LPC: A Logits and Parameter Calibration Framework on Continual Learning

Anonymous ACL submission

Abstract

Deep learning based pre-trained natural language processing (NLP) models typically pre-train on large unlabeled corpora first, then fine-tune on new tasks. When we execute such a paradigm on continuously sequential tasks, the model will suffer from the catastrophic forgetting problem (i.e., they forget the parameters learned in previous tasks when we train the model on newly emerged tasks). Inspired by the idea of how humans learn things, we aim to maintain the old knowledge when we transfer to novel contents and calibrate the old and new knowledge. We propose a Logits and Parameter Calibration (LPC) framework to reduce the catastrophic forgetting in the continual learning process. The proposed framework includes two important components, the Logits Calibration (LC) and Parameter Calibration (PC). The core idea is to reduce the difference between old knowledge and new knowledge by doing calibration on logits and parameters so that the model can maintain old knowledge while learning new tasks without preserving data in previous tasks. First, we preserve the parameters learned from the base tasks. Second, we train the existing model on novel tasks and estimate the difference between base logits and parameters and novel logits and parameters. Third, we drift from the base tasks to novel tasks gradually. Furthermore, we integrate the logtis and parameter calibration into a brand-new optimization algorithm. Finally, we do experiments on 7 scenarios of the GLUE (the General Language Understanding Evaluation) benchmark. The experimental results show that our model achieves state-of-the-art performance on all 7 scenarios.

1 Introduction

Predicting labels for a large number of instances occurring continuously is a crucial problem in many real-world applications like online tweets/news summary, online product classification in e-commerce systems, and online dialogue learning systems. In these scenarios, we not only require the model to learn from its own experiences, but also expect the model to be capable of continuously acquiring, fine-tuning, and transferring knowledge over time (Parisi et al., 2019), which is also known as continual learning. One of the most essential existing challenges we aim to solve in the continual learning is the catastrophic forgetting problem (McCloskey and Cohen, 1989; Kirkpatrick et al., 2017a). The forgetting typically happens when we apply the pre-trained model (e.g., BERT (Devlin et al., 2018)) on newly emerged tasks, the model usually forgets the parameters it learned from previous tasks when we train it on new incoming tasks.

Existing works trying to solve the catastrophic forgetting problem are varied, which can be divided into two categories: (1) storing exemplars from previous classes (Rebuffi et al., 2017; Rolnick et al., 2019); (2) regularizing the parameters when we fine-tune the model on new tasks (Kirkpatrick et al., 2017b; Li and Hoiem, 2017; Aljundi et al., 2018). Such methods aim to transfer or store the knowledge of previous tasks to the newly emerged tasks and preserve the knowledge learned previously. Memory Aware Synapses (MAS) (Aljundi et al., 2018) is an advanced approach by computing the importance of the neural network parameters in an unsupervised and online manner. MAS assigns more weights on the parameters that are most important to the model and allows the model to selectively forgets those weights that are not so essential. Also, Lee et al. (Lee et al., 2020) successfully reduce the catastrophic forgetting during the fine-tuning step by randomly mixing pre-trained parameters into a downstream model in a dropout-style.

The methods mentioned above address the catastrophic forgetting through multi-task learning. They typically require storing the data from old or pre-trained tasks, and replay them during the fine-tuning. However, this learning pattern does not consider the constraint of memory resource or privacy issues, e.g., the data of old tasks is often inaccessible or too large for the continual adaptation setting. Unlike the multi-task learning strategy, here we focus on the calibration of knowledge gap between different tasks, which can reduce the catastrophic forgetting without any old data/task replay. The proposed calibration framework is used for both encoder parameters and output classifiers, we first train the novel model and evaluate the base model on novel tasks. Specifically, we add the logits calibration that can amplify the softmax output of the base model, and overcome the bias towards the novel category (Zhao et al., 2020), which can simultaneously enforce the model to preserve previous knowledge via explicit weight constraints. Also, for the calibration on encoder parameters, we encourage the model to maintain previously learned knowledge by simulating the training objective using the parameters of the base model. Then, during the training on novel tasks, the model will calibrate parameters with target drift form the base tasks to the novel tasks to balance new task learning and old knowledge maintenance. It allows the model to focus on novel tasks by making the learning objective drifting from the base tasks to novel tasks gradually.

Accordingly, we propose a Logits and Parameter Calibration framework LPC (shown in Figure 1) on continual learning scenario, which is used to reduce catastrophic forgetting without further data storage. The proposed calibration mechanism includes two components for both model encoder parameters and output logits, we finally integrate these two calibrations into a brand-new optimization algorithm by decoupling them from the gradient updates in Adam optimizer. We do experiments on the GLUE benchmark with pre-trained models BERT-base and ALBERT-xxlarge and achieve state-of-the-art performance.

The contributions of our work are three folds. First, we propose LPC, a novel continual learning framework, which can reduce catastrophic forgetting effectively without storing previous instances. Second, we develop a new mechanism by calibrating the logits and parameters with target drift from base tasks to novel tasks, thereby alleviating the catastrophic forgetting during the model updating. Third, combining with a parameter regularization based approach, our model achieves state-of-the-art performance while addressing the old knowledge forgetting without data storage. Therefore, the newly proposed LPC is feasible for researchers to use for further explorations in this field.

