NETWORK ADAPTATION STRATEGIES FOR LEARNING NEW CLASSES WITHOUT FORGETTING THE ORIGINAL ONES

Hagai Taitelbaum[1] *Gal Chechik*[2] *Jacob Goldberger*[1]

[1] Faculty of Engineering, Bar-Ilan University, Ramat-Gan, Israel
[2] The Gonda Brain Research Center, Bar-Ilan University and NVIDIA Research

ABSTRACT

We address the problem of adding new classes to an existing classifier without hurting the original classes, when no access is allowed to any sample from the original classes. This problem arises frequently since models are often shared without their training data, due to privacy and data ownership concerns. We propose an easy-to-use approach that modifies the original classifier by retraining a suitable subset of layers using a linearly-tuned, knowledge-distillation regularization. The set of layers that is tuned depends on the number of new added classes and the number of original classes.

We evaluate the proposed method on two standard datasets, first in a language-identification task, then in an image classification setup. In both cases, the method achieves classification accuracy that is almost as good as that obtained by a system trained using unrestricted samples from both the original and new classes.

Index Terms— Catastrophic forgetting, learning privacy, knowledge distillation

1. INTRODUCTION

Many classification learning procedures require the addition of new classes to an existing classification system while maintaining the system's performance on the original classes. When the original training data are available, the classifier can simply be retrained from scratch with samples from both the new and the original classes. However, in a wide variety of machine learning problems, training data from the original classes cannot be used. This happens for example when the training dataset contains sensitive information that is not provided, or due to concerns over data ownership and sharing (e.g., GDPR). In these cases, it is impossible to simply retrain a model without compromising privacy [1].

Here, we address the problem of extending a classifier to new classes when the training data used to train the original classifier are *not available*. The only available data are from the new classes to be added to the system. We also assume that we have access to the system parameters, which were pre-trained for the original classes.

One naive approach would be to "fine-tune" the classifier with samples from the new classes. Sadly, it is well-known that this approach induces *catastrophic forgetting* (CF), namely, the tendency of an artificial neural network to forget previously learned information upon learning new information [2, 3]. CF has two related aspects: it involves forgetting the representation that was learned and also involves a performance deterioration for old tasks. The current setup is related to similar problem in continual learning, where the aim is to incrementally train a single network to learn multiple tasks. In this setup, each task is evaluated solely on the data from its own dataset (see e.g. [4, 5]), and it assumes that an oracle classifier is available, which determines the appropriate task for the given instance. In our case, the classification system needs to classify an object without information on whether it belongs to the original task or to the new class set.

Most algorithms designed to avoid CF are motivated by computational and memory constraints, rather than by privacy concerns. As a result, these algorithms operate under the assumption that samples from the original classes are available. Under these assumptions, one natural approach stores a subset of samples from the original classes and uses them during retraining [6]. Other approaches train generative models using the samples from the first phase, and use them to generate samples in the retraining phase [7, 8, 9]. [10] described a combined approach that both stores samples from the original classes and trains a generative model based on the original classes. All these approaches assume they can access the data used to train the original classification system, and thus do not apply to the problem addressed in this paper. More recently, [11] described an approach addressing the problem discussed here. Unfortunately, it leads to inferior performance because it has no representation learning during the retraining phase.

Here, we propose a method for adding new classes to an existing classifier, while addressing CF.As our main novel contribution, we show that a good representation can be relearned in the retraining phase, by training a limited but suitable subset of layers of the classifier using linearly tuned knowledge distillation regularization. The decisions as to which layers to retrain, and how to set the regularization weight depend on the ratio between the number of original classes and the number of classes in the extended class-set. Evaluations of the proposed method on standard datasets shows significant improvement compared to previously suggested methods.

2. NETWORK ADAPTATION

The learning setup

We consider a multi-class problem where a classifier was trained to map an input feature vector $\mathbf{x}$ to one class in a set of k classes $A = \{1, ..., k\}$. We are further given labeled training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ from m new classes from the set $B = \{k+1, ..., k+m\}$. We want to create an extended classifier that can correctly map a new test sample to a class in $A \cup B$. We further assume that the original model parameters are available, but there is no access to the training data used to train the original classifier. Our goal is to train a classifier for the set $A \cup B$, using the parameters of the original classifier and the new training data labeled by classes from B.

