LPC: A Logits and Parameter Calibration Framework for Continual Learning

Anonymous ACL submission

Abstract

When we execute the typical fine-tuning 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 training the model on newly emerged tasks). Existing replay-based methods need extra storage for old data to update the parameters of the previous classifier to overcome catastrophic forgetting. Our work aims to achieve the sequential/continual learning of knowledge without accessing the old data. The core idea is to calibrate the parameters and logits (output) so that preserving old parameters and generalized learning on new concepts can be solved simultaneously. Our proposed framework includes two major components, the Logits Calibration (LC) and Parameter Calibration (PC). The LC focuses on calibrating the learning of novel models with old models, and PC aims to preserve the parameters of old models. These two operations can maintain the old knowledge while learning new tasks without storing previous data. We do experiments on 9 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 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 three categories: (1) storing exemplars from previous classes (Rebuffi et al., 2017a; Rolnick et al., 2019); (2) regularizing the parameters when we fine-tune the model on new tasks (Kirkpatrick et al., 2017b; Li and Hoiem, 2017a; Aljundi et al., 2018); (3) dedicating different model parameters to each task to prevent any possible forgetting (Mallya and Lazebnik, 2018; Serra 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.

Typical replay methods 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 replay 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 focuses on both encoder parameters and output classifiers: when the new/current task comes, we evaluate the previous model on current task, and then train a new model on current task, the differences of output and parameter from the two models (old, new) are used for calibration. Specifically, for learning new concepts, we add the logits calibration that can amplify the softmax output of the previous model, and overcome the bias towards the current category (Zhao et al., 2020). Also, for encouraging the model to maintain previously learned knowledge, we propose to calibrate the encoder parameters, which simulating the training objective using the parameters of the previous model. Then, during the training on current tasks, the model will calibrate parameters with target drift from the previous tasks to the current tasks to balance new task learning and old knowledge maintenance. It allows the model to focus on current tasks by making the learning objective drifting from the previous tasks to current

tasks gradually.

Our proposed Logits and Parameter Calibration based continual learning framework (LPC) is shown in Figure 1, it reduces catastrophic forgetting without further data storage. The 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. Second, we develop a new mechanism by calibrating the logits and parameters with target drifting from previous tasks to current 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

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 online learning process, data sometimes arrives continuously in a non i.i.d. way, tasks may change over time, and entirely new tasks can emerge. (Nguyen et al., 2017). 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 with a generative model. iCaRL (Rebuffi et al., 2017b) store a subset of exemplars per class. GEM (Lopez-Paz and Ranzato, 2017) projects the estimated gradient direction on the feasible region outlined by previous task gradients through first order Taylor series approximation. A-GEM (Chaudhry et al., 2018) relaxes the problem to project on one direction estimated by randomly selected samples form a previous task data buffer. Regularization-based methods eschews storing raw inputs, prioritizing privacy, and alleviating memory requirements. Instead, an extra regularization term is introduced in the loss function, consolidating previous knowledge when learning on new data. LwF (Li and Hoiem, 2017b) uses the previous model output as soft labels for previous tasks. EWC (Kirkpatrick et al., 2017a) penalizes changes to important parameters. MAS (Aljundi et al., 2018) suggests unsupervised importance estimate, allowing increased flexibility and online user adaptation. Parameter isolation methods dedicate different model parameters to each task to prevent any possible forgetting. PackNet (Mallya and Lazebnik, 2018) iteratively assigns parameter subsets to consecutive tasks by constituting binary masks. HAT (Serra et al., 2018) requires only one training phase, incorporating task-specific embeddings for attention masking.

Our method is an advanced regularization-based method based on LCwoF (Kukleva et al., 2021) and RecAdam (Chen et al., 2020). LCwoF revises the original cross entropy loss by adding the summation of the exponential logits of the previous classes classifier to the denominator to change the normalization scale. However, the normalization part of LCwoF is flawed as the summation of all the normalization items is not 1. RecAdam is a method based on EWC. However, RecAdam treats each parameter the same, which ignores that different parameters weigh differently in a neural network.

