Contrastive Vision Transformers

Applied ML
Nitish Shukla

Goal

Develop an end-to-end image classification pipeline that learns representation of images in unsupervised fashion.

The feature extractor is a vision transformer which is trained using unlabelled data leveraging the SimCLR framework.

Result : 2% + jump in accuracy compared to training only with the labelled test data.

SimCLR

A simple idea: maximizing the agreement of representations under data transformation, using a contrastive loss in the latent/feature space.

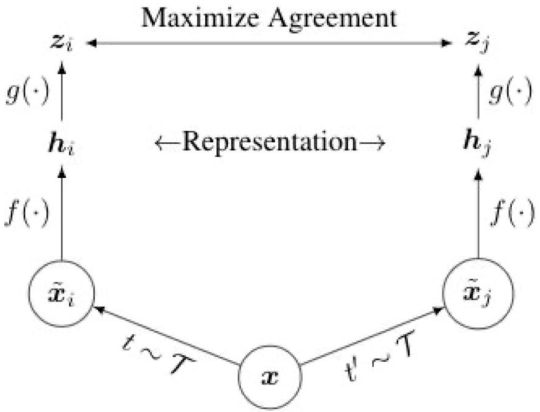
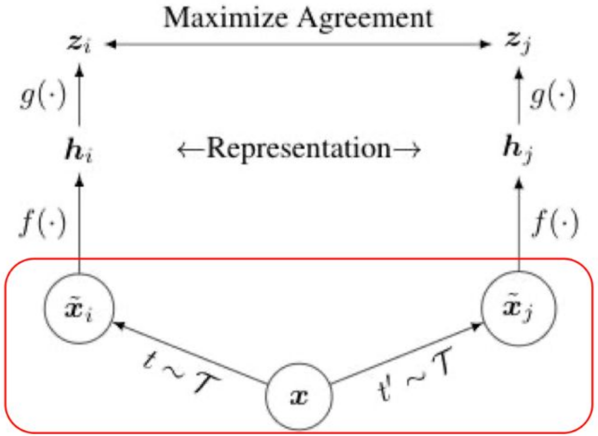
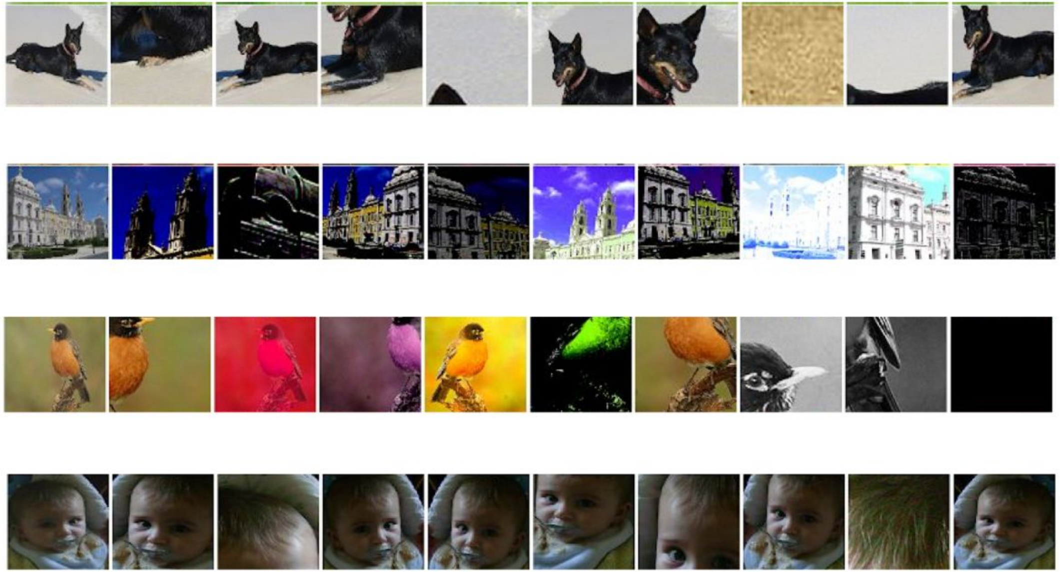


Figure 2. A framework for contrastive representation learning. Two separate stochastic data augmentations $t, t' \sim \mathcal{T}$ are applied to each example to obtain two correlated views. A base encoder network $f(\cdot)$ with a projection head $g(\cdot)$ is trained to maximize agreement in *latent representations* via a contrastive loss.

