Regularizing Deep Neural Networks by Enhancing Diversity in Feature Extraction

Babajide O. Ayinde, *Student Member, IEEE*, Tamer Inanc, *Senior Member, IEEE*, and Jacek M. Zurada, *Life Fellow, IEEE*

***Abstract*—This paper proposes a new and efficient technique to regularize neural network in the context of deep learning using correlations among features. Previous studies have shown that over-sized deep neural network models tend to produce a lot of redundant features that are either shifted version of one another or are very similar and show little or no variations; thus resulting in redundant filtering. We propose a way to address this problem and show that such redundancy can be avoided using regularization and adaptive feature dropout mechanism. We show that regularizing both negative and correlated features according to their differentiation and based on their relative cosine distances yields network extracting dissimilar features, with less overfitting, and better generalization. The concept is illustrated with deep multilayer perceptron, convolutional neural network, Gated Recurrent Unit (GRU), and Long short-term memory (LSTM) on MNIST digits recognition, CIFAR-10, ImageNet, and Stanford Natural Language Inference data sets.**

***Index Terms*—Deep learning, feature correlation, feature clustering, regularization, cosine similarity, redundancy elimination.**

I. Introduction

THE expressiveness of deep neural networks, usually with huge number of trainable parameters, sometimes comes at a disadvantage when trained on limited amount of data due to their susceptibility to overfitting. To circumvent this problem, a plethora of regularization and initialization methods such as weight decay, dropout [1], and weight initialization [2] have been purported to ameliorate overfitting and convergence problems resulting from data scarcity and network size [3]. Moreover, recent advances in deep learning for image classification [4], [5], language processing [6], [7], speech synthesis and recognition [8–10] have been attributed to efficient regularization of randomly initialized deep and complex models trained with Stochastic Gradient Descent (SGD).

Over the last few decades, research focused on strategies for reducing overfitting and improving the capabilities of deep neural network. Examples of such strategies include Batch Normalization [11] that aims to minimize the internal covariance shift. Also, keeping similar variance at each layer's input and output of deep network using initialization has shown to preserve signal propagation and improve generalization [2], [12]. Orthonormal initialization coupled with output variance normalization has also been shown as decorrelating neural network's initial weights for better convergence [13].

Another important and popular paradigm for reducing overfitting is regularization. In general, the two most commonly used regularization paradigms utilize the hidden activations, weights or output distribution. The first family of regularization strategy aims to extenuate the model complexity by using weight decay [14], [15] to reduce the number of effective model parameters, or using dropout [1] to randomly drop hidden activations, or using DropConnect [16] to randomly drop weights during training. Even though these methods have shown improvements on generalization, they regularize in a manner that under-utilizes the capacity of the model. The second family of regularization methods focuses on improving the generalization without undermining the model's capacity. For instance, [17] presented pre-training algorithms to learn decorrelated features and [18] discusses decorrelated activations using incoherent training.

Other mechanisms that also fall in the second category are those that regularize the output distribution. In this sense, entropy-based regularizer with deterministic annealing was applied to train multilayer perceptrons for the purpose of avoiding poor initialization, local minima, and for improving model generalization [19]. Regularization has also been applied in form of label smoothing for estimating the marginalized effect of label-dropout during training. This, in effect, reduces overfitting by restricting the model from assigning full probability to each training sample and maintaining a reasonable ratio between the logits of the incorrect classes [20]. Label smoothing has also been achieved using a teacher model [21] instead of smoothing with uniform distribution as in [20]. Injecting label noise has equally been shown to have a tremendous regularizing effect [22]. Moreover, models with high level of overfitting have recently been shown to assign all output probability to a single class in the training set, thus giving rise to output distributions with low entropy - a phenomenon referred to as overconfidence [20]. Methods for effective regularization of overconfident networks have also been reported that penalize the confident output distribution [23].

Our observations and those of other researchers [3], [24], [25] indicate that over-sized deep neural networks are typically prone to high level of overfitting and usually rely on many redundant features that can be either shifted version of each

B. O. Ayinde is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY, 40292 USA (e-mail: babajide.ayinde@louisville.edu).

T. Inanc is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY, 40292 USA, (e-mail: t.inanc@louisville.edu).

J. M. Zurada is with the Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY, 40292 USA, and also with the Information Technology Institute, University of Social Science,Łódz 90-113, Poland (Corresponding author, e-mail: jacek.zurada@louisville.edu).

This work was supported by the NSF under grant 1641042.

other or be very similar with little or no variations. For instance, this redundancy is evidently pronounced in features learned by popular deep neural network architecture such as AlexNet [26] as emphasized in [3], [27]. To address this redundancy problem, layers of deep and/or wide architectures have to be trained under specific and well-defined sets of constraints in order to remove this limitation during training.

The most closely related work to ours is the recently introduced regularization technique known as OrthoReg [3] that locally enforces feature orthogonality by removing interference between negatively correlated features. The key idea addressed is to regularize positively correlated features during training. In effect, OrthoReg reduces overfitting by enforcing higher decorrelation bounds on features. Our algorithms, on the other hand, aim at regularizing both negatively and positively correlated features according to their differentiation and based on their relative cosine distances. This way we only penalize features that are correlated above a certain correlation threshold, thus strengthening feature diversity as training progresses. This approach affords the flexibility of choosing the absolute correlation bound. Hence, the proposed method leads to elimination of redundancy and better generalization.

Other related work is [28], which aims at training neural networks for classification with few training samples by constraining the hidden units to learn class-wise invariant features, with samples of the same class having the same feature representation. It is remarked that our methods have the flavor of the two aforementioned families of regularization in the sense that we aim to improve generalization without undermining the model's capacity by bounding the pairwise correlation of features and at the same time temporarily drop redundant feature maps during training.

The problem addressed here is four-fold: (i) we propose an optimized algorithm that inhibits learning of redundant filters, thereby enforcing the extraction of diverse features (ii) we further propose to use hierarchical agglomerative clustering (HAC) to drop activations (or feature maps) of redundant features during training, (iii) we propose a heuristic that eliminates the computational overhead introduced by HAC for very deep and/or wide neural networks by using the pairwise feature correlation to compute the fraction of the feature maps to be dropped during training, and lastly (iv) we show that the proposed regularization methods improve state-of-the-art models across many benchmark learning tasks and datasets. The paper is structured as follows: Section II introduces the notion of diversity regularization for extracting dissimilar features during training. Section III introduces novel online redundant feature detection and dropout using agglomerative hierarchical clustering. Section IV discusses the experimental designs and presents the results. Finally, conclusions are drawn in Section V.

II. Enhancing Feature Diversity by enforcing Dissimilar Feature Extraction

Our objective is to enforce constraints on the learning process by simply encouraging diverse feature learning and preventing the extraction of redundant features that are very similar or shifted version of one another. A symptom of learning replicated or similar features is that two or more processing units extract very similar and correlated information. From an information theory standpoint, similar or shifted versions of filters do not add extra information to the feature hierarchy, and therefore should be possibly suppressed. In other words, the activation of one unit should not be predictable based on the activations of other units of the same layer. Enforcing feature dissimilarity in traditional way can be generally involved and would require computation of huge joint probability table and batch statistics which can be computationally intractable [3].

One tractable way of computing correlation between two features is by evaluating the cosine similarity measure (SIM_C) between them:

$$SIM_C(\mathbf{w}_1, \mathbf{w}_2) = \frac{< \mathbf{w}_1, \mathbf{w}_2 >}{\| \mathbf{w}_1 \| \| \mathbf{w}_2 \|} \tag{1}$$

where $< \mathbf{w}_1, \mathbf{w}_2 >$ is the inner product of arbitrary feature vectors $\mathbf{w}_1$ and $\mathbf{w}_2$. The similarity between two feature vectors corresponds to correlation between them, that is, the cosine of the angle between the feature vectors in the feature space. Since the entries of the vectors can take both negative and positive values, SIM_C is bounded by [-1,1]. It is 1 when $\mathbf{w}_1$=$\mathbf{w}_2$ or when $\mathbf{w}_1$ and $\mathbf{w}_2$ are identical. SIM_C is -1 when the two vectors are in exact opposite direction. The two filter vectors are orthogonal in feature space when SIM_C is 0. The corresponding distance measure is given as $D_C = 1 - SIM_C$.

