CLIP-based Multi-category Visual Classification Task by Few-shot Learning

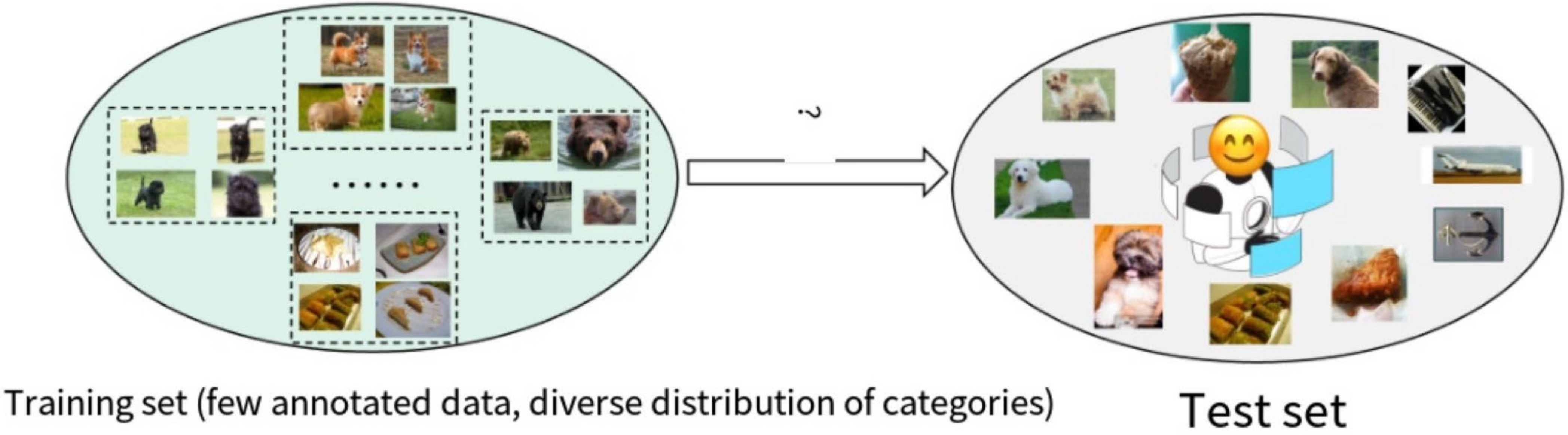
> Presenter: 王子轩 物31
> Instructor: 穆太江

- Introduction
    - Problem Set-up
    - Baseline: CLIP
- Method & Result
    - JCLIP-Adapter
- Related Work & Exploration
    - Modeling predicted probability distribution

Question introduction

With the rapid development of artificial intelligence, visual language model has become an important technology in the field of computer vision. These models perform well in multitasking and show wide application potential. However, with the high cost of data annotation, they need to improve their performance in the face of domain-specific challenges. Therefore, how to improve the performance of the model in a specific field with the support of a small amount of data has become a hot issue in current research. At the same time, using small amounts of data to enhance performance in multiple areas is more challenging.

Therefore, this competition requires participants to use a few multi-domain training samples to explore innovative model training strategies in the era of large models, so as to achieve accurate classification of the test set composed of multi-domain data.



Few-shot Learning (FLS)

- Problem formalization :

$$D_T = (D_{train}, D_{test}) = \{X_T, P(X_T)\}$$

$$D_{train} = \{(x_i, y_i)\}_{i=1}^{N_{train}} \qquad D_{test} = \{x_j\}$$

$$x_i, x_j \in X_T \subset X \qquad y_i \in Y_T \subset Y$$

> D_{train} contains C classes with only K samples per class.

- Goal:

$$\text{find} \quad f \in \mathcal{H} : X \to Y \qquad \mathcal{H} : \text{Hypothesis space}$$

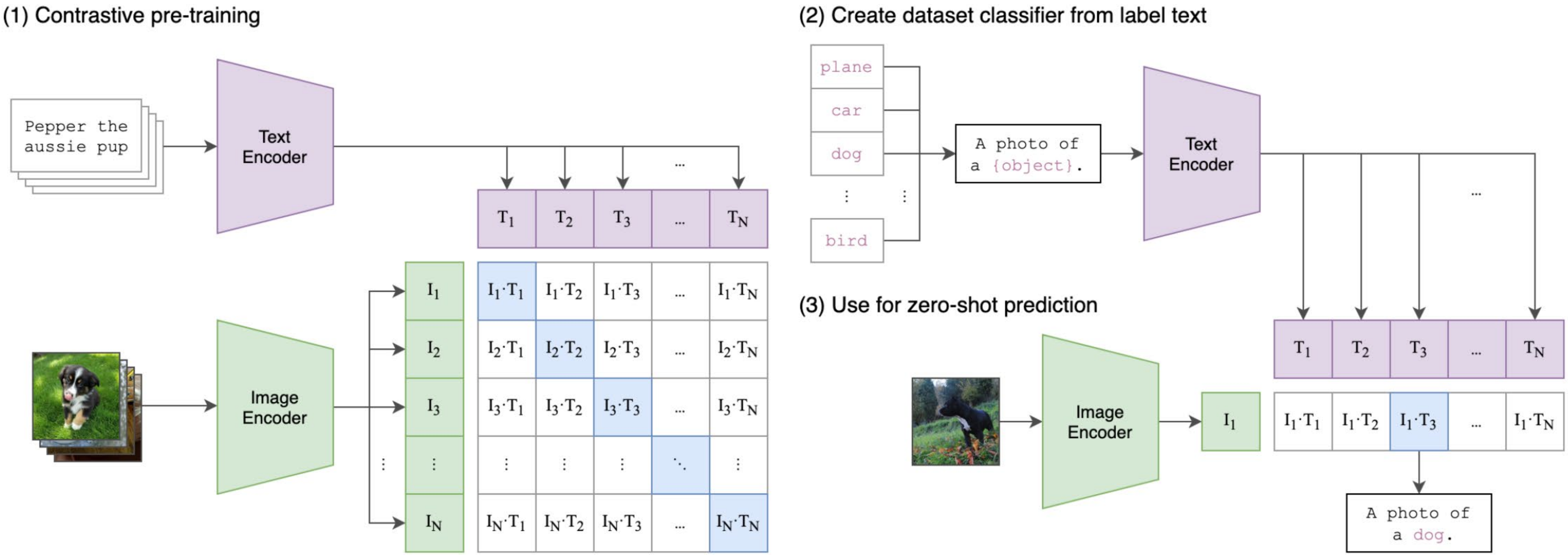
$$\min_f \quad \varepsilon = \mathbb{E}_{(x,y)\sim\mathcal{D}_{train}} L(f(x), y)$$

