Learning Independent, Class-Correlated Features

Anshuman Suri
University of Virginia
anshuman@virginia.edu

Abstract

Machine learning models learning highly correlated and redundant features is a well-known result in the literature. However, this redundancy is useful only when the learned features are truly independent. Multiple features that focus on the same concept (in terms of human knowledge) are not useful at all. Instead, features that are highly correlated, when given the class label, are much more useful. In this project, I aim to study the correlation of features learned by existing machine learning models, and experiment with some methods that can be used to learn concepts that are independent without any prior, but highly correlated for a given class label via variants of multi-task learning.

1. Introduction

When training a classification model with a labeled dataset, the only objective that drives the model is the classification loss. Nothing stops the model from learning repeated features. For instance, for the CIFAR-10 [8] dataset, the model may learn only to focus on tires, and in theory, such a classifier can achieve performance as good as one that looks at distinct features like tires, windows, doors, etc. Much research on feature-independence and correlation analysis [2] has shown that models often learn sparse and highly correlated representations.

Although work in existing literature aims towards decorrelation, focusing purely on feature decorrelation is at odds with robust learning [7]. Features like doors and tires may appear together in the CIFAR-10 dataset, and thus over this dataset are highly correlated. Nonetheless, a model learning to identify these two is arguably better than a model that learns redundant representations of the same inherent features.

2. Feature Independence and Robustness

One way to get a better understanding of how correlated neurons are in a model is to note variations in the model's features with small changes in the input. Using the CIFAR-10 test set, I generated PGD-based adversarial examples [9] (which is equivalent to adding noise within a specific budget) for 20 iterations, and then calculating correlation coefficients over these observations.

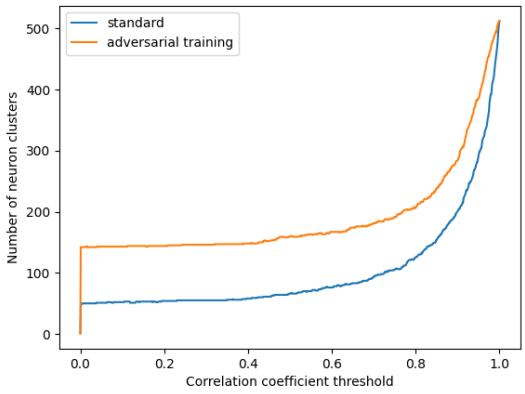

Figure 1. Number of independent features in the penultimate layer of various VGG-19 modelS (standard training and adversarial training) for varying thresholds of similarity, when observed over perturbations on the CIFAR-10 validation set.

As visible in Figure 1, when combining neurons within a specific coefficient threshold, a significant fraction of neurons are highly correlated. For instance, when clubbing neurons with 0.8 coefficient threshold or more, only 150 neurons (out of 512) are seemingly independent. This is somewhat in agreement with overall results with the success of dropout; high redundancy ensures that randomly dropping out features does not impact performance much. However, there is a noticeable difference between standard and adversarial training (which gives robust models that have gradients that perceptually align better). Specifically, using adversarial training leads to much less class-correlated neurons. This observation serves as empirical evidence for the claim of independent features helping with robustness: training for robustness does seem to give seemingly more autonomous features.

Table 1. Description and statistics of class-wise concepts for the CIFAR-10 dataset.

Class	Concepts	Average Images per Concept	
		Before Filtering	After Filtering
bird	beak, eyes, feather, feet, tail, wings	567	225
car	bumper, door, headlight, side view mirror, tail light, tire, window	740	303
cat	claw, ear, eye, nose, paw, tail	601	221
deer	antler, eyes, mouth, tail	760	221
dog	ears, eyes, mouth, nose, tail	540	282
frog	eyes, foot	597	207
horse	belly, eyes, hoof, legs, mane, mouth	550	223
plane	cockpit, fuselage, jet engine, landing gear, tail, wing	559	178
ship	hatch, hull	597	151
truck	bumper, door, lights, tire, window	636	230

