Breaking Inter-Layer Co-Adaptation by Classifier Anonymization

Ikuro Sato [1] **Kohta Ishikawa** [1] **Guoqing Liu** [1] **Masayuki Tanaka** [2]

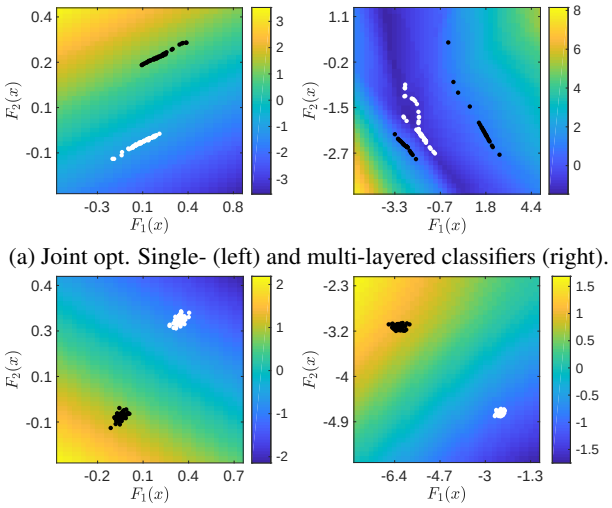
Abstract

This study addresses an issue of co-adaptation between a feature extractor and a classifier in a neural network. A naïve joint optimization of a feature extractor and a classifier often brings situations in which an excessively complex feature distribution adapted to a very specific classifier degrades the test performance. We introduce a method called Feature-extractor Optimization through Classifier Anonymization (FOCA), which is designed to avoid an explicit co-adaptation between a feature extractor and a particular classifier by using many randomly-generated, weak classifiers during optimization. We put forth a mathematical proposition that states the FOCA features form a point-like distribution within the same class in a class-separable fashion under special conditions. Real-data experiments under more general conditions provide supportive evidences.

1. Introduction

When specific signal patterns are repeatedly delivered by hidden neurons in a neural network during training, the network parameters are updated in a strongly tied way, or *co-adapted*, so that the network becomes vulnerable against small input perturbations (Hinton et al., 2012; Srivastava et al., 2014). To discourage co-adaptation, Hinton *et al.* proposed a method called Dropout that randomly deactivates neurons during training. Properties of Dropout training have been intensively studied (Helmbold & Long, 2015; Baldi & Sadowski, 2013; Gal & Ghahramani, 2016; Wager et al., 2013; Ren et al., 2016; Warde-Farley et al., 2013; Bengio et al., 2013); whereas there is a critique saying it does not necessarily yield co-adaptation prevention ability (Helmbold & Long, 2018).

Yosinski *et al.* studied the degrees of inter-layer co-adaptation by examining test performance that mid-layer features can yield (Yosinski et al., 2014). In part of their experiments, they split an end-to-end trained network into two blocks of layers, initialized the second-block parameters with random numbers, and trained the second block from scratch with the first-block parameters held fixed. They found that there are often cases where the secondary optimization degrades the test performance compared to the preceding primary joint optimization, despite that Dropout is adopted. In these cases, inter-layer co-adaptation (or *fragile co-adaptation*, in their words) happens between two blocks. Potentially, there is a chance that the secondary optimization finds the same minimum achieved by the primary optimization; however, in reality the chance rate is usually not high. Excessively complex feature distribution, like the ones shown in Fig. 1 (a), would be a major factor that induces inter-layer co-adaptation. Yosinski *et al.* also showed that inter-layer co-adaptation tends to cause negative effects



(a) Joint opt. Single- (left) and multi-layered classifiers (right).

(b) FOCA. Single- (left) and multi-layered classifiers (right).

Figure 1. Visualization of typical 2D features of two-class training data. (a) A naïve joint optimization of a feature extractor and a classifier; (b) FOCA (ours). Features in (b) form nearly point-like distributions per class, whereas those in (a) form more complex distributions. An L2 loss is minimized in each case. Black (white) dots indicate $+(-)1$-class data points, and the colored maps indicate the classifiers' outputs, where in (b) "averaged" outputs of 256 weak classifiers are shown.

[1]Denso IT Laboratory, Inc., Japan [2]National Institute of Advanced Industrial Science and Technology, Japan. Correspondence to: Ikuro Sato <isato@d-itlab.co.jp>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

in the cross-domain transfer.

Is there a way to fundamentally avoid inter-layer co-adaptation? Based on a thought that naïve joint optimization of a feature extractor and a classifier would result in unwanted co-adaptation between them, we seek a more fundamental approach to break the adhesion, rather than searching best-performing feature layers empirically (Yosinski et al., 2014; Kobayashi, 2017). The questions we try to answer in this work are: a) *Is it possible to train a feature extractor without inter-layer co-adaptation to a particular classifier?* b) *After such training, what kind of characteristics along with robustness against unwanted inter-layer co-adaptation does the feature extractor acquire?*

Regarding the first problem, we introduce a particular feature-extractor optimization method called Feature-extractor Optimization through Classifier Anonymization (FOCA) in Section 2. FOCA is designed so that the feature extractor does not explicitly co-adapt to a particular classifier. Instead, it uses randomly generated, weak classifiers during the feature-extractor training. FOCA belongs to a family of network randomization methods (Srivastava et al., 2014; Wan et al., 2013; Zeiler & Fergus, 2013; Huang et al., 2016; Singh et al., 2016), but is different from others in terms that FOCA does not employ a joint optimization of a feature extractor and a classifier. The classifier part is *anonymized* by marginalizing independently generated, weak classifiers; in this way explicit co-adaptation to a particular classifier is avoided.

Regarding the second problem, we obtained intriguing mathematical proposition (Section 3) and experimental evidences about simplicity of FOCA feature distributions. Let us suppose class-c features form a point-like distribution in a class-separable fashion. In that case a strong classifier for a partial dataset must be also strong for the entire dataset. This characteristics is largely confirmed for the FOCA features (Section 4.1). The distance between large-dataset solution and small-dataset solution in the classifier parameter space is indeed very small when FOCA is adopted (Section 4.2). Low-dimensional analyses of the FOCA features exhibit nearly point-like distributions (Section 4.3).

