Advance Techniques for Object Classification

Lesson of Content

1. SOTA of Image Classification
2. One-shot and Few-shot Learning
3. How to build demo

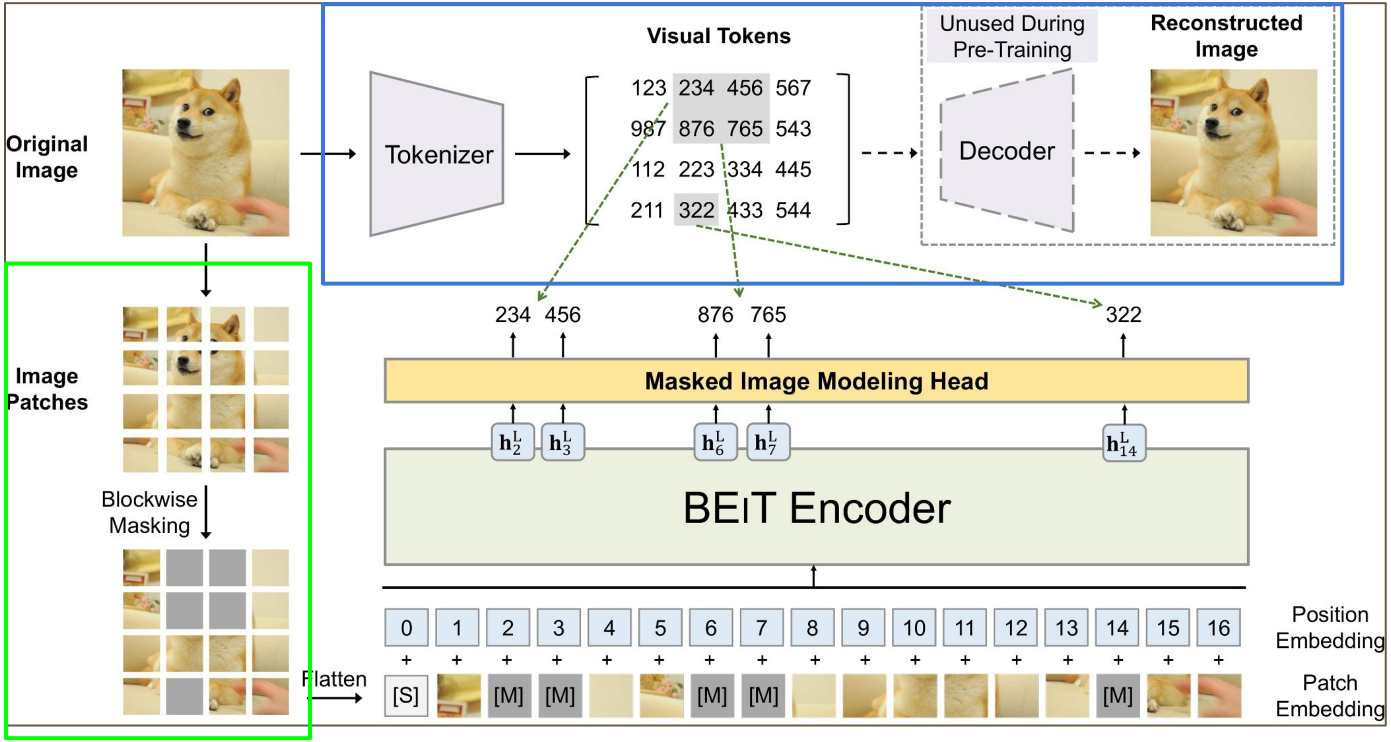
Model: Beit (**BE**RT Pre-Training of **I**mage **T**ransformers)
Paper: https://arxiv.org/pdf/2106.08254.pdf
Huggingface: https://huggingface.co/docs/transformers/main/model_doc/beit

Models	**CIFAR-100**
Training from scratch (i.e., random initialization)	
ViT_{384} [DBK+20]	48.5*
Supervised Pre-Training on ImageNet-1K (using labeled data)	
ViT_{384} [DBK+20]	87.1
DeiT [TCD+20]	90.8
Self-Supervised Pre-Training on ImageNet-1K (without labeled data)	
DINO [CTM+21]	91.7
MoCo v3 [CXH21]	87.1
BEIT (ours)	90.1
Self-Supervised Pre-Training, and Intermediate Fine-Tuning on ImageNet-1K	
BEIT (ours)	**91.8**

