Deep Learning, 2021

Evaluation of CLIP image feature extractors

Denis Kuznedelev,
Denis Rollov,
Ilya Dubovitskii,
Mark Zakharov,
Nikolay Goncharov

Skoltech, May 28 2021

Introduction: problem statement

Large scale
human-labeled datasets

Expensive

Restricted to
training
classes

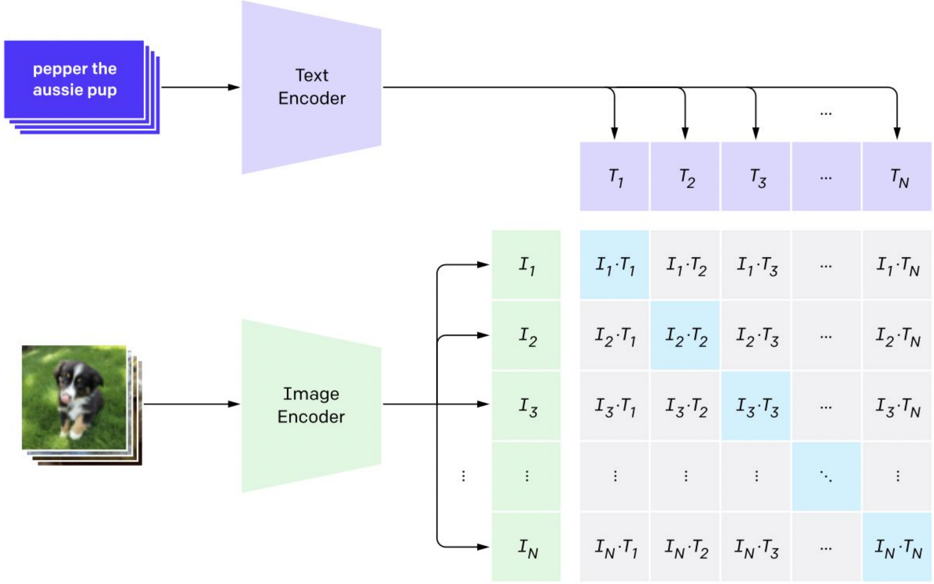
Pic. 1. Example: ImageNet dataset

Introduction: CLIP approach

Dataset: (image, caption)
Image embedding: image_encoder(image)
Caption embedding: text_encoder(caption)

Goal:

1. Calculate distance between each image and caption embedding
2. Maximize the distance for the related (image, caption) pairs
3. Minimize the distance for unrelated pairs



Pic. 2. CLIP training procedure

Introduction: CLIP approach

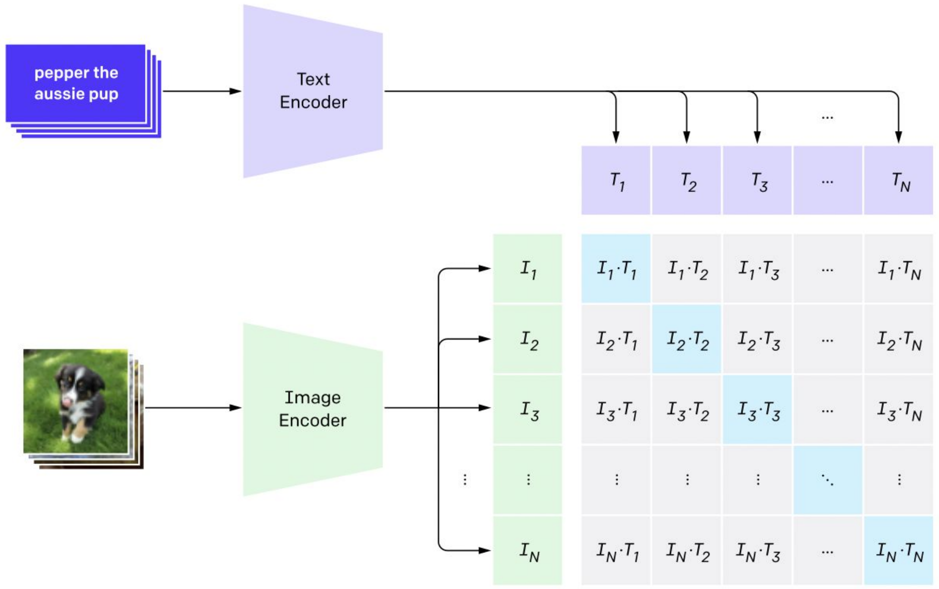
1.
$$I_f = V_{\theta_1}(I)$$
$$T_f = T_{\theta_2}(T)$$