A. Diversity Regularization

In order to minimize the extraction of redundant features during training, it is necessary to maximize the information encoded by each processing hidden units by incorporating a penalty term into the overall learning objective, here referred to as diversity regularization (divReg). The constraints induced as a result of diversity regularization term need to be reconciled with usual regularization through a judicious choice of appropriate penalty function. The diversity regularization cost (J_D) for a single layer of the deep network is thus defined as:

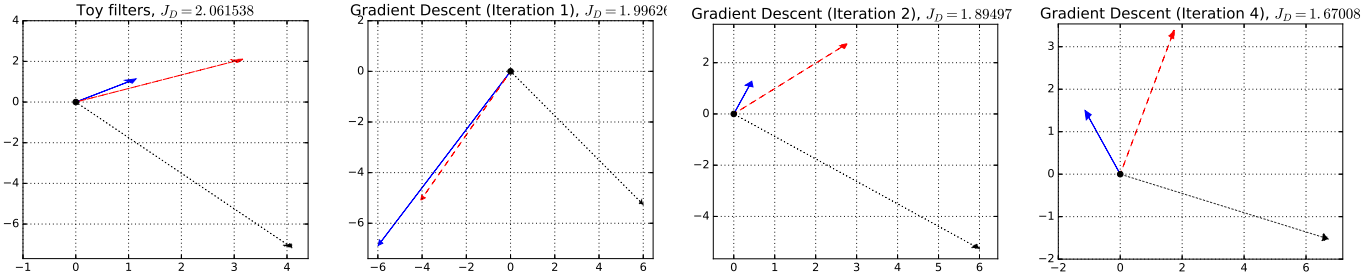
$$J_D(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{n'} \sum_{j=1, j \neq i}^{n'} m_{i,j} (SIM_C(w_i, w_j))^2 \tag{2}$$

where w_i are the weights connecting the activations of layer $l-1$ to i^{th} neuron of layer l, n' is the number of neurons in layer l. $m_{i,j}$ is a binary variable defined as in

$$m_{i,j} = \begin{cases} 1 & |SIM_C(w_i, w_j)| \geq \tau \\ 0 & i = j \\ 0 & otherwise \end{cases} \tag{3}$$

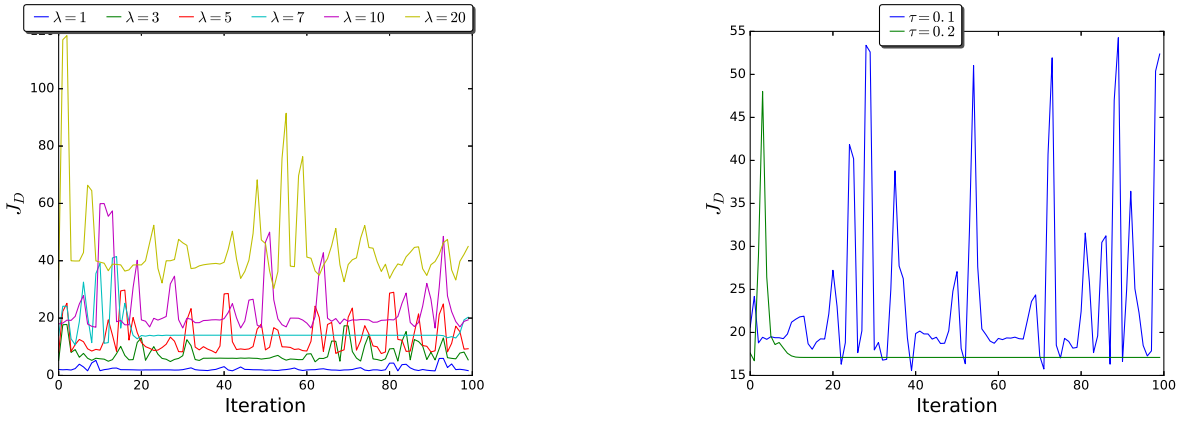
$0 \leq \tau \leq 1$ is an hyperparameter. It is worthy to note that self correlation of each feature vector w_i has been discarded in (3). Also, both negative and positive correlations above the threshold τ are taken into consideration. This implies feature pair with $|SIM_C|$ below τ will not be penalized.

It is important to also note the importance and relevance of τ in (3). Setting $\tau = 0$ results in orthogonal feature set and this is in most cases neither desirable nor practical because some features are still required to be shared. For instance, if



(a) (b) (c) (d)

Fig. 1: Illustration of effect of divReg with $\lambda = 10$ and $\tau = 0.1$ (a) on three toy filters in (b) iteration 1 (c) iteration 2 and (d) iteration 4.



(a) (b)

Fig. 2: Effect of (a) diversity penalty factor λ and (b) thresholding parameter τ on diversity regularization cost J_D

we consider a model trained on CIAFR-10 dataset [29] that has "automobiles" and "trucks" as two of its ten categories. If a particular lower-level feature describes the "wheel", then, it will not be out of place if two higher-level features describing automobile and truck share common feature that describes the wheel. The choice of τ determines the level of sharing allowed, that is, the degree of feature sharing across features of a particular layer. In other words, τ serves as a trade-off parameter that ensures some degree of feature sharing across multiple high-level features and at the same time ensuring features are sufficiently dissimilar.

By letting $\mathbf{\Phi} \in \mathbb{R}^{n \times n'}$ contain n' normalized filter vectors $\phi_i = {}^{w_i}/\sqrt{||w_i||^2}$ as columns, each with n elements corresponding to connections from layer $l-1$ to i^{th} neuron of layer l, then, J_D for all layers can be rewritten in a vectorized form as:

$$J_D(\phi) = \sum_{l=1}^{n_l} \left(\frac{1}{2} \sum_{i=1}^{n'} \sum_{j=1}^{n'} (\overset{(l)}{\Omega} \odot \overset{(l)}{M})^2 \right) \tag{4}$$

where $\overset{(l)}{\Omega} \in \mathbb{R}^{n' \times n'}$ denotes $\overset{(l)}{\mathbf{\Phi}}^{\mathbf{T}} \overset{(l)}{\mathbf{\Phi}}$ which contains the inner products of each pair of columns i and j of $\overset{(l)}{\mathbf{\Phi}}$ in each position i,j of Ω in layer l; $\overset{(l)}{M} \in \mathbb{R}^{n' \times n'}$ is a binary mask for layer l defined in (5); n_l is the number of layers to be regularized and $\odot$ is the element-wise multiplication of two matrices.

$$M(i,j) = \begin{cases} 1 & \tau \leq |\Omega(i,j)| \leq 1 \\ 0 & i = j \\ 0 & otherwise \end{cases} \tag{5}$$

In order to enforce diversity while training, the diversity regularization term (4) is added to the learning loss function $J(\theta; \mathbf{X}, \mathbf{y})$, where θ comprises of network's weights ($\mathbf{W}$) and biases ($\mathbf{b}$); $\mathbf{X}$, $\mathbf{y}$ are the data matrix and label vector, respectively. The overall cost is then

$$J_{net} = J(\theta; \mathbf{X}, \mathbf{y}) + \lambda J_D(\phi) \tag{6}$$

where λ is the diversity penalty factor experimentally chosen to be 10. The weights are updated as below using the error backpropagation:

$$W^{(l)} = W^{(l)} - \xi \frac{\partial}{\partial W^{(l)}} J_{net} \tag{7}$$

$$b^{(l)} = b^{(l)} - \xi \frac{\partial}{\partial b^{(l)}} J_{net} \tag{8}$$

where $\xi > 0$ is the learning rate and the gradient of the loss function is computed as in (9).

