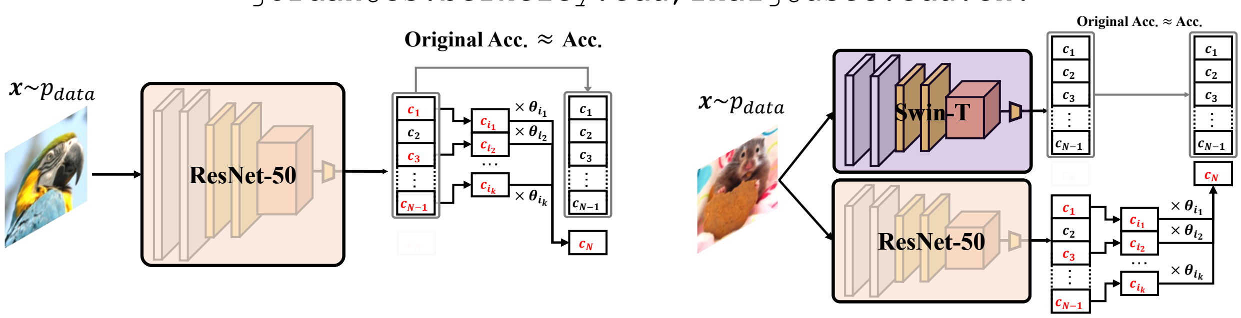
Neural Dependencies Emerging from Learning Massive Categories

Ruili Feng[1,3], Kecheng Zheng[2,1], Kai Zhu[1], Yujun Shen[2], Jian Zhao [1], Yukun Huang[1], Deli Zhao[3], Jingren Zhou[3], Michael Jordan[4], Zheng-Jun Zha[1]
[1]University of Science and Technology of China, Hefei, China
[2]Ant Group, [3]Alibaba Group, Hangzhou, China
[4]University of California, Berkeley
ruilifengustc@gmail.com,{zkcys001,kaizhu}@mail.ustc.edu.cn,
shenyujun0302@gmail.com, {zj140,kevinh}@mail.ustc.edu.cn,
zhaodeli@gmail.com, jingren.zhou@alibaba-inc.com,
jordan@cs.berkeley.edu,zhazj@ustc.edu.cn.



(a) Neural Dependency within a **ResNet-50** **(b)** Neural Dependency between **ResNet-50** and **Swin-T**

Figure 1. **Illustration of neural dependencies** that emerge (a) within a single network and (b) between two independently learned networks. Taking the intra-network dependency as an instance, the logits predicted for the category "macaw" can be *safely replaced* by a linear combination of the logits predicted for a few other categories, barely scarifying the accuracy.

Abstract

This work presents two astonishing findings on neural networks learned for large-scale image classification. 1) Given a well-trained model, the logits predicted for some category can be directly obtained by linearly combining the predictions of a few other categories, which we call ***neural dependency****. 2) Neural dependencies exist not only within a single model, but even between two independently learned models, regardless of their architectures. Towards a theoretical analysis of such phenomena, we demonstrate that identifying neural dependencies is equivalent to solving the Covariance Lasso (CovLasso) regression problem proposed in this paper. Through investigating the properties of the problem solution, we confirm that neural dependency is guaranteed by a redundant logit covariance matrix, which condition is easily met given massive categories, and that neural dependency is highly sparse, implying that one category correlates to only a few others. We further empirically show the potential of neural dependencies in understanding internal data correlations, generalizing models to unseen categories, and improving model robustness with a dependency-derived regularizer. Code to reproduce the results in this paper is available at https://github.com/RuiLiFeng/Neural-Dependencies.*

1. Introduction

Despite the tremendous success of deep neural networks in recognizing massive categories of objects [8–10, 12, 14–16, 23, 27, 29], how they manage to organize and relate different categories remains less explored. A proper analysis of such a problem is beneficial to understanding the network behavior, which further facilitates better utilization of this powerful tool.

In this work, we reveal that a deep model tends to make its own way of data exploration, which sometimes contrasts sharply with human consciousness. We reveal some underlying connections between the predictions from a well-learned image classification model, which appears as one category highly depending on a few others. In the example given in Fig. 1a, we can directly replace the logits predicted for "macaw" with a linear combination of the logits for "ostrich", "bittern", *etc.* (without tuning the network parameters) and achieve similar performance. We call this phenomenon as *neural dependency*, which automatically emerges from learning massive categories. A more surprising finding is that neural dependencies exist not only within a single model, but also between two independently learned models, as shown in Fig. 1b. It is noteworthy that these two models can even have different architectures (*e.g.*, one with convolutional neural network [12] and the other with transformer [10, 16]) and different training strategies.

Towards figuring out what brings neural dependencies and whether they happen accidentally, we make a theoretical investigation and confirm that identifying neural dependencies is equivalent to solving a carefully designed convex optimization—the Covariance Lasso (CovLasso) regression problem proposed in this paper. Such a problem owns a smooth solution path when varying its hyper-parameters [21], which has two appealing properties. First, the solution is guaranteed by a redundant covariance matrix of the category-wise logits. This condition is easily met when the model is trained on a sufficiently large number of categories [11]. Second, the solution admits elegant sparsity. It implies that a category involved in neural dependencies only relates to several instead of numerous other categories.

We further study the potential utilities of neural dependencies, as a support to our theoretical contributions. One straightforward application is to help interpret the internal data correlations, such as what categories are more likely to link to each other (Sec. 3.1). Another application is to investigate how we can generalize a well-learned model to unseen categories with the help of neural dependencies (Sec. 3.2). We also propose a regularizer to test whether discouraging the neural dependencies could assist the model in learning a more robust representation (Sec. 3.3). We believe the findings in this paper would deepen our understanding of the working mechanism of deep neural networks, and also shed light on some common rules in knowledge learning with visual intelligence systems.

