Week 11 - Pretrained Representations

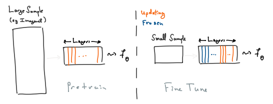
Thursday, April 3, 2025 9:53 AM

Motivation

1. In the last few weeks of this course, we'll discuss interpretability for foundation models. These models are "generalists" in the sense that they are accurate across many downstream tasks, despite never having been given task-specific training objectives. Our reasons for wanting interpretability haven't gone away. However, we will need to adapt techniques for model interpretability to this new setting.

2. While the developments in foundation models are exciting, the idea of transfer learning from large/heterogeneous datasets to small/specialized ones is not new, and many proposals for interpreting foundation model embeddings can be traced back to earlier strategies for reasoning about pretraining and transfer learning. These methods are the focus of today's lecture.

3. The transfer learning recipe is sketched below:

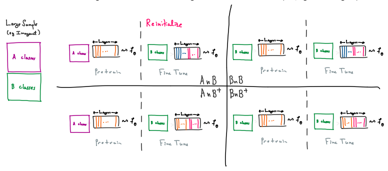


The main differences with foundation models are that, in transfer learning, the same model is being used in the pretraining and the fine-tuning stages. Also, at least some of the model parameters are updated during the fine-tuning stage.

Feature Transferability

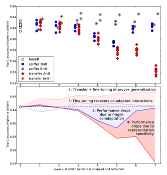
1. A first step towards understanding what happens in transfer learning is to simply evaluate prediction performance. Instead of any complicated representation analysis, we can design experiments to tease apart different mechanisms for when and why transfer learning works. Yosinski et al. (2014) was one of the first to investigate this in the context of deep learning.

2. Their main experiment has the following design:
   - Divide the data into two disjoint sets of classes. Pretrain the model on one of those sets.
   - Initialize a new model using some of the pretrained weights. Either freeze these weights or allow them to continue to be updated (top/bottom in the figure below).
   - Continue training the model on either the original or the unseen dataset (left/right in the figure).



This experiment helps identify the layers in the network that have the more general, re-usable features. The model that continues training on the original data (right-hand panel) is used as a negative control.

3. The results appear in the figure below. The notation BnB/AnB refers to whether the training continued on the original (BnB) or new (AnB) dataset. These two settings are distinguished by color. The notation "+" means that the pretrained model parameters can be updated in the second-half of training; otherwise the weights are frozen. The x-axis is the number of pretrained layers transferred, and the y-axis is test accuracy.



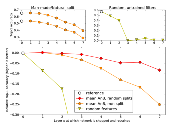
4. The decreasing trend for the dark red points is expected, because the bottom layers of the network are expected to have more general features. The top layers are more specialized to the task that they are trained on, so failing to fine-tune them leads to an out-of-distribution drop in performance. The more opaque points correspond to the fine-tuned settings, and they recover the original performance.

5. There are two surprising findings in the figure. The first is that the solid blue points have a drop near the middle of the network. Remember, these were supposed to be negative controls; we would have expected the model to achieve the same performance as training from scratch. What's happening is that when the weights for the upper half of the network are randomly re-initialized, the optimization algorithm isn't able to recover the same configuration it had before training was interrupted. It can't simply find a good solution from scratch -- it needs to find a good solution compatible with the first half of the network.

6. The second surprising thing is that AnB+ does consistently better than BnB+. I'm not sure if there has been any convincing follow-up about the mechanism behind this, but Yosinski et al. (2014) argue that transferring across domains improves the quality of the learned features and subsequent test set performance.

7. In the experiment above, the A/B split was performed by randomly splitting ImageNet classes. However, many ImageNet classes are closely related to one another. What can we say about transfer learning when the "distance" between the source and target datasets increases? The authors experiment with splitting classes using the ImageNet hierarchy as a guide. The split is based on the leaves of disjoint subtrees, one which includes all the "natural" classes and the other which includes the "man-made" ones.

8. Performance deteriorates more dramatically in this case. Nonetheless, the first few layers can still be successfully transferred without leading to much of a drop.