2 Related Works

2.1 Continual Learning

Continual learning is also named as life-long learning, sequential learning, or incremental learning. As the name suggests, continual learning aims to learn tasks in a sequential way. In the field of biology, biological neural networks exhibit continual learning in which they acquire new knowledge over a lifetime (Zenke et al., 2017). However, continual learning in deep neural networks suffers from a phenomenon called *catastrophic forgetting* (Shin et al., 2017). Thus, one of the most essential goals of continual learning systems is to achieve satisfying performance on all tasks in an incremental way. Reducing catastrophic forgetting plays a vital role to achieve it. Current continual learning approaches can be classified into the following three families (De Lange et al., 2019): (1) Replay methods, (2) Regularization-based methods, and (3) Parameter isolation methods. Replay methods store samples in a raw format or generate pseudo-samples. Regularization-based methods eschews storing raw inputs, prioritizing privacy, and alleviating memory requirements. Parameter isolation methods dedicate different model parameters to each task to prevent any possible forgetting. Our method is an advanced regularization-based method alleviating the catastrophic forgetting without data storage.

2.2 Fine-tuning

Fine-tuning is a successful method in transfer learning by the following four steps: (1) pretrain a source neural network model on the source datasets; (2) create a new neural network model which copies all model designs and their parameters on the source model except the output layer; (3) add an output layer to the target model; (4) train the target model on the target datasets. Girshick et al. (Girshick et al., 2014) propose a R-CNN to fine-tune all network parameters. Long et al. (Long et al., 2015) propose DAN only fine-tuning the parameters of the last few layers. Li et al. (Li et al., 2018) investigate several regularization schemes that explicitly promote the similarity of the fine-tuned model with the original pre-trained model.

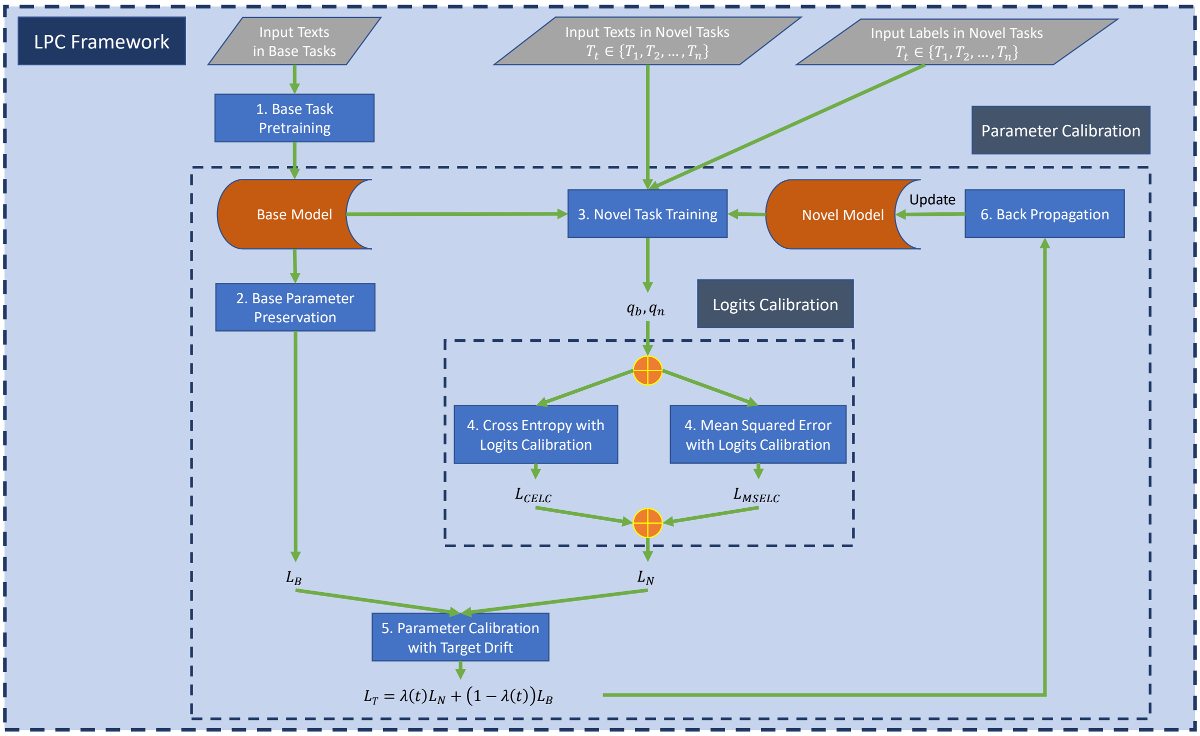

Figure 1: The Overview of LPC Framework. (1) Given the large-scale input texts from base tasks, we first pre-train a base model on these texts and initialize our model (for novel tasks) same as the base one. (2) We do base parameter preservation to preserve the parameters of the pre-trained model, and estimate the difference between base parameters and novel parameters during training, then compute the loss for the base model L_B. (3) During the novel task training, we compute logtis q_b and q_n for the base model and the novel model, respectively. (4) We do logits calibration (e.g., cross entropy with logits calibration for classification tasks) given q_b and q_n using L_{CELC} or L_{MSELC} for regression tasks as the loss for the novel model L_N. (5) In the parameter calibration with target drift, the objective function drifts from L_B to L_N gradually with the annealing coefficient $\lambda(t)$. (6) Finally, we perform back propagation to update the parameters of the novel model with parameter calibration.

Guo et al. (Guo et al., 2019) propose an adaptive fine-tuning approach SpotTune which automatically decides the optimal set of layers to fine-tune in a pre-trained model on a new task. Our method is a trade-off between multi-task learning and fine-tuning.

