Few-Shot Classification Using the OpenAI CLIP Model

Hsiao-Ju Lin
Department of Orthopedic Surgery,
Taoyuan General Hospital,
Ministry of Health and Welfare
Taoyuan, Taiwan
email: ruru1300837@gmail.com

Shi-Jinn Horng
Department of Computer Science and Information Engineering,
Asia University
Taichung, Taiwan
email: horngsj@yahoo.com.tw

Pin-Siang Huang
Agaruda Research,
Agaruda,
Taipei, Taiwan
email: sean.huang@agaruda.io

En-Ping Lu
Department of Computer Science and Information Engineering,
National Taiwan University of Sciene and Technology,
Taipei, Taiwan
email: t6233111@gmail.com

***Abstract*—With the rapid development of artificial intelligence, the application of deep learning models has matured, but it requires large datasets to improve accuracy and generalization. To reduce reliance on large datasets, this study focuses on few-shot learning based on OpenAI's Contrastive Language-Image Pre-Training (CLIP) model and adapter technology. Unlike traditional transfer learning, this method does not require fine-tuning the model but instead combines newly learned image features with existing model features through adapters. The results demonstrate that the proposed method achieved excellent accuracy across multiple validation sets, particularly on the Describable Textures Dataset (DTD), improving from 67.53% to 75.51%, thereby reducing the need for training resources.**

Keywords—Few-shot learning; Zero-shot learning; Deep learning; Machine learning.

I. INTRODUCTION

Artificial intelligence has rapidly advanced in the fields of vision, audio, and natural language. However, training practical models usually requires large amounts of data, leading to challenges with insufficient datasets. Even with powerful models, a lack of comprehensive datasets can limit their effectiveness. Additionally, fine-tuning large models requires substantial computational resources and can lead to overfitting. With the development of large models like CLIP [1], Segment Anything Model (SAM) [2], and Vision Transformer (ViT) [5] models with zero-shot learning capabilities can address many downstream tasks. This paper explores the technique of combining the CLIP model, used as the backbone network, with adapter neural networks for fine-tuning to achieve few-shot classification. This approach reduces data requirements and training resources, avoids overfitting, and enhances cost-effectiveness in enterprise applications.

The remainder of this paper is organized as follows. Section II provides a review of related work. Section III describes the proposed method. Section IV presents the experimental results. Section V concludes the paper.

II. RELATED WORK

A. CLIP

CLIP is a multimodal model developed by OpenAI. This model is pre-trained on a large-scale dataset of image-text pairs using a contrastive learning method [9], enabling it to process both textual and visual data simultaneously. Its main advantage lies in its ability to learn general representations for both images and text, allowing it to excel in a wide variety of tasks. The primary strength of the CLIP model is its powerful zero-shot learning capability, which enables efficient classification and retrieval even in the absence of labeled data. Additionally, CLIP's multimodal nature allows it to establish connections between images and text, making it highly valuable for application scenarios that require handling multiple data forms simultaneously. Typical applications include image classification, image-text retrieval, and image generation.

B. Adapter

A lightweight neural network structure is typically used to add new learning capabilities to pre-trained models. The original backbone network is frozen, and only the external network is trained, eliminating the need to retrain the entire model. This approach allows for the preservation of the pre-trained model's knowledge while enabling efficient learning with limited data. Adapters usually consist of a small fully connected layer or MultiLayer Perceptron (MLP) placed between specific layers of the pre-trained model. These adapter modules receive inputs from the pre-trained model during the forward propagation process, generate new features after the adapter's computation, and combine these features with the original ones. This way, the adapter can learn the feature representations needed for new tasks without altering the original features of the pre-trained model. Another important application of adapters is reducing the risk of overfitting. Since the number of parameters in adapters is relatively small, they effectively mitigate overfitting issues on small datasets. This technique also significantly reduces the model's training time and resource requirements, making it particularly suitable for resource-constrained applications. Initially applied in Natural Language Processing (NLP) [11] domain, it has also been proven effective in the computer vision field [10], such as the SAM adapter [12].