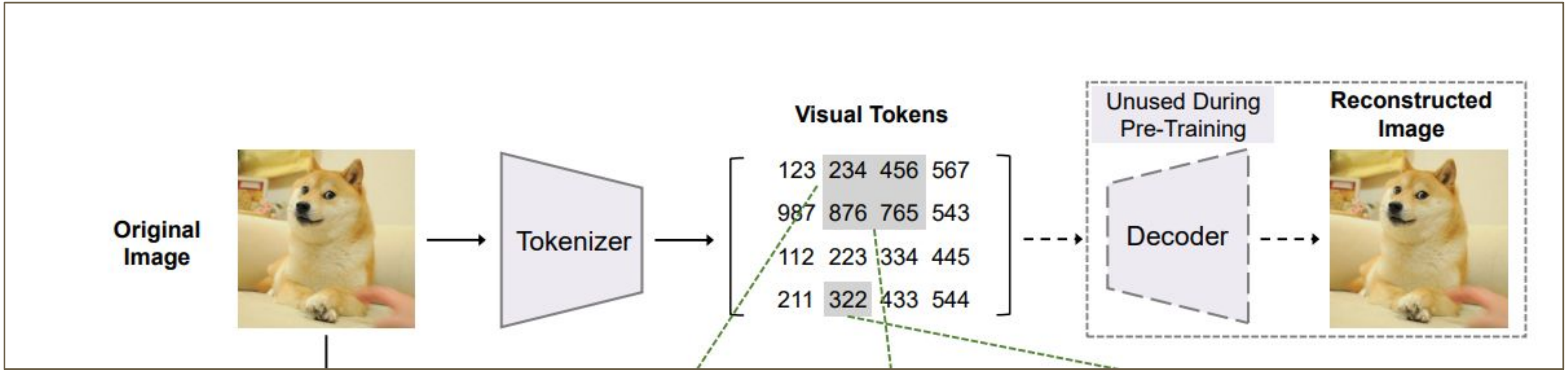
Hyperparameters	**Base Size**	**Large Size**
Layers	12	24
Hidden size	768	1024
FFN inner hidden size	3072	4096
Attention heads	12	16
Attention head size	64	
Patch size	16×16	
Training epochs	800	
Batch size	2048	
Adam ϵ	1e-8	
Adam β	(0.9, 0.999)	
Peak learning rate	1.5e-3	
Minimal learning rate	1e-5	
Learning rate schedule	Cosine	
Warmup epochs	10	
Gradient clipping	3.0	1.0
Dropout	✗	
Stoch. depth	0.1	
Weight decay	0.05	
Data Augment	RandomResizeAndCrop	
Input resolution	224×224	
Color jitter	0.4	

How it work

- Sử dụng task **MIM** (Masked Image modeling) - Một loại **self-supervised**



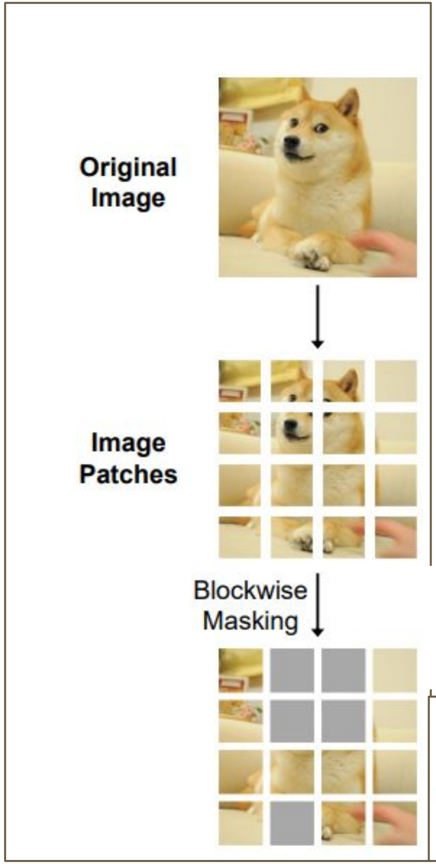
Visualize Token



- Trên thực tế mô hình chia ảnh thành 14x14 grids, mỗi grids có kích thước 16x16 (224/14 = 16 pixel).
- Dùng một bên thứ ba là DALL-E (Chức năng Encode/Tokenizer ảnh đầu vào thành ma trận 14x14, đồng thời mỗi giá trị nằm trong khoảng từ 0 -> 8191. Với **vocab_size** = 8192)
- Phần Decoder không sử dụng. Nó chỉ dùng để huấn luyện riêng mô hình DALL-E .

Image Patches

- Mục đích chia nhỏ ảnh thành 14x14 phần.
- Tiến hành masked ngẫu nhiên 40% số lượng patch để tạo dữ liệu huấn luyện. (Nó không masked ngẫu nhiên mà theo từng blocked).
- Lấy lấy phần tokenizer ở bước trước để tạo input và output cho mô hình self-supervised.



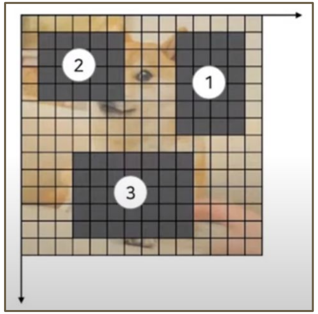
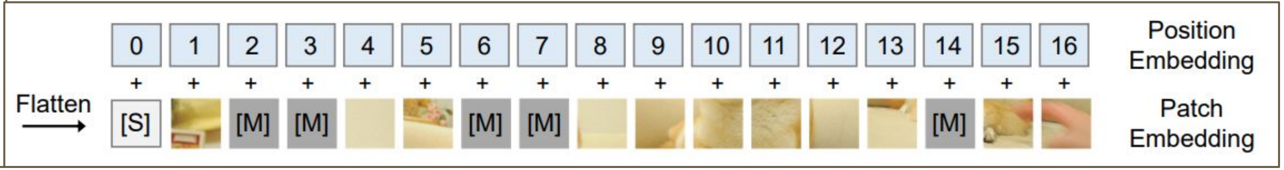