3 Proposed Approach

In this section, we introduce our proposed Logits and Parameter Calibration framework, LPC. The LPC framework includes two essential parts: (1) Logits Calibration (LC) that execute calibration on the logits to reduce the logits forgetting and increase the accuracy and (2) Parameter Calibration (PC) to do calibration on the parameters to reduce the parameter forgetting. For the Logits Calibration, we apply the Cross Entropy with Logits Calibration (CELC) for classification tasks (or the Means Squared Error with Logits Calibration (MSELC) for regression tasks). The Parameter Calibration consists of three components: (1) Base Parameter Preservation (BPP) that tries to preserve the parameters we learned from base tasks, (2) Novel Task Training (NTT) that trains the previous model on novel tasks, and (3) Parameter Calibration with Target Drift (PCTD) that focuses on drifting from the base tasks to novel tasks gradually. What is more, we introduce LPC algorithm by integrating the Logits Calibration (CELC or MSELC) and all three parts of Parameter Calibration (BPP, NTT, and PCTD) into a brand-new optimization algorithm based on the well-known Adam (Kingma and Ba, 2014) optimization algorithm.

3.1 Logits Calibration

In this section, we introduce our proposed logtis calibration, the Cross Entropy with Logits Calibration (CELC) for classification tasks. Some other loss function (e.g., the Mean Squared Error for regression) can also be combined with the Logits Calibration, in the following paragraph.

Algorithm 1 LPC

given initial learning rate $\alpha \in \mathbb{R}$, momentum factors $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, pre-trained parameter vector $\theta^* \in \mathbb{R}^n$, hyperparameter for the regularizer $\delta \in \mathbb{R}$, coefficient of the quadratic penalty $\gamma \in \mathbb{R}$, hyperparameter controlling the annealing rate $r \in \mathbb{R}$, hyperparameter controlling the timesteps $t_0 \in \mathbb{N}$.
initialize timestep $t \leftarrow 0$, parameter vector $\theta_{t=0} \in \mathbb{R}^n$, importance weights $\Omega \leftarrow \mathbf{1}$, first moment vector $m_{t=0} \leftarrow 0$, second moment vector $v_{t=0} \leftarrow 0$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$.
repeat
    $t \leftarrow t + 1$ ▷ update timestep
    $x \leftarrow \text{SelectBatch}(\mathbf{x})$ ▷ select batch data
    $q_{n,t} \leftarrow Q_{n,t}(x, \theta_{t-1})$ ▷ compute output logits for the novel model
    $q_{b,t} \leftarrow Q_{b,t}(x, \theta^*)$ ▷ compute output logits for the base model
    $\nabla(f_t(x; \theta_{t-1})) \leftarrow \nabla(L_{CELC}(q_{n,t}, q_{b,t}) \parallel L_{MSELC}(q_{n,t}, q_{b,t}))$ ▷ compute gradients
    $\Omega_t \leftarrow \Omega_{t-1}$
    for $k \leftarrow 0$ to N **do**
        $g_t(x_k) \leftarrow \nabla l_2^2(f_t(x_k; \theta_{t-1}))$
        $\Omega_t \leftarrow \Omega_t + \|g_t(x_k)\|$
    end for
    $\Omega_t \leftarrow \Omega_t / N$ ▷ compute importance weights after each update epochs
    $\lambda(t) \leftarrow 1/(1 + \exp(-r \cdot (t - t_0)))$ ▷ compute annealing coefficient
    $g_t \leftarrow \lambda(t) \nabla f_t(x; \theta_{t-1}) + 2(1 - \lambda(t))\delta\gamma\Omega_t(\theta_{t-1} - \theta^*)$ ▷ compute new gradients
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ ▷ update biased first moment estimate
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ ▷ update biased second raw moment estimate
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ ▷ compute bias-corrected first moment estimate
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ ▷ compute bias-corrected second raw moment estimate
    $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
    $\theta_t \leftarrow \theta_{t-1} - \eta_t(\lambda(t) \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon) + 2(1 - \lambda(t))\delta\gamma\Omega_t(\theta_{t-1} - \theta^*))$ ▷ update parameters
until stopping criterion is met
return optimized parameters θ_t

3.1.1 Cross Entropy with Logits Calibration

The Cross Entropy (CE) Loss (Zhang and Sabuncu, 2018) is a widely-used loss for classification tasks in deep learning. It first applies a log softmax function on the output logits of the neural network. Then, it computes the negative log likelihood (nll) loss on the output of the log softmax function. Typically, the cross entropy loss can be defined as follows:

$$L_{CE}(q) = -\sum_{i=1}^{N_C} p_i \log\left(\frac{\exp(q_{n,i})}{\sum_{j=1}^{N_C} \exp(q_{n,j})}\right) \quad (1)$$

where N_C is the total number of classes in the novel tasks. $q_{n,i}$ represents the output logits for class i of the novel model on the novel tasks. p_i can be considered as the binary label of class i. If the data input x belongs to class i, the value of p_i will be 1, otherwise, the value will be 0.

Nevertheless, the original cross entropy loss only concerns the performance of the novel model. Thus, the model will suffer the catastrophic forgetting problem with the step increasing. In order to reduce the catastrophic forgetting problem, we consider to simultaneously evaluate the base model on the novel tasks and compute the output logits of the base model q_b.

Inspired by the idea from (Kukleva et al., 2021) which revises the original cross entropy loss by adding the summation of the exponential logtis of the base classes classifier to the denominator to change the normalization scale, we add the logits information of the base model into the cross entropy loss.