3. Dataset

The main classification task is image-classification based on the CIFAR-10 dataset. A set of class-wise concepts is defined (Table 1). Although it is possible to go through the whole dataset and assign these fine-grained levels to each class, there are broadly two problems:

- Iterating through the whole dataset and assigning multiple labels to each instance can be very time consuming, especially for huge datasets like Imagenet [4].
- Each instance may contain multiple concepts in the same image. This can make it hard for the classifier to dissociate potentially correlated features. For instance, for the car class, wheels and doors will almost always appear together. An ideal classifier that holds good generalization properties should not learn the presence of these two labels as being very highly correlated.

To counter these potential problems, I used Bing[1] to automate and download images that focus on individual class-wise concepts. For each class-concept pair, queries of the form "class concept" are made, *e.g.* 'car windows', with an upper limit of 800 per class-concept pair to avoid throttling/blocking. As expected, this process yields a lot of noise in the scraped dataset. For instance, searching for "bird beaks" also gives a lot of ancient doctor images (who had bird-like masks). More statistics of the dataset are given in Table 1. After manual cleaning, resizing to $(32, 32)$, and removing corrupt/duplicate images, each of the class-concept image sets is split into training and validation sets: randomly sampled with a $70:30$ ratio.

4. Proposed Solutions

There are several possible ways to tackle the problem of learning independent, class-correlated features. I experimented with some of them (C_1, C_2, C_3), described in detail in the following sections.

[1] https://github.com/ostrolucky/Bulk-Bing-Image-downloader

4.1. Class-Wise Concept Learners

This is a naive approach to incorporate class-level information into the end-to-end classification pipeline. A VGG-19 [10] model pre-trained on the CIFAR-10 model is fine-tuned (all Dense layers towards the end of the model are trained) for each class-concept pair. For each class, a one-vs-all classifier (predicting a score in $[0, 1]$) is trained for one concept at a time. Then, the CIFAR-10 dataset is run through all of these class-concept classifiers to get prediction scores. When training these concept classifiers, it is crucial that the **negative samples** the model sees belong to the **same class**. Not only does it help the model learn features that are useful for that concept, but it also stops the model from learning other features of that class in order to get non-trivial accuracy (by just learning car v/s no car classification for instance).

4.1.1 C_1: Using Concept Classifier Scores

Using scores from each of these concept classifiers as features, a random-forest classifier is trained on the CIFAR-10 dataset. Concretely, each image x_i is passed through all the n concept classifiers H_j independently, and then resulting scores from all models are concatenated to form the feature vector z:

$$z_i = H_1(x_i)||H_2(x_i)...||H_n(x_i) \tag{1}$$

Since these concept classifiers are trained using class-wise data, information leakage (a door v/s no-door classifier learning to look at tires as well) is highly unlikely, as direct output classification scores are used as features. Such feature leakage could, however, happen if feature representations are directly used.

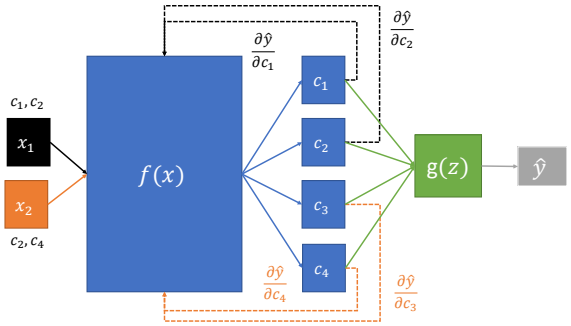

Figure 2. Flow diagram for a model that uses concept-labelled instances to learn various features for various segments of the feature classification layer.

4.1.2 C_2: Using Concept Classifier Features

An alternative approach to incorporate concept-wise information is to use feature representations of the concept classifiers instead of direct scores. Training with this approach is only slightly more computationally expensive than using direct scores and increases the expression capacity of the overall model (using all features is arguably better than using just classification scores).