2.
$$I_f \to \frac{I_f}{||I_F||}$$
$$T_f \to \frac{T_f}{||T_F||}$$

3.
$$\text{logits} = e^T I_f T_f^T$$

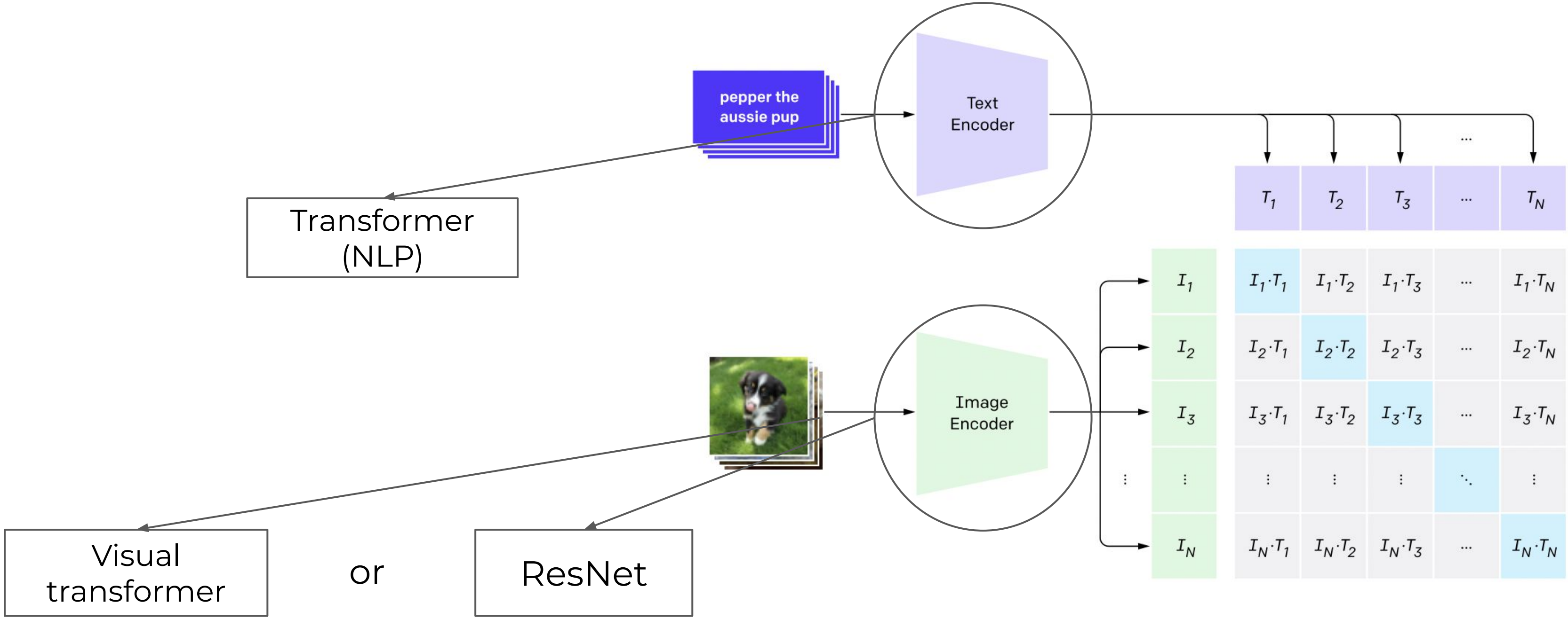
.
$$\text{labels} = (0, 1 \ldots N-1)$$

4.
$$\text{loss} = \frac{1}{2}(\text{CE}(\text{labels}, \text{logits}) + \text{CE}(\text{labels}, \text{logits})^T)$$



Pic. 2. CLIP training procedure

Introduction: CLIP architecture



Pic. 2. CLIP training procedure

Model construction

Classic fine-tuning approach ("LITE")

