Adaptive Knowledge Driven Regularization for Deep Neural Networks

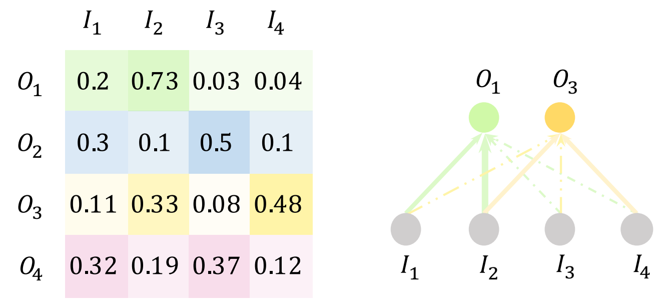
Zhaojing Luo,[1] Shaofeng Cai,[1] Can Cui,[1] Beng Chin Ooi,[1] Yang Yang[2]
[1]Department of Computer Science, School of Computing, National University of Singapore
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China
{zhaojing, shaofeng, cuican, ooibc}@comp.nus.edu.sg, dlyyang@gmail.com

Abstract

In many real-world applications, the amount of data available for training is often limited, and thus inductive bias and auxiliary knowledge are much needed for regularizing model training. One popular regularization method is to impose prior distribution assumptions on model parameters, and many recent works also attempt to regularize training by integrating external knowledge into specific neurons. However, existing regularization methods fail to take account of the interaction between connected neuron pairs, which is invaluable internal knowledge for adaptive regularization for better representation learning as training progresses. In this paper, we explicitly take into account the interaction between connected neurons, and propose an adaptive internal knowledge driven regularization method, *CORR-Reg*. The key idea of CORR-Reg is to give a higher significance weight to connections of more correlated neuron pairs. The significance weights adaptively identify more important input neurons for each neuron. Instead of regularizing connection model parameters with a static strength such as weight decay, CORR-Reg imposes weaker regularization strength on more significant connections. As a consequence, neurons attend to more informative input features and thus learn more diversified and discriminative representation. We derive CORR-Reg with the Bayesian inference framework and propose a novel optimization algorithm with the Lagrange multiplier method and Stochastic Gradient Descent. Extensive evaluations on diverse benchmark datasets and neural network structures show that CORR-Reg achieves significant improvement over state-of-the-art regularization methods.

1 Introduction

Recent years have witnessed significant improvements in machine learning and data mining technology (Zhang, Kumar, and Ré 2014; Goodfellow et al. 2014; He et al. 2015; Ooi et al. 2015; Veit, Wilber, and Belongie 2016; Wang et al. 2016). In particular, deep neural networks produce record-breaking results for a wide range of applications (Wang et al. 2015; He et al. 2016a; Zhang et al. 2018; Yang et al. 2018; Tay et al. 2018; Luo et al. 2019) due to their unprecedented model capacity and representation power. The success of deep neural networks is highly dependent on the availability of large amounts of training data (Deng et al. 2009; Krizhevsky, Sutskever, and Hinton 2012; Russakovsky et al. 2015). However, in many real-world applications, the training data is generally limited (Che et al. 2015; Luo et al. 2018). This causes over-fitting and greatly affects the model performance.



	I_1	I_2	I_3	I_4
O_1	0.2	0.73	0.03	0.04
O_2	0.3	0.1	0.5	0.1
O_3	0.11	0.33	0.08	0.48
O_4	0.32	0.19	0.37	0.12

Figure 1: Significance weight for adaptive regularization.

Regularizing training by integrating auxiliary knowledge (Mesnil et al. 2013; Lample et al. 2016) into the training of deep neural networks is widely adopted to alleviate the issue of training with insufficient data. Different techniques of injecting external knowledge into networks have been proposed, e.g., into the first or the last layer (Srivastava and Salakhutdinov 2013; Che et al. 2015; Tran et al. 2015; Brust and Denzler 2018). These methods are proposed based on the observation that the meanings of the neurons of these layers can be linked to external knowledge. While these methods have been proven to be effective in improving model training, external knowledge is typically difficult to obtain, and it requires domain knowledge to transfer such knowledge to guide the network training.

In the connectionist viewpoint (He et al. 2016b; Ren et al. 2015; Simonyan and Zisserman 2014), neurons are the basic components in the network to extract features and build up representation, although the exact interpretation of the extracted features is intractable. Further, the relation between neurons pairs, e.g., their correlation, is a strong indicator of the representation learning process and can be obtained and exploited during training. This leads us to the following question: *Can we exploit internal knowledge of the network during the training process to regularize training?*

To answer this question, we examine the interaction between neurons in the network and based on the correlation of the neuron connection, we propose CORR-Reg, an adaptive knowledge driven regularization method, to adaptively integrate such internal knowledge into the model to regularize training. The proposed regularization method is derived from the Bayesian inference framework by introducing the correlation knowledge to the prior distribution of the connection weights.

The effect of CORR-Reg is illustrated in Figure 1. We introduce the *significance weight*, which is positively correlated with the absolute value of the correlation between neurons, to discriminate the importance of incoming connections for each neuron. On the left of Figure 1, each row shows the relative significance weights of the four input neurons for a given output neuron of the current layer. The key idea of CORR-Reg is to impose weaker regularization on connections of a higher significance weight, so that neurons can attend to more important input features adaptively and thus extract more diversified and discriminative features. We illustrate the adaptive regularization with CORR-Reg on the right side of Figure 1, where a thick connection indicates a higher significance weight and weaker regularization strength correspondingly.