4.2. Feature-Components

Instead of performing an expensive overhead of training multiple concept-wise classifiers, an alternative approach is to use a single classifier and enforce structure in the latent space that utilizes concept-label information.

4.2.1 C_3: Partitioning the Latent Space

Let the latent space (a few layers before the logits) be z of size nk, where n is the number of concepts. The proposed training algorithm mixes batches of the original dataset, which are allowed to use all of z, and batches corresponding to concepts, which are only allowed to use chunks of size k from the feature space. Concretely, for examples (x_1, y_1) and (x_2, y_2, q), where the first tuple is from the original dataset, and the second one from the concept dataset (with concept q), model propagation happens as:

$$\hat{y}_1 = g(f(z_1)), z_1 = f(x_1)$$
$$\hat{y}_2 = g(f(z_2)), z_2 = f(x_2)[M(q)],$$

where $M(q)$ maps the set of indices of a chunk according to the concept q. Concretely, for a classifier with $n = 48, k = 80$, x_1 uses all 3840 features whereas x_2 uses only a chunk of size 80. Note that this chunk's start and end indices are deterministic, according to the concept. The model flow is also described in Figure 2. This way, the concept label information is implicitly passed on to the model via the chunk of the feature space. The intuition of this training algorithm is to encourage the model to associate certain features related to a concept to specific portions of the latent space. Thus, for inputs, it should be able to look at just the features for a specific concept and perform normal classification.

Note here that the mixing of the batches is a non-trivial task. One possible approach, to avoid overfitting to either of the distributions or cause too much of a domain shift, is to randomly sample concepts to train on (α out of n), as well as randomly decide if this step should be performed at any iteration (with probability β). In my experiments, I observed that the algorithm is somewhat sensitive to the choice of these parameters: $(\alpha, \beta) = (1, 0.5)$ works best, whereas variants like $(2, 0.25)$ or $(1, 0.75)$ to not allow the model to learn anything useful (random-guess accuracy).

5. Experimental Analysis

I performed experiments with the three solutions proposed in the previous sections. Additionally, as a baseline, I also trained a model a standard model trained on CIFAR-10. Additionally, for experiments that analyze adversarial robustness, I also experimented with an L_∞ robust model (trained with $\epsilon = \frac{8}{255}$). Code for all of these models and experiments is publicly available[2].

5.1. Performance on CIFAR-10

One useful baseline is reporting metrics on the CIFAR-10 test set since that is part of the end goal: concept-identification performance without overall performance is not very desirable. As visible in Table 2, the classifiers perform quite well. C_2 performs slightly worse than C_1, which may be attributed to higher input dimensionality. Notably, C_3 performs significantly worse than C_1 or C_2. This is because the structure enforced in the training process forces the models to solve a much harder task, which can potentially hinder performance.

5.2. Performance on Concept Classification

Although it is not the main classification task, it is important to see how successful these models actually are in learning concept classification. Since the concept-wise datasets are not balanced, reporting F_1 metrics is more useful than accuracy (which can be trivially very high with constant predictions). As visible in Table 2, F_1 scores are, averaged across all concepts and classes, are non-trivial. C_1 performs slight better, which my be attribute to a much larger latent space dimension (4096), allowing the model to learn robust, sparse representations, whereas C_2 has a latent dimension of size 80.

[2]All code is implemented in Python and Pytorch, and is available at `https://github.com/iamgroot42/dlvr_project`

Table 2. Performance metrics for the proposed algorithms. All metrics are reported on the test-set. Mean statistics, with interval bounds corresponding to 90% confidence intervals are reported for concept classification tasks.

