Transfer Learning Project: Two-Headed CNN with Interaction Loss

Christian Montgomery and Aksel Kretsinger-Walters
Columbia University
cm4521@columbia.edu, adk2164@columbia.edu

1 Introduction

In order to maximize performance / minimize error on the Transfer Learning project, we are going to construct a 2-headed CNN to learn the sub-class and super-class classifications. In order to ensure our model generalizes well to unseen categories, we are going to augment our training dataset with various image types not in sample (subset of the ImageNet $\sim$ 20,000 categories) and modify our training set images (translate, vertical/horizontal flip, rotations, etc.). In addition, we will incorporate various methods to improve generalization discussed in class such as regularization penalty/weight decay, early stopping, adding linear bottleneck layers and implementing dropout of units. Lastly, we plan on experimenting with an ensemble of models (bagging/boosting) or Mixture of Experts (MOE) to maximize our model performance on the class leaderboard.

2 Method / Algorithms to investigate

To initially normalize our input data, our plan is to rescale each pixel value to between [0, 1] for each input channel (RGB). Once we introduce data augmentation to the training data, we may also use batch normalization to Z-score across mini-batches to decrease training speed.

From the provided training data, we plan on performing data augmentation to ensure our model is able to generalize well. This includes extending our training set with modifications to original training images, such as horizontal/vertical flipping, rotations, translation, and noise. In addition to the provided training data, we plan on curating a subset of ImageNet representative of the wide variety of various other categories beyond the 3 provided. Thus, we will train our model to recognize any of these other broad categories as "novel" superclass. Then we plan to pull a small subset of images representing other subcategories of reptiles, birds, and dogs so that our model can identify "novel" subclasses (perhaps 10 examples per new subcategory).

We've decided to use a convolutional neural network as our base model as convolutions have been repeatedly shown to perform well on image classification tasks. In order to keep the number of parameters down, we plan on combining the CNN with linear bottleneck layers.

To ensure our CNN is robust, we plan on incorporating pooling, residual layers (skip-layer connections) and ReLU activation functions to help overcome the vanishing gradient problem. We also plan to use a weight initialization method to break symmetry during training

In order to have full control over the weights learned of our model, we are training our own CNN, rather than taking a pretrained model as a base (i.e. ResNet, AlexNet), freezing weights and then fine-tuning on our specific transfer learning problem. We may reconsider, however, if the training time of our original model is too inefficient and poor performing.

Atop the base CNN architecture, we have 2 separate softmax heads, one for each class category (superclass and subclass). We've decided to try this first instead of 2 distinct models, for simplicity and time efficiency of training. For each head, we will learn weights for that linear layer, and calculate cross-entropy loss via calculating logits and softmax for each potential superclass & subclass respectively.

However, we are aware of potential divergence between superclass and subclass predictions (for example, if the superclass prediction is bird but the subclass prediction is terrier). We'll account for this via an interaction loss term, by calculating this loss as KL divergence between 1. implied probability of a superclass (sum of corresponding subclasses) and 2. the superclass model head's calculated probability of that superclass.

Since our goal is to generalize well in order to recognize novel super/subclasses, we will incorporate additional regularization techniques such as using weight decay to learn less complex weights, and early stopping to ensure we don't overfit to the training data. For the gradient descent process itself, we intend to optimize it via Adam and learning rate decay. Additionally, we plan to use dropout during training to prevent overfitting to our training data.

3 Model Architecture

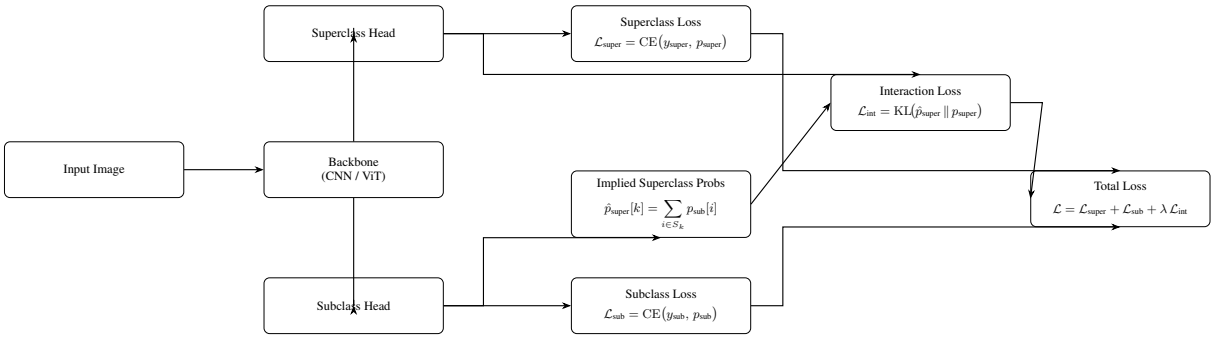

Figure 1: Model architecture and loss computation.

4 Interaction Loss / KL Divergence Calculations

For each superclass k, with subclass set S_k:

$$\hat{p}_{\text{super}}[k] = \sum_{i \in S_k} p_{\text{sub}}[i].$$

Then define interaction loss, e.g.:

$$\mathcal{L}_{\text{int}} = \text{KL}\big(\hat{p}_{\text{super}} \,\|\, p_{\text{super}}\big).$$

The Kullback–Leibler (KL) divergence between two distributions P and Q is:

$$\text{KL}(P \,\|\, Q) = \sum_k P(k) \, \log \frac{P(k)}{Q(k)}.$$

References

[1] **J. Donahue**, **Y. Jia**, **O. Vinyals**, **J. Hoffman**, **N. Zhang**, **E. Tzeng**, and **T. Darrell**. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proceedings of the 31st ICML*, 2014.

[2] **A. Krizhevsky**, **I. Sutskever**, and **G. E. Hinton**. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[3] **N. Srivastava**, **G. Hinton**, **A. Krizhevsky**, **I. Sutskever**, and **R. Salakhutdinov**. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[4] **L. Breiman**. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[5] **Y. Freund** and **R. E. Schapire**. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] **R. Caruana**. Multitask Learning. *Machine Learning*, 28(1):41–75, 1997.

[7] **G. Hinton**, **O. Vinyals**, and **J. Dean**. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.