RETROSPECTION: LEVERAGING THE PAST FOR EFFICIENT TRAINING OF DEEP NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep neural networks are powerful learning machines that have enabled breakthroughs in several domains. In this work, we introduce retrospection loss to improve performance of neural networks by utilizing prior experiences during training. Minimizing the retrospection loss pushes the parameter state at the current training step towards the optimal parameter state while pulling it away from the parameter state at a previous training step. We conduct extensive experiments to show that the proposed retrospection loss results in improved performance across multiple tasks, input types and network architectures.

1 INTRODUCTION

Large deep neural networks have enabled breakthroughs in fields such as computer vision (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), natural language understanding (Mikolov et al., 2013) and reinforcement learning (Mnih et al., 2015). Hence, in recent times, significant effort has been directed towards enhancing network efficiency through data augmentation, regularization methods and novel training strategies (Zhong et al., 2017) (Zhang et al., 2017), (Huang et al., 2017) (Noh et al., 2017), (Wang et al., 2018) (Han et al., 2016). In this work, we introduce a technique to improve performance by utilizing prior experiences of the network during training.

Humans are efficient learners with the ability to quickly understand and process diverse ideas. A hallmark of human intelligence is the capability to internalize these complex ideas by actively referencing past interpretations to continually adapt understanding. Our artificial agents should be able to do the same, learning and adapting quickly. This kind of fast and flexible learning is challenging, since the agent must effectively integrate its prior experience with a small amount of new information, while avoiding overfitting to the new data.

The idea for the retrospection loss is simple - to ensure that the predictions at a training step are more similar to the ground truth than to the predictions from a previous training step. As training proceeds, minimizing the loss constrains the network parameters to continually evolve towards the optimal state by successive constriction of the outputs into tighter spaces around the goal. The proposed retrospection loss is easy to implement and we empirically show that it works well across multiple tasks, input types and network architectures.

2 RELATED WORK

Retrospection leverages the parameter state from a previous training step as guidance to compute the direction of current gradient update. One could find similarities with efforts in optimization, that utilize information from past training steps for future weight updates as well as methods that leverage guidance from other parameter states during training.

Techniques such as SVRG (Johnson & Zhang, 2013), SARAH(Nguyen et al., 2017), ProxSARAH (Pham et al., 2019) use gradients from earlier training steps to predict better weight updates. Other optimization methods like Momentum (Sutskever et al., 2013), Adam (Kingma & Ba, 2014) Nesterov Momentum (Jin et al., 2018) accumulate past gradients to accelerate weight updates in the right direction in order to achieve faster convergence. In contrast, our work introduces an additional training objective to guide convergence, and can be used to improve performance when used with different optimizer configurations, as shown in our results.

In reinforcement learning (RL) where techniques involve optimizing using moving (evolving) targets, methods for Q-learning and policy gradients benefit from using a guidance network during training. The DQN algorithm proposed by (Mnih et al., 2015) uses an additional target network (same as online network) for Q-value updates, where parameters are updated by copying from the online network at discrete steps. Double Q-learning (Hasselt, 2010) learns two Q functions, where each Q-function is updated with a value for the next state from the other Q-function. Policy gradient methods such as TRPO (Schulman et al., 2015), PPO (Schulman et al., 2017) use a KL-divergence objective during training that constrains the loss to ensure deviation from a previously learned policy is small. In these techniques, leveraging a guidance during training results in improved convergence and sample efficiency. Note that all these efforts are constrained to the RL setting. Further, the objective in the RL setting is to control divergence from the guidance step to better handle moving targets. On the other hand, the proposed retrospection loss is formalized differently to address the supervised learning setting. To the best of our knowledge, this is the first such effort that uses an idea such as retrospection in supervised learning.

3 METHODOLOGY

We now present the formulation of our retrospective loss. Consider a neural network, $g(\cdot)$, parameterized by its weights θ. Let the optimal parameters of the neural networks at the end of training be given by θ^*. The current parameters of the network at time step T during training are given by θ^T. The objective of the retrospective loss is to leverage the past states during training, and cue the network to be closer to the ground truth than a past state at time step T_p. Given an input data-label pair $(\mathbf{x}_i, y_i)$, the retrospective loss is given by:

$$\mathcal{L}^T_{retrospective} = \kappa * ||g_{\theta^T}(\mathbf{x}_i) - y_i|| - ||g_{\theta^T}(\mathbf{x}_i) - g_{\theta^{T_p}}(\mathbf{x}_i)|| \tag{1}$$

The retrospective loss is designed such that minimizing it with respect to θ over the training steps would constrain the parameter state at each reference step θ^T to be more similar to θ^* than the parameter state from the delayed time step θ^{T_p}. The κ scaling term is required to obtain sufficient gradient signal in later stages of training when $g_{\theta^T}(\mathbf{x}_i)$ is close to y_i, and the first term becomes small.