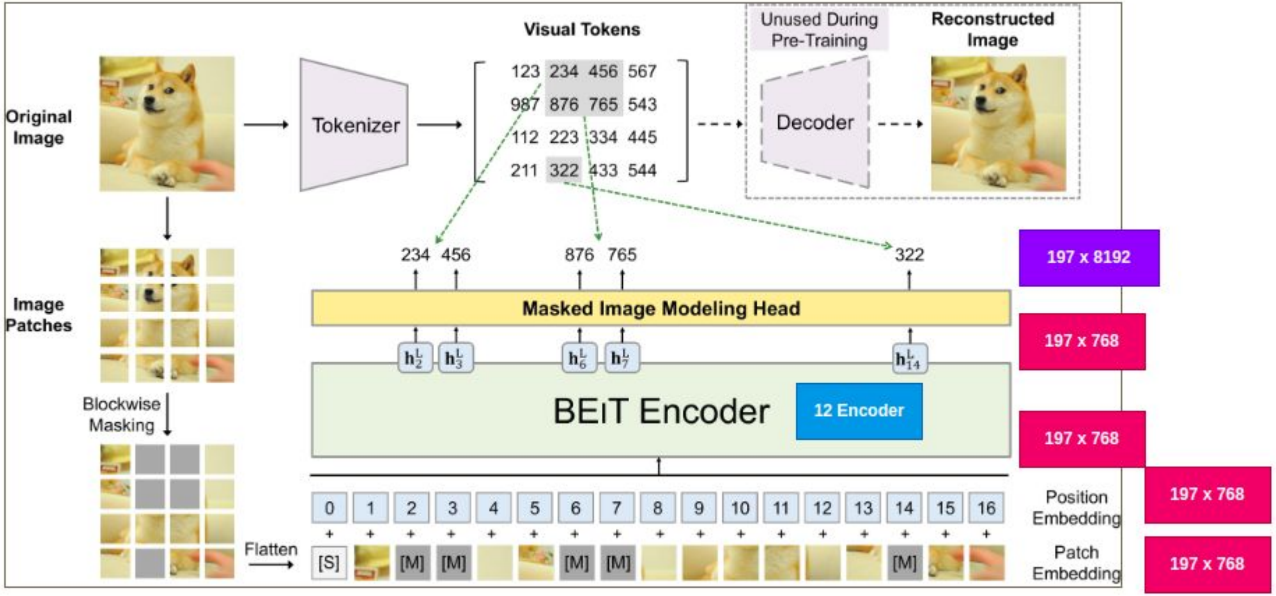
Figure 1 shows the overview of our method. As presented in Section 2.1, given an input image $\boldsymbol{x}$, we split it into N image patches ($\{\boldsymbol{x}_i^p\}_{i=1}^N$), and tokenize it to N visual tokens ($\{z_i\}_{i=1}^N$). We randomly mask approximately 40% image patches, where the masked positions are denoted as $\mathcal{M} \in \{1, \ldots, N\}^{0.4N}$. Next we replace the masked patches with a learnable embedding $e_{[\mathrm{M}]} \in \mathbb{R}^D$.

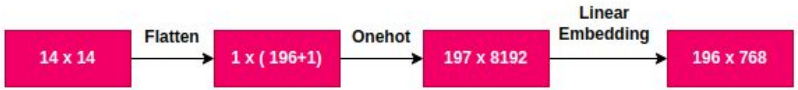


How to training

$$\max \sum_{x \in \mathcal{D}} \mathbb{E}_{\mathcal{M}} \left[\sum_{i \in \mathcal{M}} \log p_{\text{MIM}}(z_i | x^{\mathcal{M}}) \right]$$

