CS280 - Computer Vision - Spring 2019
A Tale of Two Networks

Due 11:59pm, March 27, 2019

In this assignment, we train convolutional networks with two different kinds of supervision and visualize what they learn. The CIFAR-10 PyTorch tutorial is a good reference[1], if you are unsure of how to start. This assignment does not require access to GPUs, but you can check out Google Colab[2] if you want free GPU compute.

You may do this assignment in groups of two. Please submit your code along with a PDF including your plots and visualizations.

1 Setting Up

1. **Dataset:** Obtain the CIFAR-10 dataset [3]. The easiest way to do this is as presented in the PyTorch tutorial.

2. **Network Architecture:** Construct a lighter and smaller Residual Network (ResNet) architecture. A useful reference is the original ResNet paper by He et al [1]. (Some useful PyTorch code, for reference.[4]) Start with a ResNet of depth 18 and shrink it in depth and width (number of filters).

2 Supervised Learning

Train your network to perform image classification by minimizng the cross-entropy between the network's prediction and the CIFAR-10 targets. Use your favorite optimizer to train your deep network.

1. Write out the definition of the cross-entropy loss function. Interpret it.

2. As the training proceeds, plot the loss value and the classification accuracy on training and validation set, and choose the iteration with the least error.

[1] https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
[2] https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c
[3] https://www.cs.toronto.edu/ kriz/cifar.html
[4] https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

3 Self-Supervision

The aim of self-supervised learning is to learn without relying on human annotation. We will implement a simple but effective self-supervised learning method, **Unsupervised Representation Learning by Predicting Image Rotations** [2]. The main idea is that learning to identify the rotation of image should require learning useful visual representations, i.e. of the objects that appear in the dataset.

You should not have to change much in your code to implement this method; you really only need to slightly extend the way you prepare training data, and modify your classifier.

1. Give your three favorite examples of recent work in self-supervised learning in computer vision (or robotics, NLP, ML, if you want).
2. Provide a learning curve of the training loss as a function of epochs.
3. Show the images of examples with largest and lowest loss, at convergence.

4 Visualize Your Networks

Now we experiment with a few methods for gazing into the soul of the representations learned by your two networks. **Do the following for both networks.**

4.1 Show your filters!

Visualize your filters by saving them as an image. For example, a layer with 64 filters of size 7x7x3 can be saved as 64 separate RGB images of size 7x7. Visualize these filters to identify any filters you learned about in class. Do this for two layers of the network.

1. In the report, include a single image displaying all the filters from one layer.
2. Select a few interesting ones and explain what these filters might be doing.

4.2 Class Activation Mappings

We now try to interpret the role of intermediate feature maps in classification. Implement **Class Activation Mappings** by Zhou et al [3]. This should be straight-forward with a ResNet architecture.

1. Why does the ResNet Architecture make class activation mapping easy?
2. Describe the CAM visualization algorithm in one sentence.
3. Provide your favorite class activation maps (for three classes) and one example of activations that might seem counter-intuitive.

4.3 Embedding Space Visualization with tSNE

tSNE [4] is a popular method for nonlinear dimensionality reduction. The main idea is to project high dimensional data (e.g. features) to a 2d map, such that local structure (i.e. neighbors) is preserved. We will use tSNE to visualize the mapping learned by your networks. In particular, compute features of the layer before the classifier for 100 images, and use tSNE to project them to a 2d map. You can use the implementation of tSNE in scikit-learn.

1. Plot the 2d map as a scatter plot and color the points according to their CIFAR class. Interpret it.

4.4 Love Thy Neighbours!

Another way to understand how good the learned features are is to use the penultimate layer features for kNN retrievals, and compare the results from two learned networks.

1. Visualize the 10 nearest neighbours in the training set for 5 test images (from different classes) and compare the results from the two trained networks.

Look at Section 6.1 (visualized in the right panel of Figure 4) of [5].

4.5 Compare Them

Compare the visualizations of the two networks. What was expected? What was not expected?

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[2] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[3] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[4] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.