2. Neural Dependencies

We consider the n-category classification neural network $f : \mathbb{R}^m \to \mathbb{R}^n$, which takes an input image $\boldsymbol{x} \in \mathbb{R}^m$ and outputs the logits vector of $\boldsymbol{x}$ being any of the n-categories of the task. We assume the network is well-trained and produce meaningful outputs for each category. Naively, each element of the logits vector reports the confidence of the network predicting $\boldsymbol{x}$ belonging to the corresponding category. We are curious about whether those confidences can be used to predict each other. Before we start, we formally introduce the key concept of neural dependency in this work.

Definition 1 *We say the target category c_i and categories $\{c_{i_j}\}_{j=1}^k$ have neural dependency, if and only if for almost every $\boldsymbol{x} \sim p_{\text{data}}$, there are $0 < \epsilon, \delta \ll 1$ and a few constant non-zero coefficients $\{\boldsymbol{\theta}_{i_j}\}_{j=1}^k$, $i \neq i_j \in [n], k \ll n$, such that*

$$\Pr(|f(\boldsymbol{x})_i - \sum_{j=1}^k \boldsymbol{\theta}_{i_j} f(\boldsymbol{x})_{i_j}| < \epsilon) > 1 - \delta. \tag{1}$$

Remark 1 *We do not normalize nor centralize the logits output $f(\boldsymbol{x})$ so that no information is added or removed for logits of each category. Different from usual linear dependency system (where $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$), we omit bias in the neural dependency, i.e., we require $b = 0$ if $f(\boldsymbol{x})_i \approx \sum_{j=1}^k \boldsymbol{\theta}_{i_j} f(\boldsymbol{x})_{i_j} + b$. Thus the existence of neural dependencies suggests that the network believes category c_i is nearly purely decided by categories $c_{i_1}, \cdots, c_{i_k}$ without its own unique information.*

What Does It Means? The neural dependency means that a linear combination of a few categories is in fact another category. It is natural to believe that those categories should admit certain intrinsic correlations. However, for an idea classifier, each category should hold a unique piece of information thus they should not be purely decided by other categories. What's more, we will find that some neural dependencies are also not that understandable for humans. Overall, the neural dependencies reveal a rather strong intrinsic connection of hidden units of neural networks, and are potentially interesting for understanding the generality and robustness of networks.

Between Network Dependencies. We can also solve and analyze the between network neural dependencies through the above methodology for two different neural networks f, g trained on the same dataset independently. Here we want to find a few constant non-zero coefficients $\{\boldsymbol{\theta}_{i_j}\}_{j=1}^k$ such that $\Pr(|g(\boldsymbol{x})_i - \sum_{j=1}^k \boldsymbol{\theta}_{i_j} f(\boldsymbol{x})_{i_j}| < \epsilon) > 1 - \delta$. To find those coefficients, we only need to use the i-th row of $g(\boldsymbol{x})$ to replace $f(\boldsymbol{x})_i$ in Eq. (2). The concepts of within and between network dependencies are also illustrated in Fig. 1.

Notations. We use bold characters to denote vectors and matrix, under-subscript to denote their rows and upper-subscript to denote their columns. For example, for a matrix $\boldsymbol{\mu}$, $\boldsymbol{\mu}_A^B$ denote the sub-matrix of $\boldsymbol{\mu}$ consists of the elements with row indexes in set A and column indexes in set B; for a vector $\boldsymbol{\theta}$, we use $\boldsymbol{\theta}_i$ to denote its i-th row which is a scalar. For a function $f : \mathbb{R}^m \to \mathbb{R}^n$, $f(\boldsymbol{x})_i$ denote the i-th row of vector $f(\boldsymbol{x})$, while $f_i(\boldsymbol{x})$ is some other function that connected with sub-script i. For an integer $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \cdots, n\}$. We always assume that matrices have full rank unless specifically mentioned; low-rank matrices are represented as full rank matrices with many tiny singular values (or eigenvalues for symmetry low-rank matrices).

Experiments Setup in This Section. In this section we reveal the neural dependencies empirically among some most popular neural networks, *i.e.*, ResNet-18, ResNet-50 [12], Swin-Transformer [16], and Vision-Transformer [10]. As a benchmark for massive category classification, we use ImageNet-1k [9], which includes examples ranging from 1,000 diverse categories, as the default dataset. Training details of those networks, and other necessary hyper-parameters to reproduce the results in this paper can be found in the Appendix.

2.1. Identifying Neural Dependencies through Covariance Lasso

We propose the Covariance Lasso (CovLasso) problem which will help us identify the neural dependencies in the network and play an essential role in this paper:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n,\boldsymbol{\theta}_i=-1} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\|\boldsymbol{\theta}^T f(\boldsymbol{x})\|_2^2] + \lambda\|\boldsymbol{\theta}\|_1. \tag{2}$$

Let $\boldsymbol{\theta}^*(\lambda)$ be the solution of Eq. (2) given hyper-parameter $\lambda > 0$, we can have the following observations

1. $\boldsymbol{\theta}^*(\lambda)$ will be a sparse n-dimensional vector, meaning many of its elements will be zero, due to the property of ℓ_1 penalty [20];
2. the prediction error $|f_i(\boldsymbol{x}) - \sum_{k=1}^s \boldsymbol{\theta}^*(\lambda)_{i_k} f_{i_k}(\boldsymbol{x})| = \|\boldsymbol{\theta}^{*T}(\lambda) f(\boldsymbol{x})\|_2^2$ will be very small for most $\boldsymbol{x}\sim p_{\text{data}}$, due to the property of minimization of expectation.