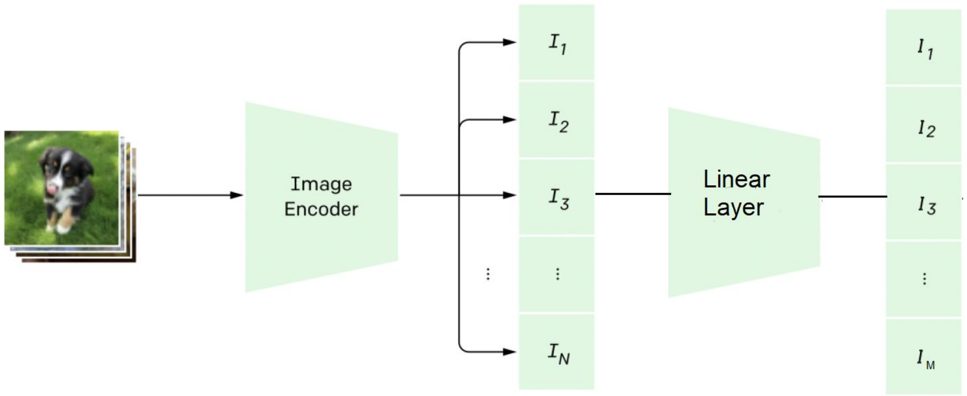
1. Place a linear layer on top of CLIP's image encoder to extract logits
2. Train some part of the resulting network and freeze the rest

Pros

- Cheaper computation
- Less consumed memory

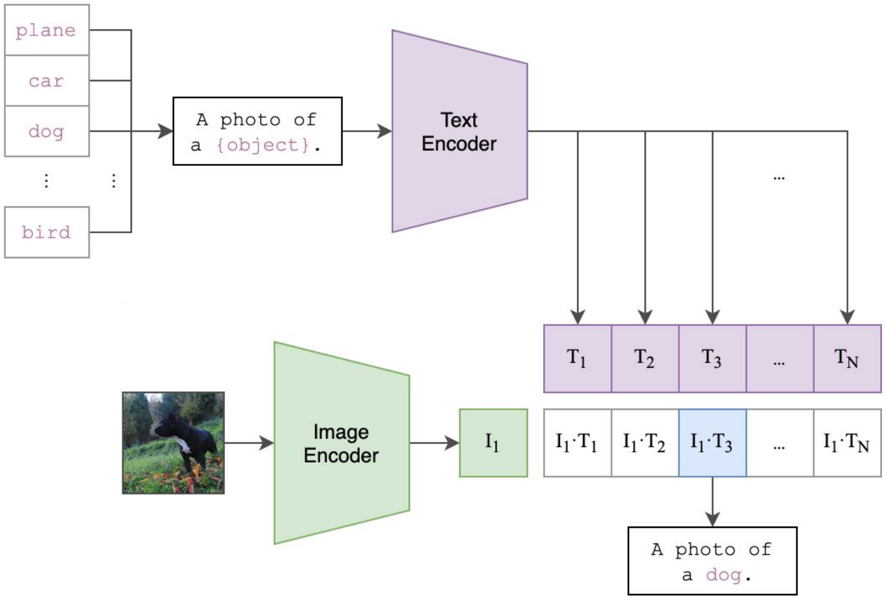
Cons

- Less flexible
- Lacks novelty



Pic. 3. "LITE" training procedure

Model construction



Pic. 4. “PRO” training procedure

Cosine similarity approach (“PRO”)

1. Calculate image embedding
2. Calculate the weighted sum of the caption encodings
3. Normalize the embeddings
4. Calculate the cosine similarity between the input image and the text embeddings for each class