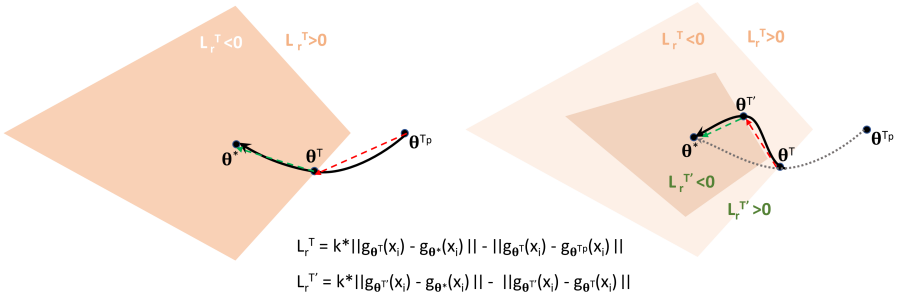

Figure 1: Geometric intuition of the working of the proposed retrospection loss. The figures show polytopes in the weight parameter space. *(Left)* For all θ^i inside the shown colored polytope, the retrospective loss is negative and is positive outside. Our objective is to push parameters of the current θ^T further inside this polygon close to θ^*; *(Right)* In a future time step $T' > T$, by design of the retrospective loss, the polytope region shrinks and our objective at this time step is to push parameters to a near-optimal region around θ^*.

Adding this loss term to an existing supervised learning task loss provides for efficient training, as shown in our experiments later. The retrospective loss is introduced to the training objective following a warm-up period wherein the neural network function can be considered stable for use of retrospective updates. The training objective at any training step T with the retrospective loss is hence defined as:

$$\mathcal{L} = \begin{cases} \mathcal{L}_{task} & T < \mathcal{I}_W \\ \mathcal{L}_{task} + \mathcal{L}^T_{retrospective} & T \geq \mathcal{I}_W \end{cases} \tag{2}$$

where $\mathcal{L}_{task}$ is the task-specific training objective and $\mathcal{I}_W$ is the number of warm-up iterations. We simply use $T_p = F * \lfloor T/F \rfloor$ as the time step for retrospection in this work, and show gains in effi-

ciency of training. One could however mine for T_p intelligently to further improve the performance. We use the L_1-norm as the norm of choice in our implementation for the retrospective loss (Eqn 1). We analyze these choices as part of our ablation studies in Section 5.

Geometric Intuition. Figure 1 illustrates the geometric intuition of the working of the retrospective loss. By design (Eqn 1), $\mathcal{L}_{retrospective}$ is negative when the current parameter state is farther away from the retrospective step, T_p, than the optimal solution (which is the desirable objective). One could view the loss term as dividing the parameter space into two regions: a polytope around the optimal θ^* where $\mathcal{L}_{retrospective} < 0$, and the region outside the polytope where $\mathcal{L}_{retrospective} > 0$. Minimizing retrospective loss pushes the network towards parameters further inside the polytope, thus helping speed up the training process. As shown on the right subfigure in Figure 1, the polytope shrinks over time, since the retrospective support, T_p, is also updated to more recent parameter states. This helps further push the parameters into a near-optimal region around θ^*. The loss term helps in improved solution in most cases, and faster training in certain cases, as shown in our extensive empirical studies in Section 4. Algorithm 1 summarizes the methodology.

Algorithm 1 Retrospective Training

Input: Training Set V, Current Model Parameters θ^T, Previous State Model Parameters θ^{T_p}, Update Frequency F, # of Warm-Up Iterations $\mathcal{I}_W$,
for Step 1 to n **do**
  $grad_{task} \leftarrow 0$ (Initialising the gradients w.r.t task-specific loss)
  $grad_{retrospective} \leftarrow 0$ (Initialising the gradients w.r.t retrospective loss)
  Training Data of minibatch size B pairs of $(X(i),\ Y(i))$.
  $L(\theta^T, X(i), Y(i)) = L_{task}(\theta^T(X(i)), Y(i))$
  $grad_{task} \leftarrow \nabla(L(\theta^T, X(i), Y(i))$
  if $Step > \mathcal{I}_W$ **then**
    $L(\theta^T, \theta^{T_p}, X(i), Y(i)) = L_{retrospective}(\theta^T(X(i)), \theta^{T_p}(X(i)), Y(i))$
    $grad_{retrospective} \leftarrow \nabla(L(\theta^T, \theta^{T_p}, X(i), Y(i))$
  end if
  if Step % F == 0 **then**
    $\theta^{T_p} \leftarrow \theta^T$
  end if
  $\theta^T \leftarrow \theta^T - \eta * (grad_{task} + grad_{retrospective})$
end for