Combining these two observations, it is easy to find out the solution of Eq. (2) naturally induces the linear neural dependencies in Definition 1. Rigorously, by Markov inequality, if $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\|\boldsymbol{\theta}^T f(\boldsymbol{x})\|_2^2] \le \epsilon\delta$, we have

$$\begin{aligned} &\Pr(|f(\boldsymbol{x})_i - \sum_{j\neq i}\boldsymbol{\theta}_j f(\boldsymbol{x})_j| < \epsilon) \\ =&1 - \Pr(|f(\boldsymbol{x})_i - \sum_{j\neq i}\boldsymbol{\theta}_j f(\boldsymbol{x})_j| \ge \epsilon) \\ \ge&1 - \frac{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\|\boldsymbol{\theta}^T f(\boldsymbol{x})\|_2^2]}{\epsilon} \ge 1-\delta, \end{aligned} \tag{3}$$

so we can have the following theorem.

Theorem 1 *The solution to Eq. (2) satisfies Definition 1 for some small ϵ and δ and appropriate λ.*

The CovLasso problem is a convex problem; we can efficiently solve it by various methods like coordinate descent or subgradient descent [4]. Finding the neural dependencies for some category c_i is now transferring into solving the CovLasso problem under the constraint $\boldsymbol{\theta}_i = -1$.

Results. Fig. 2 reports some results of both within and between network neural dependencies acquired by solving Eq. (2). In the center we report the target category and in the surroundings we enumerate those categories that emerge neural dependencies with it. We show more results in the Appendix. For the cases in Fig. 2, Tab. 1 further reports the absolute and relative errors of predicting the logits of target categories using formula $f(\boldsymbol{x})_i \approx \sum_{k=1}^s \boldsymbol{\theta}_{i_k} f(\boldsymbol{x})_{i_k}$, and the corresponding classification accuracy on this category (using the replaced logits $(f(\boldsymbol{x})_1, \cdots, f(\boldsymbol{x})_{i-1}, \sum_{j\neq i}\boldsymbol{\theta}_j f(\boldsymbol{x})_j, f(\boldsymbol{x})_{i+1}, \cdots, f(\boldsymbol{x})_n)^T$ instead of $f(\boldsymbol{x})$), tested both on positive samples only and the full validation set of ImageNet. We can find that, as claimed by Definition 1, a small number of other categories (3 or 4 in the illustrated cases) are enough to accurately predict the network output for the target category. Moreover, the predictions are all linear combinations: for example, Fig. 2f tells that for almost every image $\boldsymbol{x}\sim p_{\text{data}}$, we have

$$\begin{aligned} \text{R50}(\boldsymbol{x})_{\text{hamster}} \approx\, &3.395 \times \text{S}(\boldsymbol{x})_{\text{broccoli}} \\ &+ 3.395 \times \text{S}(\boldsymbol{x})_{\text{guineapig}} + 3.395 \times \text{S}(\boldsymbol{x})_{\text{corn}}, \end{aligned} \tag{4}$$

where R50 denotes the ResNet-50 and S denotes the Swin-Transformer. We can achieve comparable classification performance if using the above linear combination to replace the logits output for category 'hamster' of ResNet-50. For both single models and two independently trained models with different architectures, we can observe clear neural dependencies. Future work may further investigate connections and differences in neural dependencies from different networks.

Peculiar Neural Dependencies. As we have mentioned before, the solved neural dependencies are not all that understandable for humans. Fig. 2 actually picks up a few peculiar neural dependencies for both within and between network dependencies. For example, the dependencies between 'jellyfish' and 'spot' in Fig. 2a, 'egretta albus' and 'ostrich' in Fig. 2b, 'basketball' and 'unicycle' in Fig. 2c, 'komondor' and 'swab' in Fig. 2d, 'bustard' and 'bittern' in Fig. 2e, and 'hamster' and 'brocoli' in Fig. 2f. This reveals the unique way of understanding image data of neural networks compared with human intelligence that has been unclear in the past [19, 30]. Further investigating those cases can be of general interests to future works in AI interpretability and learning theory, and potentially provide a new way to dig intrinsic information in image data.

2.2. What Brings Dependencies

After identifying the neural dependencies in deep networks, we are curious about why this intriguing phenomenon can broadly exist in different architectures. So we need a further understanding of the sources of it, which can be discovered through a careful analysis on Eq. (2). This section will reveal how a redundant covariance matrix for the terminal representations induces neural dependencies.

Figure 2. Neural dependencies in popular multi-class classification networks. (a;b;c) Within-network neural dependencies in ResNet18, ResNet50, Swin-Transformer and VIT-Transformer; (e;f) Between-network neural dependencies between ResNet50 and Swin-Transformer. Much more results can be found in Appendix.

Table 1. Prediction error and classification accuracy of neural dependencies in cases in Fig. 2. Both the error of logits prediction and the loss in classification accuracy are tiny. Much more results can be found in Appendix.