Pros

- Allows for zero-shot classification
- Allows for text labels to be arbitrary

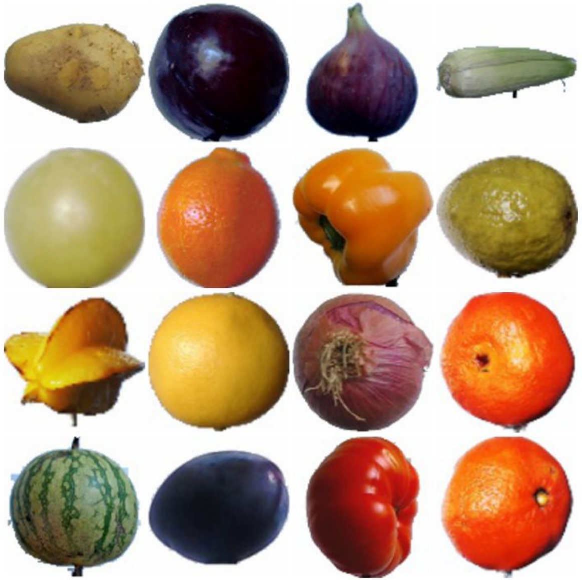
Cons

- Longer computation
- Experimental

The dataset

Fruits 360:

- 131 fruit types
- ~67k train images
- ~22k test images
- Resolution: 100 x 100
- Contains ImageNet-unseen classes
- Involves several subspecies of the same fruit



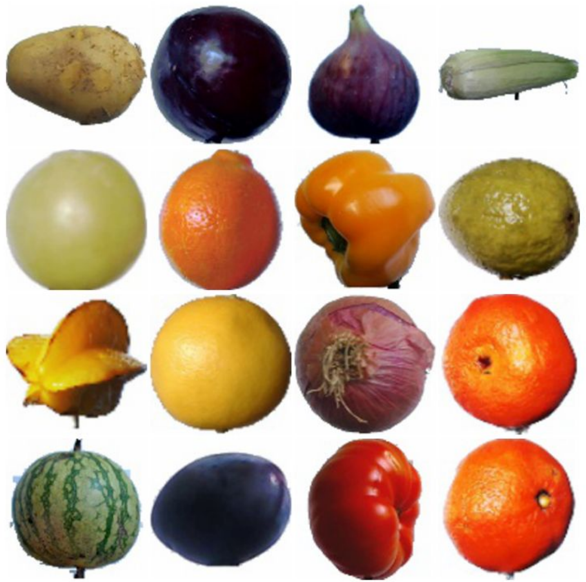
Pic. 5. Several samples from the Fruits 360 dataset

The dataset

Fruits 360:

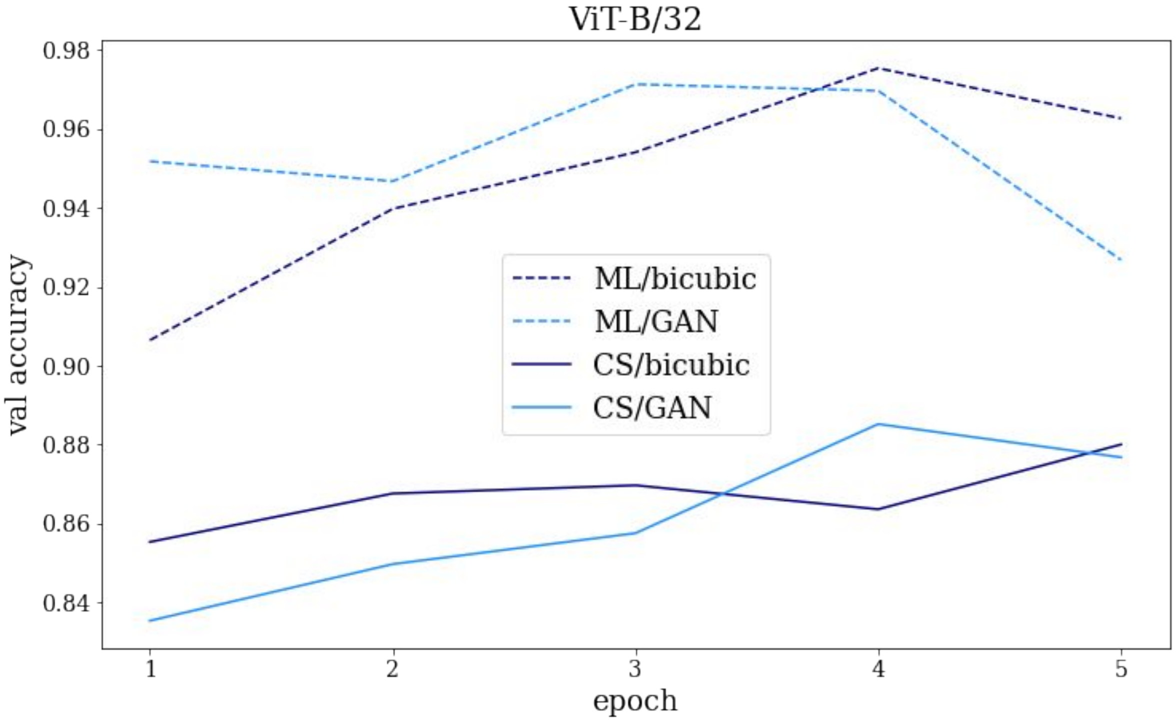
- 131 fruit types
- ~67k train images
- ~22k test images
- Resolution: 100 x 100 (need upsampling!)
- Contains ImageNet-unseen classes
- Involves several subspecies of the same fruit