C. Embedding learning

In few-shot classification tasks, one of the methods is embedding learning [13], where some samples x from the training dataset are passed through a neural network f(x) for feature extraction and converted into a vector. This process can be seen as transforming an image from a high-dimensional space to a low-dimensional space, where the similarity between classes is easier to discern. Additionally, another network g() can be trained to function similarly to f(), converting high-dimensional images into low-dimensional vectors. The classification is then performed by comparing the similarity between this vector and the class vector.

D. Transformer

The Transformer [7] is a neural network architecture based on self-attention mechanisms, introduced in 2017 and widely adopted in sequence modeling tasks. In the field of NLP [6], the introduction of the Transformer model has brought significant advancements. Traditionally, Long Short-Term Memory networks (LSTM) [4] and Recurrent Neural Networks (RNN) [3] were widely used for processing sequential data. However, these models have limitations when dealing with long-range dependencies. Additionally, autoregressive models have important applications in text generation, and the Transformer has improved upon these foundations. The self-attention mechanism of the Transformer allows for parallel processing of entire sequences, overcoming the serial computation limitations of RNNs, thereby significantly improving computational efficiency and performance. The application range of Transformers is extensive, including various NLP tasks such as machine translation, text generation [24], semantic analysis [25], text summarization [26], and question-answering systems. Furthermore, in the field of computer vision, following ResNet [8], the Transformer has also demonstrated strong capabilities. The ViT applies the Transformer's self-attention mechanism to tasks, such as image classification [5], object detection [27], and image segmentation [28], achieving remarkable results. These applications showcase the flexibility and superiority of the Transformer model in handling different data types and application scenarios.

E. Tip-Adapter [21]

The traditional CLIP-Adapter [20] integrates new knowledge with CLIP's pre-trained knowledge by fine-tuning a lightweight residual feature [8] adapter connected to the pre-trained CLIP model. This architecture allows the model to combine newly learned features with the existing visual representations of CLIP. Specifically, when an image is input, CLIP's visual encoder first extracts its features, which are then passed through the lightweight adapter to generate updated feature representations. These updated features are fused with the original CLIP features via residual connections, thus effectively combining new knowledge with pre-trained knowledge, significantly improving the performance of few-shot classification. Tip-Adapter is a training-free variant of CLIP-Adapter designed to enhance the performance of the CLIP model in few-shot classification tasks by using image cache. The principle involves constructing an image cache with a small number of images, storing the feature vectors of these images along with their corresponding labels in the cache. During inference, the model does not require additional parameter updates but instead makes classification decisions by retrieving features from the cache. The advantage of this method lies in its ability to quickly adapt to new tasks by leveraging the pre-trained knowledge of the CLIP model and performing lightweight feature adjustments with a small amount of few-shot data.

III. METHOD

A. Basical Setup

This study was conducted using Python 3.11.6 and PyTorch 2.1.2 for model development, and training was performed on an NVIDIA GeForce GTX 1080 Ti GPU. All experiments were conducted on the Ubuntu 20.04.1 LTS operating system.

B. Data Augmentation

During the adapter training, a series of data augmentation techniques were introduced to preprocess the dataset, including cropping, rotation, adding various types of noise, etc. These methods are common and effective data augmentation techniques that enhance the model's generalization ability and improve its robustness. After experimentation, the method finally chosen in this paper was to increase image contrast, as described in (1):

$$I'(x,y) = \alpha(I(x,y) - \mu) + \mu \quad (1)$$

In this paper, we use the following variables to describe the process of adjusting image contrast. Let $I(x,y)$represent the brightness value of the original image at position (x,y), and *I'(x,y)* represent the brightness value of the image after contrast adjustment at position (x,y). Here, α is the contrast adjustment coefficient. When $\alpha > 1$, the contrast increases; μ is the mean brightness value, as shown in (2):

$$\mu = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N} I(x,y) \quad (2)$$

μ is used to maintain the overall brightness of the image, where M and N represent the width and height of the image, respectively. In this paper, α is set to 2, and the enhanced image is shown in Figure 1. Contrast enhancement is a simple and effective data augmentation method. Many studies have demonstrated that it can significantly improve the model's ability to recognize low-contrast images.