3 Proposed Approach

Here we introduce our proposed Logits and Parameter Calibration framework, LPC, which 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) execute the calibration on the parameters to reduce the parameter forgetting. For the Logits Calibration, we apply the Cross Entropy with Logits Calibration (CELC) for classification

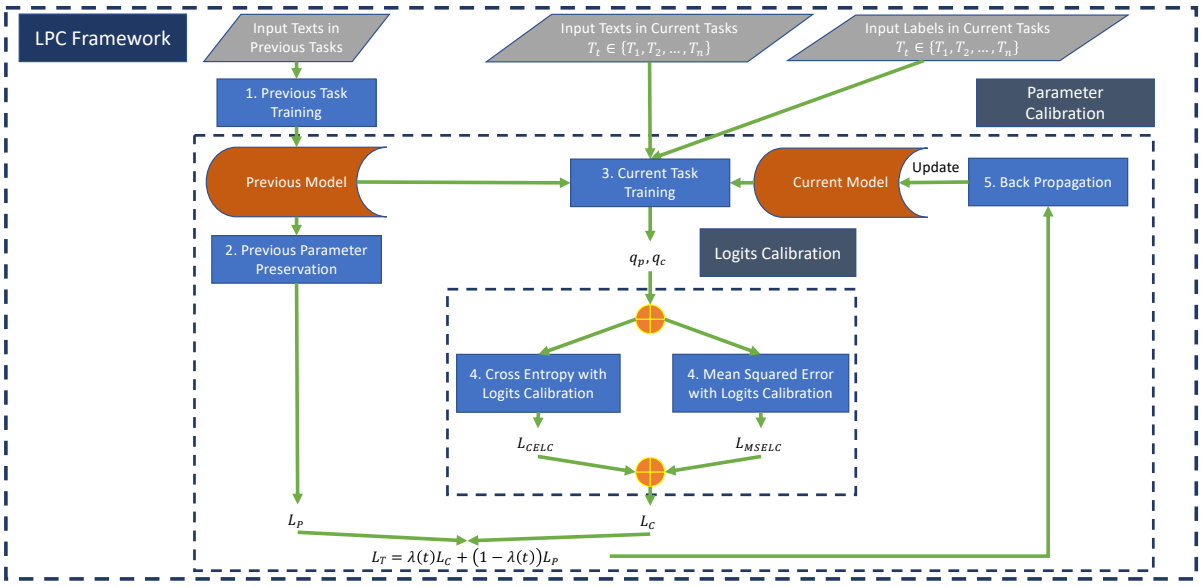

Figure 1: The Overview of LPC Framework. (1) We first train the previous model on the large-scale input texts and initialize our model (for current tasks) same as the previous one. (2) We do previous parameter preservation to preserve the parameters of the trained model, and compute the loss for the previous model L_P. (3) During the current task training, we compute logits q_p and q_c. (4) We do logits calibration (e.g., cross entropy with logits calibration for classification tasks) given q_p and q_c using L_{CELC} or L_{MSELC} for regression tasks as the loss for the current model L_C. Then, the objective function drifts from L_P to L_C gradually with the annealing coefficient $\lambda(t)$. (5) Finally, we perform back propagation to update the parameters of the current model.

tasks (or MSE with Logits Calibration (MSELC) for regression). For the Parameter Calibration (PC), it consists of two components: (1) Previous Parameter Preservation (PPP) that aims to preserve the parameters from previous tasks, and (2) Current Task Training (CTT) that to reduce the drifting from the previous tasks to current ones gradually, when update on new task. The LPC algorithm integrating the Logits Calibration (CELC or MSELC) and Parameter Calibration (PPP and CTT) into a brand-new optimization algorithm based on the Adam (Kingma and Ba, 2014) optimizer.

3.1 Logits Calibration

In this section, we introduce our proposed logits 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.

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:

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

where N_C is the total number of classes in the current tasks. $q_{c,i}$ represents the output logits for class i of the current model on the current 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 current 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 previous model on the current tasks and compute the output logits of the previous model q_p.

Inspired by the idea from LCwoF (Kukleva et al., 2021), we add the logits information of the previous model into the cross entropy loss. Different from LCwoF, we do logits calibration by adding

the difference between each logits of the current model and the previous model $(q_{c,i} - q_{p,i})$ to the corresponding output logits $q_{c,i}$ of the current model for class i. In this way, the model can preserve important output logits information of each class for the previous model in an element-wise way. Our proposed Cross Entropy with Logits Calibration (CELC) Loss is shown in Equation 2:

$$L_{celc} = -\sum_{i=1}^{N_C} p_i \log\left(\frac{\exp(q_{c,i} + \mu(q_{c,i} - q_{p,i}))}{\sum_{j=1}^{N_C} \exp(q_{c,j} + \mu(q_{c,j} - q_{p,j}))}\right) \quad (2)$$

where we multiply the difference between the logits for the current model and the previous model $(q_c - q_p)$ 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 $q_{c,i}$ is larger than $q_{p,i}$, otherwise, giving a penalty to the logits for the correct class if $q_{c,i}$ is smaller than $q_{p,i}$.