2. Optimization Method to Break Inter-Layer Co-Adaptation

In this section, we introduce FOCA that aims at training a feature extractor without inter-layer co-adaptation to a particular classifier. We first go over the basic joint optimization method, then introduce FOCA.

2.1. Joint Optimization: a Review

Let (x, t) be a pair of a d_I-dimensional input data and the corresponding d_O-dimensional target data, respectively. The training dataset $\mathcal{D}$ contains $n_{\mathcal{D}}$ such pairs. Feature extractor $F_\phi : \mathbb{R}^{d_I} \to \mathbb{R}^{d_F}$ transforms an input to a d_F-dimensional feature with parameter set ϕ, and classifier $C_\theta : \mathbb{R}^{d_F} \to \mathbb{R}^{d_O}$ transforms a feature to a d_O-dimensional output vector with parameter set θ. A joint optimization problem is given as

$$(\phi^\star, \theta^\star) = \arg\min_{\phi,\theta} \frac{1}{n_{\mathcal{D}}} \sum_{(x,t)\in\mathcal{D}} L\left(C_\theta(F_\phi(x)), t\right), \quad (1)$$

where $L(\cdot, t) : \mathbb{R}^{d_O} \to \mathbb{R}$ defines the sample-wise loss between the network output and the target.

When SGD training is naïvely applied, at each step the classifier is updated so as to become more discriminative for the presented features, no matter how complex the feature distribution is. The feature extractor, on the other hand, is updated so that the classifier at that moment becomes stronger, no matter how complex the decision boundary is. The toy example in Fig. 1 (a) demonstrates such a case, where training results in excessively complex feature distribution.

2.2. Feature-extractor Optimization through Classifier Anonymization (FOCA)

Below, we introduce FOCA for optimizing a feature extractor without explicitly co-adapting to a particular classifier. The optimization problem is defined as

$$\phi^\star = \arg\min_{\phi} \frac{1}{n_{\mathcal{D}}} \sum_{(x,t)\in\mathcal{D}} \mathbb{E}_{\theta\sim\Theta_\phi} L\left(C_\theta(F_\phi(x)), t\right), \quad (2)$$

where Θ_ϕ represents a predefined distribution function of *weak classifiers* for a given parameter set ϕ, and $\mathbb{E}_{\theta\sim\Theta_\phi}$ represents the expectation value over $\theta \sim \Theta_\phi$. The feature extractor is optimized with respect to a *set* of weak classifiers that are independently sampled from Θ_ϕ and thus is not able to co-adapt to a particular classifier, as long as Θ_ϕ generates distinct weak classifiers.

The weakness of the discriminative power of $\theta \sim \Theta_\phi$ is essential in this formulation. If $\theta \sim \Theta_\phi$ is designed to be too strong for $\mathcal{D}$, its decision boundary likely becomes fairly complex during the training, and the feature extractor would update itself to better fit the complex decision boundary, resulting in a vicious cycle. On the other hand, if $\theta \sim \Theta_\phi$ is too weak or even adversarial, the optimization process would not converge.

The marginalization over weak classifiers likely prevents the feature distribution from becoming excessively complex. Even at the end of the optimization, there is generally a large number of distinct weak classifiers, and the feature extractor is optimized with respect to the *ensemble* of these weak classifiers. Although some of the weak classifiers may have excessively complex decision boundaries, marginalization over the classifier ensemble likely smoothens those

out. This likely yields a relatively simple decision boundary and reasonably strong classification power, as is essential in the classical Classifier Bagging algorithms (Breiman, 1996) and other ensemble learning algorithms (Hara et al., 2017; Zahavy et al., 2018). Therefore, the form of the feature distribution likely becomes simple as far as the feature-extractor's description ability allows.

There is some room in defining Θ_ϕ, and here we introduce a particular definition. Let

$$\Theta_\phi = \mathcal{U}(\{\theta_{\phi,b}; b = b_1, b_2, \cdots\}), \tag{3}$$

where $\mathcal{U}(s)$ is a discrete uniform distribution function for all elements in set s, and $\theta_{\phi,b}$ is a solution that minimizes a batch-wise loss function with a norm regularization,

$$\theta_{\phi,b} = \arg\min_\theta \frac{1}{n_b} \sum_{(x,t)\in b} L\left(C_\theta(F_\phi(x)), t\right) + \lambda\|\theta\|_2^2. \tag{4}$$

Here, batch b comprises n_b training samples that cover all classes, and $\lambda > 0$. We further assume that the classifier parameters are initialized with random numbers prior to optimizing; therefore, there is almost no chance of having continuity between $\theta_{\phi,b}$ and $\theta_{\phi+\delta\phi,b}$ for $\|\delta\phi\| \ll 1$.

A solution $\theta_{\phi,b}$, a strong classifier for the batch b, is not generally strong for the entire dataset $\mathcal{D}$ for given ϕ because it does not "see" training samples other than the ones in b. However, there is no guarantee that $\theta_{\phi,b}$ is always a weak classifier to $\mathcal{D}$ in a classical sense; that is, a weak classifier performs only slightly better than random guesses. Indeed, $\theta_{\phi,b}$ can even work adversarially to $\mathcal{D}$, meaning its accuracy is below the chance rate. But, for brevity, we simply call $\theta_{\phi,b}$ a "weak classifier" in this work.

The norm regularization term in Eq. (4) helps to avoid blowups during the feature-extractor training. The scale of $\theta_{\phi,b}$ can be very large without the regularizer when two feature vectors in b stand close to each other. In such a case instability likely occurs.

After a feature extractor is obtained by Eq. (2), the following secondary optimization using the entire dataset provides a final, single classifier.

$$\theta^\star = \arg\min_\theta \frac{1}{n_\mathcal{D}} \sum_{(x,t)\in\mathcal{D}} L\left(C_\theta(F_{\phi^\star}(x)), t\right). \tag{5}$$

Here, the classifier is trained with *fixed* features. Note that the classifier architecture in this secondary optimization can differ from the one used in the primary optimization.

Our method and meta-learning (Finn et al., 2017) share a following similarity, despite that the goals are different (co-adaptation prevention vs. transferable multi-task learing). Our feature extractor acts like task-generic base network, and our classifiers act like taskwise fine-tuned models.

