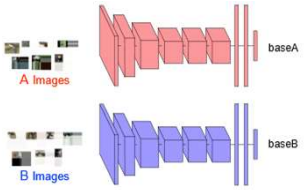
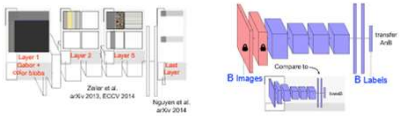
Maximizing Feature Extraction in Deep Neural Networks for Transfer Learning

Ayon Borthakur (ab2535) , Matt Einhorn (me263)

Introduction



Training on multiple datasets:
- We can train separate models for the separate data-sets
- Performance may be reduced when there is not enough training examples for data-set B.



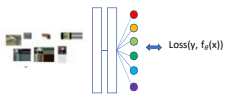
- Instead, pre-train on dataset A:
  - Initialize first few layers with weights from A
  - Randomly initialize remaining layers
  - Reduce learning rate and "fine tune" on dataset B
- This improves accuracy on dataset B **[1, 4]**

Methods

We would like to extract more features from A to increase fine-tuning performance on B.

Idea is to pose a harder problem to the network

- Given dataset **A** with data (x, y) where y∈ C={1, 2,…,N}, e.g. **{1,2,3,4,5,6}**.

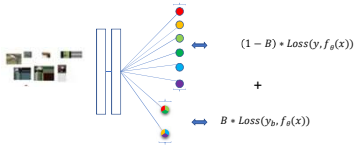


References

[1] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328.
[2] G. V. Horn and P. Perona, "The Devil is in the Tails: Fine-grained Classification in the Wild," *SciRate*, Sep. 2017.
[3] D. Arpit *et al.*, "A Closer Look at Memorization in Deep Networks," *ArXiv170605394 Cs Stat*, Jun. 2017.
[4] M. C. Kruithof, H. Bouma, N. M. Fischer, and K. Schutte, "Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers," presented at the Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII, 2016, vol. 9995, p. 99950K.
[5] B. Amos, *densenet.pytorch: A PyTorch implementation of DenseNet*. 2017.
[6] W. Yang, *pytorch-classification: Classification with PyTorch*. 2017.
[7] *vision: Datasets, Transforms and Models specific to Computer Vision*. pytorch, 2017.

Methods – con't

- **Randomly** partition C into 2 equal seized subsets, e.g. c0=**{1,3,4}**, c1=**{2,5,6}**
- New labels $y_b(y) = \begin{cases} 0, if\ y \in c_0 \\ 1, otherwise \end{cases}$



- New dataset **A'** (x, y, y_b(y))
- New loss $(1-B) * Loss(y, f_{\theta(x)}) + B * Loss(y_b, f_{\theta(x)})$

- Train the network on dataset **A'**, then
  - Initialize new network with weights from A' and remove binary layer, educe learning rate.
  - Reset final classification layer, use normal learning rate.
  - Train for reduced number of epochs on **B**

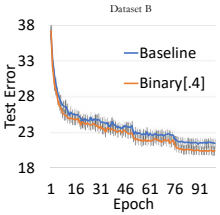
Intuition: instead of finding minimum features for classification (e.g. sphere is ball, rectangle is box), find features that relate classes to each other.

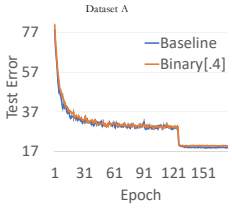
Results

Dataset:
- CIFAR-100, 32X32, 100 classes, 500 images/class for train, 100 images/class for test
- 50/50 split for A, B

Network:
- DenseNet-BC (L=100, k=12), only 0.8M

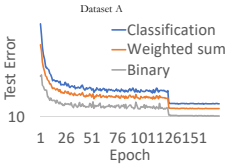
Regime:
- Train on A for 175 epochs with or without binary layer, fine-tune on B for 100 epochs (early stopping)



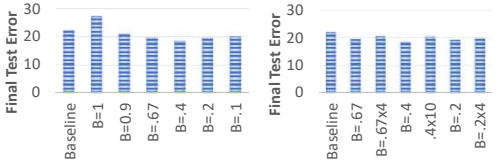


Results - con't

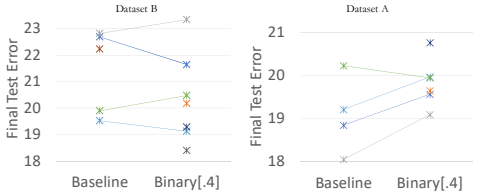
- Test error for classification and binary layers are smoothly converging



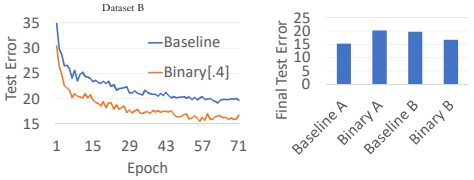
Binary layer in A improved performance on B
- Weight of binary loss affected performance
- More binary layers is worse than one binary layer



- Binary classifier generally **reduces** the test error on dataset B at the cost of an **increase** in test error on dataset A.

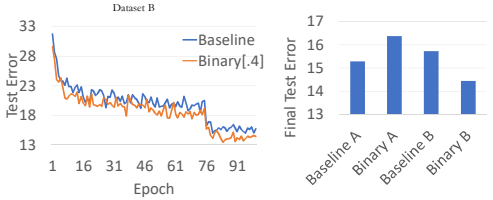


- **WideResNet** (WRN-28-10 (drop 0.3), 36.48M) on **CIFAR-100**
- Train 112 epochs on A, fine-tune 71 epochs on B
- we observe an even greater performance increase

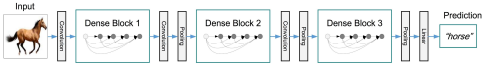


Results - con't / Discussion

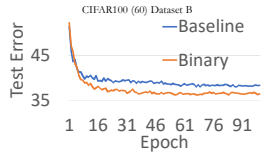
- Using a **DenseNet** (DenseNet-121 (k=32), 8M) on **ImageNet-100** we also observe a performance improvement



- There is a dependency on class **split**
- There's **always** tradeoff between **reduced** performance on A with **improved** performance on B
- Larger network (ResNet) has greater improvement, suggesting network needs sufficient **capacity**. Perhaps the DenseNet used is too small.



- Multiple binary layers reduces performance, suggesting there may not be nice minima for multiple layers
- Transferring out of domain from CIFAR-100 to CIFAR-10 or SVHN showed no improvement.
- Reducing CIFAR-100 to 60 images/class showed some improvement, but error was high and variable at different classes sizes



Conclusions / Future Work

- Observed improvement on multiple architectures and datasets with a binary layer
- Use larger and better optimized network
- Increase number of replications to verify result is not random