


A Simple Stochastic Regularization Technique for Avoiding Overfitting in Low Resource Image Classification

Y.T Ji^{†1} , B.L Wang¹, Q.D.E.J Ren¹, B Shi¹, N.E Wu¹, M Lu¹, N Liu¹, X.F Zhuang¹, X.X Xu¹, L Wang², L.J Dai², M.M Yao², X.M Li²,

¹Inner Mongolia University of Technology, China ²National Testing Center of Wool and Cashmere Quality, China

Abstract

Drop type technique, as a method that can effectively regulate the co-adaptations and prediction ability of neural network units, is widely used in model parameter optimization to reduce overfitting problems. However, low resource image classification faces serious overfitting problems, and the data sparsity problem weakens or even disappears the effectiveness of most regularization methods. This paper is inspired by the value iteration strategy and attempts a Drop type method based on Metcalfe's law, named Metcalfe-Drop. The experimental results indicate that using Metcalfe-Drop technique as a basis to determine parameter sharing is more effective than randomly controlling neurons according to a certain probability. Our code is available at <https://gitee.com/giteetu/metcalfe-drop.git>.

CCS Concepts

• **Computing methodologies** → **Image processing; Neural networks; Image representations;**

1. Introduction

The Drop type method, as a stochastic technique used for neural network training and inference, has made use of noise to improve performance recently, because neural noise plays a fundamental role in the process of learning and should be studied more thoroughly. Many outstanding works have successfully applied Drop like techniques to various applications, including neural network regularization, model compression, and in measuring the uncertainty of neural network outputs. However, for low resource image classification, serious overfitting often leads to some features being considered as noise, which can be viewed as preventing neurons from co-adapting.

In this paper, we address this question on the proposed Metcalfe-Drop method to find a flexible parameter sharing method for overfitting. To achieve this goal, the main contributions of this paper are as follows: ★ This paper proposes a Drop type method based on Metcalfe's law, Metcalfe-Drop, to explore parameter sharing methods in neural networks and reduce overfitting caused by resource scarcity. ★ We have constructed multiple strong baseline systems on specific and public datasets, including Resnet-50, VGG-16, Inception V2, and Conv Net, to validate the effectiveness of the Metcalfe-Drop method. The experimental results show that using Metcalfe-Drop to control the training neurons can achieve parameter sharing of neural networks in a new way and solve overfitting problems.

2. Methodology

The description of value and number of neurons in Metcalfe's law is intuitive and simple, as shown in formula(1-(i)). When it is as-

sociated with neural network training, formula(1) can be seen as a linear representation between the training value V and the neuron n . Setting the value coefficient K as a hyperparameter can make the Metcalfe-Drop method better adapt to a variety of loss functions. For the general low resource classification research, we suggest that it should be set as a constant of no more than 0.5. For the two datasets in this paper, we set $k=0.2$ and 0.45 as the most effective values. This paper aims to explore low resource tasks that do not rely on sufficient training. Therefore, optional, k can also be obtained through iterative training of each batch. The constructed value iteration training expects to obtain the determined number of neurons n through training value V . Therefore, under the constraint of $n \in [1, \infty]$, formula(1-(i)) can be transformed into formula(1-(ii))

$$V = K \times n^2 \quad (i) \Rightarrow n = \sqrt{\frac{V}{K}} \quad (ii) \quad (1)$$

Then, how to get the training value is the key to inference. Inspired by VI, it is intuitive and reasonable to regard training loss as training value. The effectiveness of such strategy has been fully verified in the research of NLP. It is worth noting that more than one loss function can be applied to this strategy. Here, in order to avoid a large number of normalization calculations, we take softmax loss as an example to show the transformation process from training loss to training value. Specifically, the prediction probability of the i -th sample for class j can be expressed as $p_{ij} = \frac{e^{l_{ij}}}{\sum_{m=j}^C e^{l_{i,m}}}$. Where l_{ij} represents the logit output by the neural network for the i -th sample in the j -th class. Logit represents the output of the last full connection layer. C is the total number of categories. Therefore, the

softmax loss can be expressed as (2)-(i).

$$L_{SE} = \frac{1}{n} \sum_{i=1}^n (1 - \log p_i, y_i) = \frac{1}{n} \sum_{i=1}^n \left(-\log \frac{e^{iy_i}}{\sum_{j=1}^C e^{ij}} \right) \quad (i)$$

$$L_C = \frac{1}{2} \sum_{i=1}^n \|x_i - C_{y_i}\|_2^2 \quad (ii)$$

However, the sparse features of low resource data will lead to a large gap between the feature vector and the label, which further leads to the model's lack of ability to distinguish similar categories. Therefore, we add a central loss L_C for softmax loss, as shown in formula(2-(ii)). Where C_{y_i} is the embedding center of the label corresponding to the i -th sample. N is the number of samples for the current training. Therefore, the loss of model training is expressed as $Loss = L_{SE} + L_C$. Then, two types of loss values, V_{SE} and V_C , can be obtained through value transformation. V_{SE} and V_C can be expressed as $V_{SE} = Z_{Score}(L_{SE})$ and $V_C = Z_{Score}(L_C)$. $Z_{Score}(X) = (X - X_{mean})/X_{std}$ is a common normalization method, Where X_{mean} or X_{std} , as an intermediate parameter, can be easily obtained in a batch training. Therefore, the training value can be determined by setting a hyper-parameter α expressed flexibly as $V = |\alpha V_{SE} + (1 - \alpha)V_C|$.