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 previous model on current tasks and take out the output logits q_p. We measure the difference between the output logits of the current model and the previous model by adding a squared L2 norm on the difference between logits of the current model and the previous model $(q_c - q_p)^2$ to the original function. The proposed Mean Squared Error with Logits Calibration Loss L_{MSELC} is shown in Equation 3:

$$L_{MSELC}(q) = (q_c - p)^2 + \mu(q_c - q_p)^2 \quad (3)$$

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 current model are different from the previous model by adding the squared difference between the parameters of the current model and the previous model to the training loss. It includes two parts: (1) Previous Parameter Preservation (PPP), and (2) Current Task Training (CTT).

3.2.1 Previous Parameter Preservation

As shown in Figure 1, in Previous Parameter Preservation, we try to maintain the parameters of the previous model. Here, we add a regularization to the posterior of parameters given data. The Previous Parameter Preservation method can be regarded as an improved method derived from RecAdam (Kirkpatrick et al., 2017b). Different from RecAdam, PPP measures the importance of each parameter by introducing the importance weights Ω. During training, the current 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 L_P is shown in Equation 4:

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

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_P. $H_{prior}(\theta^*)$ 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 $-\log p(\theta^*|D_P)$ as it is a constant term with respect to θ^*. $\Omega(\theta)$ is estimated by the sensitivity

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
  $q_{c,t} \leftarrow Q_{c,t}(x, \theta_{t-1})$ ▷ compute output logits for the current model
  $q_{p,t} \leftarrow Q_{p,t}(x, \theta^*)$ ▷ compute output logits for the previous model
  $\nabla(f_t(x; \theta_{t-1})) \leftarrow \nabla(L_{CELC}(q_{c,t}, q_{p,t}) \parallel L_{MSELC}(q_{c,t}, q_{p,t}))$ ▷ 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

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 current model of the connections between pairs of neurons n_i and n_j in two consecutive layers. θ^* represents parameters of the previous model, which can be assumed as a local minimum of the parameter space as shown in Equation 6:

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

3.2.2 Current Task Training with Continual Learning

In the current task training process, we train the current model and evaluate the previous model on current tasks simultaneously. In the continual learning setting, first, we train with Task T_1, then evaluate on Task T_1. Second, our current task training will cover T_2 task, and then evaluate on instances related to Task T_1 and T_2. Next, our current task training will focus on Task T_3 and then evaluate on instances related to Task T_1, T_2 and T_3 and so on. Here, we present the detail of one particular current task where how we incorporate drift from previous tasks to the current task. The function of the neural network whose output is the loss of the

Table 1: Experimental Results on Single Task. 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). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	CoLA mcc 8.5k / 1k	MNLI acc 393k / 20k	MRPC acc 3.7k / 1.7k	QNLI acc 105k / 5.4k	QQP acc 364k / 391k	RTE acc 2.5k / 3k	SST-2 acc 67k / 1.8k	STS-B corr 7k / 1.4k	WNLI acc 634 / 146	Avg acc	Avg mcc	Avg corr
BERT-base + Adam (rerun) $_{Median}$	57.1	84.2	81.3	91.0	90.7	63.9	93.1	89.2	56.3	80.1	57.1	89.2
BERT-base + EWC (rerun) $_{Median}$	54.0	84.5	83.4	91.4	90.6	67.9	92.7	89.6	33.8	77.8	54.0	89.6
BERT-base + MAS (rerun) $_{Median}$	58.0	83.5	84.9	91.2	91.0	72.2	91.9	89.5	52.1	81.0	58.0	89.5
BERT-base + SI (rerun) $_{Median}$	58.8	83.6	84.3	91.0	91.2	71.1	91.9	89.8	56.3	81.3	58.8	89.8
BERT-base + RecAdam (rerun) $_{Median}$	59.9	82.6	85.7	91.4	88.9	70.8	93.1	90.0	56.3	81.3	59.9	90.0
BERT-base + LPC $_{Median}$	**61.8**	**85.0**	**86.1**	**91.5**	**91.5**	**74.7**	**93.2**	**90.3**	**62.0**	**83.4**	**61.8**	**90.3**
ALBERT-xxlarge + Adam (rerun) $_{Median}$	70.5	88.0	88.8	93.7	81.3	72.9	91.1	92.2	69.0	83.5	70.5	92.2
ALBERT-xxlarge + EWC (rerun) $_{Median}$	70.5	88.2	85.0	94.2	88.5	74.0	93.9	91.4	63.4	83.9	70.5	91.4
ALBERT-xxlarge + MAS (rerun) $_{Median}$	71.4	89.4	88.6	94.2	92.1	84.1	94.4	92.0	76.1	88.4	71.4	92.0
ALBERT-xxlarge + SI (rerun) $_{Median}$	69.8	88.1	89.0	94.2	91.7	87.4	95.0	92.2	74.6	88.6	69.8	92.2
ALBERT-xxlarge + RecAdam (rerun) $_{Median}$	70.5	88.5	87.5	93.9	87.5	**89.5**	93.9	92.8	78.9	88.5	70.5	92.8
ALBERT-xxlarge + LPC $_{Median}$	**74.1**	**89.8**	**89.4**	**94.3**	**92.3**	**89.5**	**95.8**	**93.3**	**81.7**	**90.4**	**74.1**	**93.3**