Deep learning classifiers consist of two major components: a non-linear transformation $h(\mathbf{x})$, followed by a linear soft-max layer parametrized by a weight vector $\mathbf{w}$. Together, they yield a soft classification over the class set A:

$$p(y = i|\mathbf{x}) = \frac{\exp(\mathbf{w}_i h(\mathbf{x}))}{\sum_{j \in A} \exp(\mathbf{w}_j h(\mathbf{x}))}, \qquad i \in A. \quad (1)$$

One can extend the classifier to new classes by keeping the nonlinear transformation, and extending the softmax layer to the m new classes. The output of the *extended classifier* is

$$p(y = i|\mathbf{x}) = \frac{\exp(\mathbf{w}_i h(\mathbf{x}))}{\sum_{j \in A \cup B} \exp(\mathbf{w}_j h(\mathbf{x}))}, \quad i \in A \cup B. \quad (2)$$

The parameters of the extended classifier can be learned to maximize the likelihood:

$$\tilde{L} = \sum_{t=1}^{n} \log p(y_t|\mathbf{x}_t). \quad (3)$$

As discussed above, if (3) is maximized by only using samples from B, the classifier exhibits CF of the original class-set A [2, 3]. CF in the sense of deterioration in old tasks performance, occurs even if all the parameters of the original classifier are kept "frozen", and maximization only tunes the new parameters $\mathbf{w}_{k+1}, ..., \mathbf{w}_{k+m}$ [11].

Addressing catastrophic forgetting with regularization

One common approach to alleviating CF is to modify the objective by adding a regularization term that encourages activation of output neurons of the original classes. Following [4] and [6], we can use knowledge distillation [12] as a regularization term, yielding the modified objective:

$$L = (1-\epsilon) \sum_{t=1}^{n} \log p(y_t|\mathbf{x}_t) + \epsilon \sum_{t=1}^{n} \sum_{i \in A} q_{ti} \log p(y = i|\mathbf{x}_t), \quad (4)$$

where the hyperparameter ϵ controls the regularization term and q_{ti} is the classification results of applying the original classifier to the new data, as in Eq. (1).

Unfortunately, even if we use a regularized objective function, it is not clear which layers of the original classifier should be retrained and which parameters should be kept unchanged, to maximize the test-set performance over the entire class-set.

Our approach

Building on the work on regularized objectives, we suggest retraining several layers of the network. To decide which layers should be retrained, we analyze several cases and two benchmark datasets. We show empirically below that selecting which layers to retrain depends on multiple factors including the input features, the classifier architecture and the ratio of the number of original to the number of new classes.

For a better intuition into this problem, it is useful to consider the two extreme cases: only training the new parameters as in [11] (*ACWOD*), and retraining all the weights of the classifier (*all-weights*).

Consider first the extreme option where only the new parameters $\mathbf{w}_{k+1}, ..., \mathbf{w}_{k+m}$ are trained and all the parameters of the original classifier are kept intact. This strategy suffers from two main problems: First, the representation $x \to h(x)$ was learned for classification data labeled by classes from A and is kept fixed in the retraining phase. As a result, it generalizes poorly to classes from B. The second drawback is that the distillation loss in this case is degenerated. It can be easily verified that for every $i \in A$ and $j \in B$ the derivative of the first objective term in Eq. (4) equals $\frac{\partial}{\partial \mathbf{w_j}} \log p(y_t = i|\mathbf{x}_t) = -h(\mathbf{x}_t)p(y_t = j|\mathbf{x}_t)$, and the derivative of the regularization term in Eq. (4) is:

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}_j} \sum_{i \in A} q_{ti} \log p(y_t = i|\mathbf{x}_t) &= -h(\mathbf{x}_t)p(y_t{=}j|\mathbf{x}_t) \sum_{i \in A} q_{ti} \\ &= -h(\mathbf{x}_t)p(y_t{=}j|\mathbf{x}_t). \end{aligned}$$

The derivative of the loss therefore becomes:

$$\frac{\partial L}{\partial \mathbf{w}_j} = \sum_t ((1-\epsilon)1_{\{y_t=j\}} - p(y_t = j|\mathbf{x}_t))h(\mathbf{x}_t).$$

Note, that the derivative does not depend on the distribution of q_{ti}. Hence, no knowledge is actually distilled from the original classifier by the regularization term.