$$\frac{\partial}{\partial W^{(l)}} J_{net} = \nabla_{W^{(l)}} J(\overset{(l)}{\theta}; \mathbf{X}, \mathbf{y}) + \lambda \nabla_{W^{(l)}} J_D(\phi) \tag{9}$$

and

$$\nabla_{W^{(l)}} J_D(\phi) = \overset{(l)}{\Phi}(\overset{(l)}{\Omega} \odot \overset{(l)}{M}) \tag{10}$$

B. Implications of imposing feature diversity

The graphical illustration of impact of diversity regularization on features is shown in Fig. 1. Since this illustration does not utilize training data to update feature matrix $W^{(l)}$ in Eq.(7), we thus set $\nabla_{W^{(l)}} J(\overset{(l)}{\theta}; \mathbf{X}, \mathbf{y}) \rightarrow 0$. The three 2D filters shown as vectors in Fig. 1a were synthesized for visual illustration and both τ and λ were set to 0.1 and 10, respectively. $J_D(\phi)$ as a result of three initial filters evaluates to 2.062 using Eq.(4) with $n_l = 1$. Making a step along the gradient reduces the diversity regularization cost to 1.996 as shown in Fig. 1b. Likewise, the updated features after second and fourth iterations of gradient descent resulted in diversity regularization cost of 1.895 and 1.67 as shown in Figs. 1c and d, respectively. It is observed that at every iteration, the optimizer is forced to find features that are less similar in order to minimize $J_D(\phi)$.

Another crucial observation is that the filter far away from others in feature space is less regularized and has little influence on the regularization of other filters. The effect of both diversity penalty factor λ and thresholding parameter τ on diversity regularization cost J_D is shown in Figs.2a and b, respectively. As expected, J_D increases as the value of λ is increased for $\tau = 0.1$. The effect of τ on J_D is also explored and it can be observed in Figs.2b that when $\tau = 0.1$, the features are regularized more aggressively due to more feature-pair having similarity exceeding this threshold value and leading to a situation whereby feature vectors are heavily updated in every iteration leading to fluctuations of J_D. In contrast, when $\tau = 0.2$ features are heavily updated in the first fifteen iterations and subsequently settles into a local optimum.

III. Online Redundant Filter Detection and Dropout

This section introduces the concept of online agglomerative hierarchical clustering of features for detecting and dropping of N_r redundant features and their maps during training originally introduced in [25], [30] for pruning redundant features in unsupervised pretraining. In this section, two dropout heuristics considered both aim at online detection of redundant features and:

1) **dropping of redundant features maps** in the forward pass during training. Here, clustering of features aims at automatically detecting the features whose activations/maps will be dropped in each training epoch.
2) **random dropping of N_r feature maps** during training.

The above two criteria are alternative approaches and they both aim at temporarily dropping a set of feature maps during training. The term redundant reflects a choice of a specific chosen measure, SIM_C and of the τ value.

A. Online Filtering Redundancy Dropout

The objective here is to dropout feature maps that have identical or very similar features in weight space according to well-defined similarity measure. Achieving this involves choosing suitable similarity measures to express the inter-feature distances between vectors ϕ_i that define the features. A number of suitable agglomerative similarity testing/clustering algorithms can be applied for localizing redundant features. Based on a comparative review, a clustering approach from [30], [31] has been adapted and reformulated for this purpose as shown in Algorithm 1. By starting with each feature vector as a potential cluster, agglomerative clustering is performed by merging the two most similar clusters C_a and C_b as long as the average similarity between their constituent feature vectors is above a chosen cluster similarity threshold denoted as $\overset{*}{\tau}$ [32–34]. $\overset{*}{\tau}$ is an hyperparameter that has to be set in order to achieve optimal performance. The pair of clusters C_a and C_b exhibits average mutual similarities as follows:

$$\begin{aligned} \overline{SIM_C}(C_a, C_b) = \frac{\sum_{\phi_i \in C_a, \phi_j \in C_b} SIM_C(\phi_i, \phi_j)}{|C_a| \times |C_b|} > \overset{*}{\tau} \\ a, b = 1, ...n'; \ a \neq b; \ \ i = 1, ...|C_a|; \\ j = 1, ...|C_b|; \quad and \quad i \neq j \end{aligned} \tag{11}$$

where $0 \leq \overset{*}{\tau} \leq \tau$ It is remarked that the above similarity definition uses the graph-based-group-average technique, which defines cluster proximity/similarity as the average of pairwise similarities (that is, the average length of edges of the graph) of all pairs of features from different clusters. In this work, other similarity definitions such as the single and complete links were also experimented with. Single link defines cluster similarity as the proximity between the two closest feature vectors that are in different clusters. On the other hand, complete link assumes that cluster proximity is the proximity between the farthest two feature vectors of different clusters. Group average proximity definition empirically yielded better performance compared to the other two definitions and thus, we report experimental results using average proximity approach.

The nitty-gritty of the redundant feature dropout procedure is detailed in Algorithm 1. It first initializes weights to small random numbers by following the method introduced in [2]. Training data are shuffled and split into batches in each epoch. The loss in (6) is computed on each batch of the training samples. The backpropagation algorithm computes the gradient of the loss with respect to all the model parameters. Weights and biases are updated using the update rules in (7) and (8), respectively. At the end of every epoch, the weights connecting the activations of layer $l-1$ to neurons of layer l are examined for possible similarity using (11). The objective here is to discover n_f clusters in the set of n' original weight vectors (or simply features), where $n_f \leq n'$. Upon detecting these distinct n_f clusters, a representative feature from each of these n_f clusters is randomly sampled without replacement and the remaining set of features are tagged as redundant (S_R). This process continues for prescribed number of epochs. The detection of redundant feature vectors is generally tractable especially from practical standpoint since the number of features in each layer is reasonably sized (mostly less than

Algorithm 1 Online Redundant Feature Dropout (divReg-1)

{The parameters are: B_S - the batch size, ξ - learning rate, n' - number of filters, τ - diversity regularization correlation threshold, and $\overset{*}{\tau}$ - filter clustering similarity threshold}
{θ is the vector of concatenated weights ($\mathbf{W}^{(l)}$) and biases ($\mathbf{b}^{(l)}$). Initialize θ from a normal distribution as proposed in [2]. Initialize dropout fraction α}
$\theta \leftarrow \theta_0$ {Initial weight and biases}
for prescribed number of epochs ($nbepoch$) **do**
  permute training samples
  for all batches of B_S train samples **do**
    $J_{net} \leftarrow$ loss on batch samples from eq. (6)
    $\Delta\theta \leftarrow$ compute gradient using eq. (7) and (8)
    {Make a step along the gradient}
    $\theta \leftarrow \theta - \xi\Delta\theta$
  end for
  {Compute the set of redundant features}
  $S_R \leftarrow FilterClustering1()$
  {Drop activation maps corresponding to features in S_R}
end for
function FILTERCLUSTERING1():
  Input: $\mathbf{W}^{(l)}$, $\overset{*}{\tau}$
  Scan for cluster(s) of vectors in $\mathbf{W}^{(l)}$ with $\overline{SIM_C} > \overset{*}{\tau}$
  {Randomly sample and tag one representative features from each of the n_f clusters as non-redundant}
  Output: Set of redundant features in $\mathbf{W}^{(l)}$;
end function

a thousand). For large enough network, approximate measure of redundancy measure will be discussed in Section III(B).

B. Online Redundancy-based Dropout

The complexity of agglomerative clustering in Algorithm 1 is $O((n')^2\log(n'))$, which might sometimes make it impractical to deploy in online settings (that is, during training) especially for large n' and l. For instance, clustering 1024 feature vectors empirically takes on average on our machine (see specs in the Section V) 12 seconds and this is executed at least once in every epoch. This amounts to at least additional $(12 * l * nbepoch)$ seconds of computational overhead, where l is the number of layers and $nbepoch$ is number of epochs. However, the computational overhead is practical for relatively shallow network architectures.