Use GANs or bicubic upsampling to transform:
(100 x 100) -> (224 x 224)



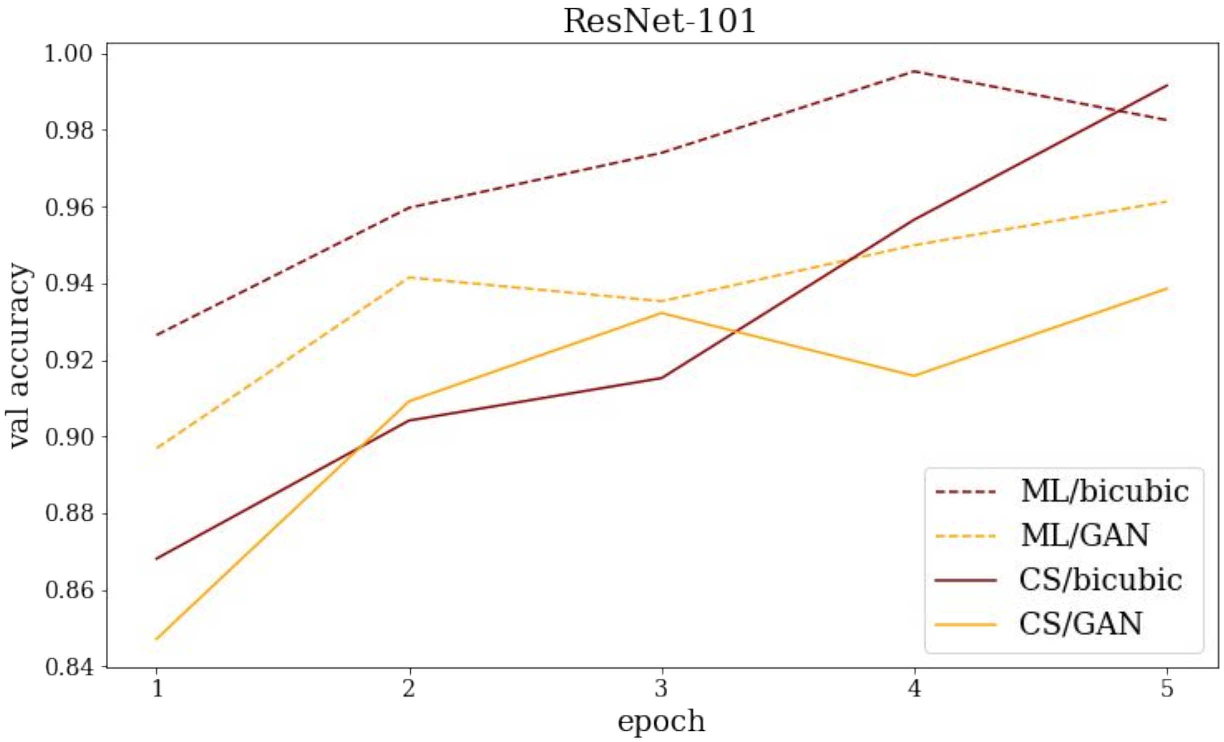
Pic. 5. Several samples from the Fruits 360 dataset