model can be represented as follows:

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

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 current model and the previous model whose output are logits with data inputs x and parameters θ_{t-1} of the model in timestep $t-1$.

To adapt the target task from previous tasks to current tasks, we introduce a method allowing the objective function to gradually drift from L_P to L_C with the annealing coefficient $\lambda(t)$:

$$L_T = \lambda(t)L_C + (1 - \lambda(t))L_P \tag{9}$$

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 $\exp(-r \cdot (t - t_0)) \to \infty$, $\lambda(t) \to 0$, $L_T = L_P$. When $t > t_0$, $-r \cdot (t - t_0)$ will be negative. In this case, if $r \to \infty$, then $\exp(-r \cdot (t - t_0)) \to 0$, $\lambda(t) \to 1$, $L_T = L_C$. Otherwise, if $r \to 0$, then $-r \cdot (t - t_0) \to 0$, $\exp(-r \cdot (t - t_0)) \to 1$, $\lambda(t) \to 0.5$, $L_T = 0.5L_C + 0.5L_P$. Finally, if $0 < r < \infty$, then $0 < \lambda < 1$. With time goes by, the objective of the model drifts from previous tasks to current tasks gradually. Finally, by doing back propagation, we update parameters of the current model with parameter calibration.

3.3 LPC Algorithm

In this section, we combine the Logits Calibration (CELC or MSELC) with Parameter Calibration (PPP and CTT) 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 16 and line 22. 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 RecAdam (Chen et al., 2020), more specific description could be viewed in Appendix.

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 Adam (Kingma and Ba, 2014), EWC (Kirkpatrick et al., 2017b), MAS (Aljundi et al., 2018), SI (Zenke et al., 2017), and RecAdam (Chen et al., 2020).

Table 2: Experimental Results on Sequentially Emerged Classification Tasks (The order of emergence: CoLA, MRPC, QNLI, QQP, RTE, SST-2, and WNLI). The results are the validation results on all the 7 classification tasks using the model sequentially trained on all the 7 classification tasks. All of results are the medians over 10 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). All other metrics are acc (Accuracy). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	**CoLA** mcc / fgt 8.5k / 1k	**MRPC** acc / fgt 3.7k / 1.7k	**QNLI** acc / fgt 105k / 5.4k	**QQP** acc / fgt 364k / 391k	**RTE** acc / fgt 2.5k / 3k	**SST-2** acc / fgt 67k / 1.8k	**WNLI** acc / fgt 634 / 146	**Avg** acc / fgt	**Avg** mcc / fgt
BERT-base + EWC (rerun) $_{Median}$	2.8 / 49.0	57.3 / 21.4	44.2 / 45.5	70.3 / 20.3	46.9 / 9.8	72.6 / 18.2	22.5 / 0.0	52.3 / 19.2	2.8 / 49.0
BERT-base + MAS (rerun) $_{Median}$	20.1 / 30.7	63.7 / 20.8	70.1 / 3.4	48.1 / 21.1	52.0 / **1.5**	67.3 / 7.9	29.1 / 0.0	55.1 / 9.1	20.1 / 30.7
BERT-base + SI (rerun) $_{Median}$	11.3 / 46.6	52.7 / 29.8	81.9 / 7.7	57.7 / 25.2	57.8 / 9.0	86.7 / 1.1	28.2 / 0.0	60.8 / 12.1	11.3 / 46.6
BERT-base + RecAdam (rerun) $_{Median}$	2.0 / 58.6	51.1 / 19.7	49.4 / 11.5	50.1 / 25.3	49.8 / 3.3	48.7 / 31.8	11.3 / 0.0	43.4 / 15.3	2.0 / 58.6
BERT-base + LPC $_{Median}$	**27.1 / 25.8**	**63.9 / 18.8**	**83.0 / 2.2**	**70.5 / 11.7**	**58.5** / 6.5	**87.3 / 0.4**	**32.4** / 0.0	**65.9 / 6.6**	**27.1 / 25.8**