> Mostly use supervised dataset to overcome the few shot.

$$D_A = \{(x_i^a, y_i^a)\}_{i=1}^{N_{aux}} \qquad x_i^a \in X_A \subset X \qquad y_i^a \in Y_A \subset Y$$

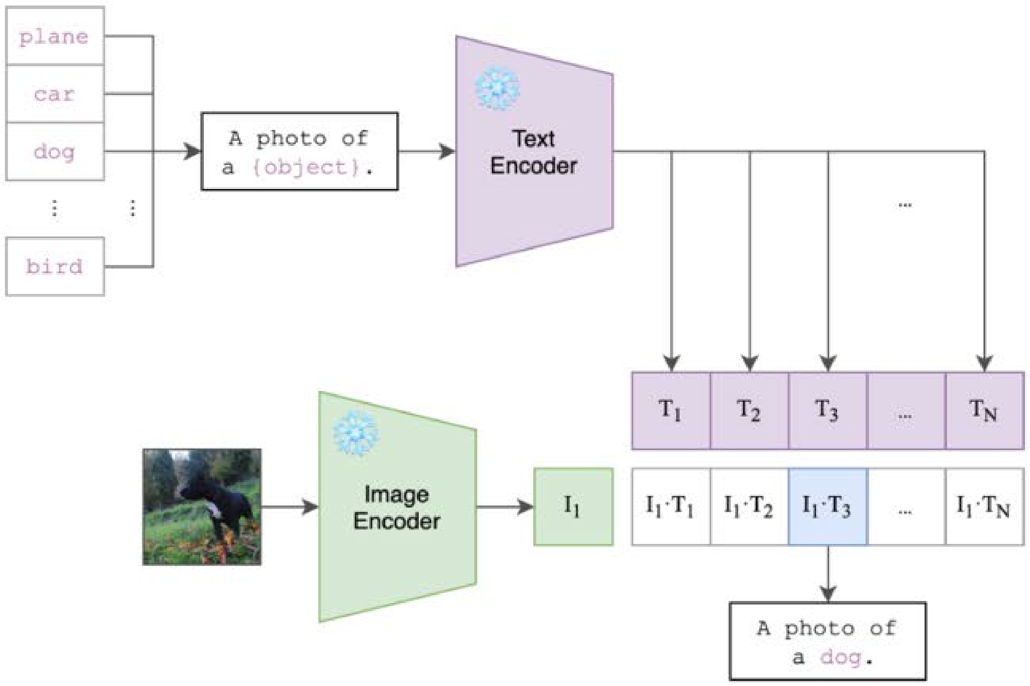
$$N_{aux} \gg N_{train}, |Y_A| \gg |Y_T|$$

Contrastive Language-Image Pre-training

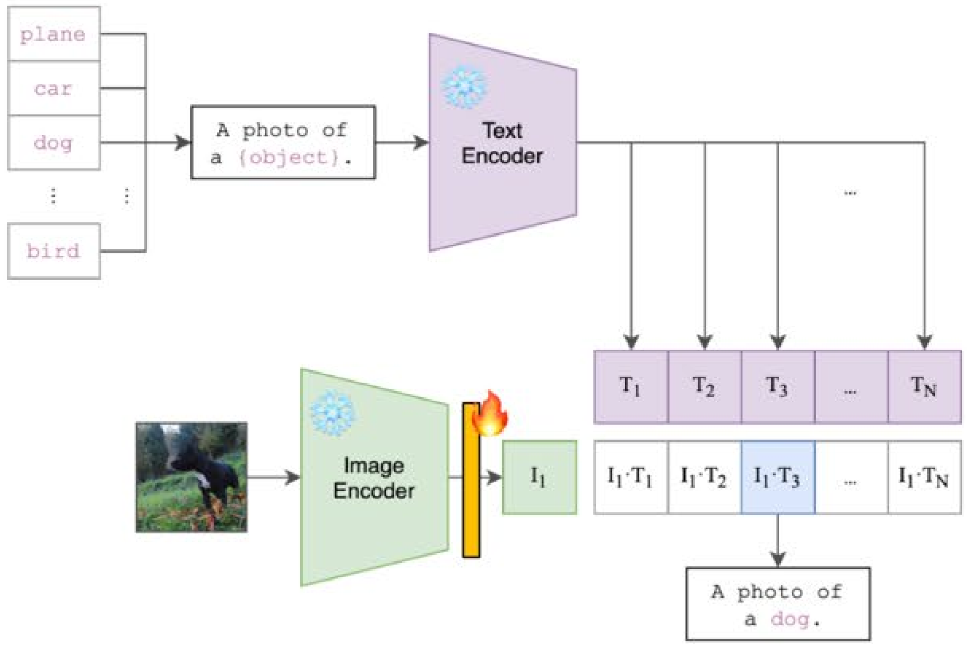


> Radford, Alec, et al. "Learning transferable visual models from natural language supervision." International conference on machine learning. ICML, 2021.

Baseline： Zero-shot performance is better than linear probe fine-tuning performance.