Medical Models and SVCCA

1. We can learn even more about transfer learning if we analyze the model's representations. This is nicely illustrated in Raghu and Zhang et al. (2019), which applies Singular Vector Canonical Correlation Analysis (SVCCA) to better understand transfer learning in medical image classification models.

2. The starting point for this paper is that, in practice, it's common for medical image classification models to start with pretrained weights from ImageNet. This has traditionally been motivated by the fact that medical image datasets are more costly to collect, which limits sample sizes. However, medical images are qualitatively different from ImageNet images (e.g., higher resolution), so it's natural to ask what exactly is the benefit of transfer learning.

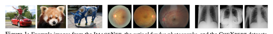
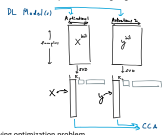


Figure 1: Example images from the IMAGENET, the *retinal fundus photographs*, and the CHEXPERT datasets, respectively. The fundus photographs and chest x-rays have much higher resolution than the IMAGENET images, and are classified by looking for small local variations in tissue.

3. The authors provide some performance comparisons, but our focus in these notes is on their application of SVCCA, a method for evaluating the similarity of pairs of model activations that's based on the multivariate statistical technique Canonical Correlation Analysis (Hotelling 1936).

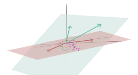
4. The starting point of the method are two sets of model activations gathered on the same collection of samples, but from different sets of neurons. The neurons might be the from the same layer at different epochs, layers, or even models. Given this input, the recipe is: (i) compute the top singular vectors for each set of activations and (ii) apply canonical correlation analysis (CCA) to measure the similarity of the resulting singular vectors.



5. CCA solves the following optimization problem,

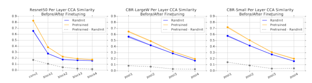
$$\underset{u,v}{\text{maximize}} \quad u^T \hat{\Sigma}_{xy} v$$
$$\text{s.t.} \quad u^T \hat{\Sigma}_{xx} u = v^T \hat{\Sigma}_{yy} v = 1$$

which can be interpreted as maximizing covariance, just like how PCA maximizes variance. Geometrically, the optimization finds pair of vectors that minimizes the angle between the subspaces spanned by the columns of the two input matrices. Smaller angles correspond to higher CCA correlations.

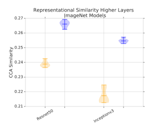


6. Aside: My impression is that step (i) is a kind of denoising heuristic -- estimating high-dimensional covariances is difficult. Arguably, a regularization penalty would have a similar effect. Moreover, the official repository implementing this algorithm doesn't seem to do the initial SVD step described in the paper.

7. Using SVCCA, the authors discover that fine-tuning doesn't have much effect on earlier layers of the network, especially for the larger Resnet50 model. This analysis gives a sense of the *learning dynamics* of transfer learning. The fact that this features aren't changing even in the randomly initialized network means that feature reuse alone isn't enough of an explanation.



8. They also argue that the representations learned after pretraining are noticeably different from those learned using random initialization. Those from different random initializations are relatively more similar to one another. The blue below come from comparing activations from pairs of randomly initialized models, while the orange compares activations after training from random vs. pretrained initializations.



This is consistent with an earlier study by Erhan et al. (2010) which used nonlinear dimensionality reduction to understand the dynamics of model activations across training when pretraining was/was not used. In a sense, pretraining serves as a regularizer.

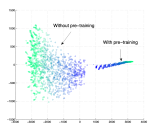
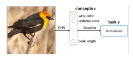


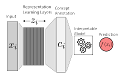
Figure 6: 2D visualization with ISOMAP of the functions represented by 50 networks with and 50 networks without pre-training, as supervised training proceeds over MNIST. See Section 6.3 for an explanation. Color from dark blue to cyan indicates a progression in training iterations (training is longer without pre-training). The plot shows models with 2 hidden layers but results are similar with other depths.