SRGAN vs bicubic upsampling: ViT-B/32



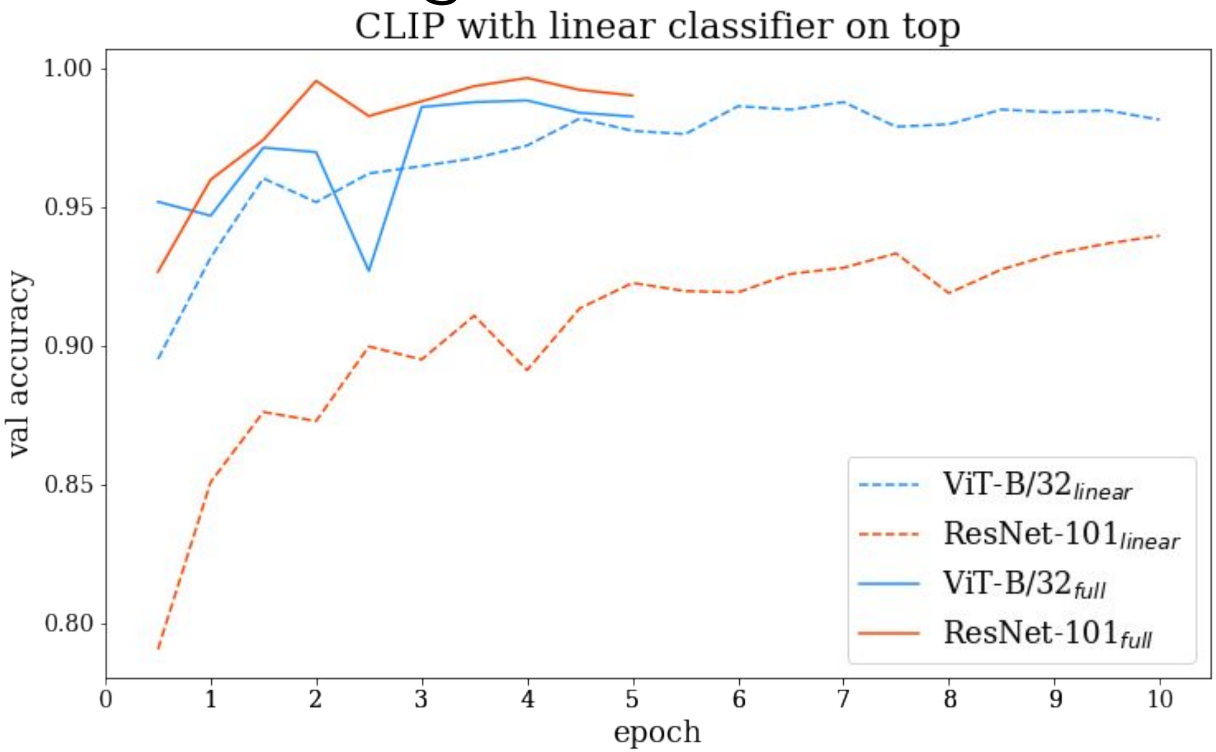
Pic. 6. Validation accuracy for ViT-B/32 backbone.
CS - cosine similarity, ML - maximum likelihood

SRGAN vs bicubic upsampling: ResNet-101



Pic. 7. Validation accuracy for ResNet101 backbone.
CS - cosine similarity, ML - maximum likelihood