Training-Free



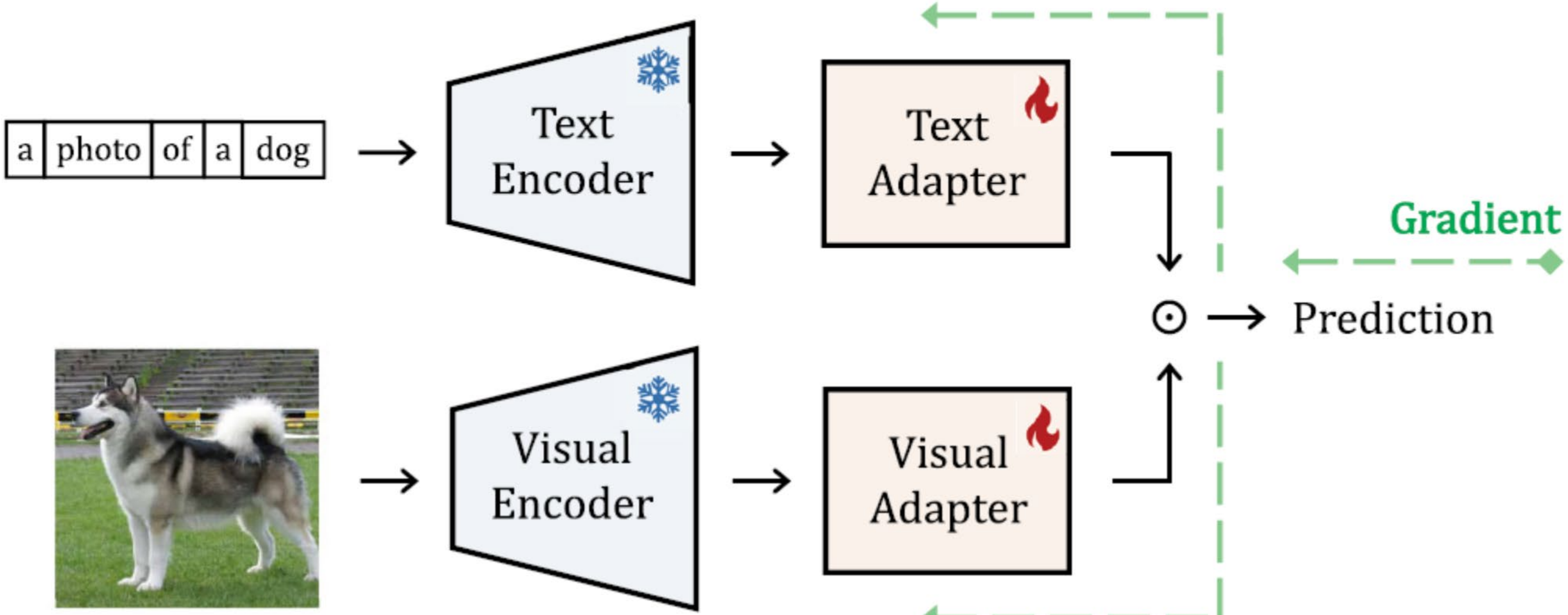
Training

> Training-Free baseline achieves **58.3%** top-1 accuracy on *test set*.
> Training baseline achieves only **51.2%** top-1 accuracy on *test set*.

How CLIP is trained?

```
# image_encoder - ResNet or Vision Transformer
# text_encoder  - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l]       - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t             - learned temperature parameter
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)
# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)
# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
```

How to properly adapt CLIP for FSL? *JCLIP-Adapter*



JCLIP-Adapter Algorithm

- *Input:* $\mathcal{I} \in \mathbb{R}^{W\times H\times C}$, $\mathcal{T}_i \in \mathbb{R}^{L_i}$
- *Feature Extraction:* $f = \text{image\_encoder}(\mathcal{I}) \in \mathbb{R}^d$, $w_i = \text{text\_encoder}(\mathcal{T}_i) \in \mathbb{R}^d$
- *Query Matrix:* $\mathcal{W} = \text{concat}(w_1, \cdots, w_K) \in \mathbb{R}^{K\times d}$
- *Adaptive Layer:* $A_v(f) = \text{ReLU}(f^T\mathcal{W}_1^v)\mathcal{W}_2^v$, $A_t(\mathcal{W}) = \text{ReLU}(\mathcal{W}^T\mathcal{W}_1^t)\mathcal{W}_2^t$
- *Res-addition:* $f^\star = \alpha A_v(f) + (1-\alpha)A_t(\mathcal{W})$, $\mathcal{W}^\star = \beta A_t(\mathcal{W}) + (1-\beta)\mathcal{W}$
- *Cos-Similarity vector:* $\mathcal{P} = \text{softmax}(\mathcal{W}^{\star\top} f^\star)$, *label* = $\text{argmax}_k \mathcal{P}_k$
- *Opimization task:* $\min \quad \mathcal{L}_{\theta=\{W_1^t, W_2^t, W_1^v, W_2^v\}} = -\frac{1}{N}\sum_i^N \log \frac{\exp\left(\mathcal{W}_i^{\star\top} f_i^\star/\tau\right)}{\sum_{j=1}^N \exp\left(\mathcal{W}_j^{\star\top} f_i^\star/\tau\right)}$

Core Method Pseudocode

JCLIP-Adapter

```
# W_1: image-adaptive network
# W_2: text-adaptive network
for _ in iteration:
    # Randomly sample images and text label
    im,labels = data_loader()
    txquery = prompt_generator(labels)
    # Extract image and txquery features
    im_ft = image_encoder(im)
    txquery_ft = text_encoder(txquery)
    #  Compute adapted image features
    im_ft = R * W_1(im_ft) + (1-R) * im_ft
    text_ft = R * W_2(txquery_ft) + (1-R) * txquery_ft
    # L2 normalizeboth
    im_ft,tx_ft = normalize(im_ft,txquery_ft)
    # Compute
    pred = softmax(im_ft @ tx_ft^T)
    loss = cross_entropy(pred,labels)
    # Backward and update linear layer
    update(W_1.params,W_2.params)
```

```python
# 引入新的残差适配网络结构 返回值为一个适配后的图像特征
class CustomJCLIP(Model):

    def __init__(self):
        super().__init__()
        # self.image_encoder = model.encode_image
        self.text_encoder = model.encode_text
        self.dtype = model.dtype
        self.adapter = Adapter()

    def execute(self, image_feature,text_features):
        # image_features = self.image_encoder(image)
        x = self.adapter(image_feature)
        global ratio # 设置学习比例
        image_feature = ratio * x + (1 - ratio) * image_feature
        image_feature = image_feature / image_feature.norm(dim=-1, keepdim=True)
        # text_features = text_features / text_features.norm(dim=-1, keepdim=True)

        predict_vector = (100.0 * image_feature @ text_features.transpose(0, 1)).softmax(dim=-1)
        _ , top_label_predicted = predict_vector.topk(1)
        predict_vector = jt.float32(predict_vector)
        # probability, top_label = text_probs[0].topk(1)
        return predict_vector, top_label_predicted # 输出这个预测的概率值分布向量

# 实例化模型并进行训练
jclip_adapter = CustomJCLIP()
loss_fn = nn.CrossEntropyLoss()
epoches = 20
batch_size = 64
```

```python
def train(jclip_adapter, dataloader,loss_fn, optimizer,epoch):
    jclip_adapter.train()
    # 保存每一个epoch的loss
    train_losses = list()
    # 记录每一个epoch的准确率
    accuracy = list()
    for batch_idx, (feature, label) in enumerate(dataloader):
    #for i in range(len(dataloader)):
        image_feature = feature
        label = label
        predict_vector = jclip_adapter(image_feature, text_features)[0]
        top_labels = jclip_adapter(image_feature, text_features)[1]
        for i in range(len(label)):
            if top_labels[i] == label[i]:
                accuracy.append(1)
            else:
                accuracy.append(0)
        accurate = sum(accuracy) / len(accuracy)
        loss = loss_fn(predict_vector, label)
        optimizer.step(loss)
        train_losses.append(loss)
        accuracy.append(accurate)
        if batch_idx == 1:
            print('in the {} training epoch, loss is {}, accuracy is {}, '.format(epoch, loss, accurate))
    return train_losses, accuracy
```