Metrics	ResNet-18	ResNet-50	Swin-T	VIT-S	R-50 $\rightarrow$ Swin-T	Swin-T $\rightarrow$ R-50
Abs Err	2.568	1.063	0.926	4.276	1.776	3.939
Rel Err (%)	18.7	6.8	10.4	29.7	20.7	21.1
Acc (Ori. Acc)	60.9 (61.0)	64.9 (64.9)	40.1 (40.1)	45.9 (45.9)	69.5 (69.5)	49.0 (49.2)
Pos Acc (Ori. Pos Acc)	72.0 (84.0)	92.0 (92.0)	94.0 (92.0)	96.0 (100.0)	94.0 (96.0)	94.0 (100.0)

Observe that $\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\|\boldsymbol{\theta}^T f(\boldsymbol{x})\|_2^2] = \boldsymbol{\theta}^T \text{Cov}\boldsymbol{\theta}$, where $\text{Cov} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})f(\boldsymbol{x})^T]$ is the (uncerntralized and unnormalized) covariance matrix of the terminal representations. Let $\boldsymbol{err}_i(\boldsymbol{\theta}) = \boldsymbol{\theta}^T\text{Cov}\boldsymbol{\theta}$ be the predicting error of using coefficient $\boldsymbol{\theta}$ for category c_i, the property of Lasso regression indicates that (see proof in Appendix) $\boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda))$ is continuous about λ and

$$\begin{aligned}\frac{\det[\text{Cov}]}{\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]} &= \boldsymbol{err}_i(\boldsymbol{\theta}^*(0)) \\ &\leq \boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda)) \leq \boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda')) \\ &\leq \boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda_{\max})) = \text{Cov}_i^i,\end{aligned} \tag{5}$$

where $\lambda \leq \lambda'$, and $\lambda_{max} = 2\|\text{Cov}_{[n]\setminus i}^i\|_\infty$ is the supremum of valid hyper-parameter λ, *i.e.*, $\boldsymbol{\theta}^*(\lambda) = -\boldsymbol{e}_i = (\underbrace{0,\cdots,0}_{i-1},-1,0,\cdots,0), \forall\lambda \geq \lambda_{\max}$, and $\boldsymbol{\theta}^*(\lambda) \neq -\boldsymbol{e}_i, \forall 0 \leq \lambda < \lambda_{\max}$.

Regardless of the sparsity, to yield neural dependency for the target category c_i, we expect a very small $\boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda))$. So if the lower bound $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0))$ is already far larger than $\epsilon\delta$, the predicting error can be too large to yield neural dependencies. Reversely, using the continuity of $\boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda))$ about λ, we can know that if the lower bound $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0))$ is very small, then there should be a small λ such that $\boldsymbol{err}_i(\boldsymbol{\theta}^*(\lambda))$ is also very small. Eq. (2) can then bring neural dependencies to category c_i. (This need to exclude a trivial case where the predicting error upper bound $\text{Cov}_i^i = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i^2]$ is already very small as it does not reveal any meaningful dependencies but that the network may be very unconfident about category c_i. While this is rare for well-trained networks, we leave the discussion of this case in Appendix.)

So to have neural dependencies, we require the term $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0))$ to be as small as possible. For term $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0))$ we can have the following observations from two different perspectives (see Appendix for deduction):

1. Information Volume: $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0)) = \frac{\det[\text{Cov}]}{\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]} = \frac{\text{Vol}(\text{Cov})}{\text{Vol}(\text{Cov}_{[n]\setminus i}^{[n]\setminus i})}$ measures the ratio between the n-dimensional volumes of the parallelotope Cov and the $n-1$ dimensional volumes of Cov removing the i-th row and i-th column; if assume Gaussian distributions of random variable $f(\boldsymbol{x}), \boldsymbol{x} \sim p_{\text{data}}$, they are also the normalizing constants of the probability density of the terminal representations with and without the i-th category; this term measures the information loss while removing the i-th category and is small if the i-th row and i-th column of Cov carry little information and are redundant;
2. Geometry: $\boldsymbol{err}_i(\boldsymbol{\theta}^*(0)) = \frac{\det[\text{Cov}]}{\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]} = (\sum_{j=1}^n \frac{\boldsymbol{\alpha}_j^2}{\sigma_j^2})^{-1}$ which will be small if some $\boldsymbol{\alpha}_j$ corresponding to tiny σ_j^2 is large, where $\sigma_1^2 \geq \cdots \geq \sigma_n^2$ are the eigenvalues of Cov and $\boldsymbol{q}_1,\cdots,\boldsymbol{q}_n$ are the corresponding eigenvectors, $\boldsymbol{\alpha}_j = \langle \boldsymbol{e}_i, \boldsymbol{q}_j\rangle, j \in [n]$; this further means that the i-th coordinate axis is close to the null space (linear subspace spanned by eigenvectors corresponding to tiny eigenvalues) of the covariance matrix Cov, which suggests the i-th category is redundant geometrically.

Let $\frac{\det[\text{Cov}]}{\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]}$ be the metric for redundancy of category c_i, both perspectives lead to the same conclusion that:

Redundancy of the target category c_i in the terminal representations brings it neural dependencies.

Remark 2 *Unfortunately, though it can help us understand the intrinsic mechanism that brings neural dependencies, this principle is only intuitive in practice—we cannot accurately calculate the value $\frac{\det[\text{Cov}]}{\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]}$ in most cases due to numerical instability.* $\det[\text{Cov}_{[n]\setminus i}^{[n]\setminus i}]$ *tends to have some tiny singular values (smaller than* $1\text{e}-3$*), making the quotient operation extremely sensitive to minor numerical errors in computation, and thus often induces* NaN *results.*

2.3. What Brings Sparsity

The last section omits the discussion of sparsity, which we want to study carefully in this section. We want to find a value that estimates whether two categories have neural dependencies, which we will show later is the (uncerntralized) covariance between the logits for two different categories.