Metrics (%)	C_1	C_2	C_3	**Standard**	**Robust**
Accuracy	94.1	94.0	80.4	94.3	87.0
F-1 on Concept Classification	37.6 ± 6.8	33.2 ± 5.2	-	-	-
L_∞ Robustness	-	-	40.7	0	53.5
Semantic Adv. Robustness	5.4	8.0	8.2	5.3	8.7
Boundary Attack++	32.1	41.6	72.7	95.7	92.9

Figure 3. Heatmaps for standard model (first heatmap), C_3 overall (second heatmap), and focus of the latent space of C_3 corresponding to two chosen concepts for that class for the images in the leftmost column.

5.3. Robustness to Adversarial Noise

One of the motivations of incorporating concept information is to help with model robustness. Thus, it is natural to test the adversarial robustness of these models. I performed three kinds of adversarial attacks: white-box L_∞ PGD [9] attack, rotation-translation semantic attacks [6], and BoundaryAttack++ [3], a gradient-free adversarial optimization method. Model accuracies on images perturbed with these attacks are then reported for all models. As expected, C_3 achieves adversarial robustness quite close to a model explicitly trained for robustness. Performance on the gradient-free attack is not as good as normal models, but is still non-trivial and much better than C_1, C_2. This suggests that the L_∞ robustness C_3 achieves is not merely via gradient obfuscation [1].

5.4. Does the Classifier really look for concepts?

Of the few possible ways to see what different parts of the model are learning, one is to look at heat-maps for specific inputs. Intuition suggests that specific parts of the latent space should correspond to focusing on different concepts in the input. I generated heatmaps for the standard model as a baseline. Additionally, I also generated concept-wise heatmaps for some concepts by using only parts of the latent space of C_3 relevant to these concepts.

As visible in Figure 3, parts of the model's latent space responsible for specific concepts do indeed end up focusing on those particular concepts in the image. For instance, the part of the latent space associated with car wheels indeed has a corresponding heatmap around wheels. Similarly, part of the heatmap for the cat associated with ears has close attention at those parts of the image. Although these are not perfect, this is very useful for model debugging and interpretability: concept-wise heatmaps can help decipher which concepts were detected in an image. Additionally, more perceptible gradients are also known to be highly correlated with model robustness [5].

6. Conclusion

Aggregating concept-level classifiers can be used to achieve near state-of-the-art performance on CIFAR-10. As we observed, even a 48-size feature vector (concept classifier scores) are sufficient to achieve 94% test accuracy. Although the F_1 scores for these concept classifiers are not very good, they do give non-trivial performance, factoring in the really small size of the dataset (less than 300 images per concept). Incorporating concept-level information via structure in the latent space also leads to non-trivial performance on the main task, making it worth-wile to further explore and perfect these techniques. Visual inspection of heat-maps, using different parts of the latent space, shows how structuring the latent space helps parts of the model focus on concepts.

Incorporating concept-level information in the classifier does seem to help with robustness against multiple kinds of attacks. Although adversarial semantic attacks still succeed nearly as much as they do on robust models, L_∞ robustness on the C_3 model is nearly as good as a model trained with an adversarial loss, which is much more computationally expensive. Robustness against even gradient-free attacks is much better than standard and adversarially robust models. These promising results, despite the gap in performance on clean data, suggest using concept-level information can be useful for achieving adversarial robustness, which is in agreement with the initial hypothesis of all these experiments.

In addition to validating these results on other datasets and model architectures, future work could potentially include inferring concept-level information without any labels, or use other methods to get noisy, self-supervised labels for concepts.

References

[1] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018.

[2] B. O. Ayinde, T. Inanc, and J. M. Zurada. On correlation of features extracted by deep neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[3] J. Chen, M. I. Jordan, and M. J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 668–685.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[5] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry. Learning perceptually-aligned representations via adversarial robustness. In *ArXiv preprint arXiv:1906.00945*, 2019.

[6] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran. On the limitation of convolutional neural networks in recognizing negative images. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 352–358. IEEE, 2017.

[7] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[8] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[10] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Machine Learning*, 2015.