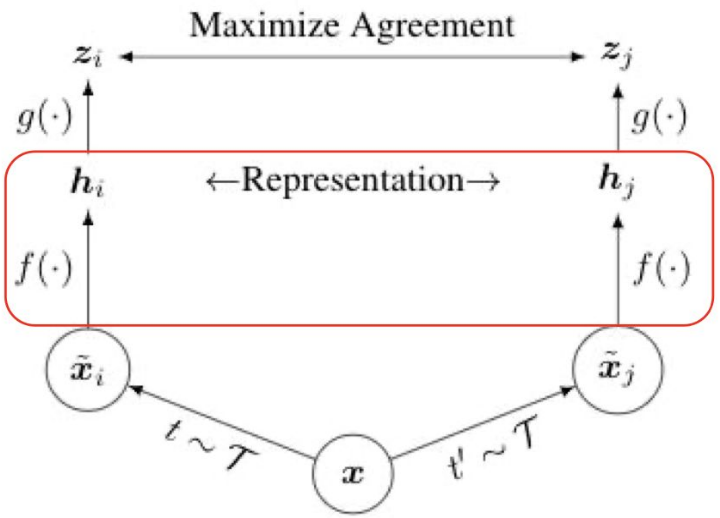


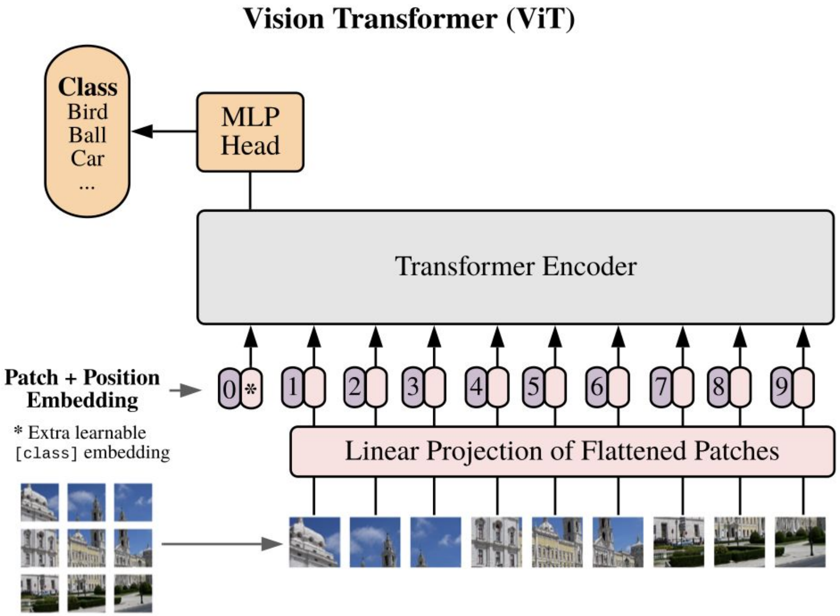
We use random crop and color distortion for augmentation.

Examples of augmentation applied to the left most images:

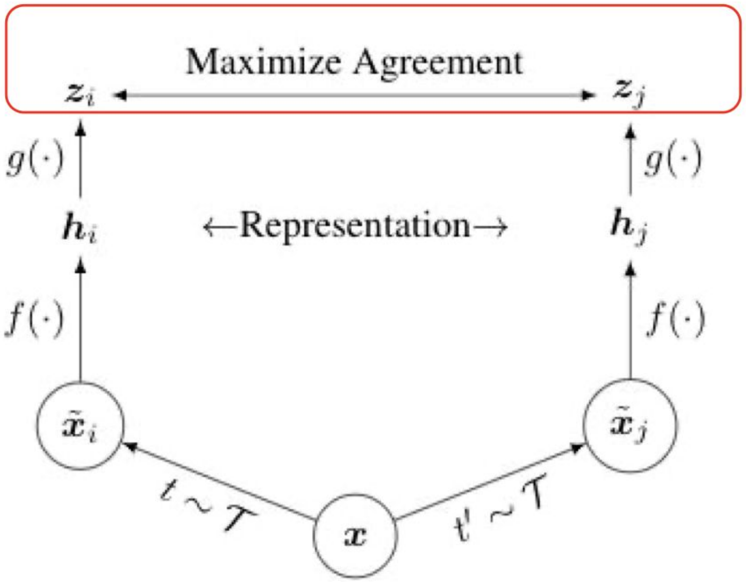


We use a pre-trained vision transformer as the embedding extractor function f.





For g(.), we use any FC network and optimize wrt the contrastive loss which minimizes the distance between images of same class and maximizes the distance among different class.



Original image | crop 1 | crop 2 | contrastive image

Loss function:

$$\text{Let } \mathrm{sim}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v} / \|\boldsymbol{u}\| \|\boldsymbol{v}\|$$