The sparsity property, *i.e.*, whether category c_j is involved in the neural dependencies with c_i, can be identified by the KKT condition of Eq. (2). Let $\hat{\text{Cov}} = \text{Cov}_{[n]\setminus i}^{[n]\setminus i}$, $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{[n]\setminus i}$, $\hat{\boldsymbol{b}} = \text{Cov}_{[n]\setminus i}^i$, and $\hat{j} = j + \mathbf{1}_{(j>i)}$ such that $\hat{\boldsymbol{\theta}}_j = \boldsymbol{\theta}_{\hat{j}}$, then Eq. (2) can be transferred into

$$\min_{\hat{\boldsymbol{\theta}}\in\mathbb{R}^{n-1}} \hat{\boldsymbol{\theta}}^T\hat{\text{Cov}}\hat{\boldsymbol{\theta}} - 2\hat{\boldsymbol{b}}^T\hat{\boldsymbol{\theta}} + \lambda\|\hat{\boldsymbol{\theta}}\|_1. \tag{6}$$

By KKT conditions, the optimal value is attained only if

$$\mathbf{0} \in \hat{\text{Cov}}\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}} + \frac{\lambda}{2}\partial\|\hat{\boldsymbol{\theta}}\|_1. \tag{7}$$

and the sparsity can be estimated by the following proposition (see detailed deduction in Appendix)

$$|\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}}_j| < \frac{\lambda}{2} \Rightarrow \hat{\boldsymbol{\theta}}^*(\lambda)_j = 0, j \in [n-1]. \tag{8}$$

This means that we can know whether two categories admit neural dependencies by estimating $|\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}}_j|$. A surprising fact is that the term $|\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}}_j|$ can actually be estimated without solving Eq. (2), but using the slope of the solution path of the Lasso problem. By convexity of Eq. (2), the slope of Eq. (2) admits the following bound.

Theorem 2 *Let* $\hat{\text{Cov}} = \boldsymbol{Q}\boldsymbol{\Sigma}\boldsymbol{Q}^T$ *be the eigenvalue decomposition of* $\hat{\text{Cov}}$*, and* $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Sigma}^{1/2}\boldsymbol{Q}^T$*, then we have for* $\lambda', \lambda'' \in [0, \lambda_{\max}]$*,*

$$
\begin{aligned}
&|\frac{\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda') - \hat{\boldsymbol{b}}_j}{\lambda'} - \frac{\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda'') - \hat{\boldsymbol{b}}_j}{\lambda''}| \\
\le& \|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2|\frac{1}{\lambda'} - \frac{1}{\lambda''}|, j \in [n-1].
\end{aligned} \tag{9}
$$

Remark 3 *Using this theorem we can also get a finer estimation of the value of* $\boldsymbol{err}_i(\hat{\boldsymbol{\theta}}^*(\lambda))$ *than Eq. (5), see Appendix for detail.*

Using triangular inequality and the closed-form solution for $\lambda_{\max}$ ($\hat{\boldsymbol{\theta}}^*(\lambda_{\max}) = \boldsymbol{0}$), we have for $j \in [n-1]$,

$$
|\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}}_j| \le \lambda|\frac{\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda_{\max}) - \hat{\boldsymbol{b}}_j}{\lambda_{\max}}| \tag{10}
$$

$$
+\lambda|\frac{\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda) - \hat{\boldsymbol{b}}_j}{\lambda} - \frac{\hat{\text{Cov}}_j\hat{\boldsymbol{\theta}}^*(\lambda_{\max}) - \hat{\boldsymbol{b}}_j}{\lambda_{\max}}| \tag{11}
$$

$$
\le\lambda|\frac{\hat{\boldsymbol{b}}_j}{\lambda_{\max}}| + \lambda\|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2|\frac{1}{\lambda} - \frac{1}{2\|\hat{\boldsymbol{b}}\|_\infty}|. \tag{12}
$$

Thus if $\lambda|\frac{\hat{\boldsymbol{b}}_j}{\lambda_{\max}}| + \lambda\|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2|\frac{1}{\lambda} - \frac{1}{2\|\hat{\boldsymbol{b}}\|_\infty}| < \frac{\lambda}{2} \Leftrightarrow |\frac{\hat{\boldsymbol{b}}_j}{\|\hat{\boldsymbol{b}}\|_\infty}| < 1 - 2\|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2|\frac{1}{\lambda} - \frac{1}{2\|\hat{\boldsymbol{b}}\|_\infty}|$, we know that $\hat{\boldsymbol{\theta}}^*(\lambda)_j = 0$ and category $c_{\hat{j}}$ is independent (meaning not involved in the neural dependencies) with c_i.