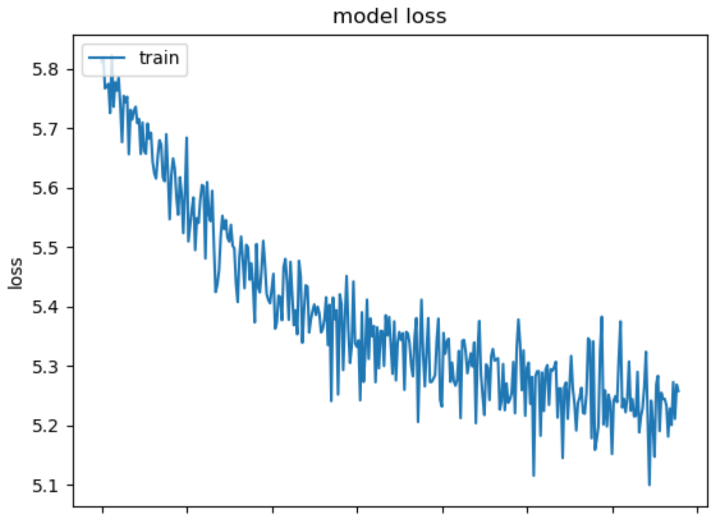
Small tricks can also improve accuracy

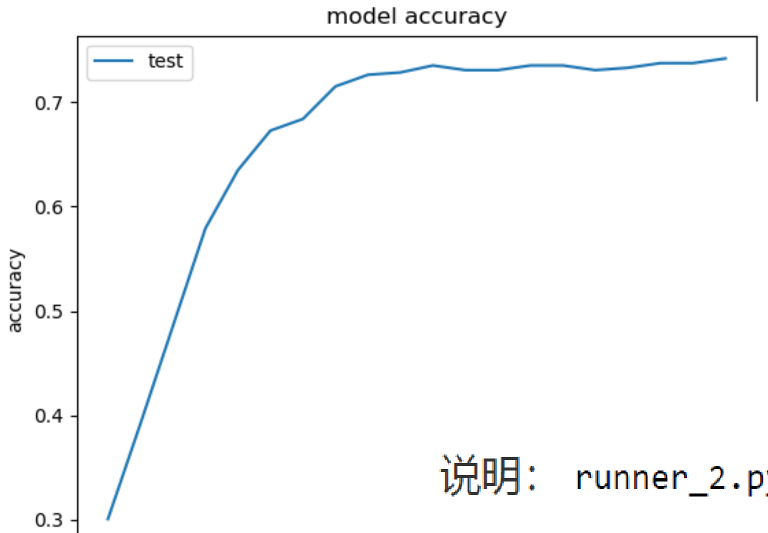
```
# 将每一类对应的样本标签进行prompt engineering
new_classes = []
for c in classes:
    c = c.split(' ')[0]
    if c.startswith('Animal'):
        c = c[7:]
        c = 'a photo of ' + c + ', a kind of an animal'
    if c.startswith('Thu-dog'):
        c = c[8:]
        c = 'a photo of ' + c + ', a category of a dog'
    if c.startswith('Caltech-101'):
        c = c[12:]
        c = 'a photo of ' + c + ', a kind of an object'
    if c.startswith('Food-101'):
        c = c[9:]
        c = 'a photo of ' + c + ', a type of food'
    new_classes.append(c)

text = clip.tokenize(new_classes)
text_features = model.encode_text(text)
text_features /= text_features.norm(dim=-1, keepdim=True)
print('text features matrix has been processed')
```

Shih Tzu (西施狗)or a dog/ a red toy car?

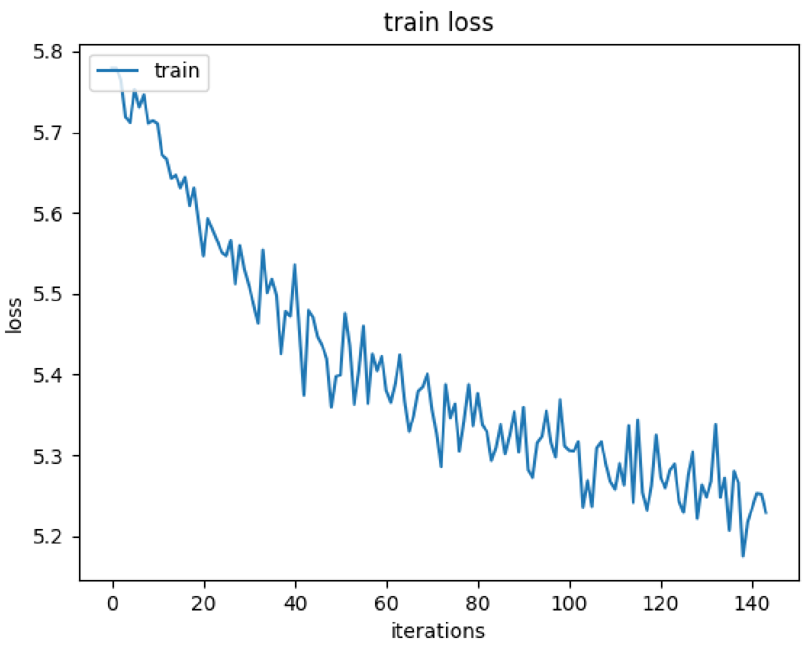
2024072512134 079904245	result (5)...	王子轩	2024-07-25 12:13:40	完成	0.6267	0.8207	下载