Approximate minimization. Regarding the number of weak classifiers used in a single update of ϕ, it is impossible to prepare a complete set of possible weak classifiers due to the huge number of distinct batches of the same size. We must adopt approximation instead. Algorithm 1 [1] gives an approximate solution of $\phi^\star$ in two senses: 1) a single weak classifier is sampled from Θ_ϕ per ϕ-update instead of taking a marginalization over Θ_ϕ, and 2) Θ_ϕ is held fixed in the computation of gradients with respect to ϕ.

Algorithm 1 Approximate minimization in Eq. (2)

Input: total number of iterations T; number of classes C; number of class-c samples $n_c (c = 1, \cdots, C)$; number of samples per class for θ-update k; total number of samples $n_\mathcal{D}$; minibatch size for ϕ-update m; learning rate η

Begin
  Initialize ϕ by random variables.
  for $t = 1 : T$ **do**
    $I_c = [\mathrm{randi}(n_1, k), \cdots, \mathrm{randi}(n_C, k)]$
    $\theta = \arg\min_{\theta'} \sum_{i\in I_c} L\left(C_{\theta'}(F_\phi(x_i)), t_i\right) + \lambda\|\theta'\|_2^2$
    $I_f = \mathrm{randi}(n_\mathcal{D}, m)$
    $\phi \leftarrow \phi - \frac{\eta}{m} \sum_{i\in I_f} \partial L\left(C_\theta(F_\phi(x_i)), t_i\right)/\partial\phi$
  end for
End

Output: feature-extractor parameters $\phi^\star = \phi$

It is worth mentioning that there is another way of generating a reasonably weak classifier: to take a batch (which can be large) and then to optimize the batch-wise loss in an incomplete fashion by stopping after a relatively small number of iterations, say 20 times. This works fine, though the definition of $\theta_{\phi,b}$ becomes mathematically less clear.

3. Mathematical Property

We now show a proposition about the simplicity of FOCA feature distributions. It will be proven that under some special conditions any two samples have exactly the same features when target classes are the same, but have different features when target classes are different. Let us first introduce a lemma about implicit optimality for individual features, and then put forth the proposition.

Lemma 3.1. *Suppose that a multi-layered feature extractor with two restrictions is used:*

1) The activation function a satisfies

$$a : \mathbb{R} \to \mathbb{R}^+, \quad \frac{\partial a(z)}{\partial z} \neq 0. \tag{6}$$

2) The last layer is fully-connected.

If $\phi^\star$ simultaneously minimizes sample-wise losses

[1] In the pseudocode, $\mathrm{randi}(i, j)$ returns a j-dimensional vector with each element being a random variable $\sim \mathcal{U}(\{1, 2, \cdots, i\})$.

$L(C_\theta(F_\phi(x)),t)$ for all $(x,t)\in\mathcal{D}$, then,

$$\frac{\partial C_\theta}{\partial F_{\phi^\star}}\frac{\partial L(C_\theta(F_{\phi^\star}(x)),t)}{\partial C_\theta}=0,\ \forall(x,t)\in\mathcal{D}. \quad (7)$$

($\frac{\partial C_\theta}{\partial F_{\phi^\star}}$ is a short-hand notation for $\frac{\partial C_\theta(f)}{\partial f}\Big|_{f=F_{\phi^\star}(x)}$. A summation symbol over C_θ indices is ignored in Eq. (7).)

Proof. Let ϕ_ℓ be the parameter set of the last weight layer in the feature extractor, and let x_ℓ be its input. Then, the i-th element of the feature layer, which is fully-connected from the previous layer, is given as $F_\phi(i)=a(\sum_j\phi_\ell(i,j)x_\ell(j))$. Let $z=\sum_j\phi_\ell(i,j)x_\ell(j)$. Then, $\frac{\partial F_\phi(i)}{\partial\phi_\ell(i,j)}=\frac{\partial a(z)}{\partial z}x_\ell(j)\neq 0$, since $\frac{\partial a(z)}{\partial z}\neq 0$ and $x_\ell>0$. The inequality $\frac{\partial F_\phi(i)}{\partial\phi_\ell(i,j)}\neq 0$, the supposition $\frac{\partial L}{\partial\phi_\ell(i,j)}\Big|_{\phi=\phi^\star}=0,\forall(x,t)\in\mathcal{D}$, and the chain rule immediately leads Eq. (7). □

In the following discussion, we assume the conditions stated below hold.

(C1) A multi-layered feature extractor with two restriction is used: 1) The activation function satisfies Eq. (6); 2) The last layer is fully-connected.

(C2) The target values are $t\in\{t_1,t_2\}$ for all samples.

(C3) A sample-wise loss function of the form $\tilde{L}_{\phi,\theta}(x,t)=(C_\theta(F_\phi(x))-t)^2$ is adopted.

(C4) A linear classifier $C_\theta(F_\phi(x))=\bar{\theta}^\top F_\phi(x)+\theta^0$ is used.

(C5) $\Theta_\phi=\mathcal{U}(\{\theta_{\phi,b};b=b_1,b_2,\cdots\})$, where $\theta_b=\arg\min_\theta\sum_{(x,t)\in b}\tilde{L}_{\phi,\theta}(x,t)+\frac{1}{2}\lambda\|\bar{\theta}\|_2^2$, and $b_1,b_2,\cdots$ are distinct batches, each of which comprises one sample from t_1 class and one sample from t_2 class.