Maximum Log-Likelihood

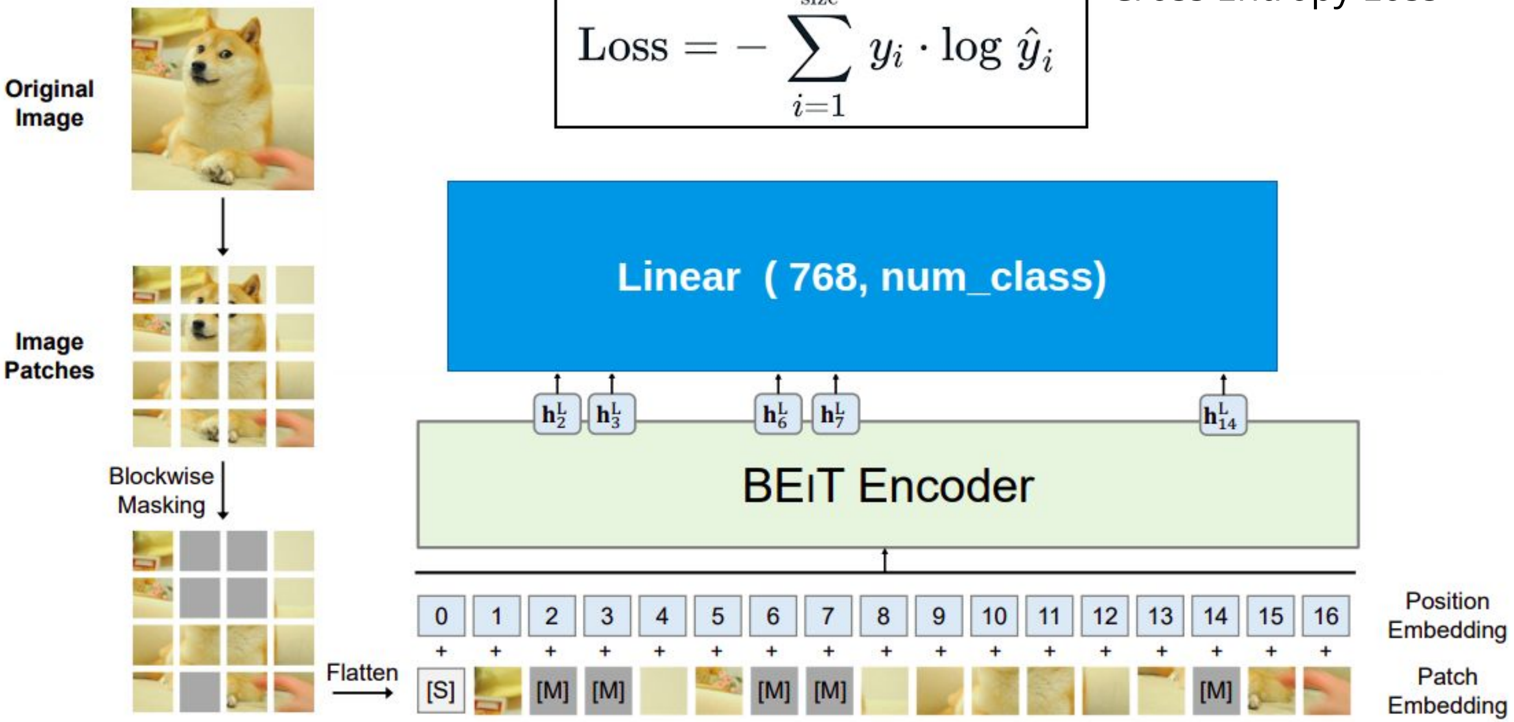




Train Classification with Beit

$$\text{Loss} = -\sum_{i=1}^{\substack{\text{output}\\ \text{size}}} y_i \cdot \log \hat{y}_i$$