Theorem 3 *When* $0 < \lambda < \lambda_{\max}$ *and* $\hat{j} \neq i$*, if*

$$
\begin{aligned}
&\frac{|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_{\hat{j}}]|}{\max_{s\neq i}|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_s]|} \\
<& 1 - 2\|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2|\frac{1}{\lambda} - \frac{1}{2\|\hat{\boldsymbol{b}}\|_\infty}|,
\end{aligned} \tag{13}
$$

then $\boldsymbol{\theta}^*(\lambda)_{\hat{j}} = 0$ *and category* $c_{\hat{j}}$ *is independent with* c_i*.*

High dimensional vectors are known to tend to be orthogonal to each other [5], thus if we assume $\boldsymbol{A}_j$ is nearly orthogonal to $\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}$, then $\|\boldsymbol{A}_j\|_2\|\boldsymbol{A}^{-T}\hat{\boldsymbol{b}}\|_2 \approx |\hat{\boldsymbol{b}}_j|$ and we can further simplify the above sparsity criterion as

Conjecture 1 *When* $0 < \lambda < \lambda_{\max}$ *and* $j \neq i$*, if*

$$
\begin{aligned}
&|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]| < \frac{\lambda}{2} (\textit{equivalent to} \\
&\frac{|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]}{\max_{s\neq i}|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_s]|}| < \frac{\lambda}{\lambda_{\max}}),
\end{aligned} \tag{14}
$$

then $\boldsymbol{\theta}^*(\lambda)_j = 0$ *and category* c_j *is independent with* c_i*.*

In practice we find that this conjecture is seldom wrong. Combining with Theorem 3, they together tell us that the covariance of terminal representations has an important role in assigning neural dependencies: more correlated categories

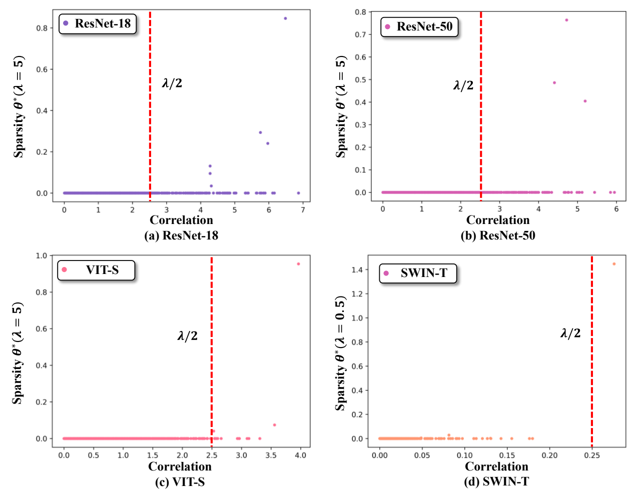


Figure 3. Relation between correlations and dependency coefficients.

tend to have neural dependencies, while weakly correlated categories will not have neural dependencies. They also describe the role of the hyper-parameter λ in Eq. (2): it screens out less correlated categories when searching neural dependencies, and larger λ corresponds to higher sparsity of dependencies. In conclusion, let $|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]|$ be the metric for correlations between category c_i and c_j, we can say that

Low covariance between categories in the terminal representations brings sparsity of dependencies.

Numerical Validation. We validate the above principle, *i.e.*, Conject. 1 in Fig. 3. Each subfigure picks up one target category c_i and solves Eq. (2) to calculate the corresponding coefficients $\boldsymbol{\theta}^*_j, j \neq i$ for all the remaining 999 categories of the ImageNet. $\boldsymbol{\theta}^*_j = 0$ implies no neural dependency between category c_i and c_j, and vice versa. We plot the relation between the covariance of c_i, c_j, $|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]|$, and the corresponding dependency coefficient $\boldsymbol{\theta}^*_j$. We can clearly find out that a small correlation corresponds to no neural dependency. Specifically, when the correlation between c_i, c_j is smaller than $\frac{\lambda}{2}$, c_i and c_j admit no neural dependency. In most cases, the bar $\frac{\lambda}{2}$ does exclude a considerable amount of zero dependency categories, which makes it a good indicator for the existence of neural dependency. This validates our principle for the source of sparsity.

Controlling Neural Dependencies. Conject. 1 also points out that we can disentangle neural dependencies by regularizing the covariance term, as tiny covariance indicates no neural dependency. We will discuss this later in Sec. 3.3.

2.4. Between Network Neural Dependencies

The general math property of the between network neural dependencies shows no essential difference from the within network ones. Let f, g be two different classification neural networks trained on p_{data} independently. We want to use the logits of f to predict the logits of the c_i category of g. Let $\tilde{f}(\boldsymbol{x}) = (f(\boldsymbol{x})_1, \cdots, f(\boldsymbol{x})_{i-1}, g(\boldsymbol{x})_i, f(\boldsymbol{x})_{i+1}, \cdots, f(\boldsymbol{x})_n)^T$, and $\tilde{\text{Cov}} = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\tilde{f}(\boldsymbol{x})\tilde{f}(\boldsymbol{x})^T]$, then we know that

- if $\frac{\det[\tilde{\text{Cov}}]}{\det[\tilde{\text{Cov}}]^{[n]\setminus i}_{[n]\setminus i}} = \frac{\det[\tilde{\text{Cov}}]}{\det[\text{Cov}^{[n]\setminus i}_{[n]\setminus i}]}$ is small, then category c_i of network g have neural dependencies with some other categories of network f;
- if $|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_j g(\boldsymbol{x})_i]|$ $(j \neq i)$ is small, then the c_j category of f is independent with the c_i category of g.

3. Potentials of Neural Dependencies

In this section, we discuss some interesting potentials and inspirations of neural dependencies in general scenarios of modern machine learning.

3.1. Visualization Neural Dependencies

We are curious about the intrinsic data relations revealed by neural dependencies. Specifically, if we have some base classes in the coordinate space, can we plot the relative position of the target classes that can be linearly decided by those classes through neural dependencies? Fig. 4 gives such an example for ResNet-50 in ImageNet. In the surroundings are 88 base categories and in the center are 10 target categories that can be linearly predicted by them using neural dependencies. The length of the arc between two categories gives their dependency coefficient. This result illustrates the relative relationship of different categories acknowledged by the neural network. It may be of potential interests to multiple domains like data relation mining, visualization, and interpretability of deep networks.