Post-Hoc Concept Bottlenecks

1. This lecture might feel like déjà vu, because the activation analysis developed by Raghu and Zhang et al. (2019) is quite related to concept activation vector (CAV) technique from Week 7. Now that we additional perspective on activation analysis, let's consider a recent development in the topic, Post-Hoc Concept Bottleneck Models (Yuksekgonul et al. 2023).

2. PCBM's are an alternative to CBMs, a concept-based model that imposes a "bottleneck" on the learned representations. Instead of directly predicting the response, the model must first predict an intermediate set of concepts -- detailed, interpretable features that describes training sample. An intrinsically interpretable model is then used to predict the final output based on predicted concepts. Intuitively, the layers from the inputs to the concepts can be viewed as automating the extraction of a manually-defined set of features.

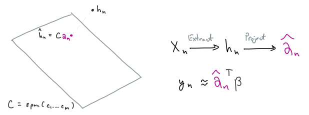




3. The main limitation of CBMs is that it's expensive to obtain concept labels. E.g., an expert has to go in and provide labels for wing color, undertail color. A second limitation is that, unlike CAV, it doesn't apply to arbitrary models -- it's a specific architecture that has to be trained from scratch. PCBMs makes several modifications that help address these problems. It defines a new architecture like CBMs, but which benefits from the flexibility of CAVs.

4. The first trick is to augment the usual CAV vectors with cheaply derived ones. Remember that CAV vectors are vectors in the activation space that distinguish samples with certain concepts (e.g., stripes). These are usually learned by training a linear classifier. In contrast, Yuksekgonul et al. (2023) propose using multimodal models, whose embeddings are associated with both text and images. This allows them to embed a phrase (e.g., "Yellow Wing") and get a vector in the activation space that can serve as a CAV.

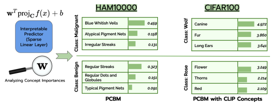
1. The second idea makes it possible to add a concept bottleneck to arbitrary, already trained models. Specifically, the arbitrary model's activations are projected onto the subspace spanned by the concepts. The coordinates of the projected activations with respect to the underlying concepts can be used to interpret the sample. These projected activations are then fed into an interpretable model, like lasso regression.


h_n
$\hat{h}_n = C a_n$
$C = \text{span}(c_1 \ldots c_m)$
$X_n \xrightarrow{\text{Extract}} h_n \xrightarrow{\text{Project}} \hat{a}_n$
$y_n \approx \hat{a}_n^T \beta$

6. A final extension is used to boost performance: A small black box model is trained on the residuals of the previous, "bottlenecked" model. This makes it possible for the model to leverage patterns that are not obviously present in any existing concept annotations. This residual learning is done after the initial linear model on concept projections has been trained. (Aside: I wonder what happens if you project onto the orthogonal complement of the concept subspace).

$$e_n := y_n - \hat{a}_n^T \hat{\beta}$$
$$\hat{g} = \underset{g \in \mathcal{G}}{\arg\min}\, L(e_n, g(x_n))$$

7. Linear models on concept activations are easy to reason about, and they give examples on both natural and medical images. They argue that these bottlenecks support model control: if there are concepts that we don't want the model to use (e.g., race or gender), we can remove them by setting the associated linear model coefficient to zero. This is illustrated in a user study where users are asked to edit bottleneck coefficients to improve out-of-domain generalization. Note, however, that it's still possible for a concept to leak through the residual learning component.


$\mathbf{w}^T \text{proj}_{\mathbf{C}} f(x) + b$
Interpretable Predictor (Sparse Linear Layer)
W
Analyzing Concept Importances
HAM10000
Class: Malignant
Blue Whitish Veils 0.459
Atypical Pigment Nets 0.158
Irregular Streaks 0.131
Class: Benign
Regular Streaks 0.323
Regular Dots and Globules 0.151
Typical Pigment Nets 0.091
PCBM
CIFAR100
Class: Wolf
Canine 4.972
Fur 3.860
Long Ears 3.641
Class: Rose
Flower 3.249
Thorns 2.214
Red 2.109
PCBM with CLIP Concepts