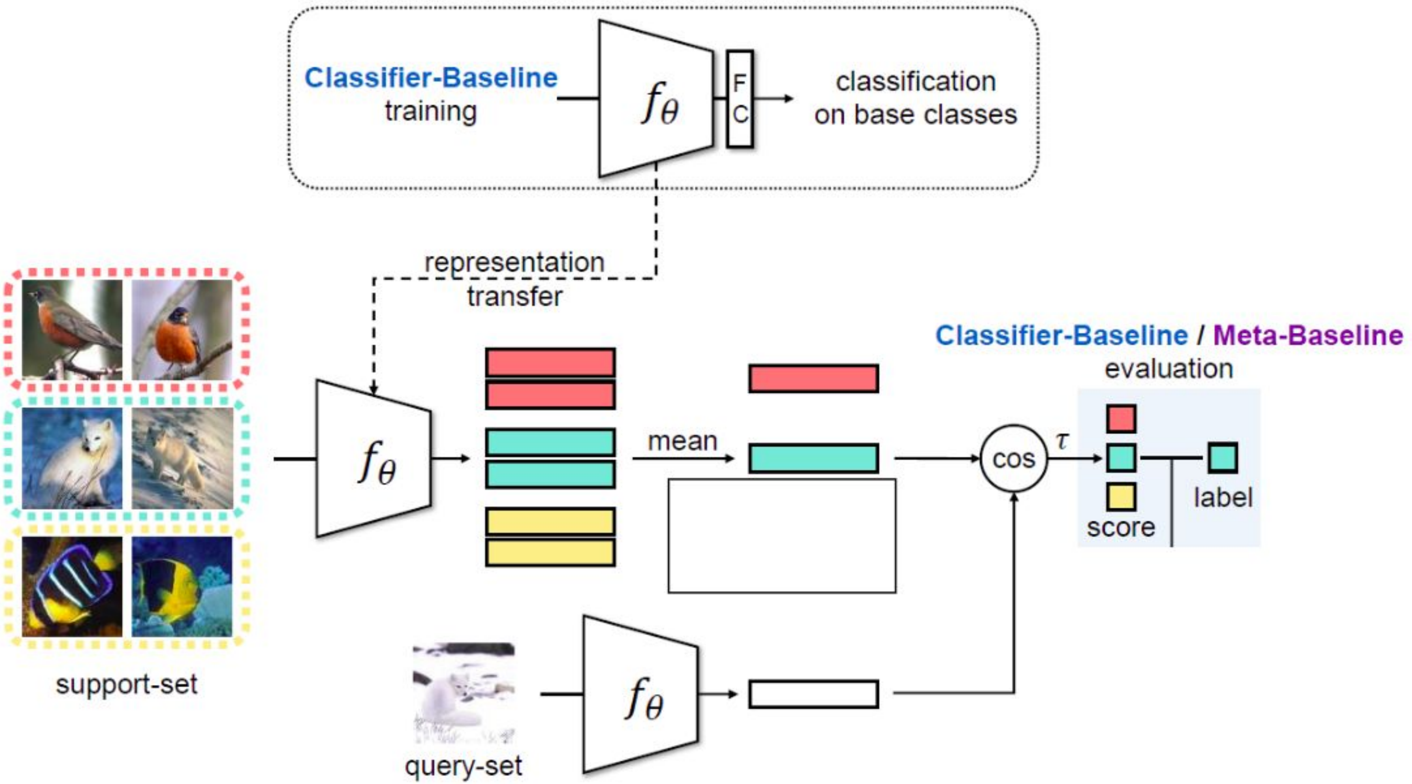
Cross Entropy Loss



What Problem with Image Classification

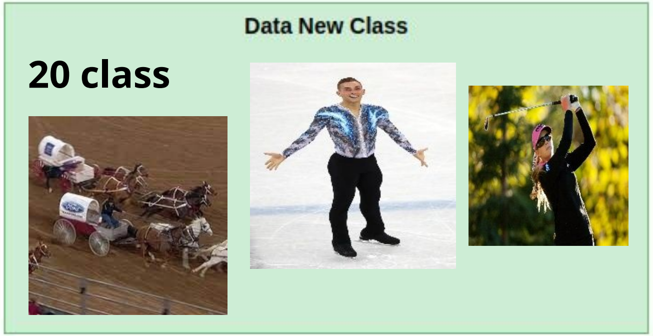
- Khi tập dữ liệu của bạn hiếm hoi cho một hoặc một số class nào đó.
- Khi bạn muốn thêm lớp mới vào mô hình mà không muốn huấn luyện lại mô hình.