$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

Implementation details

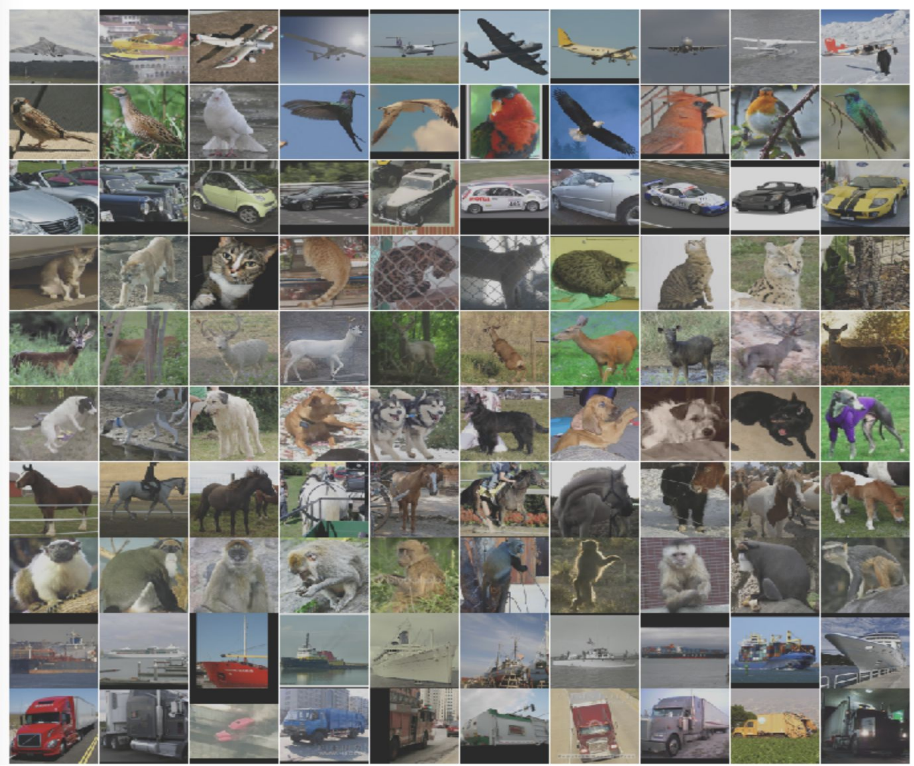
Dataset used: STL10

#unlabelled images :50K

#labelled training images :5K

#labelled test images : 8K

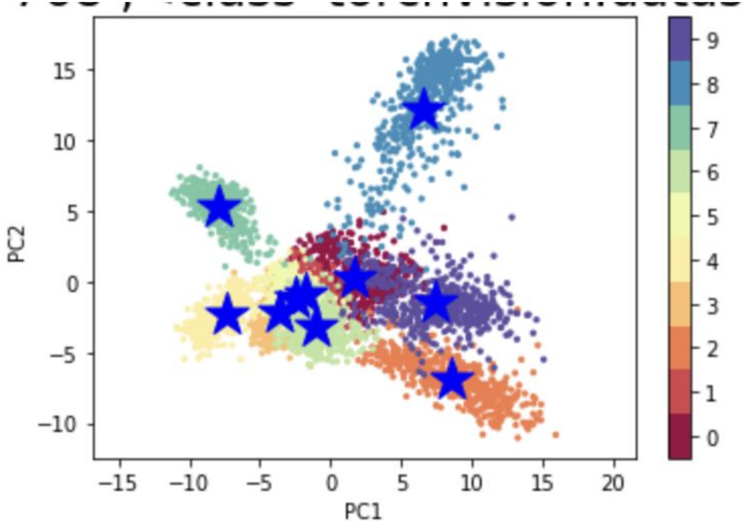
#classes :10



- Unlabelled images are used to train the ViT using contrastive loss.
- After training, we discard the projection head g(.) and use f(.) as feature extractor.
- We extract the embeddings of 5K training images and train Logistic regressor on them.

Results

Embeddings extracted from f(.) are very well separated by virtue of contrastive loss.



Results

Accuracy results:
Baseline model: Vanilla ViT trained with 5k images and tested on 8k images

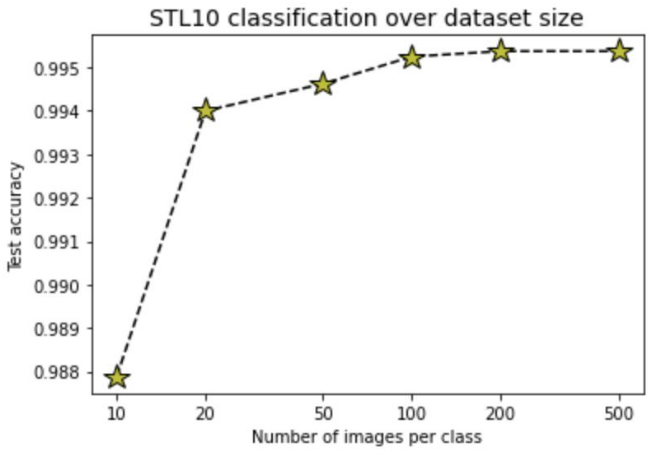
```
trainer.test(vit_model,test_loader)
```

```
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Error displaying widget: model not found
--------------------------------------------------------------------------------
DATALOADER:0 TEST RESULTS
{'test_accuracy': 0.981374979019165}
--------------------------------------------------------------------------------
```

```
[{'test_accuracy': 0.981374979019165}]
```

Results

Accuracy results:
ViT + SimCLR + Logistic Regression




```
Test accuracy for  10 images per label: 98.79%
Test accuracy for  20 images per label: 99.40%
Test accuracy for  50 images per label: 99.46%
Test accuracy for 100 images per label: 99.52%
Test accuracy for 200 images per label: 99.54%
Test accuracy for 500 images per label: 99.54%
```

With only 10 images per class, 98.79% accuracy can be achieved which is impossible if we only train with 10x10 images.

With entire training dataset, 99.54% accuracy is achieved.

Thanks!