The benefits of CORR-Reg can be understood from the perspective of representation learning (Bengio, Courville, and Vincent 2013). During the model training process, each neuron performs feature extraction and learns higher-level representation from input neurons in the preceding layer. Correlation measures the degree of the dependence of the neuron on its input neurons. Consequently, discriminating the strength of regularization based on such correlation guides neurons in learning more discriminative representations by focusing on those more informative neurons. Meanwhile, the adaptive regularization strength help neurons attend to different subsets of input neurons, which prevent neurons from learning redundant representation (Srivastava et al. 2014; Cogswell et al. 2015; Zhu, Zhou, and Li 2018).

We derive CORR-Reg with the Bayesian inference framework and propose a novel optimization algorithm with the Lagrange multiplier method and Stochastic Gradient Descent (SGD). Extensive evaluations on diverse benchmark datasets and neural network structures show that CORR-Reg achieves significant improvement over existing regularization methods. We summarize our contributions as follows:

- We propose a general adaptive knowledge driven regularization method based on correlation, CORR-Reg, to integrate internal knowledge into neural networks for better representation learning.
- We design an efficient algorithm to update significance weights by applying the Lagrange multiplier method and update the model parameters via SGD. The proposed algorithm effectively updates the two sets of parameters alternately.
- We conduct extensive experiments on real-world benchmark datasets and various neural network structures. Results show that our proposed CORR-Reg regularization achieves consistently better performance than state-of-the-art regularization methods.

2 Related Work

2.1 Regularization Methods for Neural Networks

The most representative regularization method is weight decay (Wang, Sun, and Liu 2016) that corresponds to the Gaussian prior on model parameters (Williams 1995; Kneib, Konrath, and Fahrmeir 2011), which imposes L2 norm regularization on model parameters (Krogh and Hertz 1991). Another common regularization method is L1-norm regularization (Williams 1995; Meinshausen and Bühlmann 2006), which corresponds to a Laplacian prior on model parameters and leads to sparse models. Max-norm regularization (Srebro, Rennie, and Jaakkola 2005; Lee et al. 2010; Srebro and Shraibman 2005) regularizes the values of the model parameters similar to L1-norm and L2-norm regularization by constraining the norm of model parameters to be bounded by a constant. Dropout (Hinton et al. 2012; Srivastava et al. 2014) regularizes the neural networks by changing the model architecture instead of constraining the values of the model parameters. Dropout is arguably the most effective and widely adopted regularization method for neural networks. These regularization methods impose the same regularization strength over the model parameters. In contrast, our proposed CORR-Reg imposes customized regularization strengths on different model parameters by taking account of the correlation knowledge.

2.2 Incorporating Knowledge for Regularizing Neural Networks

One of the first methods that incorporate knowledge to regularize deep neural networks is introduced in (Srivastava and Salakhutdinov 2013), where tree hierarchy knowledge is used to impose a prior regularization over the model parameters of the topmost layer of the deep neural network. Similarly, (Che et al. 2015) makes use of medical ontology knowledge to regularize the model parameters of the topmost layer. Different from (Che et al. 2015), (Tran et al. 2015) introduce knowledge to the first layer of the deep neural network for healthcare applications. These methods make use of knowledge which is limited to specific application domains, e.g., healthcare or image recognition. In (Che et al. 2015), a Laplacian graph-based prior framework is designed to incorporate any relational information that can be represented as a weighted graph, e.g., co-occurrence, which is general to different application domains. (Brust and Denzler 2018) makes use of WordNet and encodes the properties of class hierarchy into a probabilistic model for regularizing the deep neural network. Since WordNet is a general lexical database for English, this method is also general to different application domains.

Existing methods incorporate knowledge into either the first or the topmost layer of the deep neural networks because the neurons of these two layers can be linked to the knowledge. For hidden layers, since the neurons are higher-level abstractions of preceding layers, the existing methods can not be readily used to incorporate knowledge. Also, external knowledge is typically difficult to obtain and these methods require domain knowledge to transfer such knowledge to guide the model training. Different from existing methods, the adaptive knowledge driven regularization in our paper is designed to integrate internal knowledge into the hidden layers of the network as a general regularization method.

3 Bayesian Interpretation of Regularization

In this paper, we propose to exploit correlation knowledge by integrating it into the model regularization during training based on the Bayesian inference framework. Before introducing our approach, we first briefly describe the Bayesian viewpoint of understanding of the regularization. In Bayesian inference, regularization corresponds to a prior distribution over the model parameters $\boldsymbol{w}$. According to Bayes' theorem, the posterior probability of model parameters $\boldsymbol{w}$ is $p(\boldsymbol{w}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})/p(\mathcal{D})$. Here, $\mathcal{D}$ denotes the observed data, $\boldsymbol{w}$ denotes the model parameters, $p(\mathcal{D}|\boldsymbol{w})$ is the likelihood function and $p(\mathcal{D})$ is a constant. Model parameters $\boldsymbol{w}$ are typically estimated via maximum a posteriori (MAP) estimation (Kneib, Konrath, and Fahrmeir 2011), which is formulated as $\boldsymbol{w}_{MAP} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \, (-\log p(\mathcal{D}|\boldsymbol{w}) - \log p(\boldsymbol{w}))$. The term $\log p(\boldsymbol{w})$ is the log of model parameter prior distribution, and it is typically interpreted as the regularization term. In our work, we assume $p(\boldsymbol{w})$ is related to the correlation between neuron pairs, and based on this assumption, we derive a knowledge driven regularization method with correlation as the internal knowledge.