In addition, Gaussian noise was applied for data augmentation. This method perturbs images with zero-mean Gaussian noise of standard deviation σ, as formulated in (3).

$$I'(x,y) = I(x,y) + N(0,\sigma^2) \quad (3)$$

Here, *N(0, σ^2)* represents Gaussian noise with a mean of 0 and a variance of σ^2. In this paper, $\sigma^2$is set to a random value between 25 and 75. By adding Gaussian noise to the images, we can simulate more real-world scenarios, further enhancing the model's generalization ability and robustness.

Figure 1. Comparison of Contrast Enhancement: Left (Original data from Oxfordflower 17 [14]) and Right (Enhanced in this paper).

Figure 2. Effect of Gaussian Noise Augmentation, Original Data from Oxfordflower 17 [14].

The addition of Gaussian noise is illustrated in Figure 2. The augmentation effect of Gaussian noise on the image dataset has also been widely studied and proven. We will increase the number of samples in the training dataset to twice the original amount.

C. Image cache

Assume the training dataset has N classes. Let I_{i_j} represent a randomly selected image from the i^{th} class in the training dataset, with i serving as the key for that image. The selected image I_{i_j} is then input into our CLIP image encoder, which outputs a 512-dimensional vector $fo(I_{i_j})$. Using the output vector $fo(I_{i_j})$, we construct an image cache according to the classes, as shown in Figure 3. After feature extraction, the training images are stored as key-value pairs in a database. By applying the t-SNE [19] dimensionality reduction technique, we can visualize the distribution of different classes in the space, as illustrated in Figure 4, using 17 classes of images from Oxford Flower 102 to build the image cache. This is the starting point of this paper: using these distributions for classification. In our experimental design, we observe from the TIP-Adapter that the accuracy improves as the number of images increases. Therefore, we directly select 16 images per class to construct this cache. If there are N classes, the cache can be represented as a matrix of size [16×N, 512]. Cache models have been applied to improve some issues in NLP field, such as Knn-LMs [23], which also demonstrate the power of cache models. This approach allows for the initial rapid categorization of targets, further enhancing the adaptability and flexibility of the model.

There are several advantages to using this method within our model. Since the dataset contains a limited number of samples, a small amount of data can be used to construct this cache. Directly using a complex neural network could easily lead to overfitting. Therefore, we use the vectors generated by the CLIP image encoder, which has strong generalization capabilities. Similar vectors tend to cluster in closer regions within the space. By leveraging this characteristic, we avoid overfitting, reduce the impact of outliers, and decrease the number of learnable parameters, thereby increasing training speed.

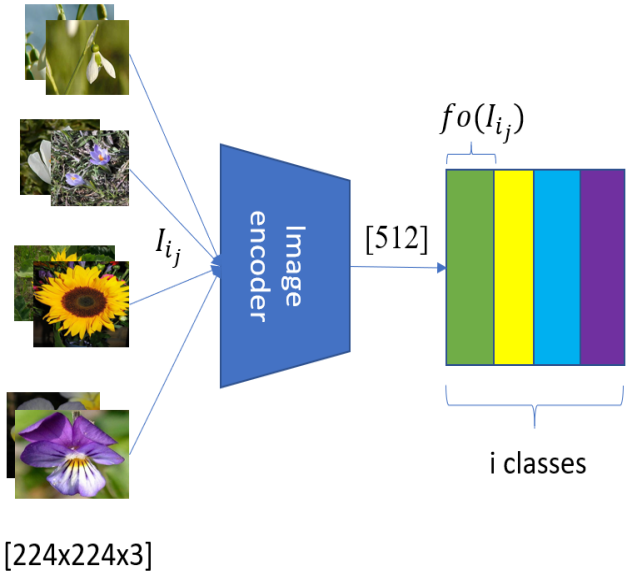

Figure 3. Image Cache Example Diagram.