Proposition 3.2. *Suppose that $\phi^\star$ simultaneously minimizes the classifier-anonymized, sample-wise losses $\mathbb{E}_{\theta\sim\Theta_\phi}\tilde{L}_{\phi,\theta}(x,t)$ in a class-separable fashion for all $(x,t)\in\mathcal{D}$. Then, samples from the same class share the same features; i.e., $F_{\phi^\star}(x)=F_{\phi^\star}(x'),\forall x,x'\in\mathcal{X}_c$, but samples from different classes do not; i.e., $F_{\phi^\star}(x)\neq F_{\phi^\star}(x'),\ \forall x\in\mathcal{X}_c,\ \forall x'\in\mathcal{X}_{c'\neq c}$.*

Proof. Lemma 3.1 about the implicit optimality of individual features with respect to sample-wise losses yields,

$$\mathbb{E}_\theta\frac{\partial C_\theta}{\partial F_{\phi^\star}}\frac{\partial\tilde{L}_{\phi^\star,\theta}(x,t)}{\partial C_\theta}=0,\ \forall(x,t)\in\mathcal{D}, \quad (8)$$

where $\mathbb{E}_\theta$ is a short-hand notation for $\mathbb{E}_{\theta\sim\Theta_\phi}$. By taking the partial derivatives in Eq. (8), one obtains,

$$\left(\mathbb{E}_\theta\bar{\theta}\bar{\theta}^\top\right)F_{\phi^\star}(x)=\mathbb{E}_\theta\bar{\theta}(t-\theta^0),\ \forall(x,t)\in\mathcal{D}. \quad (9)$$

The singular-value decomposition of the matrix consisting of column vectors $\bar{\theta}_b$ sampled from $\Theta_{\phi^\star}$ yields $\left[\bar{\theta}_{b_1},\bar{\theta}_{b_2},\cdots\right]=USV^\top$, where diagonal elements of the positive diagonal matrix S consists of the singular values aligned in decreasing order. Then, $U^\top\bar{\theta}_b=[\bar{\theta}_b^{n\top},0,\cdots,0]^\top$, where $\bar{\theta}_b^n$ is the non-singular components of $\bar{\theta}_b$. Taking the non-singular part in Eq. (9),

$$\left(\mathbb{E}_\theta\bar{\theta}^n\bar{\theta}^{n\top}\right)F^n_{\phi^\star}(x)=\mathbb{E}_\theta\bar{\theta}^n(t-\theta^0),\ \forall(x,t)\in\mathcal{D}. \quad (10)$$

Here, $U^\top F_{\phi^\star}(x)=\left[F^{n\top}_{\phi^\star}(x),F^{s\top}_{\phi^\star}(x)\right]^\top$, where $F^n_{\phi^\star}(x)$ is the corresponding non-singular part, meaning $F^n_{\phi^\star}(x)$ and $\bar{\theta}^n$ sharing the same dimension. The matrix $\mathbb{E}_\theta\bar{\theta}^n\bar{\theta}^{n\top}$ is obviously invertible; therefore, Eq. (10) can be solved for $F^n_{\phi^\star}(x)$. It then tells us that (a): $F^n_{\phi^\star}(x)$ depends only on t and θ_b's. On the other hand, the minimum-norm solution $\bar{\theta}_b$ satisfies,

$$[(f_1-f_2)(f_1-f_2)^\top+\lambda\mathbb{I}]\bar{\theta}_b=(t_1-t_2)(f_1-f_2), \quad (11)$$

where $\mathbb{I}$ is the identity matrix and $f_{1(2)}=F_{\phi^\star}$ with the target $t=t_1(t_2)$ as a short-hand notation. Taking the non-singular part in Eq. (11), the minimum-norm solution $\bar{\theta}^n_b$ satisfies (b):

$$[(f^n_1-f^n_2)(f^n_1-f^n_2)^\top+\lambda\mathbb{I}]\bar{\theta}^n_b=(t_1-t_2)(f^n_1-f^n_2), \quad (12)$$

where superscript n denotes the non-singular part. Given the definition that $[\bar{\theta}^n_{b_1},\bar{\theta}^n_{b_2},\cdots]$ is full-rank, statements (a) and (b) do not contradict only if

$$\exists v\in\mathbb{R},\ \bar{\theta}^n_b=v,\ \forall b. \quad (13)$$

It is, $\bar{\theta}^n_b$ is one-dimensional and constant for all b. Then, statement (a) yields, $F^n_{\phi^\star}(x)=F^n_{\phi^\star}(x'),\forall x,x'\in\mathcal{X}_c$. Note that $F^n_{\phi^\star}(x)\neq F^n_{\phi^\star}(x'),\forall x\in\mathcal{X}_c,\forall x'\in\mathcal{X}_{c'\neq c}$; otherwise $\phi^\star$ is not a class-separable solution. Because $\theta^0_b=1/2\sum_{(x,t)\in b}(t-\bar{\theta}^{n\top}F^n_{\phi^\star})$, θ^0_b must be the same for all b. Since $\bar{\theta}_b=U[\bar{\theta}^{n\top}_b,0,\cdots,0]^\top$, $\bar{\theta}_b$ must be the same for all b also. The fact that the minimum-norm solutions are the same for all combinations of $t=t_1$ and $t=t_2$ data points tells that $F_{\phi^\star}(x)=F_{\phi^\star}(x'),\forall x,x'\in\mathcal{X}_c$ and $F_{\phi^\star}(x)\neq F_{\phi^\star}(x'),\forall x\in\mathcal{X}_c,\forall x'\in\mathcal{X}_{c'\neq c}$. □

According to this proposition, if the feature extractor has an enough representation ability under certain conditions, all the input data of class c are projected to a single point in the feature space in a class-separable way. The left side of Fig. 1 (b) visualizes 2D features of the toy data optimized by FOCA under the conditions (C1)-(C5)[2]. Features of the same class are confined in a vicinity, the size of which is much smaller than the distance between the class centroids. It is intriguing to observe such a "point-like" distribution

[2]Here $\theta_{\phi,b}$ is the *analytical* solution.

property, even though we do not explicitly impose a thing like maximization of the between-class scatter with respect to the sum of within-class scatters. Although we have not succeeded in proving for multi-layered classifier cases, the toy experiment still exhibits the point-like distribution property as well; see the right side of Fig. 1 (b).

4. Experiment

Let us first state our motivations for a series of experiments. We saw in Section 3 that the FOCA features obey a point-like distribution per class under the special conditions. The question we try to answer in this section is, *Do the FOCA features form a point-like distribution or some similar distribution under more realistic conditions?* Let us suppose that features form a point-like distribution. Then, following secondary classifier optimization with the feature extractor held fixed should yield a similar decision boundary no matter what subset of the entire dataset it learns, as long as all classes are covered. We indeed confirmed that high test performances are achieved by FOCA, even when the secondary optimization uses smallest possible partial datasets; namely, only one data from each class (see Section 4.1). When FOCA is used, the secondary optimization leads the classifier parameter vector to almost the same point regardless of the size of the partial dataset it uses (see Section 4.2). Lastly, low-dimensional analyses revealed that FOCA features projected onto a hypersphere form a nearly a point-like distribution in a class-separable fashion (see Section 4.3).