4 Adaptive Knowledge Driven Regularization

Our main intuition is that the correlation of activation values of neurons from adjacent transformation layers can be interpreted as the relative importance of their connection. The rationale to exploit the correlation as a regularization term is that a higher correlation of the connection between neuron pairs indicates that the output neuron is more dependent on the corresponding input neuron, and thus a relatively smaller regularization strength should be imposed.

To exploit the correlation as the internal knowledge to regularize training, we introduce a variable *significance weight* to the prior distribution of the model parameters. The significance weight $\theta_{i,j}$ between the i-th output neuron O_i in the current layer and the j-th input neuron I_j in the preceding layer represents the importance of I_j to O_i. During the model training process, each output neuron O_i extracts higher-level representations from the J input neurons, i.e., the J input neurons contribute as a group to the learning of neuron O_i. Based on this observation, we place the constraints of $\theta_{i,j} \geq 0$ and $\sum_{j=1}^{J} \theta_{i,j} = 1$ to discriminate the relative importance of different input neurons adaptively. Therefore, significance weights correspond naturally to the parameters of Multinomial distribution.

Since correlation captures the importance between neurons of adjacent layers, it can be informative to learn the significance weights $\overrightarrow{\theta}_i$ for each neuron O_i. In order to incorporate correlation knowledge for learning $\overrightarrow{\theta}_i$, we propose to take advantage of the conjugate prior for $\theta_{i,j}$, i.e., Dirichlet distribution $\text{Dir}(\overrightarrow{\theta}_i|\overrightarrow{\alpha}_i)$, by setting the hyperparameters $\overrightarrow{\alpha}_i$ with the correlation information adaptively.

In the remainder of this section, we will introduce the designed prior distribution for model parameters and the resultant overall loss function in detail.

4.1 Correlation Regularization Term

Prior Distribution for Model parameters The prior distribution is designed to capture the relation between neurons from adjacent layers via the significance weight. Given the significance weight $\theta_{i,j}$ and the model parameter $w_{i,j}$ that connects the output neuron O_i with the input neuron I_j, we define the generation probability of model parameters $\overrightarrow{w}_i$ that connect O_i with all the J input neurons as follows:

$$p(\overrightarrow{w}_i|\overrightarrow{\theta}_i) = C\prod_{j=1}^{J}\{\theta_{i,j}^{\lambda w_{i,j}^2}\}, \tag{1}$$

where C is the normalization coefficient, and λ is the hyperparameter that controls the regularization strength of model parameter $w_{i,j}$. This equation can be written as $C\prod_{j=1}^{J}\{\exp(-\frac{1}{2}(\frac{w_{i,j}}{\frac{1}{\sqrt{-2\lambda\log(\theta_{i,j})}}})^2)\}$, which shows the prior distribution for model parameter $w_{i,j}$ is a Gaussian distribution with variance $\frac{1}{-2\lambda\log(\theta_{i,j})}$. Note that the variance of the Gaussian distribution is different for different model parameters $w_{i,j}$, depending on the significance weight $\theta_{i,j}$, leading to the adaptive Gaussian distribution. To be specific, if I_j is significant to O_i, which is indicated by a larger significance weight $\theta_{i,j}$, the variance of the Gaussian prior distribution, namely, $\frac{1}{-2\lambda\log(\theta_{i,j})}$, is relatively larger.

To incorporate correlation for learning $\theta_{i,j}$, Dirichlet distribution, which is the conjugate prior for $\theta_{i,j}$ is employed and the joint distribution for $\overrightarrow{w}_i$, $\overrightarrow{\theta}_i$ of O_i then can be formulated as:

$$p(\overrightarrow{w}_i, \overrightarrow{\theta}_i|\overrightarrow{\alpha}_i) = p(\overrightarrow{w}_i|\overrightarrow{\theta}_i)p(\overrightarrow{\theta}_i|\overrightarrow{\alpha}_i), \tag{2}$$

where $p(\overrightarrow{\theta}_i|\overrightarrow{\alpha}_i)$ is the Dirichlet distribution $\text{Dir}(\overrightarrow{\theta}_i|\overrightarrow{\alpha}_i)$. The hyperparameter $\overrightarrow{\alpha}_i$ is designed to integrate the correlation knowledge via:

$$\overrightarrow{\alpha}_i = 1 + \lambda\beta\overrightarrow{r}_i, \tag{3}$$

where β is used to trade off between correlation $\overrightarrow{r}_i$ and model parameters $\overrightarrow{w}_i$, which will be explained in the following sections. Thereby, correlation $\overrightarrow{r}_i$ is designed to be positively correlated with both $\overrightarrow{\alpha}_i$ and therefore $\overrightarrow{\theta}_i$. To be specific, a larger value of a specific dimension in $\overrightarrow{r}_i$ leads to a larger value in the corresponding dimension in $\overrightarrow{\alpha}_i$ and further $\overrightarrow{\theta}_i$. The intuition of integrating the internal correlation knowledge in such a way is that we want to impose weaker regularization on the model parameter $w_{i,j}$ if the corresponding connection is more important, which is measured by the significance weight $\theta_{i,j}$.

Calculation of Correlation For $\overrightarrow{r}_i$ of neuron O_i, each dimension is a normalized correlation value, and each $r_{i,m}$ is calculated by:

$$r_{i,m} = \frac{|corr_{i,m}|}{\sum_{j=1}^{J}\{|corr_{i,j}|\}}, \tag{4}$$

where $corr_{i,j}$ is the *Pearson* correlation value between O_i and I_j. In CORR-Reg, correlation is calculated batch-wise. We use N to denote the batch size, for each mini-batch, we obtain N pairs of activation values for neurons O_i and I_j. Thus, the correlation between O_i and I_j can be calculated using the N pairs of activation values. However, calculating the correlation batch-wise is not stable. To alleviate this issue, we propose to use the exponential moving average: $corr_{i,j} = \gamma corr_{i,j} + (1-\gamma) corr\_curr_{i,j}$, where $corr\_curr_{i,j}$ is the correlation value calculated by using the current mini-batch. $corr_{i,j}$ is initialized to zero and accumulates the correlation values calculated from each mini-batch. Empirically, we set γ to a large value, e.g., 0.9, so that $corr_{i,j}$ is less sensitive to recent changes and returns relatively stable values.

