University of Michigan
EECS 442/504: Computer Vision
Fall 2022. Instructor: Andrew Owens.

Problem Set 9: Representation Learning

Posted: Wednesday, Nov 16, 2022 **Due:** Wednesday, Nov 30, 2022

Please convert your Colab notebook to a PDF file and submit the PDF file to Gradescope. We have included the PDF conversion script at the end of the notebook. Nothing needs to be submitted to Canvas.

The starter code can be found at:
https://drive.google.com/file/d/1ffp2gnfSS0nE54oRfKxEV3LvZpj-Ebq0/view?usp=share_link

We recommend editing and running your code in Google Colab, although you are welcome to use your local machine instead.

Problem 9.1 *Autoencoders* (5 pts)

We'll start by implementing a simple self-supervised learning method: an autoencoder. The autoencoder is composed of an encoder and a decoder. The encoder often compresses the original data with a funnel-like architecture, i.e., it throws away redundant information by reducing the layer sizes gradually. The final output size of the encoder is a *bottleneck* that is much smaller than the size of the original data. The decoder will use this limited amount of information to reconstruct the original data. If the reconstruction is successful, the encoder has arguably captured a useful, concise representation of the original data.

Such representations could help with downstream tasks such as object recognition, semantic segmentation, etc. Here, to test the usefulness of the representation, we'll train the encoders on the STL-10 dataset, which is designed to evaluate unsupervised learning algorithms. This dataset contains 100,000 unlabeled images, 5,000 labeled training images, and 8,000 labeled test images. To keep training time short, we'll use 10,000 unlabeled images to learn representations.

We will then use the feature representation that we learned to train an object recognition model (a simple linear classifier) on the 5,000 labeled training images. If the learned representations are useful, we should obtain a performance improvement over only using the small, labeled training set.

1. We will build a small convolutional autoencoder and train it on the STL-10 dataset. The conv layers in the autoencoder all have kernel size = 4x4, stride = 2, padding = 1.

2. With the trained autoencoder, we freeze the parameter of the encoder and train a linear classifier on the autoencoder representations, i.e., the output of the encoder. You will compare

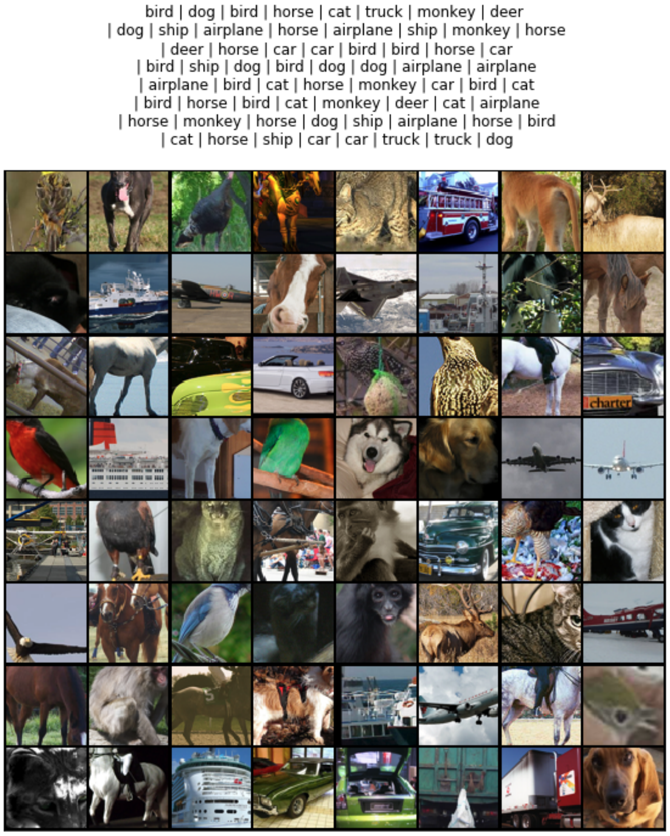
bird | dog | bird | horse | cat | truck | monkey | deer
| dog | ship | airplane | horse | airplane | ship | monkey | horse
| deer | horse | car | car | bird | bird | horse | car
| bird | ship | dog | bird | dog | dog | airplane | airplane
| airplane | bird | cat | horse | monkey | car | bird | cat
| bird | horse | bird | cat | monkey | deer | cat | airplane
| horse | monkey | horse | dog | ship | airplane | horse | bird
| cat | horse | ship | car | car | truck | truck | dog
charter

Figure 1: Sample images from STL-10 dataset.

the accuracy of the linear classifier with two other linear classifiers. One is trained together with the encoder and the other one is trained on top of a randomly initialized encoder. Confirm that the unsupervised pretraining improves the classification accuracy compared to the random baseline. Method I should achieve about 30% accuracy on the test set. Method II should achieve above 40% accuracy. Method III performs the worse among these three.