However, different from Kukleva's method, we do logtis calibration by adding the difference between each logits of the novel model and the base model ($q_{n,i} - q_{b,i}$) to the corresponding output logits $q_{n,i}$ of the novel model for class i. In this way, the model can preserve important output logits information of each class for the base model in an element-wise way. Our proposed Cross Entropy with Logits Calibration (CELC) Loss is shown in Equation 2:

$$L = -\sum_{i=1}^{N_C} p_i \log\left(\frac{\exp(q_{n,i} + \mu(q_{n,i} - q_{b,i}))}{\sum_{j=1}^{N_C} \exp(q_{n,j} + \mu(q_{n,j} - q_{b,j}))}\right) \tag{2}$$

where we multiply the difference between the logits for the novel model and the base model ($q_n - q_b$) by a weight item $\mu \in [0, 1]$ to control the calibration degree. By employing this new loss function, we can also increase the accuracy of the model through training process by giving a reward to the logits for the correct class if $q_{n,i}$ is larger than $q_{b,i}$, otherwise, giving a penalty to the logits for the correct class if $q_{n,i}$ is smaller than $q_{b,i}$.

3.1.2 Mean Squared Error with Logits Calibration

Mean Squared Error (MSE) Loss (Fisher, 1922) is the most commonly-used loss function for regression tasks. It computes the squared L2 norm between output logits and the true values and takes the mean of the full batch. Follow the idea of the logits calibration on cross entropy loss, we evaluate the base model on novel tasks and take out the output logtis q_b. We measure the difference between the output logits of the novel model and the base model by adding a squared L2 norm on the difference between logits of the novel model and the base model $(q_n - q_b)^2$ to the original function. The proposed Mean Squared Error with Logits Calibration Loss L_{MSELC} is shown in Equation 3:

$$L_{MSELC}(q) = (q_n - p)^2 + \mu(q_n - q_b)^2 \tag{3}$$

3.2 Parameter Calibration

In this section, we introduce the second module of our model, Parameter Calibration (PC). Our proposed Parameter Calibration method can effectively reduce the catastrophic forgetting by giving a penalty to the prediction if the parameters of the novel model are different from the base model by adding the squared difference between the parameters of the novel model and the base model to the training loss. It includes three parts: (1) Base Parameter Preservation (BPP), (2) Novel Task Training (NTT), and (3) Parameter Calibration with Target Drift (PCTD).

3.2.1 Base Parameter Preservation

As shown in Figure 1, in Base Parameter Preservation, we try to maintain the parameters of the base model like BERT (Devlin et al., 2018). Here, we add a regularization to the posterior of parameters given data. The Base Parameter Preservation method can be regarded as an improved method derived from EWC (Kirkpatrick et al., 2017b) and MAS (Aljundi et al., 2018). Different from EWC, BPP measures the importance of each parameter by introducing the importance weights Ω. During training, the novel model preserves the information of the most important parameters to a great extent by penalizing the changes to those important parameters more severely. The detailed derivation of our proposed loss function is shown in Equation 4:

$$\begin{aligned}
L_B &= -\log p(\theta|D_B) \\
&\approx -\log p(\theta^*|D_B) + \delta(\theta - \theta^*)^T H(\theta^*)\Omega(\theta)(\theta - \theta^*) \\
&\approx \delta(\theta - \theta^*)^T H(\theta^*)\Omega(\theta)(\theta - \theta^*) \\
&\approx \delta(\theta - \theta^*)^T (NF(\theta^*) + H_{prior}(\theta^*))\Omega(\theta)(\theta - \theta^*) \\
&\approx \delta N \sum_{ij} F_{ij}\Omega_{ij}(\theta_{ij} - \theta^*_{ij})^2 \\
&\approx \delta NF \sum_{ij} \Omega_{ij}(\theta_{ij} - \theta^*_{ij})^2 \\
&= \delta\gamma \sum_{ij} \Omega_{ij}(\theta_{ij} - \theta^*_{ij})^2
\end{aligned} \tag{4}$$

where δ is a hyperparameter for the regularizer. $H(\theta^*)$ is the Hessian matrix of the optimization objective with respect to θ^*. We can approximate $H(\theta^*)$ with the empirical Fisher information matrix $F(\theta^*)$ (Martens, 2014). N is the total number of data inputs in D_B. $H_{prior}(\theta^*)$ is the Hessian matrix of the negative log prior probability $-\log p(\theta)$. EWC ignores $H_{prior}(\theta^*)$ and approximates $H(\theta^*)$ by assigning the diagonal values of $F(\theta^*)$ to $H(\theta^*)$. Thus, we replace NF with a constant value γ at the end of the derivation. We can consider γ as a coefficient of the quadratic penalty. During the derivation, we can simply ignore $-\log p(\theta^*|D_B)$ as it is a constant term with respect to θ^*. $\Omega(\theta)$ is estimated by the sensitivity of the squared L2 norm of the function output to their changes. We can obtain Ω_{ij} by accumulating the gradients over the given data points by Equation 5:

$$\Omega_{ij} = \frac{1}{N}\sum_{k=1}^{N}\|g_{ij}(x_k)\| \tag{5}$$

where $g_{ij}(x_k) = \frac{\partial[l_2^2(f(x_k;\theta))]}{\partial\theta_{ij}}$ is the gradients of the squared L2 norm of the learned neural network with respect to the parameter θ_{ij}. The output of $f(x_k;\theta)$ is the loss of the network.

In Equation 4, θ_{ij} is the parameter of the novel model of the connections between pairs of neurons n_i and n_j in two consecutive layers. θ^* represents the pre-trained parameters, which can be assumed as a local minimum of the parameter space as shown in Equation 6:

$$\theta^* = \arg\min_{\theta}\{-\log p(\theta|D_B)\} \tag{6}$$

3.2.2 Novel Task Training

In the novel task training process, we train the novel model and evaluate the base model on novel tasks simultaneously. The function of the neural network whose output is the loss of the model can be represented as follows:

$$L_N = f_t(x;\theta_{t-1}) \tag{7}$$

where t is the timestep. We compute the loss by the proposed Cross Entropy with Logits Calibration (CELC) for classification tasks and Mean Squared Error with Logits Calibration (MSELC) for regression tasks as follows:

$$f_t = L_{CELC}(Q(x;\theta_{t-1})) \parallel L_{MSELC}(Q(x;\theta_{t-1})) \tag{8}$$

where $Q(x;\theta_{t-1})$ represents the function of the novel model and the base model whose output are logits with data inputs x and parameters θ_{t-1} of the model in timestep $t-1$.