Overall Loss Function Deriving the loss function from the MAP estimation over model parameters $\boldsymbol{w}$, we obtain:

$$G = -\log p(\mathcal{D}|\boldsymbol{w}) - \sum_{i=1}^{I}\{\log p(\overrightarrow{w}_i, \overrightarrow{\theta}_i|\overrightarrow{\alpha}_i)\}, \tag{5}$$

where I is the number of all output neurons. For Equation 5, the first term is the negative log-likelihood function, and the second term is the correlation regularization term.

4.2 Optimization for CORR-Reg

For CORR-Reg method, both model parameters $\boldsymbol{w}$ and significance weights $\overrightarrow{\theta}_i$ of O_i need to be updated. We propose to update both of them jointly from the joint distribution defined in Equation 5. Specifically, given the updated value of the model parameters $\boldsymbol{w}$, by maximizing the joint distribution with respect to significance weights $\overrightarrow{\theta}_i$, we can obtain the updated value for $\overrightarrow{\theta}_i$. Subsequently, we can use the updated $\overrightarrow{\theta}_i$ to further optimize the model parameters $\boldsymbol{w}$. This process suggests a natural alternate update algorithm over $\boldsymbol{w}$ and $\overrightarrow{\theta}_i$ until convergence.

For model parameters, SGD is used as the update method, which is a conventional method for updating parameters of different models; for significance weights, since the condition $\sum_{j=1}^{J}\theta_{i,j} = 1$ must be satisfied, Lagrange multiplier method is therefore adopted. In this section, we introduce the calculation of the model parameter gradient and the update equations for significance weights.

Gradient Descent for Model Parameters When significance weights $\overrightarrow{\theta}_i$ of O_i are fixed, gradient descent method is adopted to update model parameters $\boldsymbol{w}$. According to Equation 5, the gradient for $w_{i,j}$ with respect to loss G is

$$\frac{\partial G}{\partial w_{i,j}} = \frac{\partial - \log p(\mathcal{D}|\boldsymbol{w})}{\partial w_{i,j}} - 2\lambda \log(\theta_{i,j}) w_{i,j}. \tag{6}$$

Equation 6 shows CORR-Reg imposes a regularization similar to L2-norm regularization. Since $\theta_{i,j}$ indicates the importance of I_j to O_i, the strength of the regularization on $w_{i,j}$ is thus related to the importance weight. Specifically, the larger the significance weight $\theta_{i,j}$ is, the weaker the regularization for $w_{i,j}$. In other words, CORR-Reg tends to impose weaker regularization on the model parameter $w_{i,j}$ for more important input neurons.

Update for Significance Weights After the model parameters are updated, the significance weights $\overrightarrow{\theta}_i$ of O_i are updated in turn. Since the condition $\sum_{j=1}^{J}\theta_{i,j} = 1$ must be satisfied, Lagrange multiplier is introduced to the overall loss function. We introduce μ_i as the Lagrange multiplier for $\overrightarrow{\theta}_i$. The Lagrangian of the loss function is

$$L = G + \sum_{i=1}^{I}\{\mu_i(\sum_{j=1}^{J}\theta_{i,j} - 1)\}. \tag{7}$$

After setting the gradient of $\theta_{i,m}$ and $\mu_i$with respect to L to zero, we can get

$$\theta_{i,m} = \frac{(\alpha_{i,m} - 1) + \lambda w_{i,m}^2}{\sum_{j=1}^{J}\{(\alpha_{i,j} - 1) + \lambda w_{i,j}^2\}}. \tag{8}$$

Equation 8 shows the significance weight of I_m to O_i is determined by two factors. The first is $\alpha_{i,m}$, which is positively correlated with the absolute value of the correlation between I_m and O_i. The second is the model parameter $w_{i,m}$. Specifically, a larger value of $w_{i,m}^2$ leads to a larger value of $\theta_{i,m}$, which indicates that I_m is more significant to O_i. $\theta_{i,m}$ is then used to determine the strength of regularization.

4.3 Lazy Update for CORR-Reg

The update of significance weights is time-consuming because it involves calculation of correlation values. In order to reduce the overall computational costs, we devise a lazy update algorithm. The key idea is that both correlation values and significance weights do not change too much after the first few epochs. As a result, they do not need to be calculated every epoch after the first few epochs. Algorithm 1 shows the overall procedure for updating model parameters and significance weights with lazy update.

$\boldsymbol{A}$ denotes the Dirichlet hyperparameters for all the hidden neurons, which is related to the correlation knowledge. Θ denotes the significance weights for all hidden neurons and is uniformly initialized. lr is the learning rate for SGD. E is the number of the first few epochs without lazy update, and B is the number of mini-batches in the training dataset. T_s is the update interval for updating significance weights and it trades off between computational time and model performance. A larger T_s decreases the overall computational time while typically leads to lower model performance. Empirically, E is set to 3 and T_s is set to 10.