Datasets. We use the CIFAR-10 dataset, a 10-class image classification dataset having 5×10^4 training samples, and the CIFAR-100 dataset, a 100-class image classification dataset having the same number of samples (Krizhevsky & Hinton, 2009). Both datasets have similar properties except for the number of classes (10:100) and the number of samples per class (5000:500). The idea is to see how these differences affect feature distribution properties.

Methods. In each experiment, FOCA is compared with other methods below. **Plain**: a vanilla mini-batch SGD training. **Noisy** (Graves, 2011): the same training rule as in Plain, except that a zero-mean random Gaussian noise is added to each of the classifier parameters during training. **Dropout** (Hinton et al., 2012): adopted only to the classifier part. **Batch Normalization** (Ioffe & Szegedy, 2015): adopted to the entire architecture.

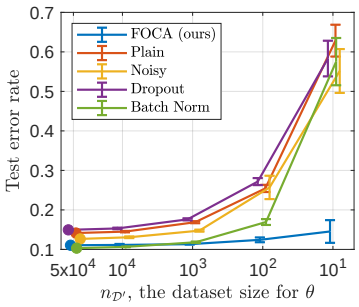
Dropout is claimed to reduce co-adaptation (Hinton et al., 2012), though there is a counterpoint to this view (Helmbold & Long, 2018). FOCA, Dropout and Noisy share the same characteristics in a sense that the classifier's descriminative power is weakened and the classifier ensemble is implicitly taken during training. Apart from FOCA, Dropout and Noisy employ joint optimization, and we are interested to see how this affects the robustness against inter-layer co-adaptation. Batch Normalization is included in comparison based on a thought that the way it propagates signals from one layer to the other may have some functionality mitigating inter-layer co-adaptation.

Architecture. The architecture that we use for the primary optimization in each CIFAR-10 experiment is the one introduced in (Lee et al., 2016), except that we replaced the last two layers by three fully-connected (FC) layers of the form: 4096(feature dim.)-β-β-10, where $\beta = 1024$ for Dropout and $\beta = 128$ otherwise. The architecture for the secondary optimization is 4096-128-128-10 for all methods. The architecture for the CIFAR-100 primary optimizations is VGG-16 (Simonyan & Zisserman, 2015), except that the last three FC layers are replaced by 512(feature dim.)-β-β-100, where $\beta = 512$ for Dropout and $\beta = 128$ otherwise. The architecture for the secondary optimization is 512-128-128-100 for all methods.

Training details. SGD with momentum is used in each baseline experiment. In each FOCA experiment, the feature-extractor part uses SGD with momentum, and the classifier part uses gradient descent with momentum. In each training, we tested a couple of different initial learning rates and chose the best-performing one in the validation. A manual learning rate scheduling is adopted; the learning rate is dropped by a fixed factor 1-3 times. The weak classifiers are randomly initialized each time by zero-mean Gaussian distribution with standard deviation 0.1 for both CIFAR-10 and -100. Cross entropy loss with softmax normalization and ReLU activation (Nair & Hinton, 2010) are used in every case. No data augmentation is adopted. The batch size b used in the weak-classifier training is 100 for the CIFAR-10 and 1000 for the CIFAR-100 experiments. The number of updates to generate θ is 32 for the CIFAR-10 and 64 for the CIFAR-100 experiments. Max-norm regularization (Srivastava et al., 2014) is used for the FOCA training, to stabilize the training. We found that the FOCA training can be made even more stable when updating the feature-extractor parameters u times for a given weak classifier parameters. We used this trick with $u = 8$ in the CIFAR-100 experiments.

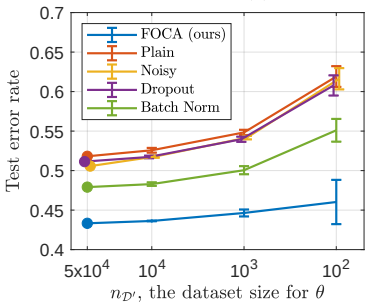
4.1. Test Performances of Classifiers Trained on Partial Datasets

Motivation. We are interested in two aspects. 1) Is the test performance produced by the primary joint optimization reproducible by the secondary classifier optimization? (FOCA is excluded here because it is not a joint optimization method.) 2) How does the test performance degrade when smaller datasets are learned in the secondary optimization? Remember that a point-like distributed features are expected to demonstrate little degradation.



	end-to-end	secondary @5x10^4
FOCA	-	11.06
Plain	13.51	14.15
Noisy	11.87	12.66
Dropout	13.43	14.95
Batch Norm	9.65	10.32

(a) CIFAR-10



	end-to-end	secondary @5x10^4
FOCA	-	43.33
Plain	44.04	51.81
Noisy	43.77	50.57
Dropout	43.24	51.14
Batch Norm	39.28	47.91

(b) CIFAR-100

Figure 2. Test error rates of classifiers trained on partial datasets. For each method, partial datasets $\mathcal{D}'$ are constructed $r(n_{\mathcal{D}'})$ times, where $r(5 \times 10^4) = 1$, $r(10^4) = 5$, $r(10^3) = 15$, $r(10^2) = 50$, and $r(10) = 150$. The solid lines indicate the mean values and the error bars indicate ± 1 standard deviations of test error rates. The numbers in the right side indicate the test error rates (%) of the end-to-end optimizations and the secondary optimizations for $n_{\mathcal{D}'} = 5 \times 10^4$ (also shown by the bullet points).

Experimental procedure. We trained feature extractors on the entire training dataset using FOCA and the other methods. All methods except for FOCA employ the end-to-end learning scheme. Then, we detach the feature extractors and the classifiers after learning. Next, for all methods, we fix the feature-extractor parameters and train classifiers from scratch by the orthodox backpropagation on *reduced datasets* $\mathcal{D}'$ of size $n_{\mathcal{D}'}$. For CIFAR-10 experiments, $n_{\mathcal{D}'} = 5 \times 10^4, 10^4, 10^3, 10^2, 10$ (10 is the smallest possible dataset size). For CIFAR-100 experiments, $n_{\mathcal{D}'} = 5 \times 10^4, 10^4, 10^3, 10^2$ (10^2 is the smallest possible).