3.2.3 Parameter Calibration with Target Drift

Multi-task learning tries to achieve satisfying performance on both of the base tasks and novel tasks. However, one of the most essential problems of multi-task learning is that it is inconsistent with adaptation (Chen et al., 2020). To deal with this problem, we introduce Parameter Calibration with Target Drift, a method allowing the objective function to gradually drift from L_B to L_N with the annealing coefficient $\lambda(t)$:

$$L_T = \lambda(t)L_N + (1-\lambda(t))L_B \tag{9}$$

where t refers to the timestep during the training process. We compute $\lambda(t) = \frac{1}{1+\exp(-r\cdot(t-t_0))}$ as the sigmoid annealing function (Kiperwasser and Ballesteros, 2018), where r is the hyperparameter controlling the annealing rate and t_0 is the hyperparameter controlling the timesteps.

When $t < t_0$, $-r\cdot(t-t_0)$ will be positive. In this case, if $r \to \infty$, then $\exp(-r\cdot(t-t_0)) \to \infty$, $\lambda(t) \to 0$, $L_T = L_B$. When $t > t_0$, $-r\cdot(t-t_0)$ will be negative. In this case, if $r \to \infty$, then $\exp(-r\cdot(t-t_0)) \to 0$, $\lambda(t) \to 1$, $L_T = L_N$. At this moment, our method can be regarded as fine-tuning. Otherwise, if $r \to 0$, then $-r\cdot(t-t_0) \to 0$, $\exp(-r\cdot(t-t_0)) \to 1$, $\lambda(t) \to 0.5$, $L_T = 0.5L_N + 0.5L_B$. In this case, our method can be regarded as multi-task learning. Finally, if $0 < r < \infty$, then $0 < \lambda < 1$. In this case, our method can be regarded as a trade off between fine-tuning and multi-task learning. With time goes by, the objective of the model drifts from base tasks to novel tasks gradually. Finally, by doing back propagation, we update parameters of the novel model with parameter calibration.

3.3 LPC Algorithm

In this section, we combine the Logits Calibration (CELC) with all three parts of Parameter Calibration (BPP, NTT, and PCTD) into a brand-new optimization algorithm as shown in Algorithm 1. The Logits Calibration (LC) part is shown from line 6 to line 8. The Parameter Calibration (PC) part is shown from line 9 to line 18 and line 24. Here, we introduce LPC Algorithm which integrates the quadratic penalty with importance weights and the annealing coefficient into a complete optimization algorithm by decoupling them from the gradient update in Adam optimization algorithm (Kingma and Ba, 2014). The orange part in Algorithm 1 depicts how LPC is different from Adam.

From line 9 to line 16, we show how we calculate Ω by initializing Ω as a tensor filled with the scalar value one. The size of Ω are the same as that of parameter size of the base model and the novel model. From line 11 to line 14, we accumulate the gradients of the squared L2 norm of the learned neural network over the given data inputs to obtain

Table 1: Experimental Results. All of results are the medians over 5 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). The metric for STS-B is corr (Average of Pearson and Spearman Correlation Coefficient). All other metrics are acc (Accuracy).

Model	**CoLA** mcc 8.5k	**MRPC** acc 3.7k	**QNLI** acc 105k	**RTE** acc 2.5k	**SST-2** acc 67k	**STS-B** corr 7k	**WNLI** acc 634	**Avg** acc	**Avg** mcc	**Avg** corr
BERT-base + PALs (Stickland and Murray, 2019)	51.2	84.6	90.0	76.0	92.6	85.8	N/A	N/A	51.2	85.8
BERT-large + Adapters (Houlsby et al., 2019)	59.5	89.5	90.7	71.5	94.0	86.9	N/A	N/A	59.5	86.9
BERT-large + Diff pruning (Guo et al., 2020)	60.5	87.0	92.9	68.1	93.8	83.5	N/A	N/A	60.5	83.5
BERT-base + Adam (rerun) $_{Median}$	57.1	81.3	91.0	63.9	93.1	89.2	56.3	77.1	57.1	89.2
BERT-base + RecAdam (rerun) $_{Median}$	59.9	85.7	91.4	70.8	93.1	90.0	56.3	79.5	59.9	90.0
BERT-base + LPC $_{Median}$	**61.8**	**86.1**	**91.5**	**74.7**	**93.2**	**90.3**	**62.0**	**81.5**	**61.8**	**90.3**
ALBERT-xxlarge + Adam (rerun) $_{Median}$	70.5	88.8	93.7	72.9	91.1	92.2	69.0	83.1	70.5	92.2
ALBERT-xxlarge + RecAdam (rerun) $_{Median}$	70.5	87.5	93.9	**89.5**	93.9	92.8	78.9	88.7	70.5	92.8
ALBERT-xxlarge + LPC $_{Median}$	**74.1**	**89.4**	**94.3**	**89.5**	**95.8**	**93.3**	**81.7**	**90.1**	**74.1**	**93.3**

importance weights Ω_{ij} for parameter θ_{ij}. In line 15, we compute the mean value of Ω_{ij} by dividing it by N. Here, N is the total number of data inputs at a given phase. In line 18, we compute the gradients of the loss function as a weighted combination of the gradients of L_N and L_B. In line 24, we update the network parameters θ by the gradient descent method.

