Dynamic Model Construction for Efficient Classification

Jaejun Lee
School of Computer Science
University of Waterloo
Waterloo, ON, N2L 3G1
j474lee@uwaterloo.ca

Abstract

One of the major drawback of neural network in the domain of classification is that retraining is unavoidable when the target set of class changes. For this reason, networks are often designed to be wide and deep. However, this leads to increase the necessary computations which has a direct impact on the efficiency of the model. In this work, I study how different combination of loss function and last activation function affects the classification output and present a new way to add or remove a class from target class set without retraining. With this technique, a model can be adjusted to classify any combination of target classes and the minimal resource usage is guaranteed as the adjusted model involves the same amount of computations as the model trained to classify the same set of class explicitly.

1 Introduction

Current issues

In this section you are going to present a brief background and motivation of your project. Why is it interesting/significant? How does it relate to the course?

2 Related Works

The main challenge with this problem is on model flexibility and resource usage. Even though these two aspects are quite related, they are generally considered as two separate problem. As a result, this problem can be categorized differently depending on which direction one approaches. The three most related domains are: ensemble learning, multi-task learning, and transfer learning

2.1 Ensemble Learning

Ensemble learning is a common technique in the field of machine learning which achieves greater performance by combining outputs from multiple models which are independent. The most famous techniques are voting, weighting, bagging and boosting [1–3]. Even though ensemble learning is considered to be easy to implement, most of the ensemble techniques assume that each model has different architecture. As a result, ensemble learning often requires significant computations as it requires every model to process the input.

Submitted to Introduction to Machine Learning (CS698, 2017 Fall). Do not distribute.

2.2 Multi-task Learning

On the other hand, multi-task learning takes different approach; combining multiple models into a single model. The key assumption is that if tasks are related, sharing information among tasks throughout training will increase the performance on each task. There are numbers of variations depending on the assumptions. Key design choices include: what kind of information to share and how much information to share among tasks. Two main categories of multi-task learnings are parameter sharing and feature sharing [4–7] which differ by type of the information that are shared. Even though multi-task learning has the similar architecture with the desired solution, it is hard to adopt it as a solution for our problem, due to it's assumption that the each task are distinct.

2.3 Transfer Learning

Transfer learning is inspired by the same assumption as in multi-task learning; if tasks are related, sharing knowledge improves the performance. However, the key difference between two problem is that transfer learning are designed to share the same model architecture for different tasks. First, a model is pre-trained with a task and learns to select important features. Then the model is fine-tuned for the target task, putting all of its effort to produce the best result for the target task exploiting selected features [8]. With its architectural flexibility, transfer learning has been adopted in wide domains and has shown its strengths [9–11].

3 Dynamic Model Construction with Class-level Transfer Learning

3.1 Approach

In this section, I introduce how to achieve dynamic model construction which adopts to the change in target class set.

One of the common technique used for multi-task learning is to use distinct fully-connected layer for each task [12–14]. In this setting, each task is trained independently while weights for upstream layers are shared among classes. Once model is trained, a task can be discarded from populating output if it is no longer necessary. This is achieved simply by removing corresponding fully-connected layer.

Next consider a case of transfer learning. Pre-training a model and fine-tuning for target task can leads to increase in performance. There is no change in model architecture throughout this process. Therefore, fine-tuned model uses the same amount of computation as the model which is explicitly trained to accomplish the same task.

Exploiting these characteristics in two distinct domains, the proposed algorithm can be summarized as following:

1. pre-train a model as if it is targeting multi-classification problem
2. freeze the model parameters
3. fine-tune a model to obtain class-specific weights (step $4 \sim 7$)
4. construct a dataset which labels corresponding class as positive and the others as negative.
5. replace the last full-connected layer with two classes
6. fine-tune the last layer with the constructed data
7. store weights for positive class
8. at the time of the inference, the last fully-connected layer is dynamically constructed to classify the given set of class by loading corresponding weights obtained from fine-tuning

For simplicity, I will call this algorithm as composing algorithm, the model trained for multi-class as base model, models constructed from step $4 \sim 7$ as fine-tuned models and the model constructed from the last step as composed model.

to be raised in introduction

Recall that the desired composed model should satisfy the following conditions:

1. preserve accuracy
2. Dynamic addition and removal of a class must be possible
3. Minimal computation is desired for an inference

Addition and removal of a class is possible with the composing algorithm as it can be achieved simply by adding or removing corresponding fully-connected connections. With such a simply addition, retraining the base model is no longer necessary, which is unavoidable in the previous setting. Furthermore, note that even though a class does not contributes to step 1, it can be fine-tuned and added for step 8 as long as the same base model is used for fine-tuning.

