Reduce Misjudgment by Using Threshold Filtering Method in Multi Classification Tasks

Hongrui Lu

Research School of Computer Science, Australian National University, Canberra Australia

u7094915@anu.edu.au

Abstract. *For some classification tasks, even the state-of-the-art models can not achieve high accuracy. Low accuracy means more misjudgment. However, in some application scenarios, a small amount of misjudgment is enough to cause serious consequences. In this work we present an automatic threshold filtering method for multi classification tasks. This algorithm can filter the outputs with "low confidence" and mark them as "uncertain" classification results. By using the threshold filtering method, users can increase the accuracy of network output at the cost of reducing the recognition rate. We tested the effect of threshold filtering on a well trained ResNet-18 network. The classification accuracy of the network was improved from 92.5% to 98.0%, and 69% of the samples can still be recognized.*

1 Introduction

Our research goal is to develop a general threshold filtering mechanism for neural networks, which can sift out more reliable outputs of neural networks in real time.

A deep learning model with low classification accuracy is difficult to be applied to real scenes, even if it is state of the art in the field. Because in some actual tasks, a wrong judgment can lead to serious adverse consequences. In automatic driving task, misclassification of obstacles may cause catastrophic traffic accidents. In the automatic alarm system, if the alarm is often mistakenly triggered, the rescue resources may be wasted. In the robot mine clearance task, the wrong classification of mine types may cause the mine clearance robot to be damaged.

For a neural network, not every output is reliable. Neural networks with low accuracy may not work unsupervised, but they can still assist humans in these tasks. In this case, the neural network only outputs highly reliable classification results, and those "uncertain" samples are handed over to human beings for classifying, so that the neural network can share part of the task for human beings on the premise of ensuring the accuracy of classification.

Manual output threshold adjustment[1] has previously been used in binary classification task to reduce false positive results. Inspired by this, we tried to apply threshold filtering to multi classification task. It is difficult to adjust the threshold manually for multi classification task, especially for a deep neural network. Therefore, an algorithm for automatically calculating threshold group is proposed. The core idea of the algorithm is to find the highest error response of the network to each class in the training set. The detailed calculation method will be explained in the "method" section.

In order to verify the effectiveness of threshold filtering method, a pre trained ResNet-18 network was fine tuned on the Vehicle-X dataset and the effect of threshold filtering was tested on the test set.

Vehicle-X is a large-scale virtual vehicles dataset generated by the Unity rendering engine[2]. Its training set contains 45438 pictures, the test set contains 14936 pictures, and the validation set contains 15142 pictures. Each image in the dataset was resized in advance, with a size of 256 * 256 pixels, 96 dpi. The dataset provides a variety of tags, so the models can carry out vehicle recognition task or vehicle type classification task.

Figure 1: The original vehicle images in the dataset, dis-played in Unity.

Because the virtual vehicle images in the Vehicle-X dataset are very close to the real photos, the model trained on it can be applied to real vehicle classification through transfer learning.

2 Methodology

The data preprocessing method, the structure of ResNet-18 and the process of automatic threshold calculation are described in detail in this section.

2.1 Data Pre-processing & Data Augmentation

Normalization

The normalization process makes the values of of each pixel in each channel be mapped between 0-1. In this process, the proportional relationship between the values remains unchanged. The expression of min-max normalization is:

$$x' = \frac{x - \min(X_c)}{\max(X_c) - \min(X_c)}$$

Where X_c is a color channel of the image. x' is the normalized value, x is the original value. Min( ) and Max( ) are functions for finding the maximum and minimum value in 2D matrix.

Standardization

The standardization process reorganizes the data distribution into standard normal distribution. The formula for image standardization is as follows:

$$x' = \frac{x - \mu_c}{\sigma_c}$$

Where μ_c is the mean of one channel of all images in the dataset, σ_c is the standard deviation of one channel of all images in the dataset. x' is the standardized value, x is the original value.

Random Horizontal Flip

This process simply flips the input image left and right with a 50% probability. This step can introduce more randomness to the dataset, improve the generalization ability of the model and prevent overfitting.