Different fine-tuning modes



Pic. 8. Validation accuracy for maximal likelihood
with different backbones

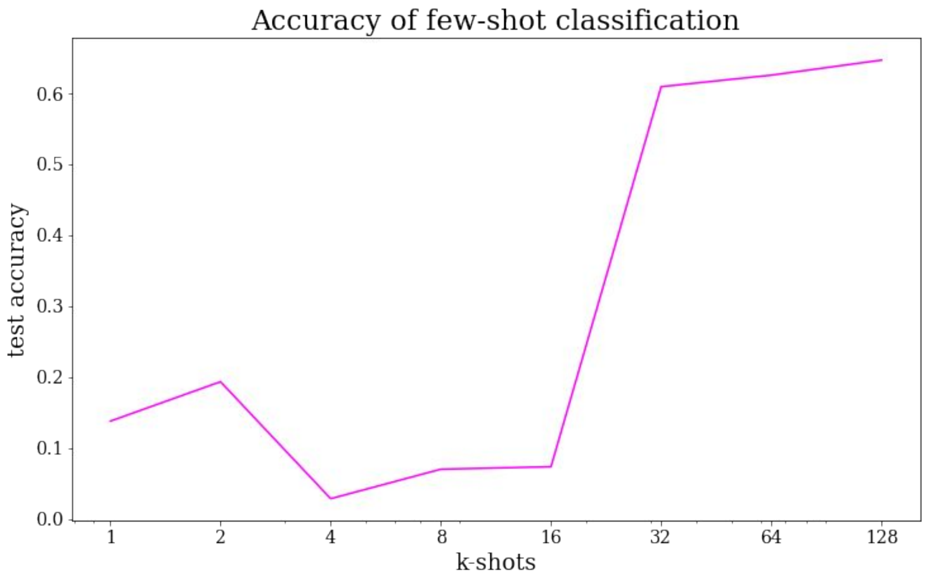
Zero-shot classification

Model	Upsampling Type	Accuracy
ResNet-101	Bicubic	0.2
ResNet-101	GAN	0.181
ViT	Bicubic	0.238
ViT	GAN	0.214

Accuracy of random or constant classifier ~ 0.01

Table 1. Comparison of zero-shot accuracy of
models trained on different types of upsampling

K-shot classification



Accuracy deteriorates for the small K in K-shot classification and then starts to improve.

Pic. 9. Accuracy of k-shot classification on the pretrained ResNet-101 backbone

Common mistakes

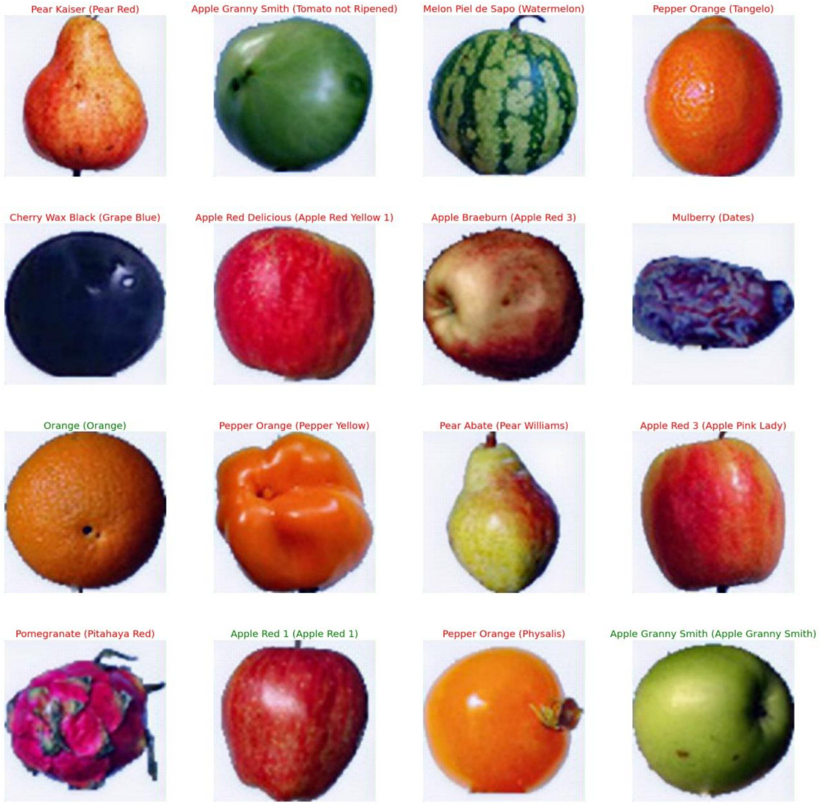
IoU metric:

$$\mathrm{IoU}(c_1, c_2) = \frac{m_{c_1 c_2} + m_{c_2 c_1}}{n_{c_1} + n_{c_2}}$$

True class	Pred class	Class IoU
Grape Blue	Cherry Wax Black	0.67
Tomato 1	Tomato Cherry Red	0.48
Pepper Yellow	Pepper Orange	0.42
Pear Forelle	Pear Abate	0.46
Grapefruit White	Lemon Meyer	0.50

Table 2. Top-5 most frequently confused classes and corresponding IoU.