3.2. Generalizability

Now that the logits of one category can be well predicted by the logits of some others, we are curious about whether we can learn a cluster of base categories, and then predict new classes purely using linear combinations of the logits of those base categories. Especially, can the overall performance of this setting be comparable to training the baseline model on the whole dataset? This problem is of general interest to many machine learning scenarios. 1) *Incremental Learning.* In incremental learning [6,17,28] we need to learn to predict novel categories using a pretrained network on old categories. Typical methods will finetune the pretrained network in the new categories to achieve this goal, which then arouses concerns of damaging the knowledge of the old categories. Using our setting we can explore the potential of keeping the pretrained network unchanged and learning merely a small weight matrix to handle novel categories, which is cheap and efficient to train and deploy in various devices and realistic scenarios. 2) *Transfer Learning.* A similar but different occasion is transfer learning [18, 25, 26], where we seek to take advantage of knowledge of old domains to improve performance in new data domains. While categories are also instances of domains, our setting also explores a new way of knowledge transfer among domains. 3) *Representation Learning.* Our setting can partially reveal how representations [2] of base knowledge help classifications in out-of-distribution data (new categories). Future studies of this setting may reveal the source of the generalizability of neural networks from the perspective of neural dependencies.

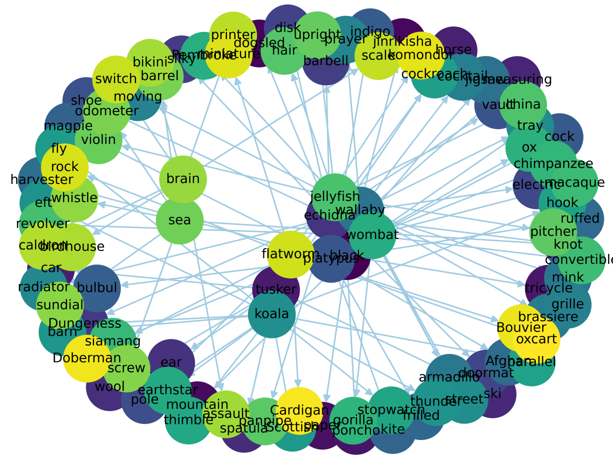


Figure 4. The graph visualization of neural dependencies in a pretrained ResNet-50. Please refer to Sec. 3.1 for detail.

To implement our setting, we may first train a deep classification network $f_{\text{base}} : \mathbb{R}^m \to \mathbb{R}^{n_1}$ on the n_1 base categories. Then we learn a coefficient matrix $\boldsymbol{\Theta} \in \mathbb{R}^{n_1 \times n_2}$ by fixing the parameters of f_{base} and minimizing the training loss of $f_{\text{new}} = f_{\text{base}}\boldsymbol{\Theta}$ on the training set of the new categories. We then concatenate $f_{\text{all}} = [f_{\text{base}}, f_{\text{base}}\boldsymbol{\Theta}]^T$ to form a new classifier for all the categories. We sample 500 samples per category from the training set of ImageNet-1K as our training data; the remains are used for constructing a balanced binary testing set we will use later. We evaluate the following three settings: 1) from 900 base classes to 100 new classes ($900 \to 100$), 2) from 950 base classes to 50 new classes ($950 \to 50$), and 3) from 999 base classes to 1 new class ($999 \to 1$) within a dataset. To approach a binary classification scenario, for $999 \to 1$ case we additionally test on 500 positive and negative sample pairs from the remained training set of ImageNet as the $999 \to 1(pos\&neg)$ setting. The baselines f_{baseline} are backbone models trained on the whole 1,000 category training data.

Table 2. Classification accuracy of baselines and learning new categories through neural dependencies (ours). While much simpler, learning new categories through neural dependencies barely loses accuracy. All figures are the mean of five independent runs.

Backbone	$900 \rightarrow 100$			$950 \rightarrow 50$			$999 \rightarrow 1$			$999 \rightarrow 1(pos\&neg)$		
	Baseline	*Ours*	*Impro*	*Baseline*	*Ours*	*Impro*	*Baseline*	*Ours*	*Impro*	*Baseline*	*Ours*	*Impro*
ResNet50	68.47±*0.25*	68.03±*0.89*	-0.44	68.47±*0.25*	68.45±*0.67*	-0.02	68.47±*0.25*	68.46±*0.41*	-0.01	60.70±*0.16*	61.50±*0.38*	+0.80
Swin-T	71.49±*0.14*	71.486±*0.17*	-0.004	71.49±*0.21*	71.578±*0.34*	+0.088	71.49±*0.09*	71.56±*0.13*	+0.07	76.20±*0.27*	78.00±*0.24*	+1.80

Table 3. Metrics of using (ours) and not using (baselines) the dependency regularization. All figures are mean of five independent runs.

Backbone	ImageNet Acc. (↑)			Dependency Coefficients (↓)			ImageNet-O AUPR (↑)		
	Baseline	*Ours*	*Impro*	*Baseline*	*Ours*	*Impro*	*Baseline*	*Ours*	*Impro*
ResNet18	69.83±*0.033*	70.12±*0.084*	+0.29	0.70	0.02	+0.68	15.15±*0.04*	15.48±*0.09*	+0.33
ResNet50	76.37±*0.25*	76.66±*0.13*	+0.29	1.10	$4.5e^{-4}$	+1.10	13.98±*0.05*	14.07±*0.02*	+0.09
Vit-S	80.67±*0.305*	81.52±*0.212*	+0.85	0.1	$3.1e^{-3}$	+0.1	28.54±*0.11*	31.14±*0.10*	+2.60
Swin-T	82.16±*0.046*	82.18±*0.062*	+0.02	0.39	0.01	+0.38	27.66±*0.08*	28.13±*0.06*	+0.47

Table 4. Classification accuracy in base (900) and new (100) categories separately. While much simpler, learning new categories through neural dependencies outperform baselines if only considering the performance in the new categories. All figures are the mean of five independent runs.