Table 3: The Results of Ablation Study on Adam, Logits 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). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	**CoLA** mcc 8.5k / 1k	**MRPC** acc 3.7k / 1.7k	**RTE** acc 2.5k / 3k	**SST-2** acc 67k / 1.8k	**STS-B** corr 7k / 1.4k	**WNLI** acc 634 / 146	**Avg** acc	**Avg** mcc	**Avg** corr
BERT-base + Adam (rerun) $_{Median}$	57.1	81.3	63.9	93.1	89.2	56.3	73.7	57.1	89.2
BERT-base + Adam + LC $_{Median}$	61.2	82.8	66.8	92.3	89.3	56.3	74.6	61.2	89.3
BERT-base + PC $_{Median}$	61.4	85.3	72.2	92.8	90.2	57.7	77.0	61.4	90.2
BERT-base + LPC $_{Median}$	**61.8**	**86.1**	**74.7**	**93.2**	**90.3**	**62.0**	**79.0**	**61.8**	**90.3**

4.1 Dataset

We evaluate our approach LPC on the GLUE benchmark, which is a collection of resources for training, evaluating, and analyzing in the 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[1], 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). We perform our experiments on 9 out of 9 corpora.

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

4.2 Experimental Setup

We perform the experiments based on deep pre-trained language models BERT-base[2] (Devlin et al., 2018) and ALBERT-xxlarge (Lan et al., 2019), respectively. BERT is a multi-layer bidirectional Transformer encoder using bidirectional self-attention to learn a Transformer encoder for representing texts. 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 have two different experimental settings: (1) single task setting, and (2) continual learning setting. In the single task setting, we treat the pretraining as the previous task, and choose one of the tasks as the current task. In the continual learning setting, we train the model on several different tasks sequentially in a given order. Each time after training, we evaluate the current model on all the tasks we have trained on. For example, if our current model is trained on QNLI and previously trained with CoLA and MRPC. During evaluation, we use instances related to all 3 tasks (CoLA, MRPC, and

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

Table 4: Different Task Sequences Used for Continual Classification Setting

Order	# Task Sequence
1	CoLA → MRPC → QNLI → QQP → RTE → SST-2 → WNLI
2	QNLI → QQP → CoLA → MRPC → RTE → SST-2 → WNLI
3	RTE → SST-2 → WNLI → QNLI → QQP → CoLA → MRPC

QNLI) so far we observed. After the last task training as a current model, we report the evaluation results based on all the tasks that we have seen so far. For the continual learning setting, we focus on evaluating the overall performance on classification as same as existing work (Kirkpatrick et al., 2017b), so we not use STS-B (regression) under this setting. The forgetting metric (Chaudhry et al., 2019) fgt for a given task is measured by the difference between results of the validation metrics (e.g., accuracy) when the task is first validated and last validated.