To circumvent this problem for very deep and wide networks, we propose Algorithm 2 to estimate the dropout fraction based on the number of feature pairs that are correlated above a set threshold $\overset{*}{\tau}$. It uses cosine similarity with thresholding mechanism to dynamically set the dropout fraction of conventional dropout regularizer. This incorporates the redundancy information in the dropout mechanism. It is worth motivating and mentioning that Algorithms 1 and 2 are alternative approaches and should be used independently. The main difference between these algorithms is that Algorithm 1 uses hierarchical agglomerative clustering to detect and drop out the exact redundant features in each epoch, while Algorithm 2 estimates the number of feature maps to be randomly dropped at each epoch. Computationally, the dropout fraction of layer l in each epoch in Algorithm 2 is computed as the mean of the upper (or lower) triangular part of matrix $\overset{*}{M}$ as in (12) below:

$$\alpha^{(l)} = \frac{\sum_{i=1}^{n'}\sum_{j=1}^{n'} \overset{*}{M}^{(l)}(i,j)}{(n')^2 - n'} \tag{12}$$

where

$$\overset{*}{M}(i,j) = \begin{cases} 1 & \overset{*}{\tau} \leq |\Omega(i,j)| \leq 1 \\ 0 & i = j \\ 0 & otherwise. \end{cases} \tag{13}$$

It must be noted that both Algorithms 1 and 2 are adaptive in the sense that they adapt accordingly in every epoch to varying number of redundant filters. Another crucial detail about Algorithm 2 is the initialization of α. We tried different initialization values [0, 0.25, 0.5, 0.75], and we found different value works best for different datasets as will be detailed in Section IV. Unlike conventional dropout [1] that randomly drops a fixed number of units throughout the training process, the number of units dropped during training using Algorithms 1 and 2 adapts accordingly as training progresses. In this paper, we denote training under the diversity regularization in (4) without dropout as divReg while training using Algorithms 1 and 2 is denoted as divReg-1 and divReg-2, respectively.

Each of divReg-1 and divReg-2 can be used in tandem with the diversity regularization introduced in the previous section. However, they could also be deployed as stand-alone regularization tools in which case the regularization term in (4) is discarded by setting $\lambda = 0$. It must be noted that when using any of these procedures in conjunction with diversity regularization term (when $\lambda \neq 0$), the width of the similarity bound [-τ, τ] must be chosen as large as possible to allow the detection of some similar features and also $\overset{*}{\tau} \leq \tau$.

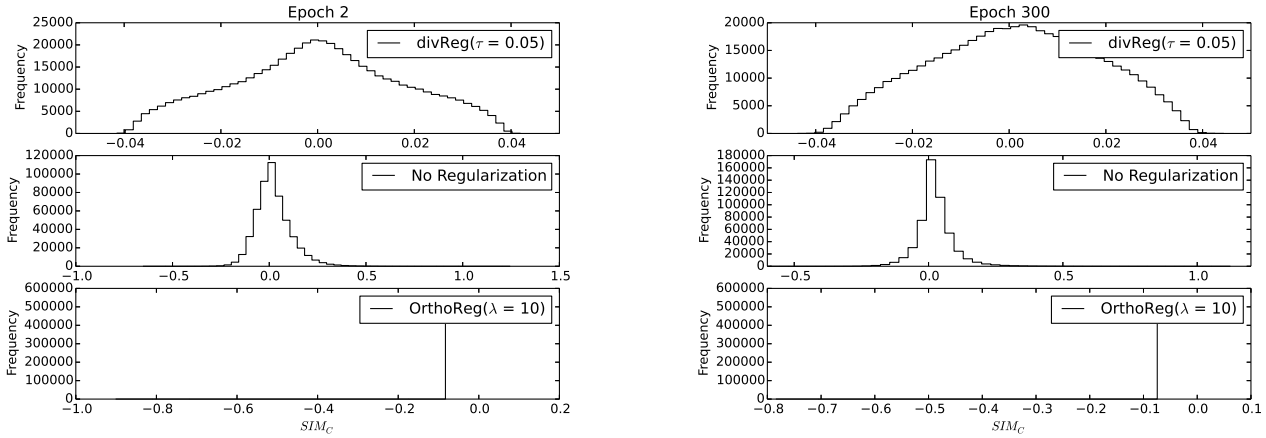
IV. EXPERIMENTS

Diversity regularization (divReg) was evaluated on MNIST dataset of handwritten digits [35], CIFAR-10 [29], and Stanford Natural Language Inference (SNLI) Corpus [36]. All experiments were performed on Intel(r) Core(TM) i7-6700 CPU @ 3.40Ghz and a 64GB of RAM running a 64-bit Ubuntu 14.04 edition. The software implementation has been in Keras library [1] with Tensorflow [37] backend on two Titan X 12GB GPUs. Implementation of divReg can be found in https://github.com/babajide07/Diversity-Regularization-Keras-Implementation.

A. Feature Evolution during Training

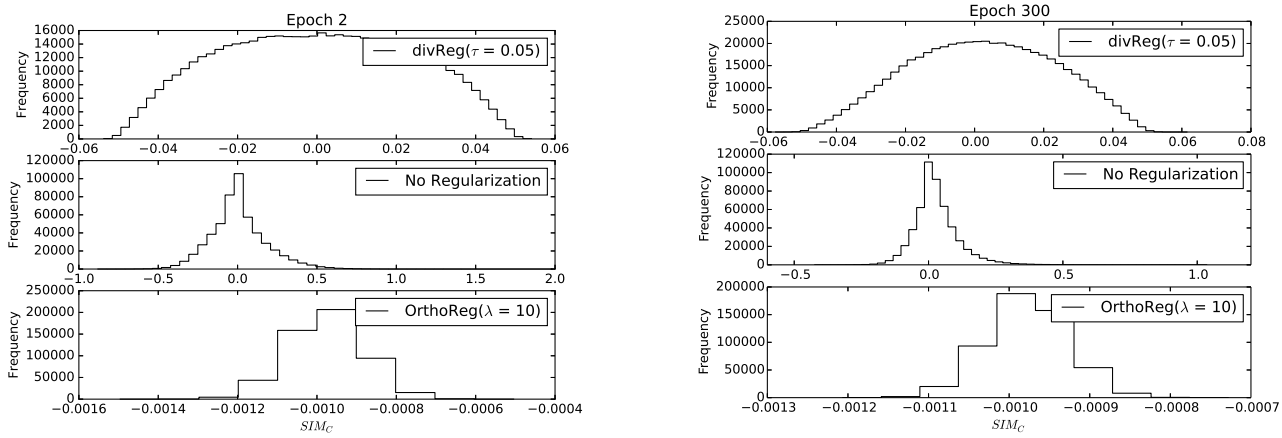
In the preliminary experiment, a multilayer perceptron with two hidden layers was trained using MNIST digits. The standard MNIST dataset has 60000 training and 10000 testing examples. Each example is a grayscale image of an handwritten digit scaled and centered in a 28 $\times$ 28 pixel box.

[1]https://keras.io/keras-the-python-deep-learning-library



(a) (b)

Fig. 3: The distribution of pairwise feature correlation ($\overset{(1)}{\Omega}$) in first hidden layer at (a) epoch 2 (b) epoch 300



(a) (b)

Fig. 4: The distribution of pairwise feature correlation ($\overset{(2)}{\Omega}$) in second hidden layer at (a) epoch 2 (b) epoch 300

Each layer has 1024 ReLU-activated hidden units and Adam optimizer [38] with batch size of 128 was used to train the model for 300 epochs and τ and ξ in divReg was both set to 0.05.