Algorithm 1 first calculates the gradients for model parameters using Equation 6 and updates $\boldsymbol{\Theta}$ using Equation 8. SGD is then used to update the model parameters $\boldsymbol{w}$ of the neural network by backpropagation. Note that in line 6, the update of significance weights $\boldsymbol{\Theta}$, which involves the calculation of correlation values, is carried out every T_s iterations instead of each iteration to reduce the computational cost.

Algorithm 1 Lazy Update for CORR-Reg

```
    Input: $\boldsymbol{w}$, $\boldsymbol{A}$, $\Theta$, $lr$, $E$, $B$, $T_s$
1:  initialize: $it \leftarrow 0, epoch\_it \leftarrow 0$
2:  while not converged do
3:      Compute $\frac{\partial - \log p(\mathcal{D}|\boldsymbol{w})}{\partial \boldsymbol{w}}$
4:      Compute $\frac{\partial G}{\partial \boldsymbol{w}}$ based on Equation 6
5:      if $epoch\_it < E$ or $it \; mod \; T_s = 0$ then
6:          Compute $\boldsymbol{\Theta}$ for all hidden neurons based on Equation 8
7:      end if
        /* SGD step */
8:      $\boldsymbol{w}^{(it+1)} \leftarrow \boldsymbol{w}^{(it)} - lr \frac{\partial G}{\partial \boldsymbol{w}}$
9:      $it \leftarrow it + 1$
10:     if $it \; mod \; B = 0$ then
11:         $epoch\_it \leftarrow epoch\_it + 1$
12:     end if
13: end while
```

4.4 Analysis and Discussion

In each iteration, with fixed significance weights $\boldsymbol{\Theta}$, CORR-Reg reduces to L2-Reg with different customized regularization strengths for different model parameters. But instead of manually designing the regularization strength for each model parameter prior to training, we propose to learn an adaptive regularization strength for each model parameter using the correlation knowledge during training.

In terms of efficiency, compared with L2-Reg, the majority of the extra computational overhead of CORR-Reg comes from the calculation of correlation. With lazy update, the overall computation time is on par with other existing regularization methods. During inference, CORR-Reg incurs no additional computational overhead.

Layer Name	**25% of E**	**50% of E**	**75% of E**	**100% of E**
MNIST-AE-1	96.59%	95.76%	95.91%	95.47%
MNIST-AE-3	97.27%	95.70%	95.31%	96.68%
MNIST-AE-5	96.83%	97.17%	97.02%	96.68%
MIMIC-III-MLP-0	94.28%	92.87%	91.20%	90.71%

Table 1: Ratio of pairs of neurons with t-values exceeding the critical values.

5 Experiments

This section evaluates the effectiveness of our CORR-Reg with diverse benchmark datasets and neural networks. The baseline regularization methods include L1-norm regularization (L1-reg) (Williams 1995), L2-norm regularization (L2-reg) (Hastie, Tibshirani, and Friedman 2001), Maxnorm (Srebro, Rennie, and Jaakkola 2005; Lee et al. 2010; Srebro and Shraibman 2005) and Dropout (Hinton et al. 2012; Srivastava et al. 2014). To understand the adaptive regularization of CORR-Reg, we also visualize the model parameter distribution during training.

5.1 Datasets

- **MNIST Dataset** [1]**:** This is a public database of handwritten digits which consists of 70000 28×28 images in 10 classes. The training and test datasets contain 60000 and 10000 images respectively.
- **CIFAR-10 Dataset**[2]**:** This is a public benchmark image classification dataset which consists of 60000 32×32 colour images in 10 classes, with 6000 images per class. There are 50000 training and 10000 test images.
- **MIMIC-III Dataset (Johnson et al. 2016):** This is a public benchmark dataset that includes various types of medical features of the patients. We study an 80-class classification problem that predicts the diagnoses of a patient given the medical history of 90 days. This dataset consists of 16248 samples, which we divide into 12379 samples for training and 3869 samples for testing. Each of the samples has 4351 features. We transform the irregular medical time series data into a regular one through resampling the data into 9 disjoint windows, taking the counts of the features within each 10-day window.
- **Sentence Polarity Dataset (Pang and Lee 2005):** This is a public benchmark dataset for sentiment analysis. It includes both positive and negative processed sentences, where each sentence is labeled with its overall sentiment polarity. We perform a binary classification task of sentiment analysis on this dataset. Each sentence is a sample with 5229 features and there are 5788 training samples and 1809 test samples.

Among the four datasets, MNIST and CIFAR-10 are standard benchmark image datasets on which mainstream NNs are evaluated. MIMIC-III and Sentence Polarity are datasets of real-world healthcare and natural language processing applications, which consist of complex raw features and require sophisticated NNs that are prone to overfitting.

5.2 Neural Network Configurations

Four types of neural network models are employed in the experiment. The first type is a simple Multilayer perceptron (MLP) model with two hidden layers of size 128. The second type is a Long short-term memory (LSTM) model with two hidden layers of size 128. The third type is a complex deep AutoEncoder (AE) with hidden layers of size 256, 128, 64, 32, 16, 32, 64, 128, 256. The last type corresponds to deep convolutional neural networks including LeNet and VGG. For LeNet, we construct a seven layer neural network according to (LeCun et al. 1998). The VGG model adopted has 16 weight layers (i.e., configurations D in (Simonyan and Zisserman 2014)). CORR-Reg is imposed on the model parameters between the hidden fully-connected layers.

In all experiments, the optimizer is SGD with a momentum of 0.9. The batch size is 128. We adopt training epochs 500 for MLP and LSTM, 200 for autoencoder and LeNet, and 300 for VGG.