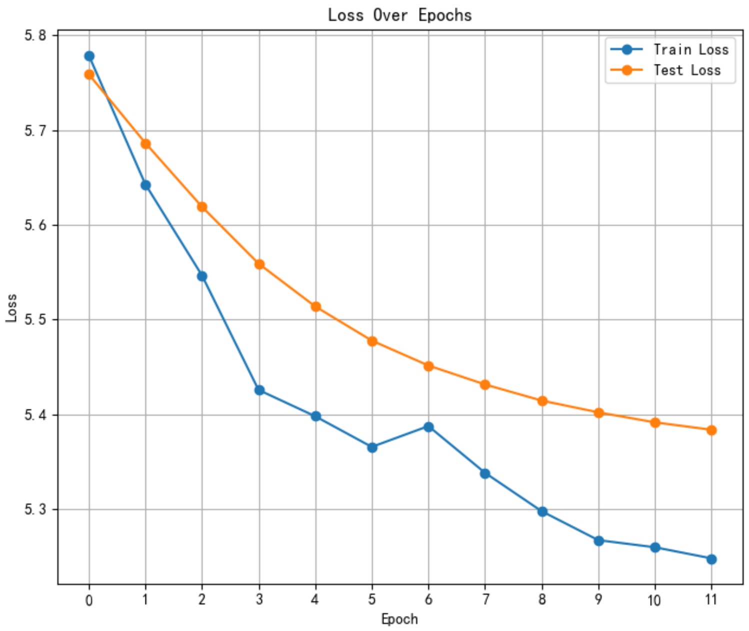


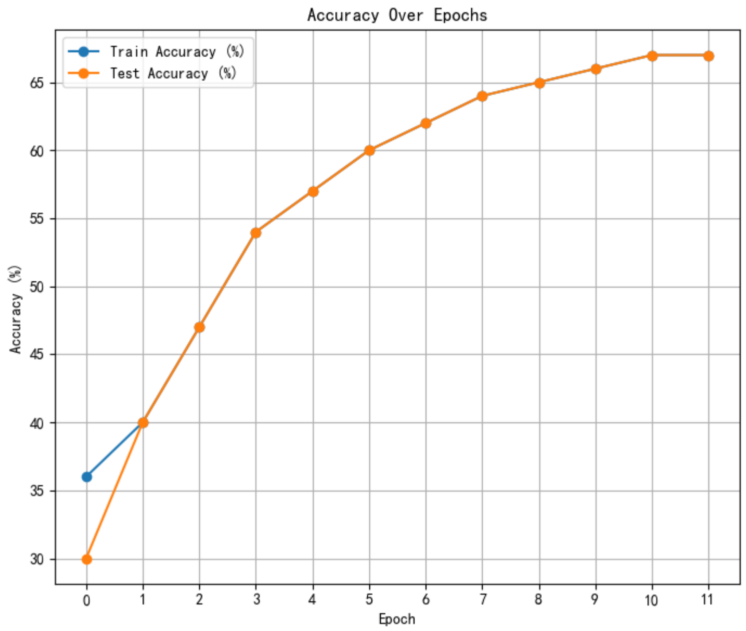


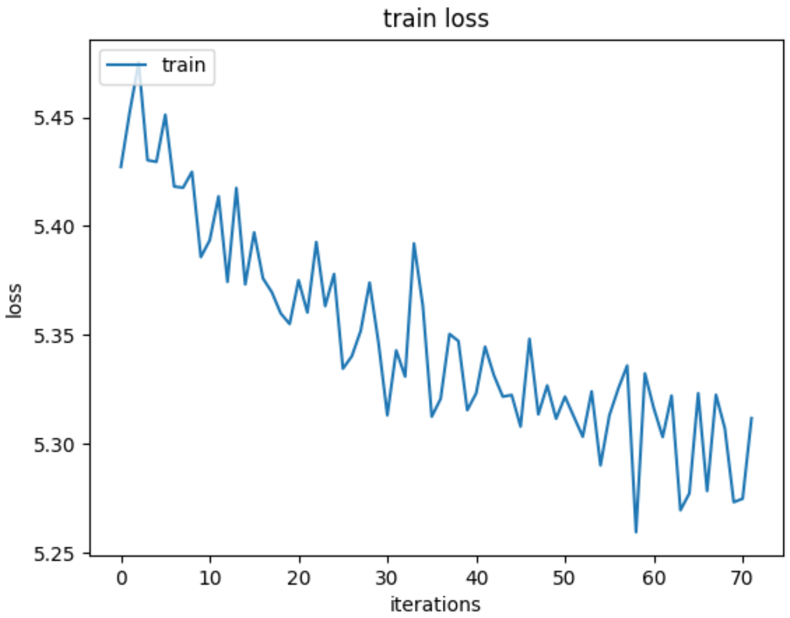
模型	$top1$	$top5$
baseline.py	0.58	0.80
baselin_ft.py	0.51	0.74
runner_2.py	0.63	0.82