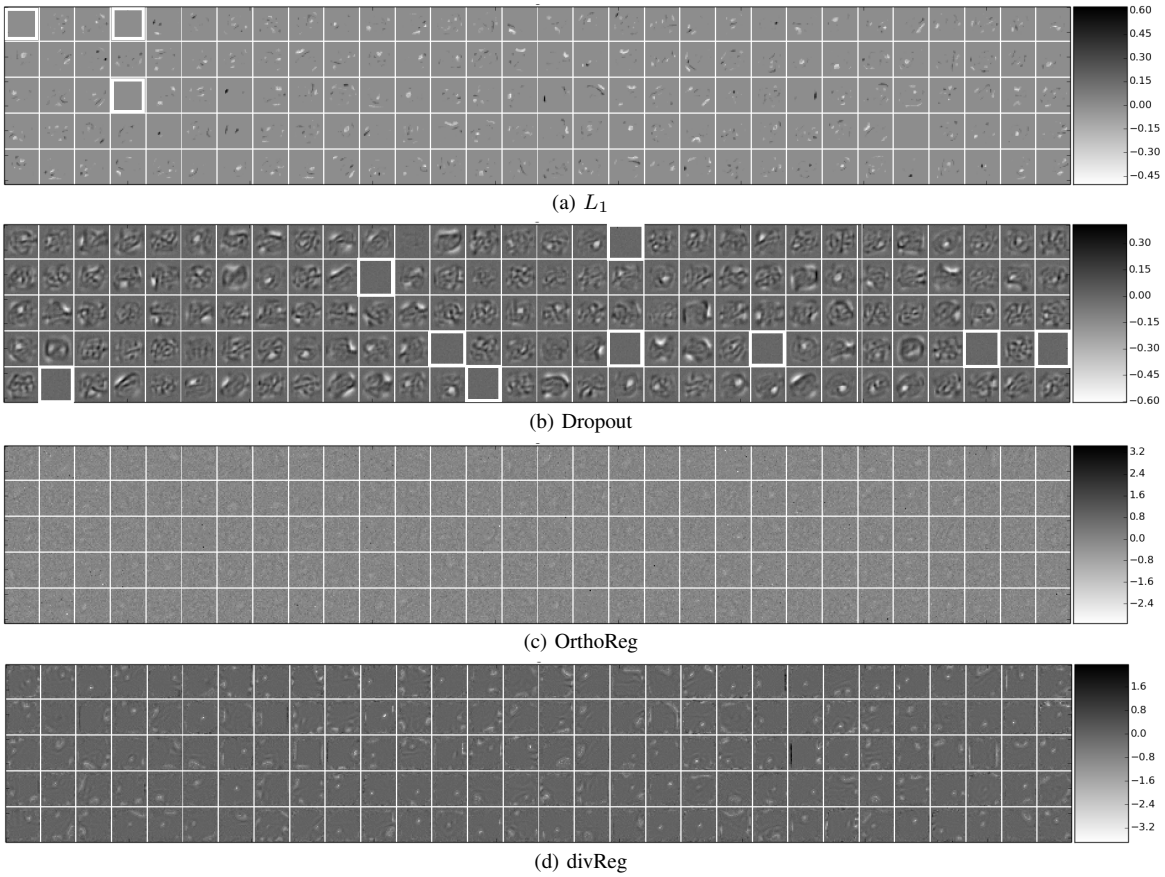
The hyperparameters of OrthoReg was set as reported in [3]. Figs. 3a and b show the distribution of pairwise correlation of first hidden layer features ($\overset{(1)}{\Omega} = \overset{(1)}{\mathbf{\Phi}}^{\mathbf{T}}\overset{(1)}{\mathbf{\Phi}}$) in the beginning and end of training, respectively. It can be observed that divReg was able to constrain the pairwise feature correlations between the desired bound (-0.05 and 0.05) compared to the highly correlated features extracted by unregularized counterpart. Although OrthoReg was able to eliminate all the positively correlated features using exponential squashing function, but it did so in a more rigid way which could lead to extraction of noisy features. Similarly in Figs. 4a and b, the pairwise feature correlations of the second hidden layer have been bounded by the set threshold for divReg, unconstrained for unregularized model, and negatively correlated with tight bound for OrthoReg.

Table I reports the performance of divReg along with four other regularization techniques. All reported results are average performance over 5 independent trials alongside with their standard deviation. The results are separated and compared on the basis of the class of the regularization technique. It can be observed that Dropout outperforms L_1 in terms of test error and also in terms of generalization (as measured by the test-train error gap). The performance improvement of Dropout technique in terms of generalization over L_1 is statistically significant as shown by the p-value. Similarly, the performance of divReg is better than both L_2 and OrthoReg with respect to test error and generalization. Another keen observation is that the test-train error gap for divReg and L_2 regularization is very similar as inferred by the p-value, but the improvement in absolute test performance does seem to be statistically significant.

For qualitative comparison, an autoencoder (AE) with 256

Regularization	train (%)	test (%)	test-train (%)	p-value
L_1	0.8791 $\pm$ 0.0947	1.9367 $\pm$ 0.0666	1.0575 $\pm$ 0.1411	0.0331
Dropout ($\alpha = 0.5$) [1]	0.4875 $\pm$ 0.0530	1.2250 $\pm$ 0.0071	0.7375 $\pm$ 0.0460	-
L_2	0.4550 $\pm$ 0.2280	1.7375 $\pm$ 0.1115	1.2825 $\pm$ 0.1505	0.0812
OrthoReg [3]	0.0167 $\pm$ 0.0212	1.6950 $\pm$ 0.0495	1.6783 $\pm$ 0.0283	0.0176
divReg	**0.0535 $\pm$ 0.0658**	**1.3150 $\pm$ 0.0212**	**1.2615 $\pm$ 0.0445**	-

TABLE I: Test-train error gap on MNIST



(a) L_1

(b) Dropout

(c) OrthoReg

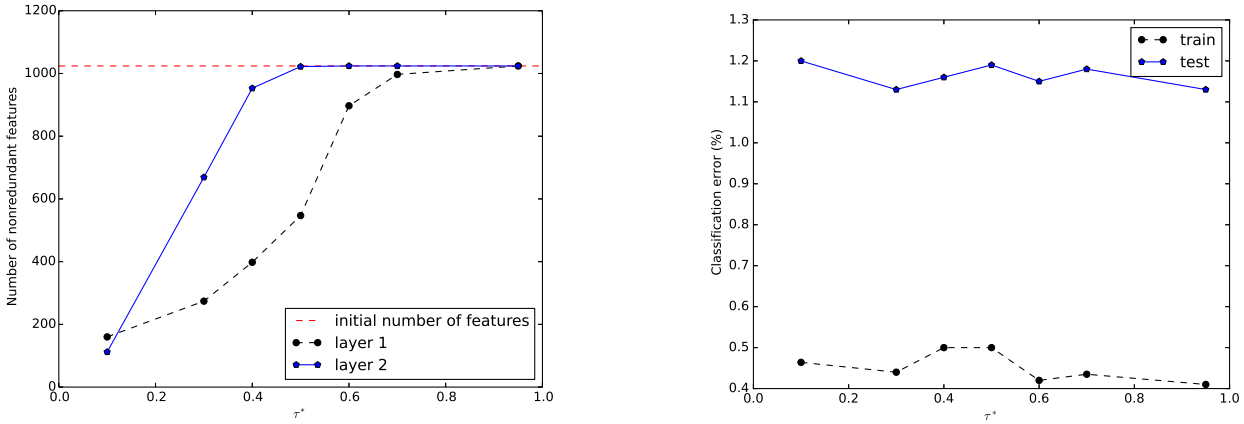
(d) divReg

Fig. 5: First 150 encoding features (left) learned from MNIST digit data set using (a) L_1, (b) Dropout (c) orthoReg, and (d) divReg. The range of weights are scaled and mapped to the graycolor map (right).

ReLU-activated encoding units and 784 sigmoid-activated decoding units was trained on raw pixels of MNIST digits. The weights were initialized randomly by sampling from Gaussian distribution with zero mean and standard deviation of 0.003 based on [12]. The AE model was regularized with L_1 (using decay parameter 10^{-4}), dropout ($\alpha = 0.5$), OrthoReg (using angle-of-influence of 10), and divReg (τ=0.4) regularization techniques and compared in terms of quality of the features learned. The features learned using each of the regularization method are shown in Fig. 5. One key observation is that L_1 and dropout regularization resulted in some dead filters as highlighted in Figs. 5a and b. Whereas, the representations learned with OrthoReg looks noisy compared to those learned with divReg regularization.