4 Evaluations

In this section, we evaluate LPC on the General Language Understanding Evaluation (GLUE) (Wang et al., 2018) benchmark. We compare our model with PALs (Stickland and Murray, 2019), Adapters (Houlsby et al., 2019), Diff Pruning (Guo et al., 2020), Adam (Kingma and Ba, 2014), and RecAdam (Chen et al., 2020).

4.1 Experimental Setup

We perform the experiments based on deep pre-trained language models BERT-base[1] (Devlin et al., 2018) and ALBERT-xxlarge (Lan et al., 2019), respectively. BERT aims to learn a Transformer encoder for representing texts. BERT is a multi-layer bidirectional Transformer encoder. In BERT, the Transformer uses bidirectional self-attention. ALBERT is an advanced deep pre-trained language model with lower memory consumption and faster training speed than BERT. ALBERT improves BERT using parameter reduction techniques and employing self-supervised loss for sentence-order prediction (SOP).

We evaluate our approach LPC on the GLUE benchmark. GLUE benchmark is a collection of resources for training, evaluating, and analyzing natural language understanding (NLU) systems (Wang et al., 2018). It contains the following 9 different scenarios: (1) Single-Sentence Scenarios: **CoLA** The Corpus of Linguistic Acceptability (Warstadt et al., 2019), and **SST-2** The Stanford Sentiment Treebank (Socher et al., 2013); (2) Similarity and Paraphrase Senarios: **MRPC** The Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005), **QQP** The Quora Question Pairs dataset[2], and **STS-B** The Semantic Textual Similarity Benchmark (Cer et al., 2017); (3) Inference Scenarios: **MNLI** The Multi-Genre Natural Language Inference Corpus (Williams et al., 2017), **QNLI** The Stanford Question Answering Dataset (Rajpurkar et al., 2016), **RTE** The Recognizing Textual Entailment datasets (Dagan et al., 2005) (Haim et al., 2006) (Giampiccolo et al., 2007) (Bentivogli et al., 2009), and **WNLI** The Winograd Schema Challenge (Levesque et al., 2012).

4.2 Results

We perform experiments on 7 scenarios of the GLUE benchmark as shown in Table 1. From the experimental results with BERT-base model, we outperform BERT-base with Adam and BERT-base with RecAdam (Chen et al., 2020) models on 7 out of 7 scenarios of the GLUE benchmark and achieve 2.5% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 3.2% improvements measured by mcc on CoLA, and 0.3% improvements measured by corr on STS-B compared with RecAdam. Especially, we achieve significant improvements on WNLI corpus (+10.1%) and CoLA corpus (+5.5%). From the experimen-

[1] https://huggingface.co/transformers/model_doc/bert.html

[2] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Table 2: The Results of Ablation Study on Adam, Logtis Calibration (LC), Parameter Calibration (PC), and Logits and Parameter Calibration (LPC). All of results are the medians over 5 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). The metric for STS-B is corr (Average of Pearson and Spearman Correlation Coefficient). All other metrics are acc (Accuracy).

Model	**CoLA** mcc 8.5k	**MRPC** acc 3.7k	**QNLI** acc 105k	**RTE** acc 2.5k	**SST-2** acc 67k	**STS-B** corr 7k	**WNLI** acc 634	**Avg** acc	**Avg** mcc	**Avg** corr
BERT-base + Adam (rerun) $_{Median}$	57.1	81.3	91.0	63.9	93.1	89.2	56.3	77.1	57.1	89.2
BERT-base + LC $_{Median}$	61.2	82.8	**91.5**	66.8	92.3	89.3	56.3	77.9	61.2	89.3
BERT-base + PC $_{Median}$	61.4	85.3	**91.5**	72.2	92.8	90.2	57.7	79.9	61.4	90.2
BERT-base + LPC $_{Median}$	**61.8**	**86.1**	**91.5**	**74.7**	**93.2**	**90.3**	**62.0**	**81.5**	**61.8**	**90.3**

tal results on ALBERT-xxlarge model, we outperform ALBERT-xxlarge with Adam and ALBERT-xxlarge with RecAdam models on 7 out of 7 scenarios of the GLUE benchmark and achieve 1.6% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 5.1% improvements measured by mcc on CoLA, and 0.5% improvements measured by corr on STS-B compared with RecAdam. Specifically, we achieve great improvements on CoLA corpus (+5.1%) and WNLI corpus (+3.5%). What is more, there is no obvious relationship between the size of the datasets and the results. Namely, our model performs well on both large datasets and small datasets.

4.3 Ablation Study

As we have mentioned, our model (LPC) has two important components, Logits Calibration (LC) and Parameter Calibration (PC). Thus, we do ablation study on these two components separately with BERT-base pre-trained model on the 7 scenarios of the GLUE benchmark. The results of albation study is shown in Table 2. We can see both of LC and PC achieve better results than the baseline Adam. LPC achieves the best results among all three models. Compared with Adam, LC achieves 1.0% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 7.2% improvements measured by mcc on CoLA, and 0.1% improvements measured by corr on STS-B. Compared with Adam, PC achieves 3.6% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 7.5% improvements measured by mcc on CoLA, and 1.1% improvements measured by corr on STS-B. Compared with Adam, LPC achieves 5.7% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 8.2% improvements measured by mcc on CoLA, and 1.2% improvements measured by corr on STS-B.