From fine-tuning and composing, we only work with the last full-connected layer; when a model is fine-tuned for a class, the weights we store are from the fully-connected layer. Since we do not use weights trained for negative class, the architecture of the composed model is same as the base model, assuming that the composed model is designed to classify all classes which base model is trained for. Therefore, we can conclude that the composed model does not add any computations compared to the base model.

4 Accuracy Preservation

The most challenging part with the composable model problem is assuring that the composed model suffers from the minimal change in accuracy. In this section, I discuss how such condition can be satisfied with the algorithm presented in the previous section.

4.1 Limitations with cross entropy loss

State of the art loss for multi-classification problem is cross entropy (CE) loss, which transforms output using softmax and calculates loss using negative log likelihood (NLL). Softmax and NLL loss are defined as following:

$$Softmax(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

$$NegativeLogLikelihood(y, t) = -\frac{1}{N}\sum_{i=1}^{N}[t_i \cdot \log y_i]$$

where x is the output of the network and t is the target. target is one hot encoded vector as in common classification problem.

Softmax is normalized logits computed from the output of the network. Therefore each index has value between zero and one and sum is designed to be one. Since multi-class classifier rely on the mutually exclusive assumption with about the classes, CE loss is the most suitable as it promotes the positive class while suppressing the negative classes.

However, such assumption does not hold for composed model and CE loss can cause indeterministic behaviour with composed model. When CE loss is used throughout the composing algorithm, the loss which used in fine-tuning is computed with respect to the output of negative class. However, for a composed model, it is not fair to select the class with the greatest value as final class because model outputs for each class are not constructed with respect to each other.

For example, let us say there are 3 classes: A, B, and C. For fine-tuning, we construct a dataset which has two classes: one for target class (positive class) and the other constructed with the other two classes (negative class). Once models are fine-tuned for each class, we have weights for positive class and negative class. For simplicity, positive weights are referred with lower case (a, b, and c) and negative weights with lower case with prime (a', b', and c'). During fine-tuning, loss is calculated in pairs: a – a', b – b', c – c'. However, when we construct a composed model, the last layer consists of weights a, b and c. As a results, selecting the highest value as a final class is not a valid approach.

4.2 Binary cross entropy with sigmoid

We now understand the limitation of CE loss caused by mutually exclusive assumption among classes. To overcome such issue, I propose binary cross entropy (BCE) loss with sigmoid.

$$Sigmoid(y_i) = \frac{1}{1 + e^{-x_i}}$$
$$BinaryCrossEntropy(y, t) = -\frac{1}{N}\sum_{i=1}^{N}[t_i \cdot \log y_i + (1 - t_i) \cdot \log(1 - t_i)]$$

Unlike softmax in CE Loss, the only input for sigmoid is the output from the network at the corresponding index. Furthermore, while CE loss considers mutually exclusive assumption as part of its calculation, BCE loss considers each index independently and is calculated with respect to its target. As a result, weights for each class in fine-tuned model no longer depend on other factors. This indicates that class with the highest output from composed model has the highest probability of being true class.

It is found that multi-label classification has raised the same issue with CE loss [15]. It is found that the independence guarantee provided by BCE loss with sigmoid is crucial for multi-label classification and enabled successful training of a model.

5 Experiments

Realising the theoritical limitation of cross entropy loss, I have conducted thorough experiments using PyTorch, reporting how different loss function affects the performance of composed model.

In this experiment, I evaluate three different loss functions. First loss function is CE loss. Since PyTorch NLL loss implementation expects log probability, log is applied after softmax but it is found that this does not affect the conclusion from the experiments. Next, I include sigmoid with BCE loss which successfully preserves accuracy between base and composed model. Lastly, I include BCE loss with softmax. This setting is known to be unstable because BCE loss assumes the independence among classes. In fact, I have observed the training collapse once it reaches a certain point. However, since we report accuracy from the best model, results from this setting is still valid and meaningful. This setting should demonstrate how crucial sigmoid is in the composing algorithm as it simply replace sigmoid with softmax from last setting.