To sum up, the feed forward operation with Metcalfe-Drop can be clearly described as $r_i^{Metcalfe} \sim \text{Bernoulli}(p^{Metcalfe})$. The presentation of neuron is transformed into $\tilde{y}_i = r_i^{Metcalfe} \bullet y_i$, and used to calculate hidden layer output $o_i = w_i \tilde{y}_i + b_i$, thereby obtaining the output of the next neuron $y_{i+1} = f(o_i)$, where $f()$ is any activation function. Here \bullet denotes an element-wise product. Let o_i denote the vector of inputs into layer, y_i denote the vector of outputs from layer ($y_0 = x$ is the input). W_i and b_i are the weights and biases at a layer. For any layer, r_i is a vector of independent Bernoulli random variables each of which has probability $p^{Metcalfe}$ of being 1. This vector is sampled and multiplied element-wise with the outputs of that layer, y_i , to create the thinned outputs. During the test time, the parameters of each unit are multiplied by $\frac{n}{N}$.

So far, we can derive the optimized number of neurons through Metcalfe value, so as to further provide a reliable probability $p^{Metcalfe} \sim \frac{n}{N} = \frac{1}{N} \sqrt{|V|K}$ for the sampling basis in Drop type methods.

3. Experiment

We trained Metcalfe-Drop neural networks for classification problems on data sets in two different domains, including extremely low resource dataset of wool and cashmere fiber image(0.8K for training set and 0.2K for test set) and common dataset CIFAR-10/100. The CIFAR dataset is divided into two versions, including the abridged version(2K for training set and 0.5K for test set) and the normal version(60K for training set and 10K for test set). The abridged version is used to test the effectiveness of the method in low resource environments, while the normal version is used to verify the commonality of the method. We found that Metcalfe-Drop improved generalization performance on all data sets compared to neural networks that use other Drop type techniques.

We constructed several strong baseline models based on a variety of methods. Table1 shows the error rate obtained by different baselines on fiber image and CIFAR-10/100.

Table 1: Error rates on fiber image dataset and CIFAR-10/100

Method	fiber image	CIFAR-10	CIFAR-100
Resnet-50(Shen et al., 2022)	17.71	39.65	52.20
Resnet-50+Dropout	13.60	38.31	50.70
Resnet-50+Dropconnect	13.88	34.43	55.10
Resnet-50+Metcalfe-Drop	12.51	34.00	46.37
Resnet-50+Metcalfe-Dropconnect	10.75	34.90	48.18
VGG-16(Simonyan et al., 2015)	23.50	41.77	63.55
VGG-16+Dropout	23.15	38.00	57.61
VGG-16+Dropconnect	21.20	37.32	58.27
VGG-16+Metcalfe-Drop	20.58	37.20	51.35
VGG-16+Metcalfe-Dropconnect	18.31	33.24	58.10
Conv Net+Dropout	17.44	39.90	55.10
(Snoek et al., 2012)			
Conv Net+Maxout	13.37	41.17	58.14
(Goodfellow et al., 2013)			
Conv Net+Metcalfe-Drop	11.56	37.22	54.50
GoogleLeNet	14.63	39.90	51.02
(Szegeedy et al., 2015)			
GoogleLeNet+Metcalfe-Drop	13.27	39.95	43.08

Note that training a network of 25 million parameters to give good generalization error is very hard with standard regularization methods and early stopping. Metcalfe-Drop prevent overfitting and does not even need early stopping. Due to the sparsity of activation units in low resource classification tasks, we observed that the sparsity of hidden unit activations on a random mini-batch taken from the test set. Figure 1 compares the sparsity for the Resnet-50 with/without Metcalfe-Drop. As we thought, a good sparse model should only be a few highly activated units for any dataset. Thus, we plot histograms to observe the sparsity performance of Metcalfe-Drop in neural network training. Comparing the results, we can see that fewer hidden units have high activations in figure1(b) and figure1(d) compared to figure1(a) and figure1(c), as seen by the significant mass away from zero that does not use Metcalfe-Drop.

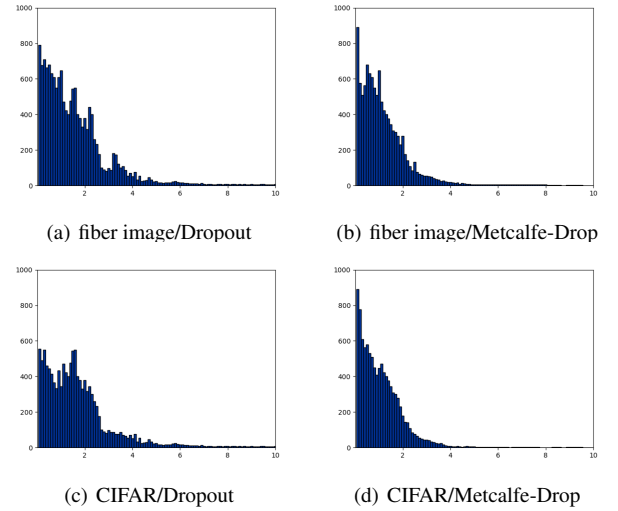


Figure 1: Effect of dropout on sparsity.

4. Conclusions

Considering that Drop-type technique was found to improve the performance of neural nets in a wide variety of application domains including object classification, digit recognition, speech recognition, document classification and analysis of computational biology data. We expect to apply Metcalfe-Drop technique to model training in other fields in the follow-up research.