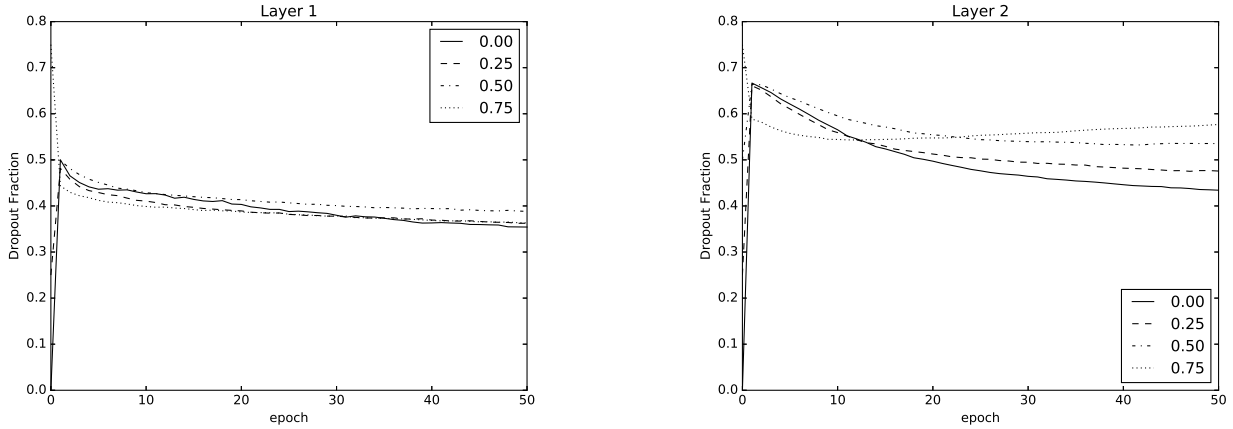
In a similar vein, we trained multilayer perceptron on MNIST with two ReLU-activated hidden layers and regularized by divReg-1. Number of hidden units per layer was set 1024 and parameters of the model was again optimized using Adam. As observed in Fig 6a, increasing τ^* yields increased number of dissimilar features because more and more features are considered occupying distinct clusters. Another interesting observation from this result is that earlier layers are generally prone to extracting more distinct features than latter layers with the same value of τ^*. Fig 6b is averaged over ten experiments to show the statistical significance. The error curves shown in Figs 6b reveal that networks trained using divReg-1 with τ^* 0.3 resulted in the lowest test-train error gap.

As mentioned earlier, a crucial step in achieving good performance with divReg-2 is not only in the choice of τ^* but also in the initialization of adaptive dropout fraction α. Figs. 7a and b show the evolution of dropout fraction for four different α initializations (0.0, 0.25, 0.50, 0.75) as training progresses for first and second layers, respectively. For MNIST dataset, α initialized to 0.75 generalizes better than other initializations as shown in Fig. 8 by the test-train classification error gap.



(a) (b)

Fig. 6: Performance of Multilayer Perceptron (with architecture 784-1024-1024-10) regularized using divReg-1 and trained on the MNIST dataset vs. threshold τ^*. (a) Number of nonredundant features for 1024 initial features. (b) percentage classification error



(a) (b)

Fig. 7: Evolution of dropout fraction (α) with divReg-2 using the MNIST dataset for four different initializations of α in (a) layer 1 and (b) layer 2

Model	**# layers**	**size**	**test** (%)
Unregularized [16]	2	800	1.40
DropConnect [16]	2	800	1.20
Dropout [1]	2	1024	1.28 $\pm$ 0.06
Dropout [1]	3	1024	1.25 $\pm$ 0.04
OrthoReg [3] (+ Dropout)	2	1024	1.38 $\pm$ 0.03
Label Smoothing [20] (+ Dropout)	2	1024	1.21 $\pm$ 0.06
Confidence Penalty [23] (+ Dropout)	2	1024	1.17 $\pm$ 0.06
DivReg-2	**2**	**1024**	**1.15 $\pm$ 0.03**
DivReg-1	**2**	**1024**	**1.10 $\pm$ 0.02**

TABLE II: Test error(%) on MNIST

However, both the test and train accuracies are not as good as that initialized to 0.5, which has the best train and test error trade-off.

B. *Image Classification*

In the first set of experiments, multilayer perceptron with two ReLU-activated hidden layers and a softmax layer for classification is again tested to ascertain if the enhanced ability to extract dissimilar features would lead to improved classification accuracy using MNIST dataset. 9000 images from MNIST data were randomly sampled from the training set as a held-out validation set for hyperparameter tuning and the network was retrained on the entire dataset using the best hyperparameter configuration. Adam optimizer with batch size of 128 was used for training the model for 300 epochs; τ and $\overset{*}{\tau}$ were set to 0.3 and 0.05 in divReg1 and divReg2, respectively. The dropout fraction α in divReg2 was initialized to 0.5. Every run of the experiment is repeated five times and averaged to combat the effect of random initialization. The classification errors of model trained with divReg were compared with state-of-the-art regularization techniques as detailed in Table II. It is observed

Model	# layers	# parameters	test (%)	# epochs
Residual CNN [5]	110	1.7M	13.63	300
Stochastic Depth Residual CNN [40]	110	1.7M	11.66	500
Densely Connected CNN (with Dropout) [39]	40	1.0M	7.00	300
Densely Connected CNN (with Label Smoothing [20] + Dropout)	40	1.0M	6.89	300
Densely Connected CNN (with Confidence Penalty [23] + Dropout)	40	1.0M	6.77	300
Densely Connected CNN-BC (with Dropout) [39]	100	0.8M	6.8 ($\pm$0.057)	150
Densely Connected CNN-BC (with OrthoReg [3] + Dropout)	100	0.8M	11.2 ($\pm$0.125)	150
Densely Connected CNN-BC (with DivReg-2)	**100**	**0.8M**	**6.3** ($\pm$0.083)	**150**

TABLE III: Test error(%) on CIFAR-10 without data augmentation

Algorithm 2 Online Redundancy-dependent Dropout (divReg-2)

{The parameters are: B_S - the batch size, ξ - learning rate, n' - number of filters, α - dropout fraction, N_r - number of redundant filters, τ - diversity regularization correlation threshold, and $\overset{*}{\tau}$ }
{θ is the vector of concatenated weights ($\mathbf{W}^{(l)}$) and biases ($\mathbf{b}^{(l)}$). Initialize θ from a normal distribution as proposed in [2].
$\theta \leftarrow \theta_0$ {Initial weight and biases}
{ Initialize α - dropout fraction}
for prescribed number of epochs ($nbepoch$) **do**
  permute training samples
  $N_r \leftarrow 0$ {Initial number of redundant features}
  for all batches of B_S train samples **do**
    $J_{net} \leftarrow$ loss on samples in the batch b from eq. (6)
    $\Delta\theta \leftarrow$ compute gradient using eq. (7) and (8)
    {Make a step along the gradient}
    $\theta \leftarrow \theta - \xi\Delta\theta$
  end for
  {Compute the binary mask $\overset{*}{M}^{(l)}$ in (13) for every layer}
  {Compute and update α in (12) for every layer l}
end for

from the result that the model trained using divReg-1 and divReg-2 outperforms all other benchmark regularizers. The results also show that dropping the maps of redundancy filters in divReg-1 leads to a better generalization but introduces computational overhead comparable to divReg-2 with similar performance.