Lets us now turn to the second extreme case of retraining all the classifier parameters, which suffers from other limitations. Specifically, when only a small number of new classes is added during the retraining phase, tuning the classifier parameters is based on a small subset of the entire dataset , thus hurting the overall performance. This effect occurs even if the weight of the regularization term is set in an optimal way.

To avoid the drawbacks of these two extreme cases, we take an intermediate approach, and train both the last layer and several layers before it. If tuned correctly, this approach gains from both worlds: on one hand, knowledge distillation about the original classes is effective; on the other, the representation h is re-learned to represent well the new classes.

How should we decide how many layers to retrain? To shed light on this question, consider the number of new classes compared to the number of original classes. When the fraction of original classes is small, the learned representation tends to be narrowly tuned to these classes, since they

do not sample well the space of classes. Intuitively, this can be viewed as overfitting to classes in the sense of transfer learning. In this case, small weight should be given to the knowledge distillation component (small ϵ) and more layers of the model should be retrained. This allows the classifier to re-learns its representation based on the new classes, which constitute most of the data.

When the fraction of original classes is large, and only few new classes are added, the learned representation is likely to reflect well the distribution over many classes. As a result, the weight ϵ should be set high, and only few layers should be retrained to keep the knowledge from the original classes, which now form most of the data.

As in [11], we set ϵ to be the linear function:

$$\epsilon = c \times \frac{|A|}{|A| + |B|} \quad . \tag{5}$$

In the experiment section, we show that Eq. (5) indeed approximates well the optimal value. The slope c depends on the task and on the classifier layers we want to retrain.

3. EXPERIMENTS

In this section we demonstrate the benefits of using the proposed learning method on two standard classification tasks: language identification and image classification.

(1) The NIST 2015 language recognition challenge [13] has labeled data from 50 target languages, with 300 speech segments per language. The speech segments were derived from conversational telephone and narrow-band broadcast speech data. Each speech segment is represented by an i-vector of 400 components [14]. The segments used to create the i-vectors had their duration sampled from a log-normal distribution with a mean of approximately 35s. For each language we used 255 speech segments for training and the remaining 45 segments for performance evaluation.

As a classification network, we used a Deep Neural Networks (DNN) with two fully-connected hidden layers with 200 and 100 neurons, and a soft-max output layer. The activation function was set to be ReLU and the optimization procedure used was mini-batch SGD with a constant learning rate of 0.01 for 80 epochs in each phase. We also used L_2 regularization to prevent overfitting. The temperature of the knowledge distillation loss was set to 1. The regularization parameter was selected using the same method as in [11].

(2) The CIFAR-100 dataset [15] consists of 32×32 color images from 100 classes. There are 500 training images and 100 testing images per class for a total of 60000 images. For experiments with CIFAR-100, we used RseNet-32 [16] based on PyTorch implementation. For the optimization process we used mini-batch SGD for 70 epochs. The initial learning rate was 0.1 and was divided by 10 after 49 and 63 epochs (7/10 and 9/10 of all epochs). We used L_2 regularization with a weight decay of 0.0005. The temperature of the knowledge distillation loss was set to 2, and we also used a Nesterov momentum [17] of 0.9.

In the experiments we first trained a DNN classifier for a subset A. Then, we used examples from the remaining classes (subset B) to extend this model to the unified class-set $|A \cup B|$. We set ϵ as described in Eq. (5). To evaluate performance, we calculated the prediction accuracy on the entire evaluation set including classes from both subsets A and B. We computed the performance results as a function of the proportion of A classes; namely, the fraction $|A|/(|A|+|B|)$.

Compared methods

As explained in Section 2, choosing the appropriate subset of layers to retrain depends on the number of original classes. To demonstrate our approach, we tested the networks using two retraining subsets. First, we retrained all parameters of the original classifier (*all-weights*). Second, we tuned a subset of high-level layers. For the language identification task, we retrained the softmax layer and one hidden layer (*one-hiddden-layer*). For the image classification, task we retrained one residual block (*residual-block*) and the softmax layer.