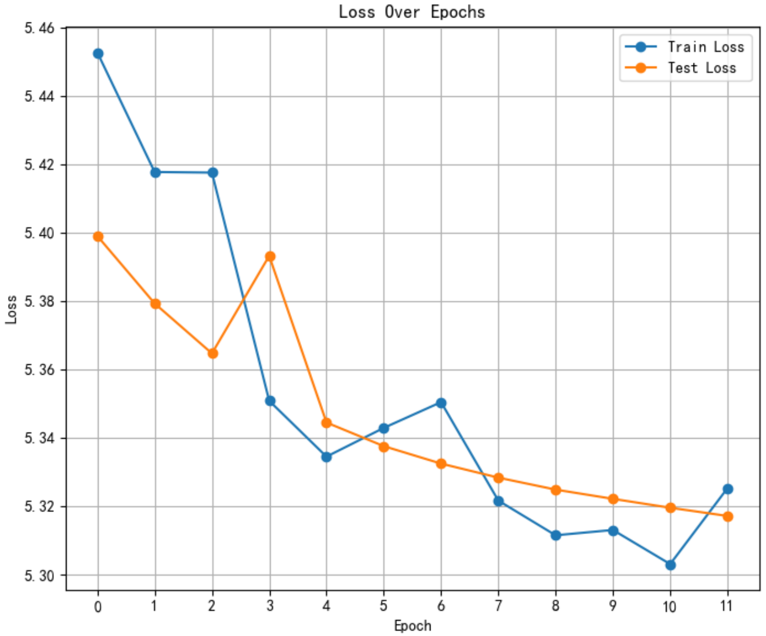
说明： `runner_2.py` 是本次实验最终提交的模型

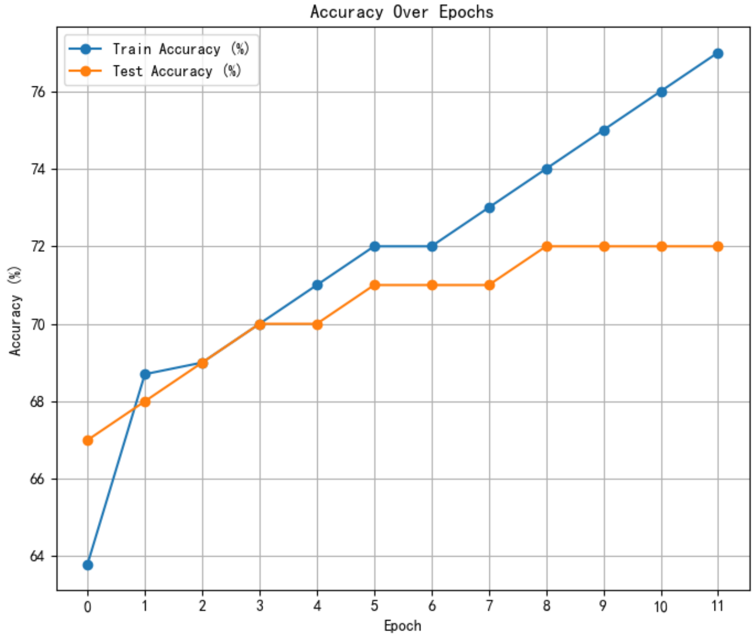

train loss
train
loss
iterations


Loss Over Epochs
Train Loss
Test Loss
Loss
Epoch


Accuracy Over Epochs
Train Accuracy (%)
Test Accuracy (%)
Accuracy (%)
Epoch


train loss
train
loss
iterations

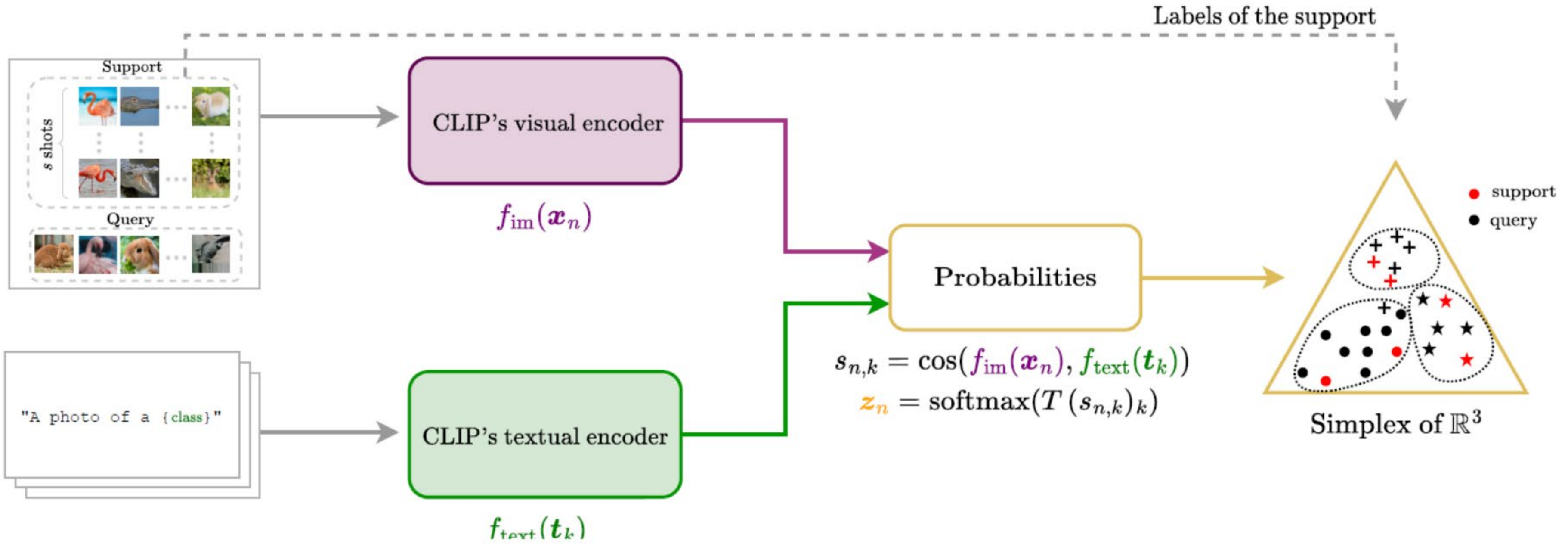

Loss Over Epochs
Train Loss
Test Loss
Loss
Epoch


Accuracy Over Epochs
Train Accuracy (%)
Test Accuracy (%)
Accuracy (%)
Epoch

Exploration

Modeling predicted probability distribution by transductive inference

> Transductive Zero-Shot and Few-Shot CLIP (CVPR2024)



$$p(z \mid \alpha_k) = \frac{1}{\mathcal{B}(\alpha_k)} \prod_{i=1}^{K} z_i^{\alpha_{k,i}-1} \quad \text{for } z \in \Delta_K$$