2.2 Fine Tuning of the ResNet-18 model

Model Structure

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2: Architecture of ResNet-18 for ImageNet, provided in the original paper of ResNet[3]

In the experimental stage, the main model used was the ResBet-18 model provided by Pytorch. The model contains pre-training weights on ImageNet.

The full connection layer of the network was replaced by a hidden layer with 256 neurons and an output layer corresponding to the total number of classes. ReLU activation function was used between the two layers, and a dropout layer[4] with p = 0.5 is added.

Loss Function

Cross Entropy Loss is a widely used loss function in multi classification tasks. Its mathematical expression is:

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -\sum_{c=1}^{M} y_{ic}\log(p_{ic})$$

Where M is the number of different classes. If sample i is correctly classified, y_{ic} is 1, otherwise it is 0. And p_{ic} is the prediction probability of sample i belonging to category c.

Optimizer

Adam Optimizer[5] based on adaptive low order moment estimation. It is easy to implement, and has high computational efficiency and low memory requirements.

2.3 Automatic Threshold Calculation

For the neural network, the output value of the output layer neuron reflects the judgment of the neural network on the category of the sample. We can obtain such information by recording the output of the model for the samples in training set: If sample X does not belong to class A, then the output value of the output neuron corresponding to class A of the model will probably not exceed which number. This "number" is the basic threshold of class A.

Our algorithm is to automatically calculate the basic thresholds corresponding to each class. The threshold array composed of basic thresholds can be used to filter the output of the model. This idea is essentially the same as the idea of manually adjusting the threshold.

Before calculating the threshold, the model should have been trained. In the process of threshold calculation, back propagation is not carried out. Suppose the number of training samples is n and the total number of classes is m.

Step 1: Calculate the output vector of the model for each sample in the training set.

$$output_i = \text{model}(input_i)$$

Step 2: Set the number corresponding to the real class in the output vector to negative infinity.

$$output_i[label_i] = -\text{infinity}$$

Step 3: Stack all the output vectors vertically to form an n* m matrix O.

$$O = \begin{bmatrix} output_0 \\ \cdots \\ output_n \end{bmatrix}$$

Step 4: Threshold array is the column maximums of the matrix O.

$$thresholds_i = \max(O[:, i])$$

$$threshold\_array = \begin{bmatrix} thresholds_0 & \cdots & thresholds_m \end{bmatrix}$$

2.4 Threshold Filtering Method

Before using the threshold method, we first introduce a new hyper-parameter, “magnification”. Multiply the base thresholds by it before filtering the model output using the threshold array, then the trade-off between accuracy and recognition rate can be adjusted by changing the value of threshold magnification.

The method of filtering model output using threshold array is very simple: If the output value of each output neuron does not exceed the corresponding threshold in the threshold array, then mark the output as “uncertain” or “unrecognized” .

$$\begin{aligned} &\text{if} \quad \max(output_i - threshold_i \times magnification) < 0 \\ &\text{then} \quad output_i = \text{uncertain} \end{aligned}$$

As with all other hyper-parameters, the value of “magnification” can be tested on the validation set until the most appropriate value is selected.

3 Results and Discussion

In the process of experiment, the accuracy of ResNet-18 was first optimized by hyper-parameter adjustment. The final best performing model was used to test the effect of threshold filtering method.

Due to the limitation of computing power, the dataset used by the model was a subset of the Vehicle-X dataset in the experiments of hyper-parameter selection, and the size is as follows: Train Set - 10000 samples, Validation Set - 3000 samples, Test Set - 3000 samples

3.1 The Selection of Hyper-Parameters

ResNet Model

Model	Requires_grad	Validation Accuracy							Average Epoch Time
		epoch 1	epoch 4	epoch 7	epoch 10	epoch 13	epoch 16	Best	
ResNet-18	True	79.93%	84.57%	85.23%	90.60%	89.93%	90.03%	90.97%	57.4223s
ResNet-50	True	64.77%	78.97%	85.93%	82.17%	85.93%	87.97%	89.47%	107.0633s
ResNet-18	False	43.03%	53.10%	56.67%	57.53%	58.33%	58.90%	60.77%	22.4918s
ResNet-50	False	43.83%	56.27%	56.77%	60.67%	61.20%	61.20%	63.23%	47.7017s
ResNeXt-50	False	44.97%	54.47%	57.20%	57.50%	60.17%	59.53%	61.93%	49.2284s

Table 1: Model selection and the setting of Requires_grad

When “Requires_grad” is False, the weight of the pre training model is frozen, and only the full connection layer participates in the back propagation.