We compared the above two variants of our approach against three baselines:

(1) Original model. This method preserves its accuracy on A classes, but always errs when classifying samples from B.

(2) ACWOD The method described in [11]. This method is based on a transfer learning approach where the weights of the original model are frozen, and only the weights that correspond to the new labels are learned.

(3) Train with all data. An upper-bound baseline where training has access to all the training data, while other models are limited to samples from B classes alone. The network is trained "from scratch" in a single training phase.

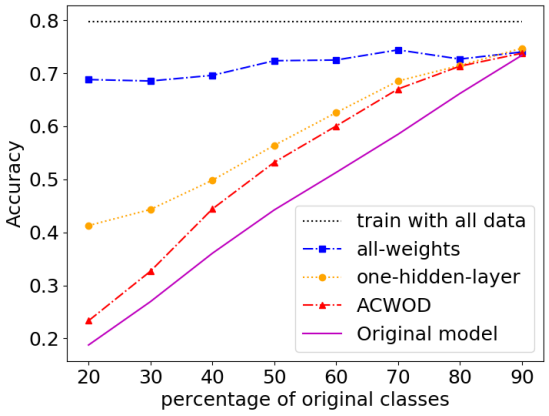

Fig. 1: NIST-2015 test-set accuracy as a function of the percentage of original classes.

Classification Results

Fig. 1 depicts the accuracy on the language identification task as a function of the fraction of the original classes in the overall class set. Here, the more layers retrained, the higher the

accuracy on the entire class-set. The *all weights* model outperformed both the *original model* and *ACWOD*. These two methods freeze all their weights, so there is no representation learning when adding the classes from B. This is especially noticeable when the original class set is small. Performance reached close to the ideal upper bound.

Fig. 2 depicts the CIFAR-100 performance as a function of the size of the original class-set. Retraining the network led to better performance than either the *original model* or *ACWOD*, when $|A|$ is small. However, as the fraction of A classes increases, the performance of *all-weights* decreases slightly and then increases slightly. In addition, *residual-block* consistently performed better than *ACWOD* and improved monotonically as a function of the number of original classes $|A|$.

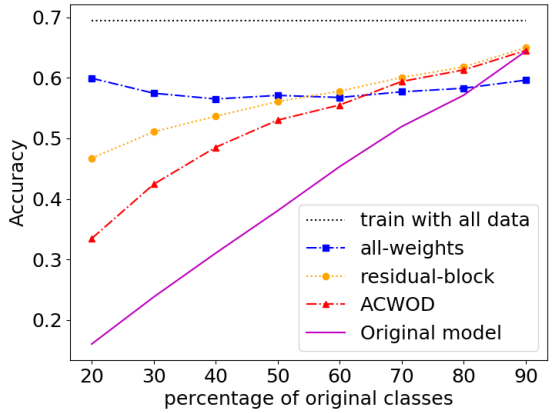

Fig. 2: CIFAR-100 test set accuracy as a function of the percentage of original classes.

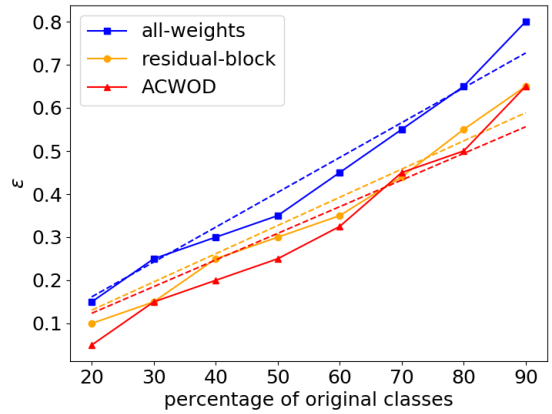

Fig. 3: The optimal weight of regularization, ϵ, as a function of the fraction of original classes for CIFAR-100.

Figures 1,2 show that the accuracy of the *all-weight* method, behaves differently as a function of the fraction of original classes, compared to the other methods. Specifically, in NIST-2015 (Fig. 1) the accuracy slightly improves as the fraction of original classes increases, and remains superior to the other methods. However, in CIFAR-100 (Fig. 2) the accuracy fluctuates around the same level, and becomes inferior as the other methods improve for fractions larger than 60%.