List of functions/classes to implement:

1. `class Encoder` (1 pt)
2. `class Decoder` (1 pt)
3. `def train_ae` (1 pt)
4. `def train_classfier` and set the supervised parameter in three methods (1 pt)
5. Report results in the Report results section at the end of the notebook (1 pt)

Problem 9.2 *Associating Images with Language* (5 pts)

We will implement a simplified version of a recent representation learning method, Contrastive Language-Image Pre-training (CLIP) [1, 2] that learns to associate images with text provided by humans[1].

[1] We covered contrastive learning in the *representation learning* and *language* lectures.

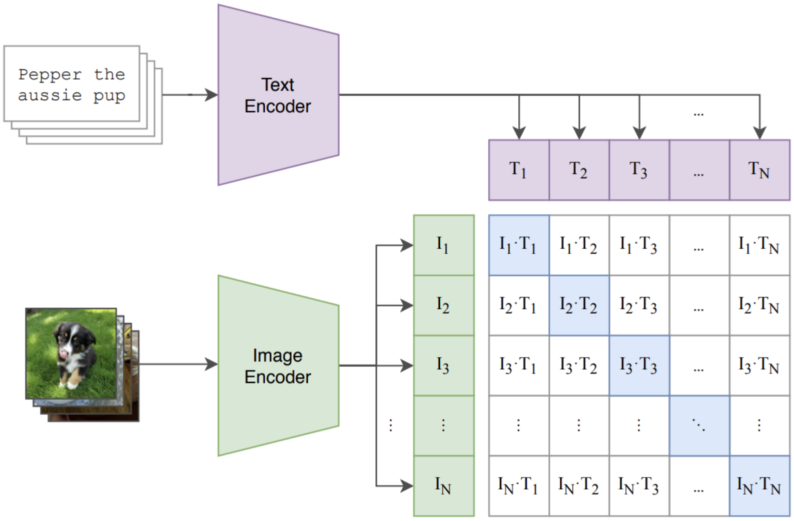

Figure 2: We learn a joint embedding between images and their corresponding text using a contrastive loss. Figure source: Radford et al. [1].

We will learn a vector embedding for images using contrastive learning: in this representation, the input image and it corresponding caption should have a large dot product, while dot products between an image and other unrelated captions should be small. We first extract the feature embeddings for the image and its respective caption using a *ImageEncoder* (ResNet50[3]) and a *TextEncoder* (DistilBert [4]) respectively. Next, we project both these embeddings into a common vector space for matching as illustrated in part Figure 2. Finally, we will use InfoNCE loss [5] to train the contrastive model:

$$\mathcal{L}^{I,T}_{\text{contrast}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(I_i \cdot T_i/\tau)}{\sum_{j=1}^{N}\exp(I_i \cdot T_j/\tau)}, \tag{1}$$

where I_i and T_i are the image and text embeddings for the i_{th} sample in a minibatch, and N denotes the batch size. The constant τ is used to control the range of logits. The denominator sums over all of the other text embeddings in the batch.

We will minimize a symmetric objective function:

$$\mathcal{L} = \frac{\mathcal{L}^{I,T}_{\text{contrast}} + \mathcal{L}^{T,I}_{\text{contrast}}}{2}. \tag{2}$$

where the second term is defined:

$$\mathcal{L}^{T,I}_{\text{contrast}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(T_i \cdot I_i/\tau)}{\sum_{j=1}^{N}\exp(T_i \cdot I_j/\tau)}. \tag{3}$$

Conceptually, $\mathcal{L}^{T,I}_{\text{contrast}}$ matches the text to the image (rather than the image-to-text, as in $\mathcal{L}^{I,T}_{\text{contrast}}$). Its denominator sums over the image embeddings. By minimizing the above loss function, we learn representations for the image and text such that the positive pairs (respective images and their captions) will produce high dot product while giving low dot product for negative samples (image and other unrelated captions).

We will complete the implementation as follows:

Contrastive learning. We will implement a simplified version of the network and loss function. In this simplified model, the language network backbone will be frozen[2], and the visual backbone will be pretrained using ImageNet.

Linear probing evaluation. We will again train a linear classifier for object recognition, using the learned network. There should be a significant performance improvement compared to the autoencoder representations[3], e.g., 75% (or higher) vs. 44%.

Text-to-image retrieval. Next, we will use our model for cross-modal retrieval. Given a text query, e.g., "dogs in the park" we will find the k best-matching images. To do this, we will find the images that have the highest dot products in our learned embedding space.

List of functions/classes to implement:

1. `class ProjectionHead` (1 pt)
2. `class CLIPModel` (2 pts)
3. Report linear classifier accuracies (1 pt)
4. `function retrieve_images` (1 pt)

Acknowledgements. The CLIP problem and code are built on top of Simple CLIP [2].

References

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[2] M. Moein Shariatnia. Simple CLIP, 4 2021.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[5] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[2] While the design of this network is outside of the scope of this class, you can learn more about it by reading Sanh et al. [4].

[3] This performnace improvement may be due to a number of factors, including the larger network size, the use of ImageNet pretraining, and the visual-language learning task.