Final choice: Model = ResNet-18, Requires_grad = True
(Unlisted hyper-parameters: Loss Function = Cross Entropy Loss, Optimizer = Adam Optimizer, Max Epochs = 30, Batch Size = 128, Learning Rate = 0.0005, Standardization = False)

Pre Training & Normalization

Pre Training	Standardization	Validation Accuracy							Average Epoch Time
		epoch 1	epoch 4	epoch 7	epoch 10	epoch 13	epoch 16	Best	
False	True	27.20%	51.60%	60.83%	65.77%	77.07%	76.13%	82.56%	57.7589s
True	True	75.50%	86.43%	87.37%	90.77%	86.77%	90.33%	92.06%	56.3381s
True	False	79.93%	84.57%	85.23%	90.60%	89.93%	90.03%	90.96%	57.4223s

Table 2: Influence of pre training and standardization on model performance

When “Pre Training” is False, the model uses randomly initialized weights at the beginning.
When “Standardization” is False, the input images are only normalized and not standardized.

Final choice: Pre Training = True, Standardization = True
(Unlisted hyper-parameters: Model = ResNet-18, Requires_grad = True, Loss Function = Cross Entropy Loss, Optimizer = Adam Optimizer, Max Epochs = 30, Batch Size = 128, Learning Rate = 0.0005)

Learning Rate

Learning Rate	Validation Accuracy							Average Epoch Time
	epoch 1	epoch 4	epoch 7	epoch 10	epoch 13	epoch 16	Best	
0.0001	69.97%	89.23%	91.97%	91.47%	91.63%	92.20%	92.87%	57.7379s
0.0005	75.50%	86.43%	87.37%	90.77%	86.77%	90.33%	92.06%	56.3381s
0.00005	59.87%	85.13%	90.83%	91.57%	91.93%	92.17%	92.60%	57.6734s
0.00001	30.57%	57.73%	71.37%	78.40%	82.07%	84.97%	89.77%	57.8589s

Table 3: Choose an appropriate learning rate

Final choice: Learning Rate = 0.0001
(Unlisted hyper-parameters: Model = ResNet-18, Pre Training = True, Requires_grad = True, Standardization = True, Loss Function = Cross Entropy Loss, Optimizer = Adam Optimizer, Max Epochs = 30, Batch Size = 128)

3.2 Model Training

Hyper-parameters:
Model = ResNet-18, Pre Training = True, Requires_grad = True, Standardization = True, Loss Function = Cross Entropy Loss, Optimizer = Adam Optimizer, Max Epochs = 29, Batch Size = 128, Learning Rate = 0.0001

(Train with the complete training set, 45437 samples)

Validation Accuracy											Average Epoch Time
epoch 1	epoch 3	epoch 5	epoch 7	epoch 9	epoch 11	epoch 13	epoch 15	epoch 17	epoch 19	epoch 21	
69.97%	86.00%	90.30%	91.97%	92.23%	90.90%	91.63%	92.50%	92.87%	92.37%	92.43%	57.7379s