5.3 Evaluation Metrics

For classification tasks on MNIST and CIFAR-10 datasets, we use error rate, the ratio of incorrect predictions, to measure the model performance. For the unsupervised learning task on MNIST dataset, we use mean squared error as

[1]http://yann.lecun.com/exdb/mnist/

[2]https://www.cs.toronto.edu/~kriz/cifar.html

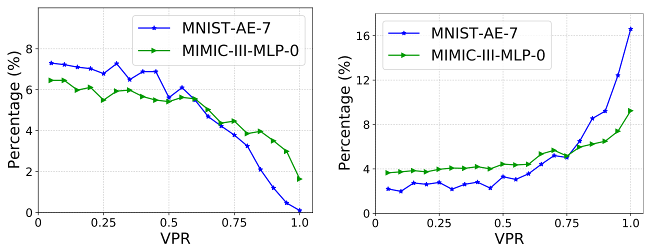
Model	Metric	L1-Reg	L2-Reg	Maxnorm	Dropout	CORR-Reg
MNIST-MLP	Error Rate (%)	1.95 ± 0.01	1.95 ± 0.01	1.94 ± 0.01	1.90 ± 0.03	**1.74 ± 0.01**
MNIST-LeNet	Error Rate (%)	0.88 ± 0.01	0.80 ± 0.01	0.80 ± 0.01	0.67 ± 0.02	**0.59 ± 0.01**
MNIST-AE	Reconstruction Error	8655.21 ± 2.69	8648.91 ± 1.81	8650.00 ± 3.76	8637.92 ± 2.58	**8619.99 ± 0.93**
CIFAR-10-VGG	Error Rate (%)	6.32 ± 0.04	6.21 ± 0.03	6.29 ± 0.06	6.23 ± 0.07	**5.77 ± 0.05**

Table 2: Performance comparison on benchmark image datasets.

Model	Metric	L1-Reg	L2-Reg	Maxnorm	Dropout	CORR-Reg
MIMIC-III-MLP	AUC (%)	86.19 ± 0.03	85.90 ± 0.01	86.22 ± 0.07	86.32 ± 0.02	**86.92 ± 0.04**
MIMIC-III-LSTM	AUC (%)	86.10 ± 0.05	86.05 ± 0.02	85.95 ± 0.06	86.19 ± 0.05	**86.60 ± 0.02**
Sentence-Polarity-MLP	AUC (%)	83.27 ± 0.05	82.42 ± 0.01	83.22 ± 0.06	83.14 ± 0.01	**83.55 ± 0.03**
Sentence-Polarity-LSTM	AUC (%)	83.47 ± 0.13	83.38 ± 0.08	83.08 ± 0.12	84.06 ± 0.13	**84.56 ± 0.07**

Table 3: Performance comparison on real-world applications.

the Reconstruction Error (RE) to measure the model performance. For both error rate and reconstruction error, smaller values indicate better performance. While for MIMIC-III and Sentence Polarity datasets, AUC (Area Under the ROC Curve) computed across all outputs is used as the evaluation metric, and larger AUC values indicate better performance.



(a) Bottom model parameters (b) Top model parameters

Figure 2: The distribution of VPR for bottom and top model parameters in MNIST-AE-7 and MIMIC-III-MLP-0.

5.4 Stability of Correlation Knowledge

Firstly, we conduct experiments to show that throughout the training process the correlation values between hidden layers are stable. At every quarter of the total training epochs E, we fix the trained model and feed the training data to the fixed trained model batch-wise. For each mini-batch, we can calculate a correlation value, $corr\_curr_{i,j}$, for each pair of the neurons. Denoting B as the number of mini-batches in the training set, for each pair with O_i and I_j, we can obtain B correlation values. Then, we perform a hypothesis testing on these B number of correlation values, to show that the correlation values are statistically significant instead of noises centered at zero.

The null hypothesis is that the mean of the B correlation values is zero. The significance level is set to 5%. We use t-values for the B correlation values for each neuron pair. Table 1 shows the ratio of pairs of neurons whose t-values exceed the critical values, i.e., whose correlation values' mean is significantly non-zero, on two representative models, namely deep autoencode on MNIST dataset (MNIST-AE) and MLP on MIMIC-III dataset (MIMIC-III-MLP). We denote the layers as "<dataset name>-<model name>-<layer number>". From the table, we can observe that throughout the training process, the correlation values of most pairs of neurons are not zero mean noises. Further, the ratio is stable. This confirms our assumption that correlation values between neurons exist and are stable, which means that such internal knowledge can be exploited to regularize the model training.

5.5 Comparison on Benchmark Image Dataset

In this section, we compare our CORR-Reg method with four state-of-the-art regularization methods, namely, L1-Reg, L2-Reg, Maxnorm and Dropout, on MNIST and CIFAR-10 datasets. For MNIST, we perform supervised learning tasks using MLP and LeNet and an unsupervised learning task using deep autoencoder. In this experiment, for both error rate and reconstruction error, the smaller the value, the better the performance. The results over three runs are summarized in Table 2. In this table, the best results are in bold and the second best results are underlined. As shown in the table, L1-Reg and L2-Reg achieve similar results since both of these two regularization methods add a regularization term which imposes the same strength of regularization for different model parameters prior to training. Dropout performs the best among the baseline regularization methods in three out of four models. CORR-Reg outperforms dropout consistently in all the models. This result confirms the superiority of our adaptive regularization which incorporates correlation knowledge to impose adaptive customized regularization on the model parameters.