In the second set of large scale image classification experiments, we trained the current state-of-the-art densely connected convolutional neural network (DenseNet) [39] on CIFAR-10 dataset to see effect of extracting dissimilar features on classification performance. The dataset contains a labeled set of 60,000 32x32 color images belonging to 10 classes: airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks. The dataset is split into 50000 and 10000 training and testing sets, respectively. Again, 6000 images were randomly sampled from the training set as a held-out validation set for hyperparameter tuning and the network was retrained on the entire dataset using the best hyperparameter configuration. Due to GPU memory constraints, we used 100-layer DenseNet with a growth rate of 12. The model was trained with stochastic gradient descent (SGD) with batch-size

Model	Top-1 (%)	Top-5 (%)	# Epochs
ResNet-34 [5]	26.77	8.56	90
ResNet-34 + OrthoReg	33.21	12.42	90
ResNet-34 + divReg	**26.33**	**8.0**	90

TABLE IV: Validation error on ImageNet

of 64 for 150 epochs and ξ initialized to 0.1 and scheduled to 0.01, 0.1, 0.01, 0.001, 0.01, 0.001, and 0.0001 in epochs 20, 24, 44, 84, 104, 114, and 130, respectively as shown in Fig. 9. For a fair comparison, the results presented in Table III are for models trained without data augmentation. Hyperparameters τ and $\overset{*}{\tau}$ in divReg2 was also set to 0.5 and 0.2, respectively. The dropout fraction α was initialized to 0.1. We limit our implementation to DenseNet with bottleneck (BC) with approximately $0.8M$ trainable parameters due to memory constraints. The classification performance of the deep networks regularized with Dropout [1], Label smoothing [20], confidence penalty [23], and OrthoReg [3] were used as benchmark. The experiments were repeated five times and averaged. It can be observed in Table III that divReg-2 outperforms all other regularizations considered - an indication that extraction of dissimilar features and redundancy-based adaptive dropout improve generalization of very deep neural network models. In the third set of experiments involving

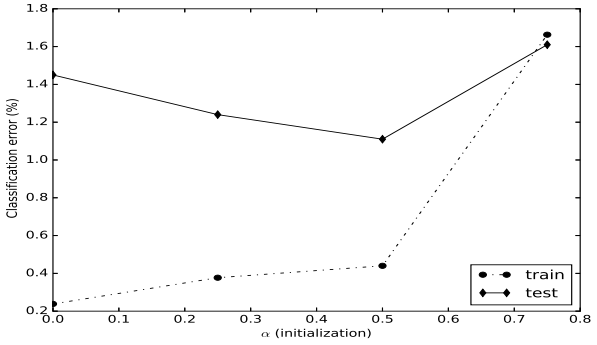


Fig. 8: Performance evaluation using divReg-2 on MNIST dataset for four different initializations of α.

large-scale image classification, experiments were performed on the 1000-class ImageNet 2012 dataset [41] which contains about 1.2 million training images, 50,000 validation images, and 100,000 test images (with no published labels). The results are measured by top1/top-5 error rates [41]. ImageNet dataset was used to train a residual network known as ResNet-34

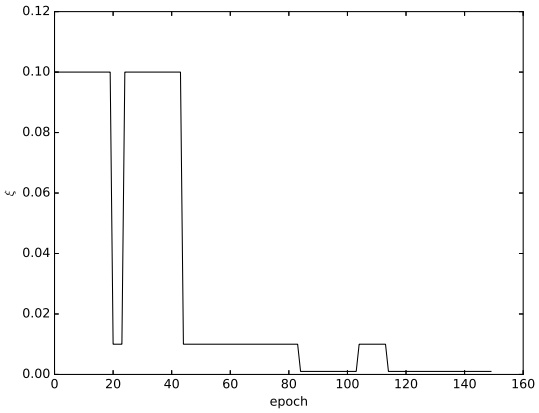


Fig. 9: Learning rate (ξ) schedule for experiments on CIFAR-10 dataset.

[5], which has four stages of residual blocks and uses the projection shortcut when the feature maps are down-sampled. The model was trained for 90 epochs, with a batch-size of 200 and a learning rate 0.1. Each layer of the model is regularized using divReg with $\tau = 0.4$. It can be observed in Table IV that divReg also outperforms both OrthoReg regularization method and unregularized counterpart.

C. Natural Language Inference

In the last set of experiments, we demonstrated the efficacy of diversity regularization in enhancing the efficiency of models used for understanding semantic relationship between two sentences and recognizing textual entailment. This task involves determining whether two observed sentences, first one is known as the premise and the other referred to as the hypothesis, are contradictory, not related (neutral) or entailing. In this series of experiments, models are evaluated on textual entailment recognition task using Stanford Natural Language Inference (SNLI) dataset [36]. The original dataset contains $550,152$ duos of premise-hypothesis sentences and their corresponding labels as training set, $10,000$ as validation set, and $10,000$ as testing set. After the removal of sentence-pair with unknown labels, we obtained $549,367$ pairs for training, $9,842$ for validation and $9,824$ for testing. Select examples from SNLI dataset are shown in Table V.

Our implementation is based on baseline *Keras SNLI models* repository [2]. From SNLI dataset, we extracted 300-dimensional word embeddings from the pretrained 300D Glove 840B vocabulary [42], each for both the premise and hypothesis sentences and fed them through a ReLU "translation" layer. The maximum sequence length was chosen to be 42 and the embeddings of words not in the vocabulary are set to zero in accordance with [43]. The pretrained Glove embedding layer contains more than 12 million parameters, which we fixed during training to avoid overfitting [44] and computational overhead. We used the LSTM model with 300 hidden units to encode the premise and hypothesis sentences and the resulting two $300D$ embeddings are concatenated and fed into three layers of fully-connected units with ReLU activations. The output of the last layer is fed into Softmax layer for classification.

The overall model was trained using Adam optimizer with batch-size of 512 for 100 epochs while τ and $\overset{*}{\tau}$ in divReg2 was also set to 0.5 and 0.2, respectively. The dropout fraction α for both recurrent (LSTM) and fully-connected layers was initialized to 0.2. The experiment was also performed by replacing LSTM with a GRU. We benchmark our results with recent sentence encoding-based models and experimental results were illustrated in Table VI. It is remarked that the parameters of the Glove embedding layer were not included in the number of parameters computed in Table VI. As can be observed from the results, diversity regularization and adaptive dropout significantly improved the performance of both the baseline LSTM and GRU models. By initializing dropout fraction of both recurrent and fully-connected units to 0.2, the model was able to figure out the suitable dropout fraction in accordance with differentiation of features. In addition, setting τ to 0.5 ensures no feature pair have cosine similarity greater than 0.5. Another important observation is that OrthoReg sometimes extracts noisy features in an attempt to decorrelate features, which explains why the performance of some models deteriorates. Deep Gated Attn BiLSTM (D-GAB) encoders [45] is the state-of-the-art sentence encoding-based model for SNLI dataset with test accuracy of 85.5%. However, we did not regularize D-GAB using diversity regularization because it has more than 11 million parameters requiring larger memory than those compared in Table VI.

V. Conclusion

This paper addresses the concept and properties of special regularization of deep neural network models that take advantage of extracting diversified features and dropping features based on select redundancy measures. The performance of the proposed regularization in terms of extracting diverse features and improving generalization was compared with recent regularization techniques on select tasks using state-of-the-art deep learning models. The results show that if not properly constrained, deep neural network models are capable of extracting very similar features thereby creating unnecessary amount of filtering redundancy. By using the proposed methods, such redundancy can be controlled, eliminated and networks are enabled to extract more distinctive features. It has also been shown on select examples that concurrent extraction of diverse features and redundant feature dropout improve model generalization. These concepts are illustrated using MNIST handwritten digits, CIFAR-10, ImageNet, and Stanford Natural Language Inference Dataset.

[2]https://github.com/Smerity/keras_snli

References

[1] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1026–1034.

Premise	Hypothesis	label
A soccer game with multiple males playing	Some men are playing a sport	E
A man inspects the uniform of a figure in some East Asian country	The man is sleeping	C
A smiling costumed woman is holding an umbrella	A happy woman in a fairy costume holds an umbrella	N

TABLE V: A select examples from SNLI dataset where E, C, and N represent Entailment, Contradiction, and Neutral, respectively.