5 Conclusion

In this paper, we propose Logits and Parameter Calibration (LPC) framework on continual learning to deal with the catastrophic forgetting problem. The proposed framework includes two important components, Logits Calibration (LC) and Parameter Calibration (PC). We propose the Cross Entropy Loss with Logits Calibration (CELC) for classification tasks and the Mean Squared Error with Logits Calibration (MSELC) for regression tasks. The Parameter Calibration consists of three components: (1) Base Parameter Preservation (BPP), (2) Novel Task Training (NTT), and (3) Parameter Calibration with Target Drift (PCTD). What is more, we introduce LPC algorithm by integrating the Logits Calibration and all three parts of Parameter Calibration (BPP, NTT, and PCTD) into a brand-new optimization algorithm based on the well-known Adam optimization algorithm. We do experiments on 7 scenarios of GLUE benchmark and achieve state-of-the-art performance on all the 7 scenarios. The limitation of our work is that our work cannot handle the online learning settings. This means that when data comes in an online manner (sometimes without labels), we have no technique to handle it. Thus, our future direction is to make our model fit the online learning settings. We also release the open-source LPC Algorithm to further benefit the continual learning research community.

References

Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2019. A continual learning survey: Defying forgetting in classification tasks. *arXiv preprint arXiv:1909.08383*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ronald A Fisher. 1922. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and William B Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL workshop on textual entailment and paraphrasing*, pages 1–9.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

Demi Guo, Alexander M Rush, and Yoon Kim. 2020. Parameter-efficient transfer learning with diff pruning. *arXiv preprint arXiv:2012.07463*.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. 2019. Spottune: transfer learning through adaptive finetuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4805–4814.

R Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Eliyahu Kiperwasser and Miguel Ballesteros. 2018. Scheduled multi-task learning: From syntax to translation. *Transactions of the Association for Computational Linguistics*, 6:225–240.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017a. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, page 201611835.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017b. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Anna Kukleva, Hilde Kuehne, and Bernt Schiele. 2021. Generalized and incremental few-shot learning by explicit learning and calibration without forgetting. *arXiv preprint arXiv:2108.08165*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. *ICLR*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer.

Xuhong Li, Yves Grandvalet, and Franck Davoine. 2018. Explicit inductive bias for transfer learning with convolutional networks. *arXiv preprint arXiv:1802.01483*.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*, pages 97–105. PMLR.

James Martens. 2014. New insights and perspectives on the natural gradient method. *arXiv preprint arXiv:1412.1193*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P Lillicrap, and Greg Wayne. 2019. Experience replay for continual learning. *NeurIPS*.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. *arXiv preprint arXiv:1705.08690*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. In *32nd Conference on Neural Information Processing Systems (NeurIPS)*.

Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. 2020. Maintaining discrimination and fairness in class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13208–13217.

A LPC Appendix

A.1 Hyperparameter Analysis

In this section, we analyze the most essential hyperparameters we set in the LPC model. δ is a hyperparameter controlling the level of regularization. Setting δ between 1 and 2 balances the level of regularization. Ω is a parameter measuring the importance of different parameters in the model. Initializing Ω as ones makes the importance of each parameter more balanced. The hyperparameter u_e controls the updating epochs of Ω. Typically, u_e is between 1 and 16. w_s is a hyperparameter controlling the number of steps of updating with low learning rate before/at the beginning of the training process. We set w_s as 0, 320 or 640. After these warmup steps, we will use the regular learning rate to train our model until convergence. In other words, we have a few steps adjustment before we actually train the model. From our experiments, we find that the hyperparameters δ, u_e, and w_s have great influences on the experimental results.

Figure 2 shows the comparison of different hyperparameter (δ, u_e, and w_s) initializations on CoLA, MRPC, and STS-B corpora with BERT-base pre-trained model.

In Figure 2 chart (1), we set $u\_e = 2$ and $w\_s = 320$. We can see when δ increases from 1 to 1.2, the performance of the model decreases a lot. After that, the performance of the model increases with δ increasing. The model achieves the best results when $\delta = 2$ on all the three corpora.

In Figure 2 chart (2), we set $\delta = 2$ and $w\_s = 320$. We can see the performance of the model varies with different values of u_e. Specifically, when u_e increases from 1 to 2, the performance of the model improves a lot. While when u_e increases from 2 to 4, the performance decreases in a large extent. However, when u_e increases from 4

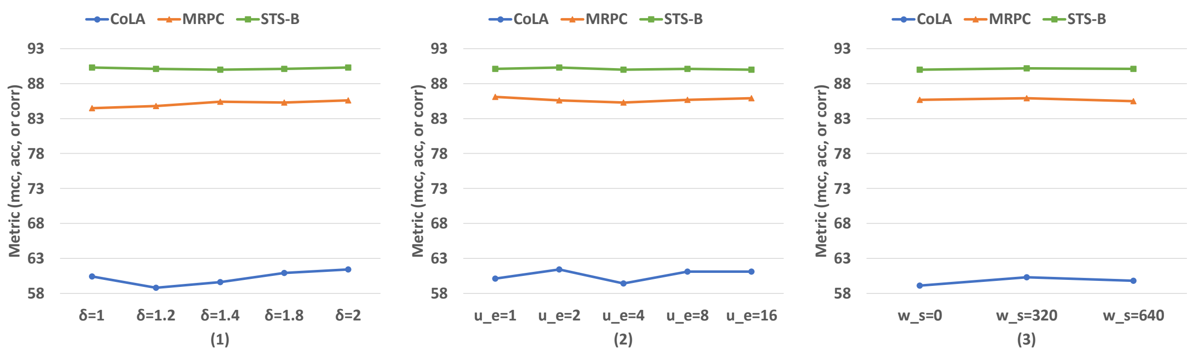

Figure 2: Comparison of Different Hyperparameter Initializations on CoLA, MRPC, and STS-B Corpora with BERT-base Pre-trained Model. The metric for CoLA, MRPC, and STS-B are mcc (Matthew Correlation Coefficient), acc (Accuracy), and corr (Average of Pearson and Spearman Correlation Coefficient), respectively.