4.3 Results

We perform single task experiments on 9 scenarios of the GLUE benchmark as shown in Table 1. From the experimental results with BERT-base model, we outperform BERT-base with Adam, EWC, MAS, SI, and RecAdam models on 9 out of 9 scenarios of the GLUE benchmark. From the experimental results on ALBERT-xxlarge model, we also outperform ALBERT-xxlarge with Adam, EWC, MAS, SI, and RecAdam on 9 out of 9 scenarios of the GLUE benchmark. In both cases, we achieve the best average acc, mcc and corr compared to the other 5 models. For the result under the continual learning setting, we try different task sequence to evaluate the performance of our work with EWC, MAS, SI, and RecAdam, the description of task sequences is in Table 4, We show the result of sequence 1 in Table 2, other results are listed in the Appendix. From the experimental results, we can see our model achieves less forgetting than EWC, MAS, SI, and RecAdam especially for older tasks like CoLA, MRPC, QNLI and QQP. In general, we achieve the best average acc on MRPC, QNLI, QQP, RTE, SST-2, and WNLI, and the best mcc on CoLA. We also achieve the least average forgetting on all the 7 classification tasks. The results of continual learning setting show that our method can achieve the best performance and forget less than other methods, which demonstrate the effectiveness of our method to address the catastrophic forgetting problem in continual learning.

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.

4.4 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 6 scenarios of the GLUE benchmark. The results of ablation study is shown in Table 3. 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 1.2% improvements on average measured by acc on MRPC, 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 4.5% improvements on average measured by acc on MRPC, 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 7.2% improvements on average measured by acc on MRPC, 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 introduce LPC algorithm by integrating the Logits Calibration and Parameter Calibration into a brand-new optimization algorithm based on the well-known Adam optimization algorithm. We do experiments with single task setting on 9 scenarios of GLUE benchmark and achieve state-of-the-art performance. We also do experiments with continual learning setting and achieve the best average accuracy and mcc, and the least forgetting. The limitation of our work is 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*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2018. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019. Efficient lifelong learning with a-gem. *ICLR*.

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.

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*.

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*.

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.

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

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

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30:6467–6476.

Arun Mallya and Svetlana Lazebnik. 2018. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773.

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.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. 2017. Variational continual learning. *arXiv preprint arXiv:1710.10628*.

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. 2017a. icarl: Incremental classifier and representation learning. In *CVPR*, pages 2001–2010.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017b. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

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

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. 2018. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR.

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.

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

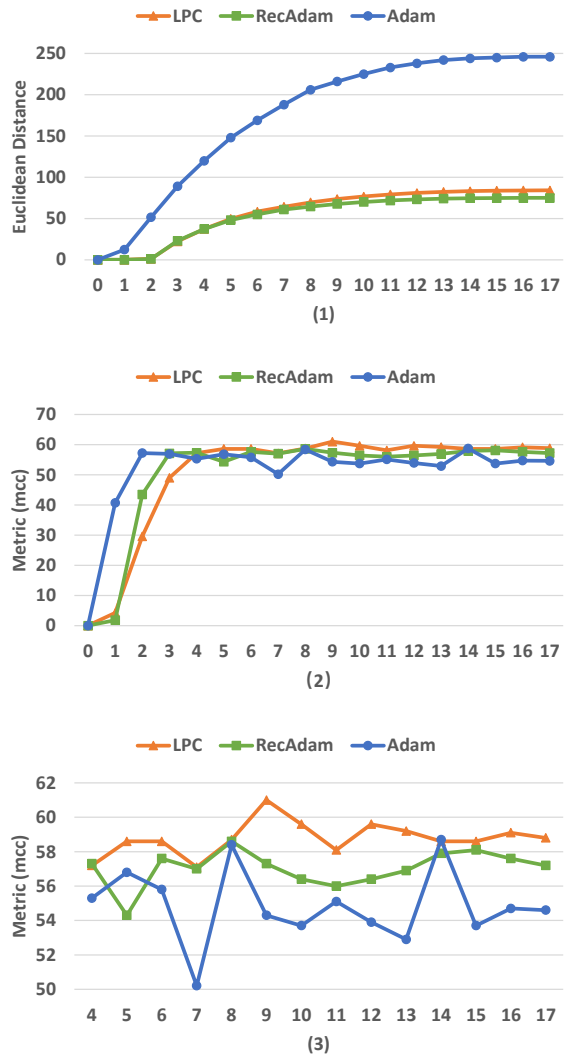

Figure 2: Comparison of Parameter Forgetting and Model Performance with the Epoch Increasing on CoLA Corpus with BERT-base Pre-trained Model.

A.1 Specific Desription of Algorithm 1

From line 9 to line 14, 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 previous model and the current model. From line 10 to line 13, we accumulate the gradients of the squared L2 norm of the learned neural network over the given data inputs to obtain importance weights Ω_{ij} for parameter θ_{ij}. In line 14, 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 16, we compute the gradients of the loss function as a weighted combination of the gradients of L_C and L_P. In line 22, we update the network parameters θ by the gradient descent method.

