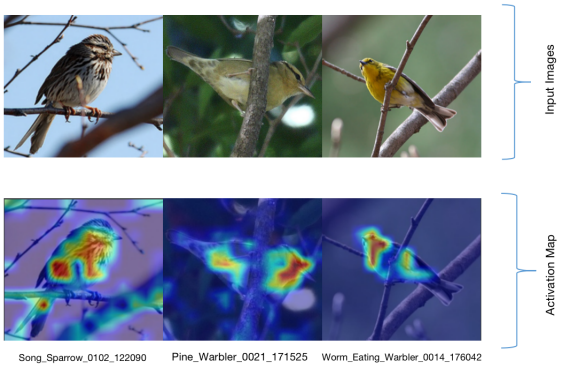

Figure 10. Activation map of centralized images

E.7. Robustness of the model

E.7.1 Method

Many machine learning models, including neural networks, consistently mis-classify adversarial examples which are formed by applying small but intentionally worst-case perturbations to examples from the dataset, such that the perturbed input results in the model outputting an incorrect answer with high confidence. This suggests that adversarial examples expose fundamental blind spots in our training algorithms. Adversarial example-based sensitivity analysis methods are methods that create adversarial examples for different kinds of data such as images or test.

FGSM algorithm [6] indicated that high-confidence neural network misclassifications that are caused by small, yet intentionally, worst-case datapoint perturbations, were not due to nonlinearity or overfitting, but instead due to neural networks' linear nature. FGSM is a fast and simple yet powerful gradient-based method of generating adversarial examples while using the max norm.

In this section, we want to use FGSM to attack our model, testing the robustness of our model. Let θ be the parameters of a model, x the input to the model, y the targets associated with x (for machine learning tasks that have targets) and $J(\theta, x, y)$ be the cost used to train the neural network. We can linearize the cost function around the current value of θ, obtaining an optimal max-norm constrained pertubation of

$$\delta = \epsilon sign(J \bigtriangledown_x (\theta, x, y)). \tag{9}$$

This is the "fast gradient sign method" of generating adversarial examples. Note that the required gradient can be computed efficiently using backpropagation. Adding this pertubatioin to the original image generate the adversarial examples, which is similar to the original image.

E.7.2 Experimental Result

We report the performance of our model testing by adversarial examples generated by FGSM algorithm as input in Figure 6. The adversarial examples hurt the performance hugely. The performance of our model decreases sharply to around 20%, indicating the potential risk of the model.

model	CUB200 Bird	Stanford Dogs
Baseline (Ours)	90.7%	87.20%
w/ FGSM Attack	22.16%	11.62%

Table 6. Explore the robustness of the model. *FGSM Attack* means adversarial examples generated by FGSM algorithm as input. We set ϵ to 0.1.

Furthermore, as shown in Figure 11, we conduct the experiments on impact of the choice of ϵ. When ϵ become larger, the generated images contain more noise. Therefore, the performance of our model become worse when ϵ increases, showing the risk of the existing training methods.

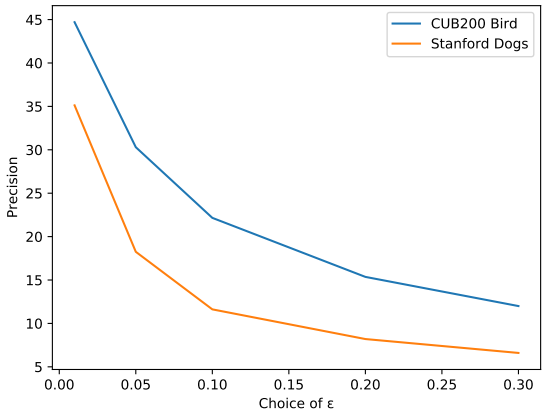

Figure 11. The impact of the choice of ϵ. We report the result of different ϵ from 0.01 to 0.3.

F. Conclusion

In this project, we finish the fine-grained visual classification task by a transformer based model. We apply multiple techniques required by this project, achieving the performance with accuracy 90.7% on CUB200 Bird dataset and 87.20 % on Stanford Dogs dataset. And we conduct comprehensive experiment to research the ability of vision transformer in the task of Fine-grained vision classification.

G. Acknowledgements

This work was completed as the final project for the Artificial Neural Networks course at Sun Yat-sen University

(Spring 2023). We would like to express our sincere gratitude to Professor Ruixuan Wang for his invaluable guidance and constructive feedback.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. 3

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. 1, 2

[4] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *CoRR*, abs/2102.05644, 2021. 1

[5] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3034–3043. Computer Vision Foundation / IEEE, 2019. 1, 2

[6] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 8

[7] Ju He, Jieneng Chen, Shuai Liu, Adam Kortylewski, Cheng Yang, Yutong Bai, and Changhu Wang. Transfg: A transformer architecture for fine-grained recognition. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 852–860. AAAI Press, 2022. 1, 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. 1

[9] 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. 6

[10] Yuheng Li, Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Mixnmatch: Multifactor disentanglement and encoding for conditional image generation. *CoRR*, abs/1911.11758, 2019. 5

[11] Chuanbin Liu, Hongtao Xie, Zheng-Jun Zha, Lingfeng Ma, Lingyun Yu, and Yongdong Zhang. Filtration and distillation: Enhancing region attention for fine-grained visual categorization. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 11555–11562. AAAI Press, 2020. 1, 2

[12] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. 7

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[15] Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. *CoRR*, abs/1811.11155, 2018. 5

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 6

[17] Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J. Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6558–6569. Association for Computational Linguistics, 2019. 2

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[19] Jun Wang, Xiaohan Yu, and Yongsheng Gao. Feature fusion vision transformer for fine-grained visual categorization. In *32nd British Machine Vision Conference 2021, BMVC 2021, Online, November 22-25, 2021*, page 170. BMVA Press, 2021. 2

[20] Zhihui Wang, Shijie Wang, Pengbo Zhang, Haojie Li, Wei Zhong, and Jianjun Li. Weakly supervised fine-grained image classification via correlation-guided discriminative learning. In Laurent Amsaleg, Benoit Huet, Martha A. Larson, Guillaume Gravier, Hayley Hung, Chong-Wah Ngo, and Wei Tsang Ooi, editors, *Proceedings of the 27th ACM*

International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019, pages 1851–1860. ACM, 2019. 2

[21] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 842–850. IEEE Computer Society, 2015. 1

[22] Bo Zhao, Xiao Wu, Jiashi Feng, Qiang Peng, and Shuicheng Yan. Diversified visual attention networks for fine-grained object classification. *IEEE Trans. Multim.*, 19(6):1245–1256, 2017. 1, 2

[23] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 6881–6890. Computer Vision Foundation / IEEE, 2021. 1, 2