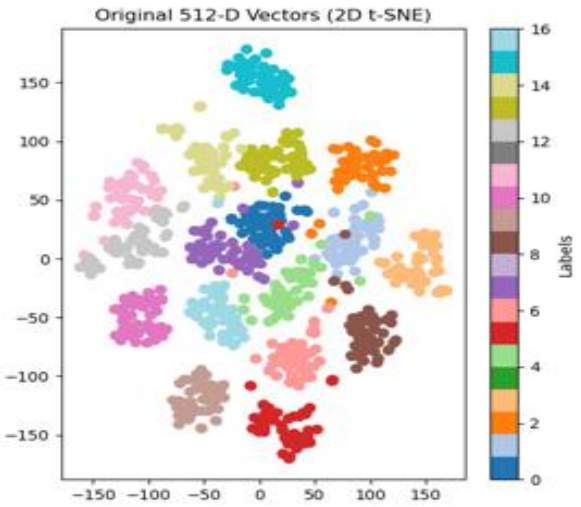

Figure 4. t-SNE Visualization of Oxford17 Data.

D. Image adapter

To make the CLIP model more effective in few-shot classification, this study introduces an adapter architecture as an external framework for the model. Adapters were initially used for fine-tuning in the natural language processing (NLP) domain and have since been widely applied across various deep learning fields. Unlike traditional deep learning neural networks, this method significantly reduces the number of training parameters and training time. It preserves the complexity of the original network while avoiding overfitting, which can occur when the amount of data is too small. The adapter achieves model fine-tuning by inserting lightweight parameter layers into the pre-trained model, without requiring extensive parameter updates across the entire network. In this paper, the adapter consists of a two-layer fully connected MLP with the activation function ReLU. The vectors f_o , obtained by

passing images through the image encoder, are used as training data to train the MLP. The parameters of the MLP are 512×128 and 128×512, respectively, which can be represented as shown in (4):

$$f_x = \mathrm{W}_2 ReLU(W_1 f_v + b_1) + b_2. \quad (4)$$

W1 and W2 are the weight matrices from the input layer to the hidden layer and from the hidden layer to the output layer, respectively. b1 and b2 are the bias vectors. We only train the adapter, freezing all the parameters of the image encoder, which reduces our training parameters to just those of the fully connected layers. During the training phase, our architecture is illustrated in Figure 5, and the following sections will introduce the process in sequence.

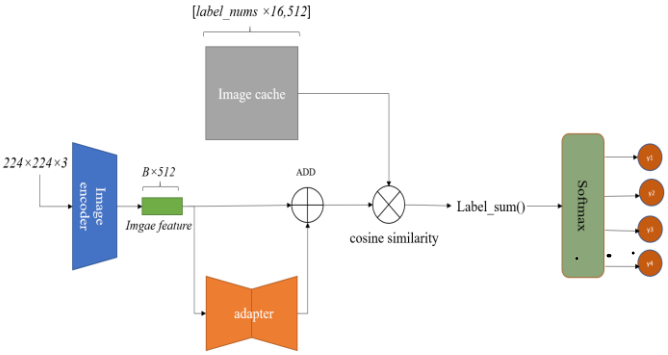

Figure 5. The proposed system architecture.

During the training phase, through the image cache, we construct the cache from other data in the training dataset, which is divided into two paths. One path passes through the MLP adapter for training, while the other remains unprocessed. The vector f_x output by the adapter is then added to the vector f_o, which bypassed the adapter, as shown in (5):

$$f_c = f_x + f_o \quad (5)$$

The vector fc is compared with the image cache database using dot product similarity. The similarities for the same category are summed to obtain the classification score for that category, as shown in (6):

$$i \in label\ ,\ y_i = \sum f_c \cdot image\_cache_i \quad (6)$$

The similarity for each category obtained by this method is then passed through a Softmax function to produce the classification result for the image, as shown in (7):

$$\hat{y}_i = \frac{e^{y_i}}{\sum_j e^{y_j}} \quad (7)$$

Compared to traditional single-point key-value lookups, our approach can better avoid the occurrence of extreme values within the image cache, thereby reducing classification errors. Our loss function is the cross-entropy loss function, as shown in (8):

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} y_i^{(n)} \log\left(\hat{y}_i^{(n)}\right) \quad (8)$$

where N represents the total number of samples, K represents the number of classes, $y_i^{(n)}$ denotes the true label of the n^{th} sample, and $\hat{y}_i^{(n)}$ represents the predicted probability of the n^{th} sample after applying the Softmax function.