Table 3: The Results of Ablation Study on Adam, Logtis Calibration (LC), Parameter Calibration (PC), and Logits and Parameter Calibration (LPC) with ALBERT-xxlarge Pre-trained Model. All of results are the medians over 5 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). The metric for STS-B is corr (Average of Pearson and Spearman Correlation Coefficient). All other metrics are acc (Accuracy).

Model	**CoLA** mcc 8.5k	**MRPC** acc 3.7k	**QNLI** acc 105k	**RTE** acc 2.5k	**SST-2** acc 67k	**STS-B** corr 7k	**WNLI** acc 634	**Avg** acc	**Avg** mcc	**Avg** corr
ALBERT-xxlarge + Adam (rerun) $_{Median}$	70.5	88.0	93.7	72.9	91.1	92.2	69.0	82.9	70.5	92.2
ALBERT-xxlarge + LC $_{Median}$	71.0	88.5	93.8	88.4	95.5	92.3	70.4	87.3	71.0	92.3
ALBERT-xxlarge + PC $_{Median}$	**74.1**	88.6	94.0	88.4	95.7	92.9	74.6	88.3	**74.1**	92.9
ALBERT-xxlarge + LPC $_{Median}$	**74.1**	**89.4**	**94.3**	**89.5**	**95.8**	**93.3**	**81.7**	**90.1**	**74.1**	**93.3**

to 8, the model performance increases again. When u_e increases from 8 to 16, there is a slight increase on the performance.

In Figure 2 chart (3), we set $\delta = 1$ and $u\_e = 1$. We can see when w_s increases from 0 to 320, there is a big increase on all three corpora. However, when w_s increases from 320 to 640, the performance decreases slightly, instead.

A.2 Forgetting Analysis

In addition to computing accuracy, we also measure the forgetting by computing the euclidean distance between the parameters of the novel model and the base model on CoLA corpus. Figure 3 shows the comparison of parameter forgetting from the first epoch to the last epoch and the corresponding accuracy after convergence with epoch increasing among LPC, RecAdam and Adam. In Figure 3 chart (1), with epoch increasing, the euclidean distance of Adam increases a lot, which means the forgetting of Adam is huge with the epoch increasing. However, our model (LPC) reduces the forgetting in a large extent compared with Adam and achieves similar forgetting with RecAdam, another baseline trying to reduce carastrophic forgetting. Here, the forgettnig of our model is a little bit worse than RecAdam is because our model tries to remember the most important parameters while forget unimportant parameters. Furthermore, in Figure 3 chart (2), we can see our model (LPC) achieves the best accuracy compared with RecAdam and Adam all the time after convergence (Epoch 4).

A.3 Ablation Study with ALBERT-xxlarge Model

As we have mentioned, our model (LPC) has two important components, Logits Calibration (LC) and Parameter Calibration (PC). In addition to doing ablation study with BERT-base pre-trained model, we also do ablation study on these two components separately with ALBERT-xxlarge pre-trained model on the 7 scenarios of the GLUE benchmark. The results of albation study with ALBERT-xxlarge pre-trained model is shown in Table 3. We can see with ALBERT-xxlarge pre-trained model, both of LC and PC achieve better results than the baseline Adam. LPC achieves the best results among all three models. Compared with Adam, LC achieves 5.3% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 0.7% im-

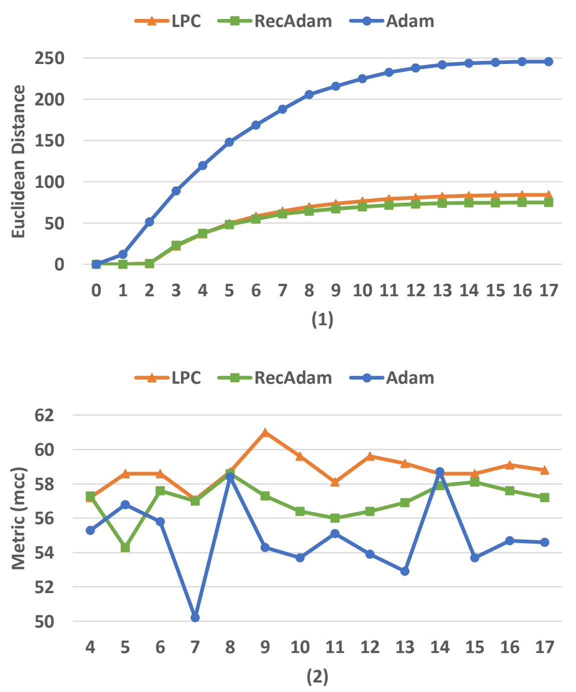

Figure 3: Comparison of Parameter Forgetting and Model Performance with the Epoch Increasing on CoLA Corpus with BERT-base Pre-trained Model.

provements measured by mcc on CoLA, and 0.1% improvements measured by corr on STS-B. Compared with Adam, PC achieves 6.5% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 5.1% improvements measured by mcc on CoLA, and 0.8% improvements measured by corr on STS-B. Compared with Adam, LPC achieves 8.7% improvements on average measured by acc on MRPC, QNLI, RTE, SST-2, and WNLI, 5.1% improvements measured by mcc on CoLA, and 1.2% improvements measured by corr on STS-B. Thus, we can conclude that our model can achieve state-of-the-art results with different pre-trained model. These results prove the scalability of our model.