One-Shot and Few-shot



Dataset

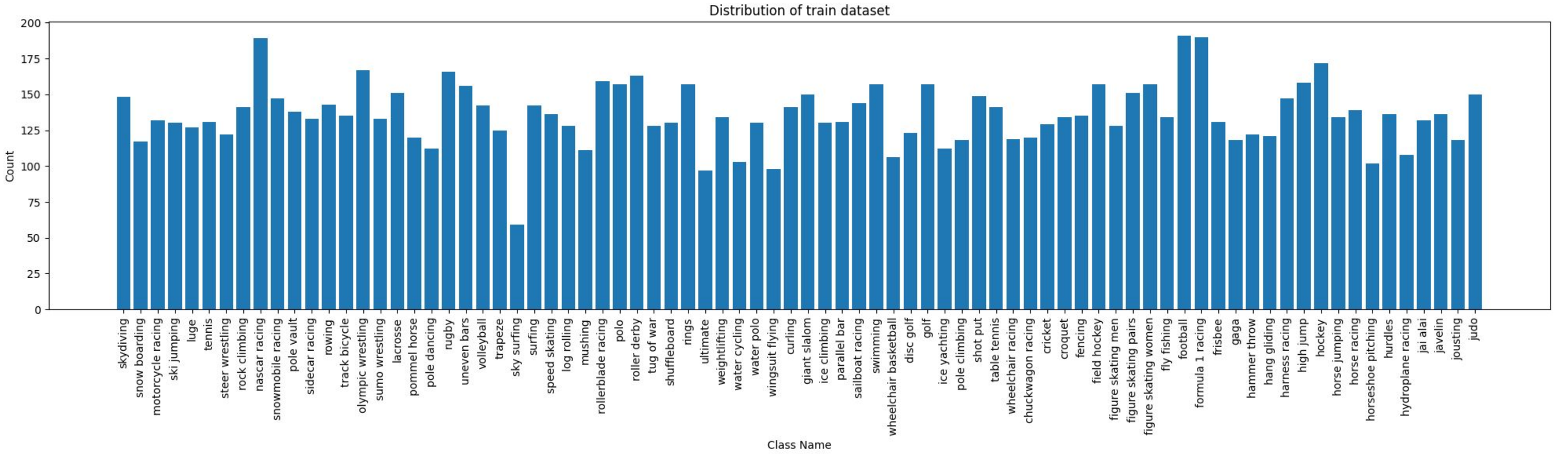

Data old Class
80 class

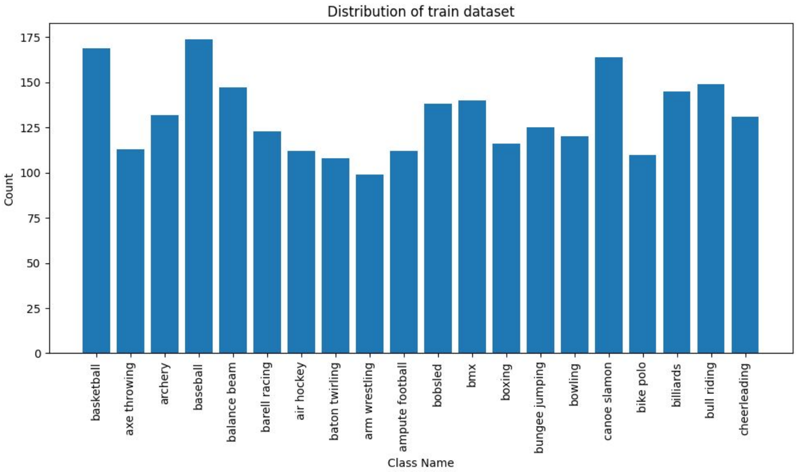

Data New Class
20 class


few shot reference
100 class
MN AXE

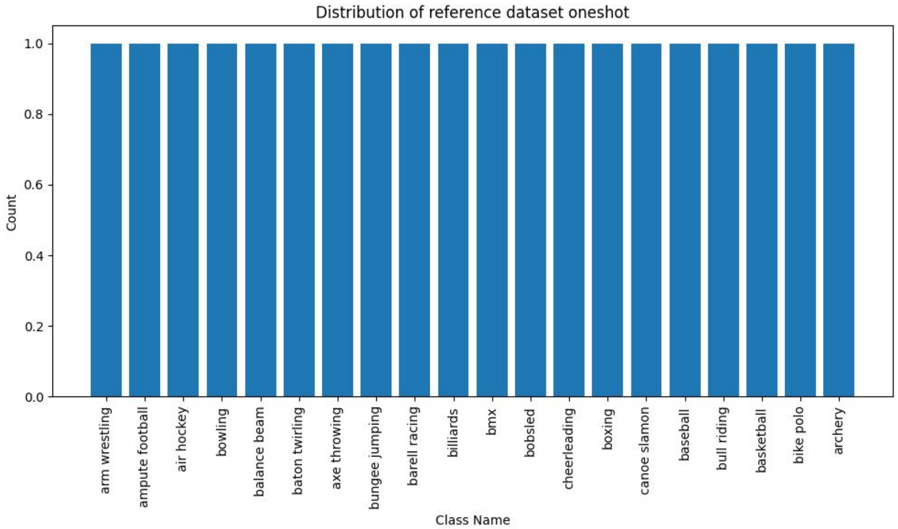

one shot reference
100 class

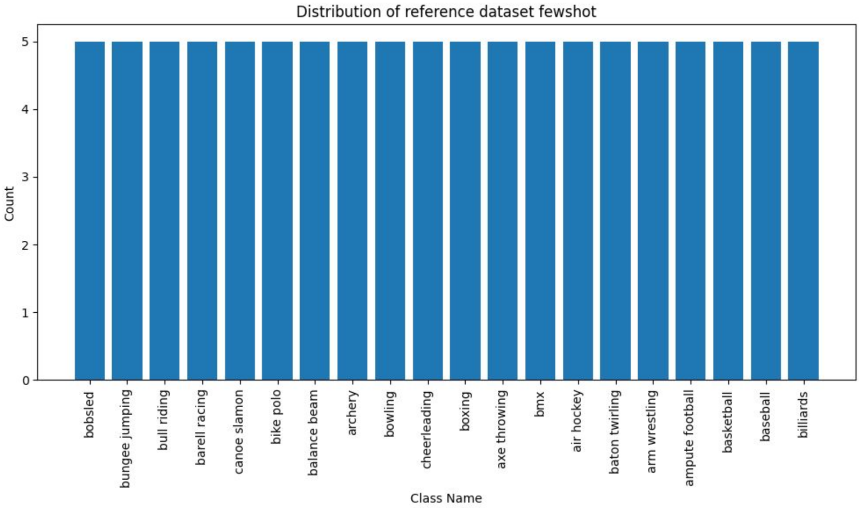
3. How to build a Demo





3. How to build a Demo





Metrics

Cosine Similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