Exploration

Modeling predicted probability distribution by transductive inference

: $\mathcal{P} = \mathrm{softmax}(\mathcal{W}^{\star\top} f^{\star})$, *label* = $\mathrm{argmax}_k \mathcal{P}_k$

$$\min \quad \mathcal{L}_{\theta=\{W_1^t, W_2^t, W_1^v, W_2^v\}} = -\frac{1}{N}\sum_i^N \log \frac{\exp\left(\mathcal{W}_i^{\star\top} f_i^{\star}/\tau\right)}{\sum_{j=1}^N \exp\left(\mathcal{W}_j^{\star\top} f_i^{\star}/\tau\right)}$$

$$p(z \mid \alpha_k) = \frac{1}{\mathcal{B}(\alpha_k)} \prod_{i=1}^{K} z_i^{\alpha_{k,i}-1} \quad \text{for } z \in \Delta_K$$

$$\mathcal{B}(\alpha_k) = \frac{\prod_{i=1}^K \Gamma(\alpha_{k,i})}{\Gamma\left(\sum_{i=1}^K \alpha_{k,i}\right)},$$

$$\mathcal{L}(u, \alpha) = \sum_{n=1}^{N} \sum_{k=1}^{K} u_{n,k} \ln\left(p\left(z_n \mid \alpha_k\right)\right),$$

$$\underset{u,\alpha}{\text{minimize}} \; -\mathcal{L}(u,\alpha) + \Phi(u) + \lambda\Psi(u),$$

$$\begin{aligned} &\text{subject to} \\ &u_n \in \Delta_K \quad \forall n \in Q, \\ &u_{n,k} = y_{n,k} \quad \forall n \in S, \forall k \in \{1, \ldots, K\}. \end{aligned}$$

$$\boldsymbol{u}_n^{(\ell+1)} = \mathrm{softmax}\left(\left(\ln \mathrm{p}\left(\boldsymbol{z}_n \mid \boldsymbol{\alpha}_k^{(\ell+1)}\right) + \frac{\lambda}{|\mathbb{Q}|}\ln(\pi_k^{(\ell+1)})\right)_k\right), \quad \forall n \in \mathbb{Q}.$$

Exploration

x : image

$y_i : \{\text{text}\}_i,\ i \in \{1, 2, \ldots, K\},$ which is queries

Our task is to model $P_{\text{real}}(y_i \mid x)$ by $P_{\text{model}}(y_i \mid x)$

CLIP models $P(x, y_i)$ by $\cos(f_{\text{in}}(x), f_{\text{tx}}(y_i))$

$P_{\text{model}}(y_i \mid x) = \frac{P_{\text{model}}(x, y_i)}{P(x)} \propto P(x, y_i)$

Only use label $= \arg\max \text{softmax}\,(\cos(f_{\text{in}}(x), f_{\text{it}}(y_i)))$

Introduce $\alpha_k,$ latent variables to strengthen the expressive ability of the model

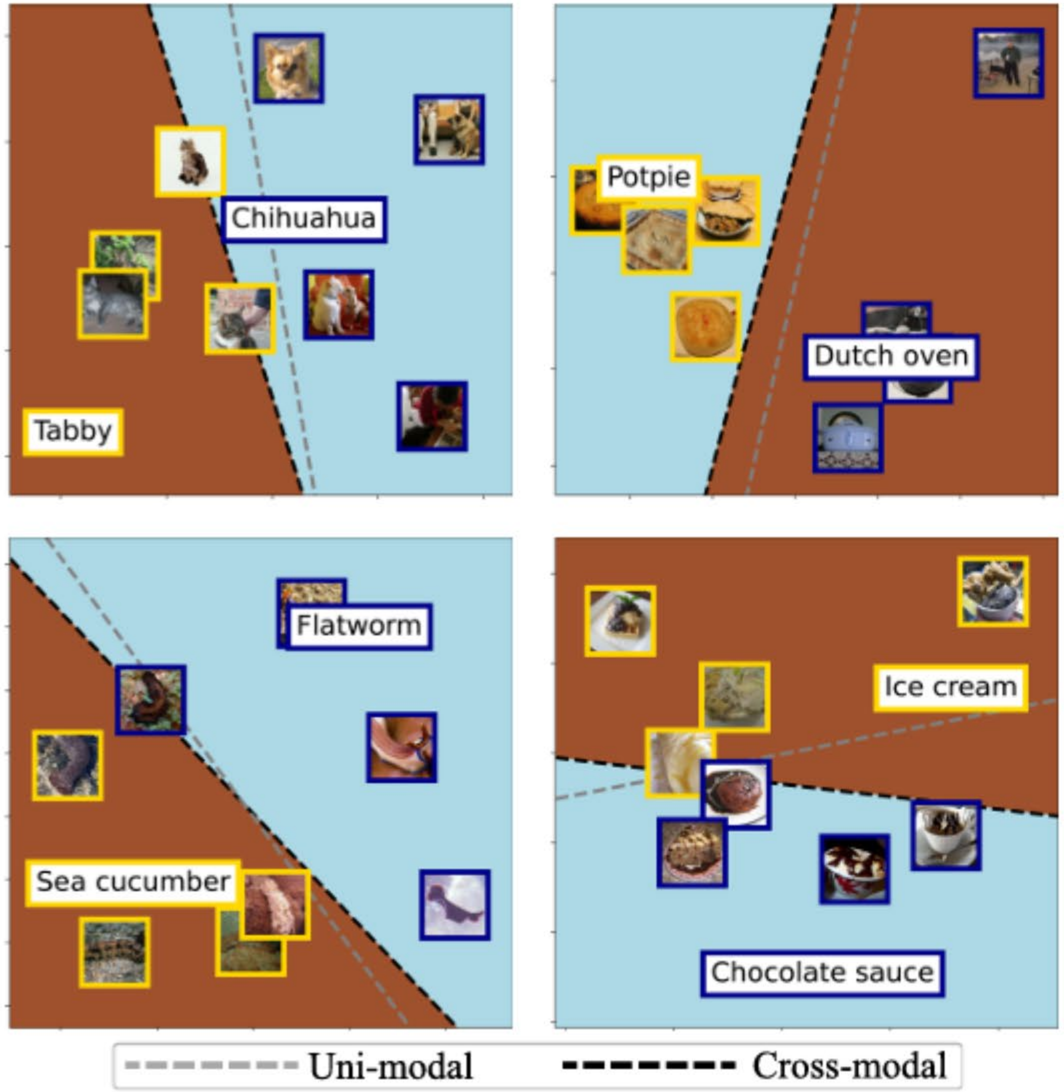
Better modeling: $\sum_i P_{\text{model}}(y_i \mid x) = 1$