Figure 2 now shows the test error rates vs. $n_{\mathcal{D}'}$.

Cases of $n_{\mathcal{D}'} = n_{\mathcal{D}}$. For all end-to-end optimization methods tested here, the test performances after the secondary optimizations at $n_{\mathcal{D}'} = 5 \times 10^4$ happen to be worse than corresponding end-to-end optimization results (see the right side of Fig. 2). This is one of the indications that inter-layer co-adaptation occurs to some degree in each of these joint optimization methods.

Cases of $n_{\mathcal{D}'} < n_{\mathcal{D}}$. For CIFAR-10, when $n_{\mathcal{D}'} \leq 10^3$, FOCA outperforms the other methods. To our surprise, the average test error rate of FOCA at $n_{\mathcal{D}'} = 10$ is 14.45%, which is indeed better than the test error rates of Dropout (14.95%) at $n_{\mathcal{D}'} = 5 \times 10^4$ Plain (14.50%) at $n_{\mathcal{D}'} = 10^4$, Noisy (14.75%) at $n_{\mathcal{D}'} = 10^3$, and Bach Normalization (16.93%) at $n_{\mathcal{D}'} = 10^2$. For CIFAR-100, FOCA outperforms the other methods at any $n_{\mathcal{D}'}$. The average error rate of FOCA at $n_{\mathcal{D}'} = 10^2$ is 46.03%, which is indeed better than the test error rate of all the other methods at $n_{\mathcal{D}'} = n_{\mathcal{D}} = 5 \times 10^4$. We think that a high test performance of FOCA for $n_{\mathcal{D}'} \ll n_{\mathcal{D}}$ is one indication of relatively simple form of feature distribution.

4.2. Approximate Geodesic Distances between Solutions

Motivation. We just saw that classifiers trained with the largest possible dataset ($n_{\mathcal{D}'} = n_{\mathcal{D}}$) and classifiers trained with the smallest possible partial dataset ($n_{\mathcal{D}'} = 10$ for CIFAR-10, $n_{\mathcal{D}'} = 100$ for CIFAR-100) exhibit similar test performances when the FOCA features are used. Let us call the former the large-dataset solution θ^{LD} and the latter the small-dataset solution θ^{SD}. The above observation drove us to investigate distances between θ^{LD} and θ^{SD}. The distance should be small when the features form a point-like distribution per class, and we expect FOCA has this characteristics. We employ the *approximate geodesic distance* here to take changes in the loss landscape into account. If θ^{LD} and θ^{SD} are virtually the "same" point with FOCA, we further expect that the test error rate is almost unchanged at any intermediate point between θ^{LD} and θ^{SD}. Since two neural networks having the same architecture and different parameters can produce the same output for an arbitrary input (Watanabe, 2009), we initialized networks with the same set of random numbers in the experiments conducted in this subsection.

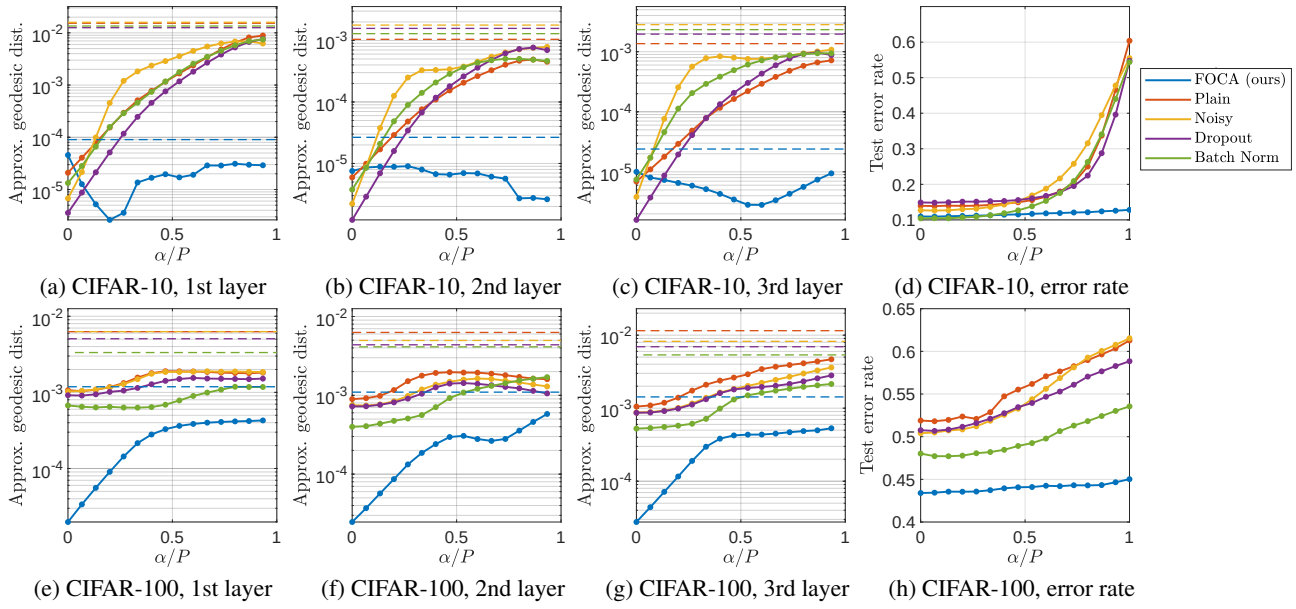
Experimental procedure. After optimizing a feature extractor on the full training dataset, θ^{LD} is optimized with features of the full dataset of size $n_{\mathcal{D}'} = n_{\mathcal{D}} = 5 \times 10^4$, and θ^{SD} is optimized with features of a smallest possible partial dataset; $n_{\mathcal{D}'} = 10$ for CIFAR-10 and $n_{\mathcal{D}'} = 100$ for CIFAR-100. To quantify the separation between θ_{LD} and θ_{SD}, we partition the straight line connecting θ_{LD} and θ_{SD} into P line segments of equal lengths in the parameter space; *i.e.*,

$$\theta^{\alpha} = \frac{\alpha\theta_{SD} + (P-\alpha)\theta_{LD}}{P}, \ \alpha = 0, 1, \cdots, P. \qquad (14)$$