IV. EXPERIMENTS

A. Experiment result

To validate the effectiveness of our method, we conducted a series of experiments. First, we trained different adapters on multiple datasets, including Flowers102 [29], Caltech-101 [16], OxfordPets [18], FGVCAircraft [17], and DTD [15]. We randomly selected a fixed number (16) of images from each class in the dataset to construct the image cache. The remaining data was then augmented and used for training, where each class has at most 40 images. Next, we trained the model. The pre-trained weights for the CLIP image encoder were ViT-B/32, which is the most commonly used weight. During training, we used the Adam optimizer and cross-entropy loss function, with an initial learning rate set at 0.001. To better adapt to changes during the training process, we reduced the learning rate to 80% of its original value every 10 epochs, for a total of 30 epochs. The batch size was set to 32. We kept the output vector dimension of the CLIP image encoder fixed, and our model achieved good accuracy on various few-shot test sets, as shown in Table I, outperforming other benchmark models in this field. These results demonstrate the excellent classification ability of our method and prove its effectiveness. In the following sections, we will provide a more detailed analysis of our experimental results.

To ensure stable performance of the model, we conducted tests and validations on several datasets and compared the results with previously proposed models, as shown. We imported the ViT-B/32 weights into the CLIP model and tested the model's performance according to the experimental setup described in Section IV.A. We also compared it with CLIP-Adapter and Tip-Adapter, both of which are benchmark models that use CLIP for classification tasks.

TABLE I. ACCURACY OF CLASSIFICATION MODELS ON DIFFERENT DATASETS

Method/ Dataset	CLIP	CLIP adapter[20]	Tip adapter[21]	Ours
Oxfordflower 102 [14]	66.35%	93.90%	94.80%	95.14%
OxfordPet [18]	85.98%	87.84%	89.70%	92.84%
DTD [15]	42.27%	65.96%	67.53%	75.51%
Caltech-101 [16]	86.01%	92.49%	92.86%	94.53%
FGVCA [17]	17.89%	32.10%	35.55%	39.85%

B. Efficacy

Since CLIP uses the similarity between text and images, we selected the category with the highest similarity as the predicted result. However, similar textual phrases can also affect

accuracy. For example, in the Oxford Flower 102 dataset, which contains 102 types of flowers, phrases like "a type of," "a type of flower on the ground," or "the picture shows a type of flower" could lead to variations in accuracy. Therefore, we standardized the input format, comparing the similarity using the English phrase "This is a photo of [specific flower]." For instance, for a sunflower, the input would be "This is a photo of sunflower."

According to our experimental data, all datasets showed a significant improvement compared to the original CLIP. Our model also showed a notable improvement on the DTD texture dataset, with accuracy increasing from 67.53% to 75.51%, outperforming Tip-Adapter. The other datasets also showed slight improvements. These results demonstrate the effectiveness and robustness of our model across these datasets.

C. Training parameters

In the field of few-shot classification, the ability to apply the model in scenarios such as data engineering has undeniable value, as it allows classification with a small amount of data. In data engineering, obtaining the first-hand manually labeled data often requires a significant amount of time. If we can reduce training time, we can save considerable time in the initial stages. As shown in Table II, only the corresponding adapters need to be trained, so there is less concern about the equipment being unable to handle the training. Additionally, with only 30 epochs per dataset, the training time can be compressed into just a few hours, significantly reducing the overall development process.

D. Ablation experiment

In this section, we conduct an ablation study to gain a deeper understanding of the contributions of various parts of the model, including the generalization ability of the original CLIP and the impact of the image cache combined with adapters. As shown in the results in Table III, there is a significant difference in accuracy across different datasets when using or not using the adapter strategy. Adding the adapter clearly demonstrates that this approach can indeed improve the model's accuracy. When only using CLIP, we still need to calculate the similarity with text vectors, but it is sometimes challenging to fully capture the features of an image with a textual description, particularly in datasets like DTD textures, where the accuracy is not high. Therefore, we also designed a method to compare each input image with every vector in the image cache, using the label of the most similar vector as the output. The results show that this paper effectively leverages CLIP's image encoder, making it more accurate in classification.