Table 4: The final training result

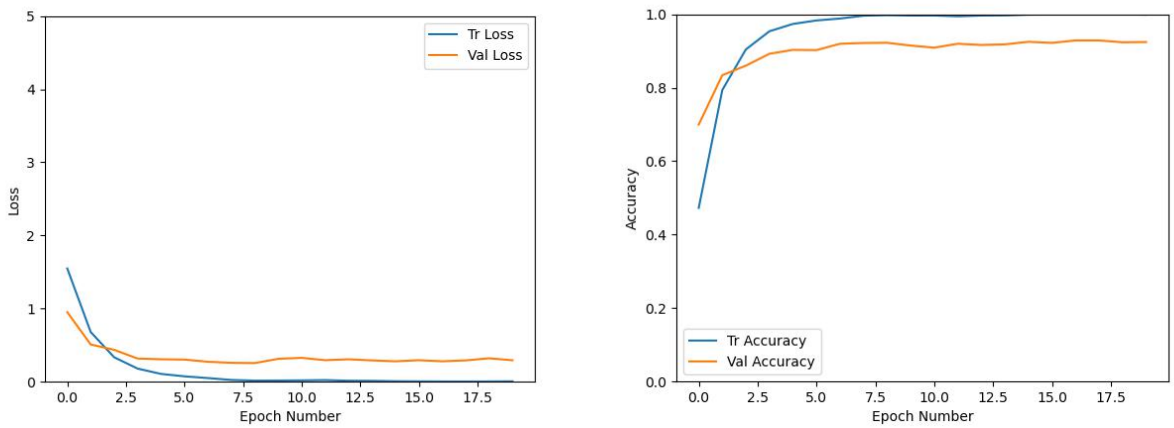


Figure 3: Accuracy and loss of the model in the training process

Training Accuracy	Validation Accuracy	Test Accuracy
100.0000%%	92.9333%	92.6333%

Table 5: Overall accuracy of the model

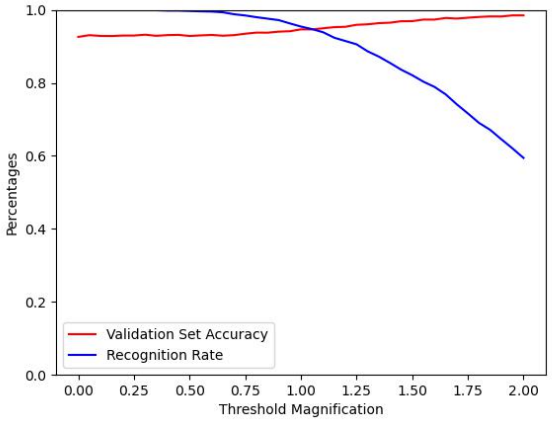
It can be seen that the model was slightly overfitted, this is because only a part of the training samples in the dataset are used in the training process.

3.3 Applying threshold filtering method

First, the threshold array of the model was calculated (11 thresholds corresponding to 11 vehicle types)

1	2	3	4	5	6	7	8	9	10	11
4.646	3.170	2.937	4.051	3.549	5.380	3.167	3.636	4.228	6.185	3.967

Then, threshold filtering was tested on the verification set to select the appropriate threshold magnification.



When Magnification = 1.8, the recognition Rate of the model is 69.0333%, and the accuracy of the model on validation set is 98.0686%. The original accuracy of the model is about 92.5%. Therefore, the thresholding method improves the accuracy by 5.5%.

4 Conclusion and Future Work

We have successfully developed an automatic thresholding method, which includes the automatic thresholds calculation algorithm and the threshold filtering method part. This threshold method can screen out the high reliability output of neural network, so as to reduce the misjudgment of neural network. The balance between accuracy and recognition rate can be adjusted by modifying the value of hyper-parameter "magnification".

In order to verify the effectiveness of the thresholding method, two neural networks trained on the Vehicle-X dataset for different classification targets are developed. Through hyper-parameters tuning, both networks achieve relatively high

accuracy. The threshold filtering method improves their classification accuracy successfully, and the threshold array calculated by the automatic threshold generation algorithm performed better than the thresholds generated randomly.

For the future work, we plan to apply threshold filtering method in more networks, packaging the algorithm as simple and clear modules. We will study more network structures to see if the threshold method is effective for each neural network.

In this paper, we have extended the application of thresholding method from binary classification problem to arbitrary multi classification problem. We want to find out if similar ideas can also be applied to regression problems.

References

[1] Milne L K, Gedeon T D, Skidmore A K. Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood[J]. training, 1995, 109(81): 0.

[2] Yao Y, Zheng L, Yang X, et al. Simulating content consistent vehicle datasets with attribute descent[J]. arXiv preprint arXiv:1912.08855, 2019.

[3] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.

[4] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. The journal of machine learning research, 2014, 15(1): 1929-1958.

[5] Kingma D P, Ba J. Adam: A method for stochastic optimization[J]. arXiv preprint arXiv:1412.6980, 2014.