$\Rightarrow$ Assume $P_{\text{model}}(y_i \mid x) \sim$ Dirichlet Distribution

$P(P(y_i \mid x) \mid \vec{\alpha_k}) = \text{Dirichlet}_{\vec{\alpha_k}}(P(y_i \mid x))$

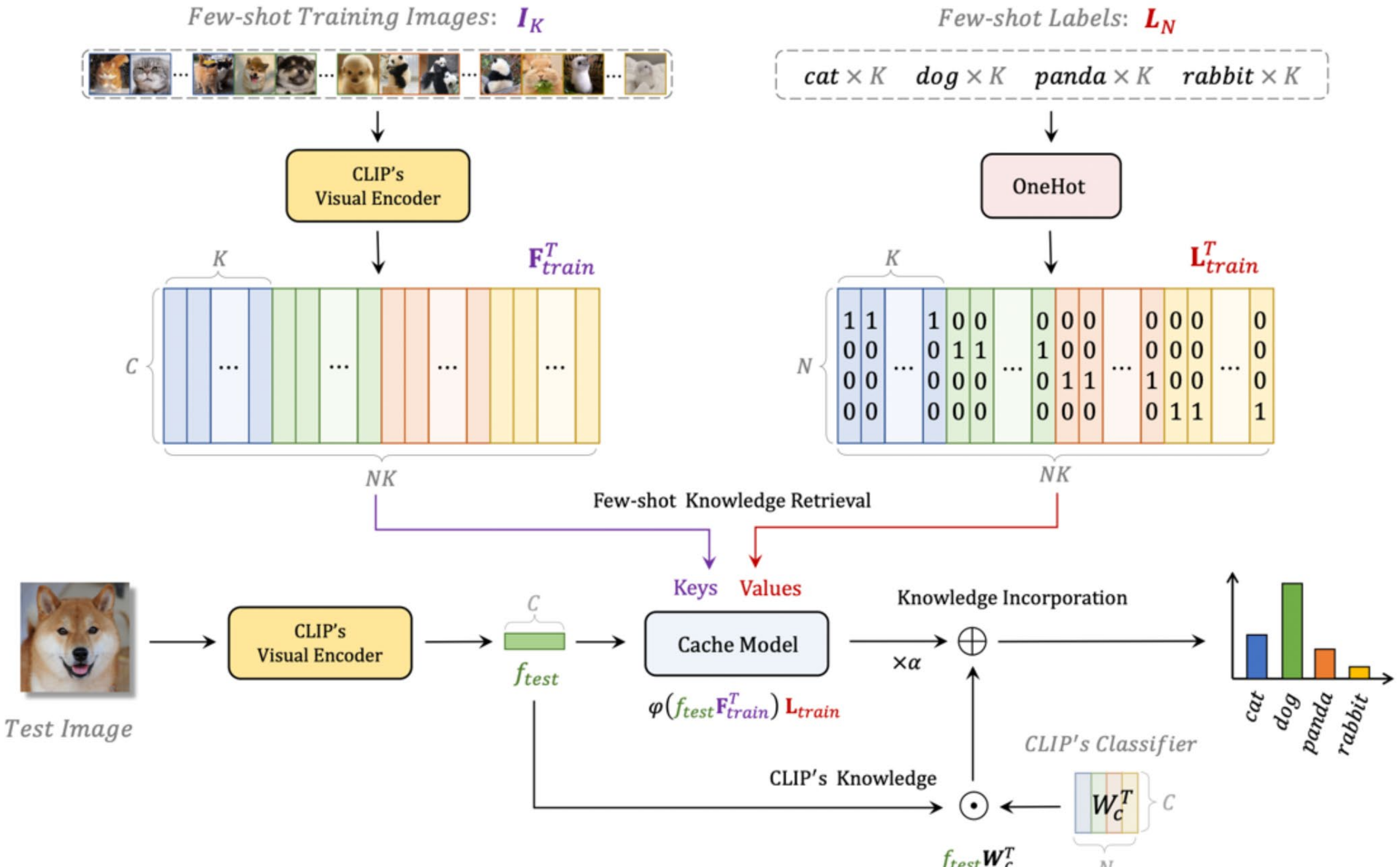
Use ME to new $P(y_i \mid x)$

Multimodality Helps Unimodality: Cross-Modal Few-Shot Learning with Multimodal Models




```
# W: multimodal processing network
for _ in iteration:
    # Randomly sample images and text label
    im,labels = data_loader()
    txquery = prompt_generator(labels)
    # Extract image and text features
    im_ft = image_encoder(im)
    txquery_ft = text_encoder(txquery)
    # L2 normalize both features
    im_f = normalize(im_f)
    tx_f = normalize(tx_f)
    # Compute multimodal loss
    im_loss = softmax_loss(W(im_f) / T, im_labels)
    tx_loss = softmax_loss(W(tx_f) / T, tx_labels)
    loss = (im_loss + tx_loss) / 2
    # backword and Update linear layer
    update(W.params)
```

Tip-Adapter：Training-free CLIP-Adapter for Better Vision-Language Modeling



Other References

- Gondal, M. W., Gast, J., Ruiz, I. A., Droste, R., Macri, T., Kumar, S., & Staudigl, L. (2024). Domain Aligned CLIP for Few-shot Classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Shao, S., Bai, Y., Wang, Y., Liu, B., & Zhou, Y. (2024). DeIL: Direct-and-Inverse CLIP for Open-World Few-Shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., & Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*.
- Tang, Y., Lin, Z., Wang, Q., Zhu, P., & Hu, Q. (2024). AMU-Tuning: Effective Logit Bias for CLIP-based Few-shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

- Martin, S., Huang, Y., Shakeri, F., Pesquet, J. C., & Ayed, I. B. (2024). Transductive Zero-Shot and Few-Shot CLIP. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Gao, P., Geng, S., Zhang, R., et al. (2024). CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *International Journal of Computer Vision* (IJCV), 132(2), 581–595.

Thanks

end of file