Experiments are conducted with MNIST, Keyword Spotting, and CIFAR100 varying on number of classes, 10, 30, and, 100 respectively. From each experiment, I report accuracy of base model, fine-tuned model, and composed model. Furthermore, as I construct composed model by adding a class at a time, I also report accuracy for each intermediary composed model to understand how how number of classes affects the accuracy of combined model. It is found that the intermediary composed model accuracy fluctuate quite a bit depending on the order each class is added. Therefore, I repeat this process 10 times for each base model as I randomly select a class to add next.

5.1 MNIST

MNIST is a standard benchmark for classification [16] which consist of images for handwritten digits. Among the wide range of model architecture solving this problem, I have conducted my experiments with `LeNet-5` [17]. `LeNet-5` is constructed with two convolutional, one dropout and two fully connected layers. The original implementation of `LeNet-5` has 10 and 20 channels for the two convolutional layers and produces accuracy of 98% on MNIST dataset. Since our goal of this experiment is to understand the difference in peformance caused by different loss functions, I intentionally limited the expressive power of the network by constructing the network with only 5 channels for both convolutional layers.

Both base model training and fine-tuning is conducted using Adam optimizer with learning rate of 0.0001. After conducting 50 iterations of the experiment, it is found that 5 epochs are enough for

Loss function	Base model
LogSoftmax + NLL	95.82
Softmax + BCE	94.86
Sigmoid + BCE	95.48

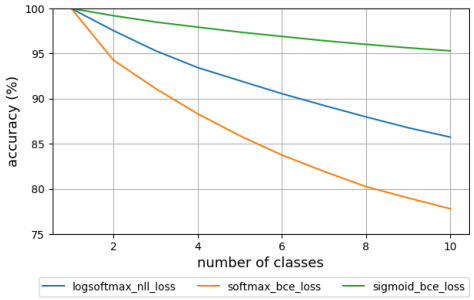

Table 1: Accuracy of base model(left) and composed model with respect to number of classes(right)

Loss function	Fine tuned model									
	0	1	2	3	4	5	6	7	8	9
LogSoftmax + NLL	99.04	99.23	97.79	98.18	98.09	98.03	98.64	97.74	97.12	96.34
Softmax + BCE	98.81	99.09	97.23	97.60	97.45	97.04	98.34	97.32	96.46	95.44
Sigmoid + BCE	99.03	99.19	97.68	98.21	98.31	98.09	98.54	97.57	97.18	96.66

Table 2: Accuracy of fine-tuned model per each class

each training to be converged and `LeNet-5` achieves average accuracy of 95% with the base model and 98% with the fine-tuned model (see Figure 1 and 2). In fact, all three loss functions present similar accuracy confirming that the comparison on the composed model is fair.

Figure 1 summarizes how the accuracy changes as number of classes increases for composed model. No matter which loss function is used for composing algorithm, decrease in accuracy is found as more classes are involved for the composed model construction. However, models with softmax based loss is found to suffer at a greater rate than models with sigmoid based loss. With CE loss, the composed model for all 10 classes has an accuracy of 85.86% introducing about 10% decrease compared to the base model. BCE loss with softmax is found to be the least effective for composing algorithm as it leads to final accuracy of 78.50%. On the other hand, BCE loss with sigmoid introduces the minimal degradation and achieves the finall accuracy of 95.29%. This proves that class independence training is crucial with the composing algorithm.

5.2 Keyword Spotting

In order to evaluate the universality of composing algorithm, I extend the idea to keyword spotting where the input data is audio. The goal of Keyword Spotting (KWS) is to detect an audio of pre-trained keywords, such as Hey Siri. Since the meaningful results introduced by [18], Neural network has been the *de facto* approach for KWS. In this experiment, I implement `res15-narrow` introduced by [19], which achieves accuracy of 94% for 12 keywords on Googles Speech Commands Dataset [20]. `res15-narrow` comprises 6 residual blocks with 19 feature maps where each residual block is composed of bias-free convolution and batch normalization layer.