A.2 Forgetting Analysis

In addition to computing accuracy, we also measure the forgetting by computing the euclidean distance between the parameters of the current model and the previous model on CoLA corpus. Figure 2 shows the comparison of parameter forgetting from the first epoch to the last epoch and the corresponding accuracy with epoch increasing among LPC, RecAdam and Adam. In Figure 2 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 2 chart (3), we can see our model (LPC) achieves the best accuracy compared to RecAdam and Adam all the time after Epoch 4. Figure 2 chart (2) shows results of all models starting from Epoch 0.

A.3 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 3 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.

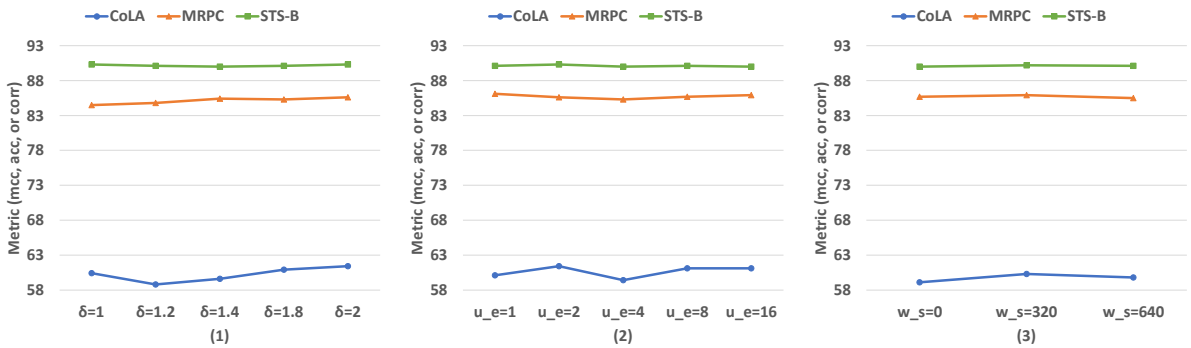

Figure 3: Comparison of Different Hyperparameter Initializations on CoLA, MRPC, and STS-B Corpora with BERT-base Pre-trained Model. The metric for CoLA, MRPC, and STS-B are mcc (Matthew Correlation Coefficient), acc (Accuracy), and corr (Average of Pearson and Spearman Correlation Coefficient), respectively.

Table 5: The Results of Ablation Study on Adam, Logits 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). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	**CoLA** mcc 8.5k / 1k	**MRPC** acc 3.7k / 1.7k	**QNLI** acc 105k / 5.4k	**RTE** acc 2.5k / 3k	**SST-2** acc 67k / 1.8k	**STS-B** corr 7k / 1.4k	**WNLI** acc 634 / 146	**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 + Adam + 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**

In Figure 3 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 on CoLA and STS-B corpora while increases on the MRPC corpus. 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 3 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 on CoLA and STS-B corpora while decreases on the MRPC corpus. When u_e increases from 2 to 4, the performance decreases in a large extent especially on the CoLA corpus. However, when u_e increases from 4 to 8, the model performance increases again. When u_e increases from 8 to 16, there is no obvious difference on the performance.

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

A.4 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 ablation study with ALBERT-xxlarge pre-trained model is shown in Table 5. 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% improvements 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,