We define in this article an approximate geodesic distance $d(\theta^{LD}, \theta^{SD})$ between θ^{LD} and θ^{SD} as

$$d(\theta^{LD}, \theta^{SD}) = \left[\sum_{\alpha=0}^{P-1} d(\theta^{\alpha}, \theta^{\alpha+1})^2 \right]^{\frac{1}{2}}. \qquad (15)$$

Here, $d(\theta^{\alpha}, \theta^{\alpha+1})$ is the distance between θ^{α} and $\theta^{\alpha+1}$ with



(a) CIFAR-10, 1st layer (b) CIFAR-10, 2nd layer (c) CIFAR-10, 3rd layer (d) CIFAR-10, error rate

(e) CIFAR-100, 1st layer (f) CIFAR-100, 2nd layer (g) CIFAR-100, 3rd layer (h) CIFAR-100, error rate

Figure 3. The approximate geodesic distances (a-c, e-g) between θ^{LD} and θ^{SD} and test error rates at θ^{α} (d, h). In (a-c, e-g), the solid lines indicate segment-wise distances $d(\theta^{\alpha}, \theta^{\alpha+1})$, $\alpha \in \{0, \cdots, P-1\}$, and the dashed lines indicate total distances $d(\theta^{LD}, \theta^{SD})$.

respect to the Fisher information metric $\mathcal{I}^{\alpha}$ evaluated at θ^{α},

$$d(\theta^{\alpha}, \theta^{\alpha+1})^2 = (\theta^{\alpha+1} - \theta^{\alpha})^{\top} \mathcal{I}^{\alpha} (\theta^{\alpha+1} - \theta^{\alpha}), \tag{16}$$

$$\mathcal{I}^{\alpha} = \mathbb{E}_{(x,t)\in\tilde{\mathcal{D}}} \left(\frac{\partial L_{\phi^\star,\theta}(x,t)}{\partial \theta} \right) \left(\frac{\partial L_{\phi^\star,\theta}(x,t)}{\partial \theta} \right)^{\top} \Bigg|_{\theta=\theta^{\alpha}},$$

where $\tilde{\mathcal{D}}$ is either $\mathcal{D}$ or a subset of $\mathcal{D}$ for ease of computation, and $L_{\phi^\star,\theta}(x,t)$ is a short-hand notation of $L(C_{\theta}(F_{\phi^\star}(x)), t)$. To compute a genuine geodesic distance, one needs to compute the sum of Fisher-metric distances between pairs of infinitesimally separated points along the curve that minimizes the squared sum. This is computationally infeasible; we instead approximate the curve by the straight line, as explained. In the experiment, we let $\tilde{\mathcal{D}}$ be a randomly chosen subset consisting of 5% of the entire training samples. We set $P = 15$. We evaluated $d(\theta^{LD}, \theta^{SD})$ layer by layer.

Figure 3 now shows the approximated geodesic distances and test error rates at θ^{α}.

Approximate geodesic distances. For CIFAR-10, FOCA exhibits some orders-of-magnitude, say 40-180 times, smaller distances $d(\theta^{LD}, \theta^{SD})$ than the other methods. For CIFAR-100, FOCA exhibits 3-9 times smaller distances than the other methods. We guess the reason why the differences are moderate for the CIFAR-100 cases is that point-like distribution is harder to obtain for CIFAR-100. Nevertheless, FOCA exhibit smallest approximate geodesic distances in all cases, and we think this is an implicit evidence that the distribution of the FOCA features is simple enough so that the discriminative function generated with $n_{\mathcal{D}'} = n_{\mathcal{D}}$ is virtually reproducible when $n_{\mathcal{D}'} \ll n_{\mathcal{D}}$.

Test error rates at θ^{α}. For FOCA, test error rates are almost constant at all θ^{α}. Together with the small $d(\theta^{LD}, \theta^{SD})$, two points θ^{LD} and θ^{SD} could be viewed as virtually the same point, when FOCA is used. For other methods, test error rates increase more rapidly toward $\theta^{SD} = \theta^{15}$.

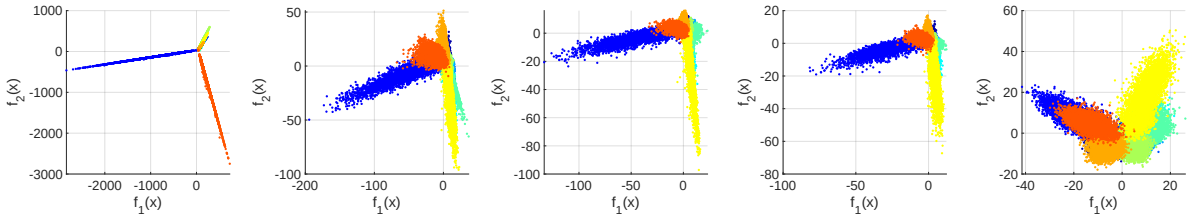
4.3. Low-Dimensional Properties

Motivation. We now use classical component analyses to clarify the low-dimensional structure of the FOCA features.

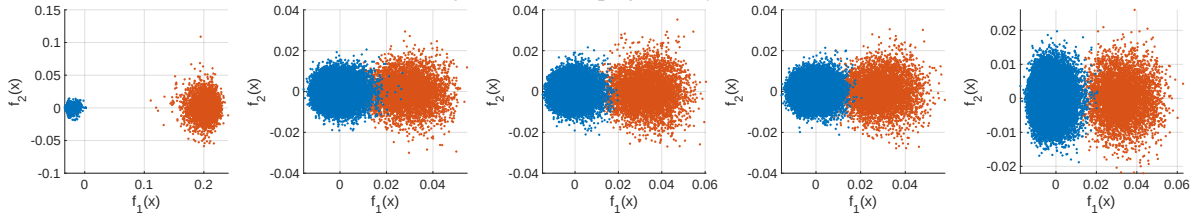
In this subsection we only show the CIFAR-10 results, because the CIFAR-100 results are qualitatively similar.