Unlike the previous studies with Googles Speech Commands Dataset, I try to classify all 30 keywords as I would like to understand how stable the copmosing algorithm with respect to the increasing number of classes. Following the standard feature extraction process for audio, I first construct Forty-dimensional Mel-Frequency Cepstrum Coefficient (MFCC) frames and stack them using 30ms windows with a 10ms shift.

Throughout 10 experiments conducted, both base model training and fine-tuning is achieved with stochastic gradient descent with decreasing learning rate from 0.1 to 0.0001, stepping by a factor of ten. The base model is trained for 30 epochs with decrease in learning rate at 10 and 20 epochs. Unlike MNIST, the standard CE loss produces the best results; average accuracy of 93.11%. It is found that the base model achieves average accuracy of 91.03% and 89.63% when trained using softmax with BCE loss and sigmoid with BCE loss respectively.

In order to obtain fine-tuned weights for each class, I further train the model for 10 epochs with the decrease at 4 and 7 epochs. As summarized in table 3, CE loss leads to the best fine-tuned models

Loss function	Base model	Fine tuned model		
		average	minimum	maximum
LogSoftmax + NLL	93.11	95.31	92.89	97.17
Softmax + BCE	91.03	91.98	89.29	95.28
Sigmoid + BCE	89.63	91.28	89.08	93.97

Table 3: Average accuracy of `res15-narrow` base and fine-tuned model

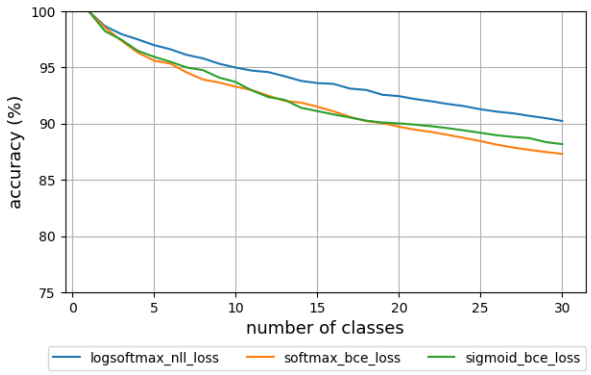

Figure 1: Accuracy of base model(left) and composed model with respect to number of classes(right)

of 95.31% accuracy. Similarly, softmax with BCE loss produces fine-tuned models with average accuracy of 91.84% and sigmoid with BCE loss produces models with 91.28%.

Suprisingly, accuracy perservation is failed on kws

5.3 CIFAR-100

CIFAR is a collection of tiny colour images from the web introduced by [21]. There exist two variations of this dataset varying on number of classes: CIFAR-10 and CIFAR-100. The experiment is constructed with CIFAR-100 which constitutes of 600 images of 100 classes.

densenet [22]

6 Conclusion

In this section please concisely describe what you are going to achieve in this project. E.g., formulate your problem precisely (mathematically), present the technical challenges (if any), discuss the tools or datasets that you will build on, state your goals, and come up with a plan for evaluation.

For your own sake, you might want to lay out a time line, so that you can keep a good track of your project.

Acknowledgement

Thank people who have helped or influenced you in this project.

References

[1] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[2] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[3] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *icml*, volume 96, pages 148–156. Citeseer, 1996.

[4] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[5] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *ICML*, 1993.

[6] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850, 2015.

[7] Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343, 2017.

[8] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[9] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[10] Toby Marshall Egan, Baiyin Yang, and Kenneth R Bartlett. The effects of organizational learning culture and job satisfaction on motivation to transfer learning and turnover intention. *Human resource development quarterly*, 15(3):279–301, 2004.

[11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[12] Wenyi Huang and Jack W Stokes. Mtnet: a multi-task neural network for dynamic malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 399–418. Springer, 2016.

[13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and S Yu Philip. Learning multiple tasks with multilinear relationship networks. In *Advances in Neural Information Processing Systems*, pages 1594–1603, 2017.

[15] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124. ACM, 2017.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20, 2015.

[18] Tara Sainath and Carolina Parada. Convolutional neural networks for small-footprint keyword spotting. 2015.

[19] Raphael Tang and Jimmy Lin. Deep residual learning for small-footprint keyword spotting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5484–5488. IEEE, 2018.

[20] Pete Warden. Launching the speech commands dataset. https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html, 2017.

[21] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.