Model	**# parameters**	**test** (%)
300D LSTM (recurrent dropout) + 3 x 600D ReLU + OrthoReg	1.9 M	77.4
300D LSTM encoders [46]	3.0 M	80.60
300D LSTM (recurrent dropout) + 3 x 600D ReLU	1.9 M	82.7
300D SPINN-PI encoders [46]	3.7 M	83.2
600D (300+300) BiLSTM encoders [44]	2.0 M	83.3
300D NTI-SLSTM-LSTM encoders [43]	4.0 M	83.4
300D LSTM (recurrent dropout) + 3 x 600D ReLU + divReg2	**1.9 M**	**83.9**
300D GRU (recurrent dropout) + 3 x 600D ReLU	1.7 M	83.0
300D GRU (recurrent dropout) + 3 x 600D ReLU + OrthoReg	1.7 M	80.8
300D GRU (recurrent dropout) + 3 x 600D ReLU + divReg2	**1.7 M**	**84.3**

TABLE VI: Test accuracy (%) on SNLI dataset.

[3] P. Rodríguez, J. Gonzàlez, G. Cucurull, J. M. Gonfaus, and X. Roca, "Regularizing cnns with locally constrained decorrelations," *arXiv preprint arXiv:1611.01967*, 2016.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[7] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *arXiv preprint arXiv:1701.06538*, 2017.

[8] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[9] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. of the 31st International Conference on Machine Learning*, 2014, pp. 1764–1772.

[10] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.

[13] D. Mishkin and J. Matas, "All you need is a good init," *International Conference on Learning Representations*, 2016.

[14] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural Computation*, vol. 4, no. 4, pp. 473–493, 1992.

[15] B. O. Ayinde and J. M. Zurada, "Deep learning of constrained autoencoders for enhanced understanding of data," *IEEE Transactions on Neural Networks and Learning Systems*, 2017, https://doi.org/10.1109/TNNLS.2017.2747861.

[16] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proc. of the 30th International Conference on Machine Learning*, 2013, pp. 1058–1066.

[17] Y. Bengio and J. S. Bergstra, "Slow, decorrelated features for pretraining complex cell-like networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 99–107.

[18] Y. Bao, H. Jiang, L. Dai, and C. Liu, "Incoherent training of deep neural networks to de-correlate bottleneck features for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6980–6984.

[19] D. Miller, A. V. Rao, K. Rose, and A. Gersho, "A global optimization technique for statistical classifier design," *IEEE Transactions on Signal Processing*, vol. 44, no. 12, pp. 3108–3122, 1996.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[21] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[22] L. Xie, J. Wang, Z. Wei, M. Wang, and Q. Tian, "Disturblabel: Regularizing cnn on the loss layer," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4753–4762.

[23] G. Pereyra, G. Tucker, J. Chorowski, Ł. Kaiser, and G. Hinton, "Regularizing neural networks by penalizing confident output distributions," *arXiv preprint arXiv:1701.06548*, 2017.

[24] A. Dundar, J. Jin, and E. Culurciello, "Convolutional clustering for unsupervised learning," *arXiv preprint arXiv:1511.06241*, 2015.

[25] B. O. Ayinde and J. M. Zurada, "Nonredundant sparse feature extraction using autoencoders with receptive fields clustering," *Neural Networks*, vol. 93, pp. 99–109, 2017.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[27] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[28] S. Belharbi, C. Chatelain, R. Herault, and S. Adam, "Neural networks regularization through invariant features learning," *arXiv preprint arXiv:1709.01867*, 2017.

[29] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical report, University of Toronto*, 2009.

[30] B. Walter, K. Bala, M. Kulkarni, and K. Pingali, "Fast agglomerative clustering for rendering," in *IEEE Symposium on Interactive Ray Tracing*, 2008, pp. 81–86.

[31] C. Ding and X. He, "Cluster merging and splitting in hierarchical clustering algorithms," in *Proc. of the IEEE International Conference on Data Mining*, 2002, pp. 139–146.

[32] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Workshop on Statistical Learning in Computer Vision*, vol. 2, no. 5, 2004, p. 7.

[33] S. Manickam, S. D. Roth, and T. Bushman, "Intelligent and optimal

normalized correlation for high-speed pattern matching," *Datacube Technical Paper*, 2000.
[34] S. Zhou, Z. Xu, and F. Liu, "Method for determining the optimal number of clusters based on agglomerative hierarchical clustering," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 12, pp. 3007–3017, Dec 2017.
[35] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.
[36] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.
[37] M. Abadi, A. Agarwal, P. Barham, and et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/
[38] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[39] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," *arXiv preprint arXiv:1608.06993*, 2016.
[40] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.
[41] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
[42] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.
[43] T. Munkhdalai and H. Yu, "Neural semantic encoders," *arXiv preprint arXiv:1607.04315*, 2016.
[44] Y. Liu, C. Sun, L. Lin, and X. Wang, "Learning natural language inference using bidirectional lstm model and inner-attention," *arXiv preprint arXiv:1605.09090*, 2016.
[45] Q. Chen, X. Zhu, Z.-H. Ling, S. Wei, H. Jiang, and D. Inkpen, "Recurrent neural network-based sentence encoder with gated attention for natural language inference," *arXiv preprint arXiv:1708.01353*, 2017.
[46] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, "A fast unified model for parsing and sentence understanding," *arXiv preprint arXiv:1603.06021*, 2016.

Babajide Ayinde (S'09) received the M.Sc. degree in Engineering Systems and Control from the King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. He is currently a Ph.D. candidate at the University of Louisville, Kentucky, USA and a recipient of University of Louisville Grosscurth Fellowship. His current research interests include unsupervised feature learning and deep learning techniques and applications.

Tamer Inanc received a B.S. degree in Electrical Engineering from the Dokuz Eylul University, Izmir, Turkiye, in 1991. He received his M.S. and Ph.D. degrees in Electrical Engineering from the Pennsylvania State University, State College, PA in 1996 and 2002, respectively, under supervision of Prof. Mario Sznaier. After receiving his Ph.D., Dr. Inanc did a postdoctoral scholarship at the California Institute of Technology, Pasadena, CA between 2002 and 2004. He started working as an Assistant Professor in 2004 at the Electrical and Computer Engineering Department at University of Louisville, Louisville, KY. He has been working as an Associate Professor at the same department since 2010. He received the 2008 Delphi Center, UofL, "Innovations in Technology Award for Teaching and Learning" and the 2006 Kentuckiana Metroversity Instructional Development Award for "Fundamentals of Autonomous Robots". His research interests center on control systems, model identification, autonomous robotics, biometrics and applications of control systems and identification to biomedical problems.

Jacek M. Zurada (M'82-SM'83-F'96-LF'14) received the Ph.D. degree from the Gdansk Institute of Technology, Gdansk, Poland. He currently serves as a Professor of electrical and computer engineering with the University of Louisville, Louisville, KY, USA. He has authored or co-authored several books and over 420 papers in computational intelligence, neural networks, machine learning, logic rule extraction, and bioinformatics cited 11,900 times, and delivered over 120 presentations throughout the world.

Dr. Zurada has been a Board Member of the IEEE, IEEE CIS and IJCNN. He was a recipient of the 2013 Joe Desch Innovation Award, the 2015 Distinguished Presidential Service Award, and five honorary professorships. He served as the IEEE V-President and the Technical Activities Board (TAB) Chair in 2014. In 2010-13 he was the Chair of the IEEE TAB Periodicals Committee and the TAB Periodicals Review and Advisory Committee. From 2004 to 2005, he was the President of the IEEE Computational Intelligence Society. He was the Editor-in-Chief of the IEEE Transactions on Neural Networks (1997-2003) and an Associate Editor of the IEEE Transactions on Circuits and Systems, Neural Networks NEURAL and the Proceedings of the IEEE. He is currently a Candidate for 2019 IEEE President-Elect.