Table 6: Experimental Results on Sequentially Emerged Classification Tasks (The order of emergence: QNLI, QQP, CoLA, MRPC, RTE, SST-2, and WNLI). The results are the validation results on all the 7 classification tasks using the model sequentially trained on all the 7 classification tasks. All of results are the medians over 10 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). All other metrics are acc (Accuracy). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	**QNLI** acc / fgt 105k / 5.4k	**QQP** acc / fgt 364k / 391k	**CoLA** mcc / fgt 8.5k / 1k	**MRPC** acc / fgt 3.7k / 1.7k	**RTE** acc / fgt 2.5k / 3k	**SST-2** acc / fgt 67k / 1.8k	**WNLI** acc / fgt 634 / 146	**Avg** acc / fgt	**Avg** mcc / fgt
BERT-base + EWC (rerun) $_{Median}$	65.4 / 25.4	60.4 / 30.0	5.9 / 38.7	36.4 / 46.3	57.0 / 5.1	66.5 / 24.6	26.6 / 0.0	52.1 / 21.9	5.9 / 38.7
BERT-base + MAS (rerun) $_{Median}$	56.2 / 34.0	57.5 / **12.2**	6.3 / **3.3**	63.2 / **3.8**	47.7 / 5.7	55.5 / 1.8	20.8 / 0.0	50.2 / **9.6**	6.3 / **3.3**
BERT-base + SI (rerun) $_{Median}$	78.1 / 12.8	52.2 / 31.9	3.5 / 43.8	40.3 / 38.8	57.4 / 9.4	83.3 / 3.3	26.8 / 0.0	56.4 / 16.0	3.5 / 43.8
BERT-base + RecAdam (rerun) $_{Median}$	49.1 / 42.2	47.0 / 28.5	0.6 / 14.6	**54.7** / 11.4	53.1 / 2.3	49.0 / 31.7	21.1 / 0.0	45.7 / 19.4	0.6 / 14.6
BERT-base + LPC $_{Median}$	**86.8 / 4.1**	**63.7** / 22.7	**15.7** / 25.3	40.9 / 37.7	**60.3 / 2.2**	**83.5 / 1.7**	**35.2** / 0.0	**61.7** / 11.4	**15.7** / 25.3

Table 7: Experimental Results on Sequentially Emerged Classification Tasks (The order of emergence: RTE, SST-2, WNLI, QNLI, QQP, CoLA, and MRPC). The results are the validation results on all the 7 classification tasks using the model sequentially trained on all the 7 classification tasks. All of results are the medians over 10 runs. The metric for CoLA is mcc (Matthew Correlation Coefficient). All other metrics are acc (Accuracy). The train-test split of the datasets is shown as (# train samples / # test samples) in the third row.

Model	**RTE** acc / fgt 2.5k / 3k	**SST-2** acc / fgt 67k / 1.8k	**WNLI** acc / fgt 634 / 146	**QNLI** acc / fgt 105k / 5.4k	**QQP** acc / fgt 364k / 391k	**CoLA** mcc / fgt 8.5k / 1k	**MRPC** acc / fgt 3.7k / 1.7k	**Avg** acc / fgt	**Avg** mcc / fgt
BERT-base + EWC (rerun) $_{Median}$	50.9 / 18.8	46.8 / 43.0	43.7 / 7.0	42.7 / 47.4	**75.8** / 15.0	41.4 / 5.6	79.5 / 0.0	56.6 / 21.9	41.4 / 5.6
BERT-base + MAS (rerun) $_{Median}$	48.7 / 16.3	**75.1 / 13.1**	38.0 / 5.0	56.9 / 17.9	44.6 / 29.6	6.1 / 3.1	68.5 / 0.0	55.3 / 13.7	6.1 / 3.1
BERT-base + SI (rerun) $_{Median}$	42.6 / 25.9	52.0 / 40.2	47.9 / 2.8	27.9 / 61.1	67.8 / 14.7	32.2 / 6.2	**83.0** / 0.0	53.5 / 24.1	32.2 / 6.2
BERT-base + RecAdam (rerun) $_{Median}$	51.6 / 18.4	50.8 / 29.5	49.3 / 7.0	51.4 / **9.7**	44.4 / 31.1	-6.3 / 18.0	64.9 / 0.0	52.1 / 16.0	-6.3 / 18.0
BERT-base + LPC $_{Median}$	**56.7 / 5.4**	67.4 / 23.4	**57.7 / 1.4**	**66.4** / 11.0	70.9 / **7.9**	**46.9 / 1.2**	79.9 / 0.0	**66.5 / 8.2**	**46.9 / 1.2**

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.

A.5 Experimental Results on Sequentially Emerged Classification Tasks with Different Orders of Emergence

For continual learning setting, in addition to the original order of emergence, we also do experiments on sequentially emerged classification tasks with different orders of emergence. The results are shown in Table 6 and Table 7. In both cases, even with different orders of emergence, LPC achieves the best average accuracy and mcc. The results demonstrate the superiority and robustness of LPC on continual learning setting.

A.6 Hyper-parameters for the Rerun of Baselines

For all the baselines, we use standard hyper-parameters. For example, for BERT-base model, the learning rate of all the baseline models is 2e-5. For ALBERT-xxlarge model, the learning rate of all the baseline models is 1e-5. The max sequence length of all the models is 128. For RecAdam, the annealing k is 0.1, the pretrain coefficient is 5000. For EWC, MAS, and SI, the regularization coefficient is 150.