Method	ResNet-50		Swin-T	
	900	*100*	*900*	*100*
Baseline	67.43±*0.16*	68.87±*0.84*	71.28±*0.29*	70.73±*1.03*
Ours	68.65±*0.13*	71.06±*1.15*	71.50±*0.40*	72.47±*0.41*

Other details can be found in Appendix.

Experimental Results. We report the performance of f_{all} and f_{baseline} in Tab. 2, where we can find both settings (ours v.s. baselines) achieve comparable performance. While our setting requires training on only a small coefficient matrix, it consumes much less computation and time resources (less than 60% time consumption of the baseline in each epoch, see Appendix for detail) compared with the baselines. We further investigate how our setting performs in the new categories. Tab. 4 reports classification accuracy in the old 900 and new 100 categories of our setting and baselines (here we choose the class with maximum logits in the 900/100 categories as the prediction results). We can find that our setting significantly outperforms the baselines in the new classes. Both results reveal the power of neural dependencies in the generalizability of deep networks.

3.3. Robustness

As we have mentioned before, some neural dependencies are not that sensible for humans. We are therefore curious about whether cutting off them can help the network and improve robustness. Here we compare two cases, the baselines, and baselines finetuned by adding the regularization term $|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]|$ where c_i, c_j are the two categories that emerge irrational neural dependencies to cut off. We use two benchmarks, ImageNet-1K and ImageNet-O [13]. ImageNet-O consists of images from 200 classes that are unseen in ImageNet-1K, and is used to test the robustness on out-of-distribution samples. This ability is usually measured by the AUPR (*i.e., the area under the precision-recall curve*) metric [3]. This metric requires anomaly scores, which is the negative of the maximum softmax probabilities from a model that can classify the 200 classes. We train the baseline models for 90 epochs and our settings for 60 epochs of regular training followed by 30 epochs of finetuning using the regularization term $|\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[f(\boldsymbol{x})_i f(\boldsymbol{x})_j]|$. We manually choose one dependency to cut off for each case. Details are in Appendix.

Experimental Results. Tab. 3 reports the results. The regularization term does cut off the neural dependencies as the dependency coefficients are approaching zero after regularization. This then results in some improvements of performance in both ImageNet and ImageNet-O for all the backbones. While here we only cut-off one dependency for each case, we believe a thorough consideration of reasonable dependencies to maintain may benefit the network more. This reveals the connection between neural dependencies and the robustness of networks.

4. Conclusion

This paper reveals an astonishing neural dependency phenomenon emerging from learning massive categories. Given a well-trained model, the logits predicted for some category can be directly obtained by linearly combining the predictions of a few others. Theoretical investments demonstrate how to find those neural dependencies precisely, when they happen, and why the dependency is usually sparse, *i.e.* only a few instead of numerous of other categories related to one target category. Further empirical studies reveal multiple attractive potentials of neural dependencies from the aspects of visualization, generalization, and robustness of deep classification networks.

This work was supported by the National Key R&D Program of China under Grant 2020AAA0105702, National Natural Science Foundation of China (NSFC) under Grants 62225207 and U19B2038, and the University Synergy Innovation Program of Anhui Province under Grants GXXT-2019-025.

References

[1] Yusuf Aytar and Andrew Zisserman. Enhancing exemplar svms using part level transfer regularization. In *British Machine Vision Conference (BMVC)*, pages 1–11, 2012. 8

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, 2013. 7

[3] Kendrick Boyd, Kevin H Eng, and C David Page. Area under the precision-recall curve: point estimates and confidence intervals. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 451–466, 2013. 8

[4] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 3

[5] Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011. 6

[6] Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 233–248, 2018. 7

[7] Minshuo Chen, Yu Bai, Jason D Lee, Tuo Zhao, Huan Wang, Caiming Xiong, and Richard Socher. Towards understanding hierarchical learning: Benefits of neural representations. *Advances in Neural Information Processing Systems*, 33:22134–22145, 2020. 8

[8] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 1, 3

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3

[11] Ruili Feng, Kecheng Zheng, Yukun Huang, Deli Zhao, Michael Jordan, and Zheng-Jun Zha. Rank diminishing in deep neural networks. *arXiv preprint arXiv:2206.06072*, 2022. 2

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 2, 3

[13] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15262–15271, 2021. 8

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 1

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, 2021. 1, 2, 3

[17] Sudhanshu Mittal, Silvio Galesso, and Thomas Brox. Essentials for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3513–3522, 2021. 7

[18] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 22(10):1345–1359, 2009. 7

[19] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. 3

[20] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 3

[21] Ryan J Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011. 2

[22] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, Yasemin Altun, and Yoram Singer. Large margin methods for structured and interdependent output variables. *Journal of machine learning research*, 6(9), 2005. 8

[23] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*, 2022. 1

[24] Gang Wang, Derek Hoiem, and David Forsyth. Learning image similarity from flickr groups using stochastic intersection kernel machines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 428–435. IEEE, 2009. 8

[25] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018. 7

[26] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016. 7

[27] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 23965–23998, 2022. 1

[28] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 374–382, 2019. 7

[29] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022. 1

[30] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016. 3