5.6 Comparison on Real-World Applications

In this section, we compare different regularization methods on two real-world applications, namely, disease prediction and sentence polarity prediction. For these two applications, the larger the AUC, the better the performance. The results are summarized in Table 3. For the LSTM model, the input sequence lengths for MIMIC-III dataset and Sentence Polarity datasets are 9 and 25 respectively. As can be observed from the table, LSTM performs better than MLP in the sentence polarity dataset. The reason is that the sentence polarity prediction task aims to predict the sentiment label of

	CORR-Reg			L2-Reg		
Layer Name	**Bottom_Avg_VPR**	**Top_Avg_VPR**	**Avg_VPR_Dist**	**Bottom_Avg_VPR**	**Top_Avg_VPR**	**Avg_VPR_Dist**
MNIST-AE-1	40.28%	61.43%	21.15%	41.82%	58.58%	16.76%
MNIST-AE-3	31.27%	76.95%	45.68%	31.62%	75.14%	43.52%
MNIST-AE-5	36.40%	66.44%	30.04%	37.58%	65.44%	27.86%
MNIST-AE-7	43.46%	57.29%	13.83%	45.70%	55.54%	9.84%
MIMIC-III-MLP-0	37.34%	68.08%	30.74%	44.02%	60.64%	16.62%

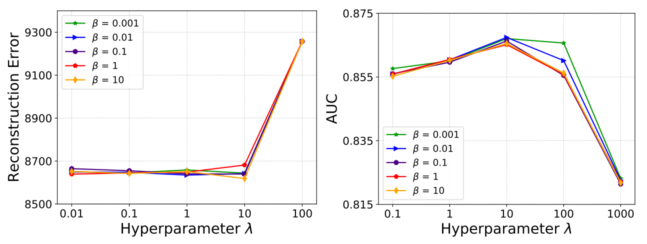
Table 4: Average VPR for bottom and top model parameters.

a sentence according to the word sequence of the sentence. As a result, LSTM, which is designed for dealing with sequential data input, performs better in such a learning task. In terms of the comparison among different regularization methods, we observe a consistent improvement of CORR-Reg over the other four state-of-the-art regularization methods, which is in line with the observations in Table 2 and again confirms the effectiveness of our knowledge-driven adaptive regularization method.

5.7 Effects of Adaptive Regularization

To verify that CORR-Reg imposes adaptive customized regularization on the model parameters during training, we visualize the model parameter distribution for MNIST-AE and MIMIC-III-MLP. At the last epoch of training, we take out the model parameters and the correlation values between the hidden layers. For each model parameter, there is a corresponding correlation value, defined in Equation 4, between the two neurons it connects. For the model parameters from the same layer, we obtain two percentile ranks for each of them. The first, denoted as Value Percentile Rank (VPR), is the percentile rank of the absolute value of the model parameter. The second, denoted as Correlation Percentile Rank (CPR), is the percentile rank of the corresponding correlation value. In each layer, we denote the model parameters that fall in the bottom 25% CPR as *bottom model parameters* and the top 25% as *top model parameters*. For the bottom and top model parameters, the distribution of their VPR for two representative layers is shown in Figure 2. In this figure, we divide VPR, which ranges from 0 to 1, into 20 bins. Each point shows the percentage of model parameters whose VPR values fall into the corresponding bin. We first focus on the VPR distributions of bottom model parameters. As can be observed, the percentage of model parameters decreases quickly as VPR increases. This is expected since for the bottom model parameters, their corresponding correlation values are smaller and thus they are imposed stronger regularization. As a consequence, most of the bottom model parameters have small VPR. The distribution of top model parameters shows a reversed trend, and the explanation is the opposite.

To further verify the effects of the correlation-based regularization, we show in Table 4 the average VPR for bottom and top model parameters, denoted as Bottom_Avg_VPR and Top_Avg_VPR respectively, and their distance (i.e., Top_Avg_VPR - Bottom_Avg_VPR), denoted as Avg_VPR_Dist, for both neural networks with CORR-Reg and neural networks with L2-Reg that imposes the same regularization strength on all model parameters. As can be observed from Table 4, one key observation is that the Avg_VPR_Dist of CORR-Reg is significantly larger than that of L2-Reg. This can be attributed to the fact that with CORR-Reg, bottom model parameters receive stronger regularization whereas top model parameters receive weaker regularization, which further confirms that the correlation knowledge is exploited to impose customized regularization on model parameters.



(a) MNIST-AE (b) MIMIC-III-MLP

Figure 3: Performance for different λ and β values.

5.8 Effectiveness of Hyperparameters

As shown in Equations 3 and 8, both λ and β controls the learning of significance weights. In this section, we investigate the effect of λ and β on the model training. Figure 3 shows the performance for different combinations of λ and β values on MNIST-AE and MIMIC-III-MLP. We can observe that CORR-Reg with smaller λ values performs better than the one with larger λ values. While a large λ, e.g., 100 and 1000, also hurts the model's performance. This is due to the fact that a larger λ incurs excessive regularization, which restricts model capacity and representation learning. While for β, we can observe that as long as the value is between 0.001 and 10, the results are promising.

6 Conclusion

In this paper, we propose an adaptive knowledge driven regularization method based on correlation, called CORR-Reg, to incorporate internal knowledge into deep neural networks. An efficient algorithm is designed to update significance weights by applying the Lagrange multiplier method and update model parameters via SGD. Experiments show that our CORR-Reg regularization achieves consistently better performance than existing state-of-the-art regularization methods for diverse neural networks.

7 Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This research is supported by Singapore Ministry of Education Academic Research Fund Tier 3 under MOE's official grant number MOE2017-T3-1-007.

References

Bengio, Y.; Courville, A.; and Vincent, P. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(8): 1798–1828.