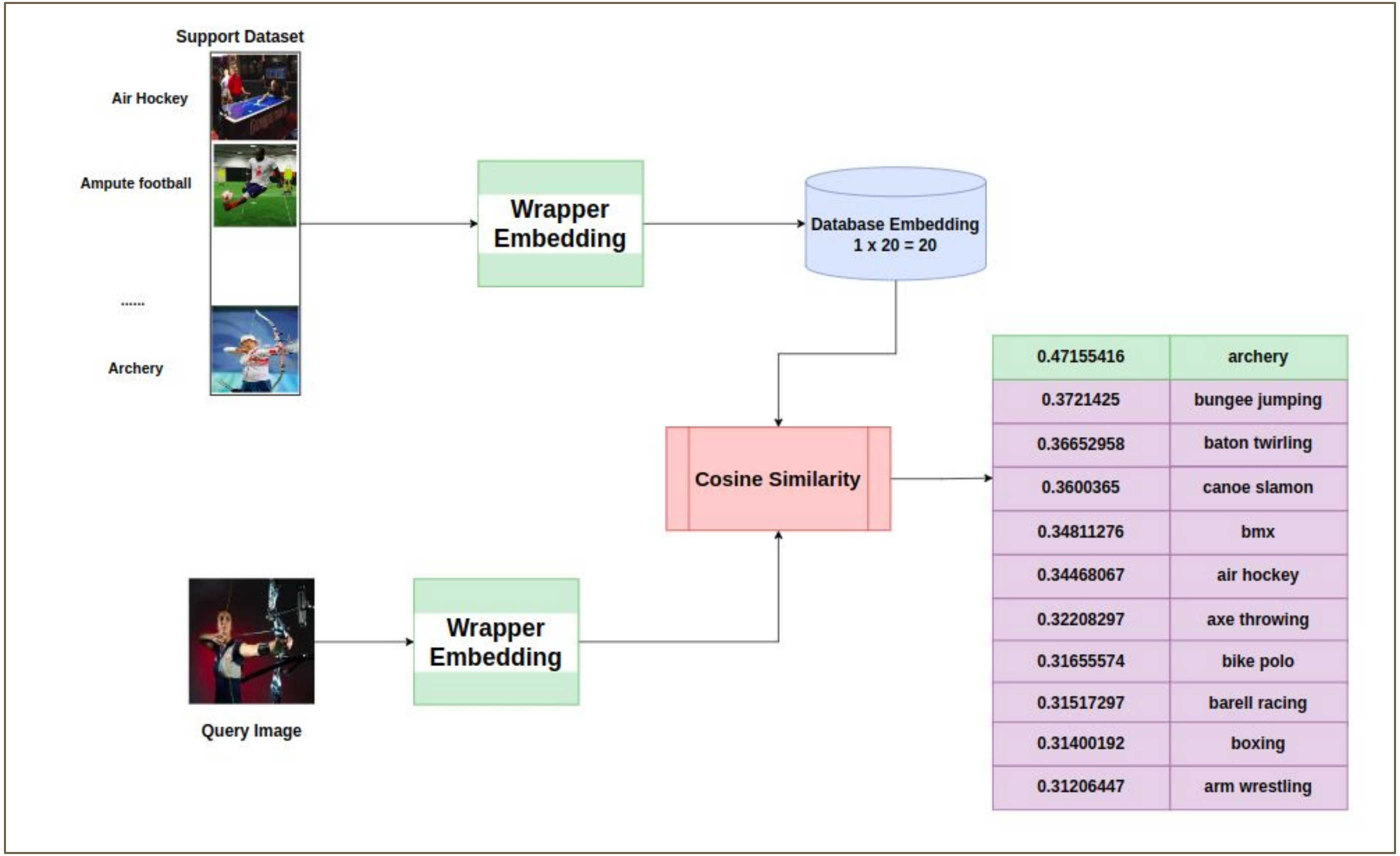
```python
import numpy as np

def cosine_similarity(vector1, vector2):
    # Chuẩn hóa vector trước khi tính cosine similarity
    vector1_normalized = vector1 / np.linalg.norm(vector1)
    vector2_normalized = vector2 / np.linalg.norm(vector2)

    # Tính cosine similarity
    similarity = np.dot(vector1_normalized, vector2_normalized)
    return similarity
```