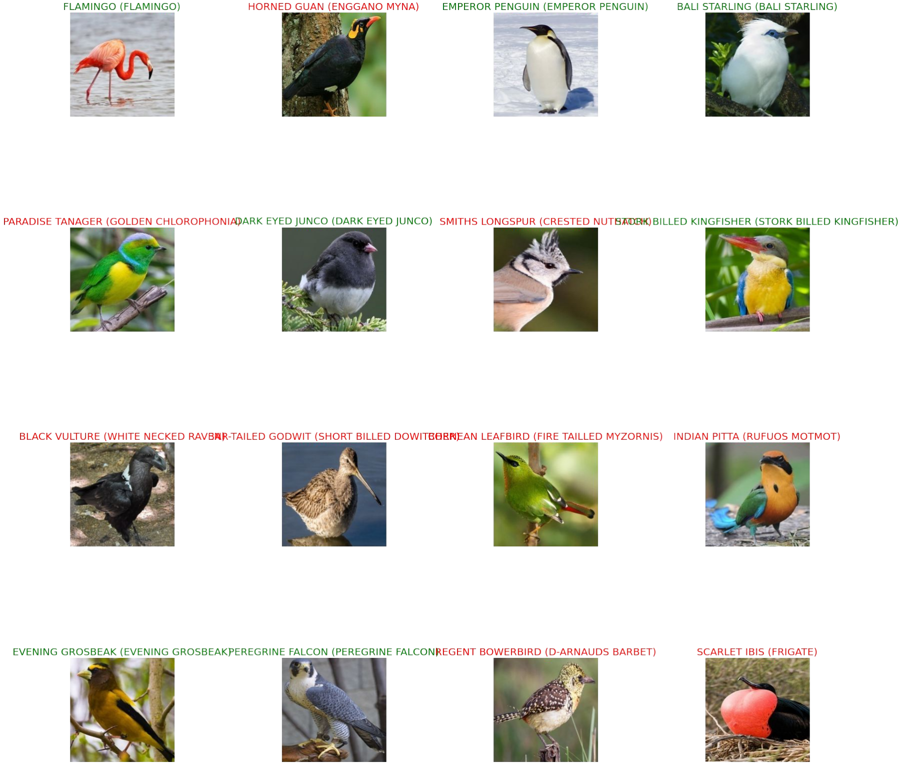
Zero-shot predictions on **Fruits 360**



Zero-shot predictions on **Simpsons characters**

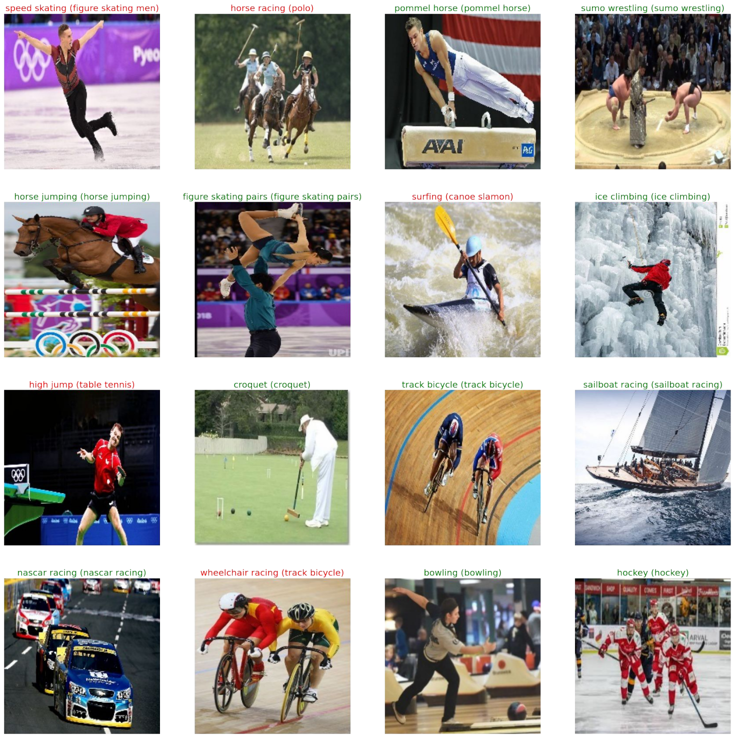
Accuracy ~0.5

Zero-shot predictions on **Birds - 270**



Accuracy ~0.5

Zero-shot predictions on **Sports-72**



accuracy ~0.79

That’s amazing! Isn’t it?