Brust, C.-A.; and Denzler, J. 2018. Integrating domain knowledge: using hierarchies to improve deep classifiers. *arXiv preprint arXiv:1811.07125* .

Che, Z.; Kale, D.; Li, W.; Bahadori, M. T.; and Liu, Y. 2015. Deep Computational Phenotyping. In *twenty-first ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Cogswell, M.; Ahmed, F.; Girshick, R.; Zitnick, L.; and Batra, D. 2015. Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068* .

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*, 248–255.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The Elements of Statistical Learning*. Springer New York Inc.

He, D.; Xia, Y.; Qin, T.; Wang, L.; Yu, N.; Liu, T.; and Ma, W.-Y. 2016a. Dual Learning for Machine Translation. In *Advances in Neural Information Processing Systems*, 820–828.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

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

Johnson, A. E.; Pollard, T. J.; Shen, L.; Li-wei, H. L.; Feng, M.; Ghassemi, M.; Moody, B.; Szolovits, P.; Celi, L. A.; and Mark, R. G. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* .

Kneib, T.; Konrath, S.; and Fahrmeir, L. 2011. High dimensional structured additive regression models: Bayesian regularization, smoothing and predictive performance. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 60(1): 51–70.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Krogh, A.; and Hertz, J. A. 1991. A Simple Weight Decay Can Improve Generalization. In *Advances in Neural Information Processing Systems*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360* .

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11): 2278–2324.

Lee, J. D.; Recht, B.; Srebro, N.; Tropp, J.; and Salakhutdinov, R. R. 2010. Practical large-scale optimization for max-norm regularization. In *Advances in neural information processing systems*, 1297–1305.

Luo, Z.; Cai, S.; Chen, G.; Gao, J.; Lee, W.-C.; Ngiam, K. Y.; and Zhang, M. 2019. Improving Data Analytics with Fast and Adaptive Regularization. *IEEE Transactions on Knowledge and Data Engineering* .

Luo, Z.; Cai, S.; Gao, J.; Zhang, M.; Ngiam, K. Y.; Chen, G.; and Lee, W.-C. 2018. Adaptive Lightweight Regularization Tool for Complex Analytics. In *thirty-fourth IEEE International Conference on Data Engineering*.

Meinshausen, N.; and Bühlmann, P. 2006. High-dimensional graphs and variable selection with the Lasso. *The Annals of Statistics* 34(3): 1436–1462.

Mesnil, G.; He, X.; Deng, L.; and Bengio, Y. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, 3771–3775.

Ooi, B. C.; Tan, K.-L.; Wang, S.; Wang, W.; Cai, Q.; Chen, G.; Gao, J.; Luo, Z.; Tung, A. K.; Wang, Y.; et al. 2015. SINGA: A distributed deep learning platform. In *Proceedings of the ACM international conference on Multimedia*, 685–688.

Pang, B.; and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the fifty-third Annual Meeting of the Association for Computational Linguistics*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision* 115(3): 211–252.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Srebro, N.; Rennie, J.; and Jaakkola, T. S. 2005. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, 1329–1336.

Srebro, N.; and Shraibman, A. 2005. Rank, trace-norm and max-norm. In *International Conference on Computational Learning Theory*, 545–560. Springer.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15(1): 1929–1958.

Srivastava, N.; and Salakhutdinov, R. R. 2013. Discriminative transfer learning with tree-based priors. In *Advances in Neural Information Processing Systems*, 2094–2102.

Tay, Y.; Luu, A. T.; Hui, S. C.; and Su, J. 2018. Densely Connected Attention Propagation for Reading Comprehension. In *Advances in Neural Information Processing Systems*, 4906–4917.

Tran, T.; Nguyen, T. D.; Phung, D.; and Venkatesh, S. 2015. Learning vector representation of medical objects via EMR-driven nonnegative restricted Boltzmann machines (eNRBM). *Journal of biomedical informatics* 54: 96–105.

Veit, A.; Wilber, M. J.; and Belongie, S. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, 550–558.

Wang, W.; Chen, G.; Dinh, A. T. T.; Gao, J.; Ooi, B. C.; Tan, K.-L.; and Wang, S. 2015. SINGA: Putting deep learning in the hands of multimedia users. In *Proceedings of the ACM international conference on Multimedia*, 25–34.

Wang, W.; Zhang, M.; Chen, G.; Jagadish, H.; Ooi, B. C.; and Tan, K.-L. 2016. Database meets deep learning: Challenges and opportunities. *ACM SIGMOD Record* 45(2): 17–22.

Wang, Y.; Sun, X.; and Liu, L. 2016. A variable step size LMS adaptive filtering algorithm based on L2 norm. In *2016 IEEE International Conference on Signal Processing, Communications and Computing*, 1–6.

Williams, P. M. 1995. Bayesian regularization and pruning using a Laplace prior. *Neural computation* 7(1): 117–143.

Yang, Z.; Hu, Z.; Dyer, C.; Xing, E. P.; and Berg-Kirkpatrick, T. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, 7287–7298.

Zhang, C.; Kumar, A.; and Ré, C. 2014. Materialization Optimizations for Feature Selection Workloads. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 265–276.

Zhang, L.; Zhu, G.; Mei, L.; Shen, P.; Shah, S. A. A.; and Bennamoun, M. 2018. Attention in Convolutional LSTM for Gesture Recognition. In *Advances in Neural Information Processing Systems*, 1953–1962.

Zhu, X.; Zhou, W.; and Li, H. 2018. Improving Deep Neural Network Sparsity through Decorrelation Regularization. In *Ijcai*, 3264–3270.