TABLE II. TRAINING PARAMETERS OF COMMON FEW-SHOT MODELS

Method	Parameters
ResNet-12	9.5630 M
EfficientNet-B0 [22]	5.3305M
Ours	0. 13171M

TABLE III. ACCURACY ON DIFFERENT TEST SETS UNDER VARIOUS CONDITIONS

	one point	Image cache sum	Image cache sum+adapter
DTD [15]	51.00%	56.87%	75.51%
FGVCA [17]	25.55%	27.52%	47.88%
Caltech-101 [16]	87.25%	84.80%	94.53%
OxfordPet [18]	62.52 %	68.87 %	92.84%
Oxfordflower102 [29]	64.55%	87.32%	95.14%

V. CONCLUSION

This study explored the few-shot learning capabilities of large models for classification tasks by designing a model based on CLIP's image encoder with adapters. Experimental results across multiple datasets demonstrate the effectiveness of the proposed method. Owing to the CLIP backbone, our model exhibits strong generalization ability. Although frozen parameters are still required during inference, the model maintains relatively low parameter and computational costs, making it feasible for edge deployment. Despite promising results, further improvements remain possible.

While our results are promising, several improvements remain possible:

1. Dataset Diversification: Future studies could use more obscure datasets to better validate generalization, as CLIP may have encountered commonly used datasets.
2. Enhanced Data Augmentation: Beyond contrast and Gaussian noise, more effective augmentation methods could be explored.
3. Architecture Enhancement: Since only CLIP's image encoder was used, future work could leverage its text encoder or explore alternative architectures to improve performance and generalization.

ACKNOWLEDGEMENT

This work was supported in part by the National Science and Technology Council under contract number 114-2221-E-468 -014 -.

REFERENCES

[1] A. Radford et al., "Learning transferable visual models from natural language supervision," *International conference on machine learning*, 2021, pp. 8748–8763.

[2] A. Kirillov et al., "Segment anything," *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[3] L. R. Medsker et al., "Recurrent neural networks," *Design and Applications*, vol. 5, no. 64–67, p. 2, 2001.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[5] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint arXiv:2010.11929, 2020.

[6] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011.

[7] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in 2006 *IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, 2006, vol. 2, pp. 1735–1742.

[10] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," *Advances in neural information processing systems*, vol. 30, 2017.

[11] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," *International conference on machine learning*, 2019, pp. 2790–2799.

[12] J. Wu et al., "Medical sam adapter: Adapting segment anything model for medical image segmentation," arXiv preprint arXiv:2304.12620, 2023.

[13] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2593–2601.

[14] M.-E. Nilsback and A. Zisserman, "A visual vocabulary for flower classification," in 2006 *IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, 2006, vol. 2, pp. 1447–1454.

[15] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3606–3613.

[16] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *2004 conference on computer vision and pattern recognition workshop*, 2004, pp. 178–178.

[17] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," arXiv preprint arXiv:1306.5151, 2013.

[18] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar, "Cats and dogs," *2012 IEEE conference on computer vision and pattern recognition*, 2012, pp. 3498–3505.

[19] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[20] P. Gao et al., "CLIP-adapter: Better vision-language models with feature adapters," *International Journal of Computer Vision*, vol. 132, no. 2, pp. 581–595, 2024.

[21] R. Zhang et al., "Tip-adapter: Training-free CLIP-adapter for better vision-language modeling," arXiv preprint arXiv:2111.03930, 2021.

[22] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International conference on machine learning*, 2019, pp. 6105–6114.

[23] U. Khandelwal, O. Levy, D. Jurafsky, L. Zettlemoyer, and M. Lewis, "Generalization through memorization: Nearest neighbor language models," arXiv preprint arXiv:1911.00172, 2019.

[24] T. Brown et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[26] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," *International conference on machine learning*, 2020, pp. 11328–11339.

[27] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *European conference on computer vision*, 2020, pp. 213–229.

[28] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[29] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, 2008, pp. 722–729.