Principal Component Analysis (PCA). Figure 4 (a) shows scatter plots of training-data features projected by 2D bases of the PCA that is applied to all features. For a given class, the projected FOCA features look nearly one-dimensional, not a point-like, per class. This one-dimensional characteristics is probably due to the use of softmax normalization at the last layer, though we have no proof so far. In contrast, the projected features of the other methods clearly span two dimensions, roughly confined in an ellipse-like region, for a given class.

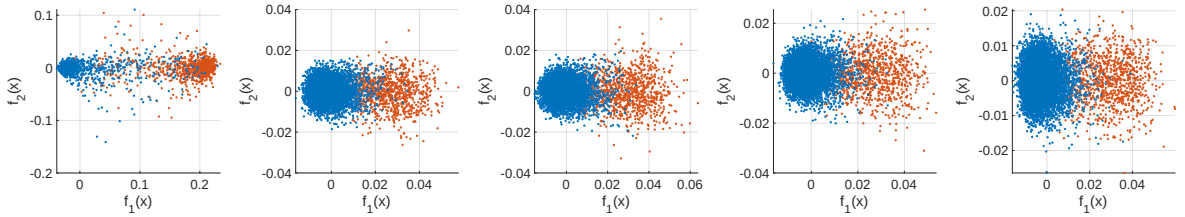
Linear Discriminant Analyses (LDA) with normalization. Next, we examine the LDA on features normalized to unit lengths. Normalization is taken based on the observation that the 2D features in Fig. 4 (a) are distributed



(a) Training-data features projected by the PCA bases.



(b) Training-data features first normalized to unit lengths then projected by the LDA bases, constructed on the class-1, normalized features vs. the rest of the normalized features.



(c) Test-data features first normalized to unit lengths then projected by the same LDA bases used in (b).

Figure 4. Two-dimensional visualization of 4096-dimensional, CIFAR-10 features. Methods are (from the left): FOCA (ours), Plain, Noisy, Dropout, and Bach Normalization. Colors indicate true classes.

Table 1. The results of the class-1-vs-rest LDA with normalization.

Method	Eigenvalue	Test error rate
FOCA (ours)	**247.28**	**2.01%**
Plain	5.74	2.71%
Noisy	7.49	2.86%
Dropout	5.81	2.78%
Bach Norm	7.28	2.43%

mostly along radial direction about a point close to the origin. Figure 4 (b) shows the 2D features that are normalized and projected by the LDA bases described above. Only class-1-vs-rest results are shown because no significant differences are observed when replacing class-1 by another class. Here, we can observe a remarkable differences; the projected FOCA features are linearly separable with a fairly large margin, compared to the characteristic scales of the class-1 distribution or of the rest-of-the-class distribution. The form of the feature distribution is close to point-like per class, somewhat similar to the observation in the toy experiment shown in Fig. 1 (b). In contrast, the other methods exhibit linearly non-separable feature distributions.

Linear separability by the LDA with normalization. The generalized eigenvalue computed in the LDA discussed above are given in Table 1. The values are the largest ratios of the between-class scatters to the within-class scatters after linear projections. FOCA exhibits orders of magnitude larger generalized eigenvalue than other methods. This supports the high level of linear separability of the FOCA features. Figure 4 (c) shows the normalized features of test data, projected by the same LDA bases, to see the generalizability. Table 1 also shows the smallest possible test error rates (class-1 vs. rest) by setting a threshold along the principal axis. FOCA yields the lowest test error rate.

5. Conclusion

A naïve joint optimization of a feature extractor and a classifier in a neural network often brings cases where both sets of parameters are tied in a so complex way that the classifier is irreplaceable without degrading the test performance. We introduced a method called Feature-extractor Optimization through Classifier Anonymization (FOCA), that is designed to break unwanted inter-layer co-adaptation. FOCA produces a feature extractor that does not explicitly adapt to a particular classifier. We gave a mathematical proposition that guarantees a simple form of feature distribution under special conditions; indeed, features form a point-like distribution in a class-separable way. Different kinds of real-dataset experiments under more general conditions provide supportive evidences.

References

Baldi, P. and Sadowski, P. J. Understanding dropout. In *Neural Information Processing Systems (NIPS)*, pp. 2814–2822, 2013.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*, 2013.

Breiman, L. Bagging predictors. *Machine Learning*, 24: 123–140, 1996.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

Graves, A. Practical variational inference for neural networks. In *Neural Information Processing Systems (NIPS)*, 2011.

Hara, K., Saitoh, D., and Shouno, H. Analysis of dropout learning regarded as ensemble learning. *arXiv:1706.06859*, 2017.

Helmbold, D. and Long, P. Surprising properties of dropout in deep networks. *Journal of Machine Learning Research*, 18:1–28, 04 2018.

Helmbold, D. P. and Long, P. M. On the inductive bias of dropout. *Journal of Machine Learning Research*, 16: 3403–3454, 2015.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Kobayashi, T. Sharing convnet across heterogeneous tasks. In *International Conference on Neural Information Processing (ICONIP)*, 2017.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical Report 4, 2009.

Lee, C.-Y., Gallagher, P., and Tu, Z. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, 2010.

Ren, Y., Zhang, L., and Suganthan, P. N. Ensemble classification and regression-recent developments, applications and future directions [review article]. *IEEE Computational Intelligence Magazine*, 11(1):41–53, 2016.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

Singh, S., Hoiem, D., and Forsyth, D. Swapout: Learning an ensemble of deep architectures. In *Neural Information Processing Systems (NIPS)*, 2016.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Wager, S., Wang, S., and Liang, P. S. Dropout training as adaptive regularization. In *Neural Information Processing Systems (NIPS)*, 2013.

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. Regularization of neural network using dropconnect. In *International Conference on Machine Learning (ICML)*, 2013.

Warde-Farley, D., Goodfellow, I. J., Courville, A. C., and Bengio, Y. An empirical analysis of dropout in piecewise linear networks. *arXiv:1312.6197*, 2013.

Watanabe, S. *Algebraic geometry and statistical learning theory*. Cambridge University Press, 2009.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Neural Information Processing Systems (NIPS)*, 2014.

Zahavy, T., Sivak, A., Kang, B., Feng, J., and Mannor, H. X. S. Ensemble robustness and generalization of stochastic learning algorithms. In *Workshop of International Conference on Learning Representations (ICLR)*, 2018.

Zeiler, M. and Fergus, R. Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representation (ICLR)*, 2013.