MSE

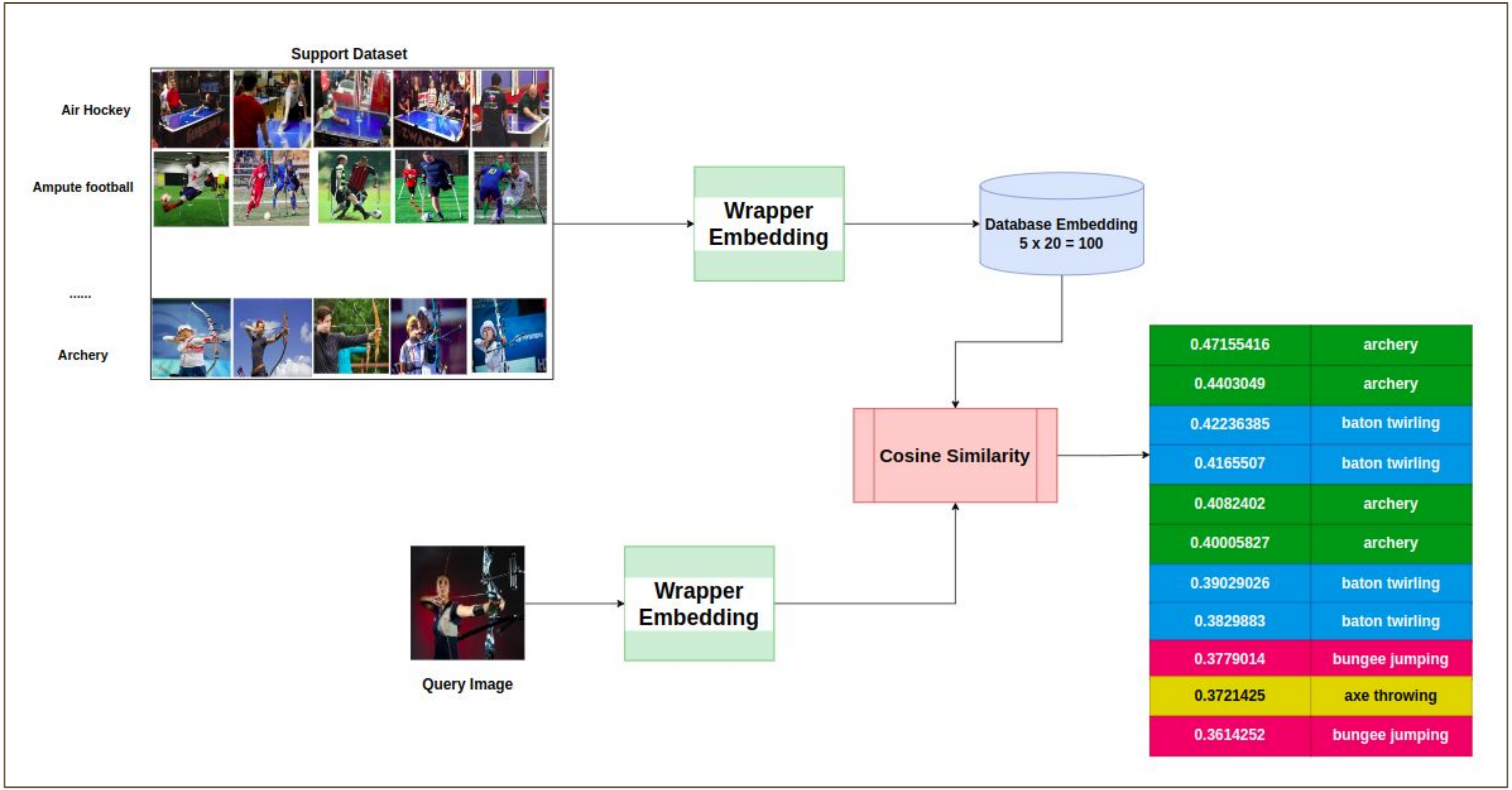
$$\text{MSE} = \underbrace{\boxed{\frac{1}{n}\sum_{i=1}^{n}}}_{\text{Mean}} \underbrace{\boxed{(Y_i - \hat{Y}_i)}}_{\text{Error}}{}^{\overbrace{\boxed{2}}^{\text{Squared}}}$$

```python
def mean_squared_error(vector1, vector2):
    mse = np.mean((vector1 - vector2) ** 2)
    return mse
```

One-Shot Diagram



Few-Shot Diagram



Some Results

Classification Report:

	precision	recall	f1-score	support
air hockey	0.74	0.98	0.85	112
ampute football	0.97	0.90	0.94	112
archery	0.94	0.91	0.93	132
arm wrestling	0.91	0.52	0.66	99
axe throwing	0.97	0.50	0.65	113
balance beam	0.99	0.88	0.93	147
barell racing	0.70	0.97	0.81	123
baseball	0.86	0.77	0.81	174
basketball	1.00	0.78	0.88	169
baton twirling	0.61	0.86	0.71	108
bike polo	1.00	0.23	0.37	110
billiards	1.00	0.97	0.99	145
bmx	0.58	1.00	0.73	140
bobsled	1.00	0.91	0.95	138
bowling	0.90	0.87	0.89	120
boxing	0.39	0.98	0.56	116
bull riding	0.94	0.65	0.77	149
bungee jumping	0.80	0.98	0.88	125
canoe slamon	0.97	1.00	0.98	164
cheerleading	1.00	0.21	0.34	131
accuracy			0.80	2627
macro avg	0.86	0.79	0.78	2627
weighted avg	0.87	0.80	0.79	2627

One-Shot Learning

Classification Report:

	precision	recall	f1-score	support
air hockey	0.90	0.98	0.94	112
ampute football	0.99	0.95	0.97	112
archery	0.97	0.87	0.92	132
arm wrestling	0.77	0.94	0.85	99
axe throwing	0.94	0.88	0.91	113
balance beam	1.00	0.88	0.93	147
barell racing	0.96	0.85	0.90	123
baseball	0.88	0.95	0.92	174
basketball	0.97	0.91	0.94	169
baton twirling	0.80	0.98	0.88	108
bike polo	0.93	0.89	0.91	110
billiards	1.00	0.98	0.99	145
bmx	0.87	0.94	0.90	140
bobsled	1.00	0.93	0.97	138
bowling	0.89	0.98	0.94	120
boxing	0.73	0.98	0.84	116
bull riding	0.88	0.97	0.93	149
bungee jumping	0.89	0.99	0.94	125
canoe slamon	0.99	1.00	1.00	164
cheerleading	1.00	0.41	0.58	131
accuracy			0.91	2627
macro avg	0.92	0.91	0.91	2627
weighted avg	0.92	0.91	0.91	2627

Few-Shot Learning