One possible explanation is that the samples in NIST-2015 are already represented using a well-tuned and high-level features of sound, namely i-vectors. This representation is fixed, not tuned by our approach, and as a result, even the *all-weight* method, does not modify that representation. It can be viewed as if there are low layers that implement a non-linear feature-extraction procedure, that is kept fixed. In contrast, CIFAR-100 is composed of raw images, and the entire representation is learned by our network. In this case, under the *all-weight* method the whole model is updated, and may hurt the low-level representation for the original classes.

Distillation weight analysis

The three network-retraining methods, *all-weight*, *residual-block/one-hidden* and *ACWOD*, may differ in two aspects: First, the number of layers that are tuned during the second training phase; Second, the value of the ϵ parameter that weighs the distillation loss in the objective function (Eq. 4). We further studied the relation between these two parameters.

For each value of $|A|$, we empirically found the value of ϵ that maximizes the accuracy of the extended classifier, as in [11]. For NIST2015, the optimal epsilon values were similar for all methods. Fig. 3 depicts the optimal ϵ values for the CIFAR-100 dataset as a function of the fraction of A, for the three studied methods (solid curves). It shows that the optimal ϵ increases monotonically as a function of $|A|$, and roughly follows a linear growth. We therefore approximate the empirical optimal ϵ values with a linear function of $|A|$ as suggested in Section 2 and discussed in [11]. The linear fit is plot in Figure 3 as dashed lines for each of the three methods.

When comparing the slopes of the three dashes curves in Fig. 3, we observe that methods that update more layers have a steeper slope. For instance, *all-weights* tunes all layers of the network, and reaches the highest ϵ values. This suggests that methods that are restricted to tune only small part of the network (like *ACWOD*, and to a lesser extent *residual-block*) counter-act this constraint, by reducing the weight of the distillation loss. This allows them to adapt more strongly those layers of the network that are not fixed.

Conclusions

We proposed a method to learn new classes incrementally when the data used to train the original classifier are not available. We showed that the relative proportion of original and new classes dictates the best network adaptation strategy and controls the number of network layers that should be trained and not remained fixed.

4. REFERENCES

[1] M. Abadi, U. Erlingsson, I. Goodfellow, H. McMahan, I. Mironov, N. Papernot, K. Talwar, and L. Zhang, "On the protection of private information in machine learning systems: Two recent approaches," in *IEEE Computer Security Foundations Symposium*, 2017.

[2] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *The Psychology of Learning and Motivation*, vol. 24, pp. 109–164, 1989.

[3] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[4] Z. Li and D. Hoiem, "Learning without forgetting," in *European Conference on Computer Vision*, 2016.

[5] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, pp. 3521–3526, 2017.

[6] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental Classifier and Representation Learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017.

[7] C. Atkinson, B. McCane, L. Szymanski, and A. Robins, "Pseudo-recursal: Solving the catastrophic forgetting problem in deep neural networks," *arXiv preprint arXiv:1802.03875*, 2018.

[8] Y. Wu, Y. Chen, L. Wang, Y. Ye, Z. Liu, Y. Guo, Z. Zhang, and Y. Fu, "Incremental classifier learning with generative adversarial networks," *arXiv preprint arXiv:1802.00853*, 2018.

[9] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Advances in Neural Information Processing Systems*, 2017, pp. 2990–2999.

[10] C. He, R. Wang, S. Shan, and X. Chen, "Exemplar-supported generative reproduction for class incremental learning," in *29th British Machine Vision Conference*, 2018, pp. 3–6.

[11] H. Taitelbaum, E. Ben-Reuven, and J. Goldberger, "Adding new classes without access to the original training data with applications to language identification," in *Proc. Interspeech*, 2018, pp. 1808–1812.

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *NIPS Workshop on Deep Learning*, 2014.

[13] "NIST language recognition i-vector machine learning challenge," https://ivectorchallenge.nist.gov/, 2015.

[14] N. Dehak, P. A. Torres-Carrasquillo, D. Reynold, and R. Dehak, "Language recognition via Ivectors and dimensionality reduction," in *Interspeech*, 2011.

[15] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.

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

[17] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Int. Conference on Machine Learning*, 2013, pp. 1139–1147.