Connection with Triplet Loss. The triplet loss ((Chechik et al., 2010; Schroff et al., 2015; Hoffer & Ailon, 2015)) has been proposed and used extensively over the last few years to learn high-quality data embeddings, by considering a triplet of data points, $\mathbf{x}_a$ (anchor point), $\mathbf{x}_p$ (point from the positive/same class as the sample under consideration), and $\mathbf{x}_n$ (point from the negative class/class different from the sample under consideration). The loss is then defined as:

$$\max\left(\|g_a - g_p\|^2 - \|g_a - g_n\|^2 + m, 0\right) \tag{3}$$

where g is the neural network model, and m is a minimum desired margin of separation. The triplet loss, inspired by contrastive loss (Hadsell et al., 2006), attempts to learn parameters θ of a neural network in such a way that data points belonging to the same class are pulled together closer than a data point from another class. One could view the proposed retrospection loss as a triplet loss in the parameter space. While the traditional triplet loss consider a triplet of data samples, we consider a triplet of parameters, θ^T, θ^*, and θ^{T_p}. We however believe that retrospection captures the proposed loss better, since we consider previous parameter states in time.

Connection with Momentum. Viewing retrospection from the perspective of previous gradients in the training trajectory, one can connect it to the use of momentum, although more in a contrasting sense. The use of momentum and variants such as Nesterov momentum (Jin et al., 2018) in training neural networks use the past gradient, say at θ^{T-1} or the gradient over the previous few steps, at $\{\theta^{T-q}, \cdots, \theta^{T-1}\}, q > 0$), while updating the parameters in the current step. This assumes local consistency of the direction of the gradient update in the training trajectory, and that one can use these previous directions to get a more robust estimate of the gradient step to be taken currently. In contrast, retrospection leverages the same idea from the opposite perspective, viz., consistency of the direction of the gradient update is *only* local, and hence the parameter state, θ^{T_p} farther away from the current state θ^T, provides a cue of what the next parameter must be far from. This raises interesting discussions, and the possibility of analyzing retrospection as a thrust obtained from an undesirable parameter state, as opposed to momentum. We leave these as interesting directions of future work, and focus this work on proposing the method, and showing its effectiveness in training neural networks.

4 EXPERIMENTS AND RESULTS

We conduct experiments using retrospection on the following tasks: image classification (Sec 4.1), image generation (Sec 4.2), speech recognition (Sec 4.3), text classification (Sec 4.4) and few-shot image classification (Sec 4.5). During experimentation, the original (without retrospection) and retrospective (with retrospection) configurations are trained using same weight initialization, to ensure consistency of comparison. For all experiments, L_1-norm is the default choice of norm for retrospection. When retrospection is used without warm-up, the guidance parameters, θ^{T_p}, are initialized at random.

4.1 IMAGE CLASSIFICATION

We perform image classification experiments using Fashion-MNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) and CIFAR-10 (Krizhevsky, 2009) datasets. The retrospection loss, for classification, uses activations of the softmax layer. The default hyperparameter configurations for retrospection include a warm-up period of zero epochs and a retrospective update frequency of fifty steps. The parameter, K, is initialized at 4 and increased by 2% at each retrospective update. Quantitative results for image classification are compiled in Table 1.

Fashion-MNIST. For experiments on Fashion MNIST, we use LeNet (Lecun et al., 2001) and ResNet-20 (He et al., 2016) architectures. Models in each experiment are trained to convergence using the SGD optimizer (lr=0.1, momentum=0.5, mini-batch=32) running over 70,000 steps. Results in Figure 2 (a)-(b) show that using the retrospective loss results in improved training.

SVHN. For experiments on SVHN, we use VGG-11 (Simonyan & Zisserman, 2014) and ResNet-18 (He et al., 2016) architectures. Models in each experiment are trained to convergence using the SGD optimizer (lr=0.001, momentum=0.9, mini-batch=100) running over 200,000 steps. Results in Figure 2 (c)-(d) show that using the retrospective loss results in more efficient training.

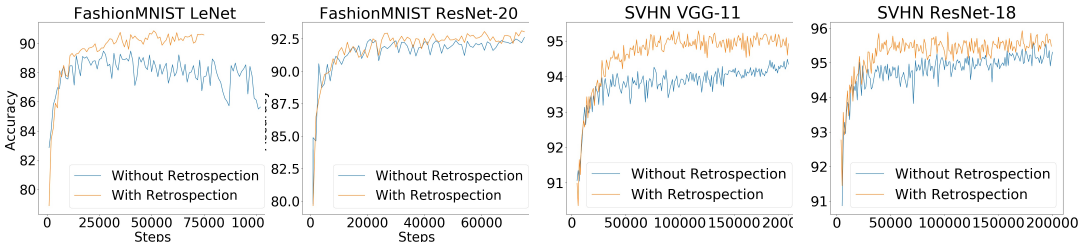

Figure 2: Classification performance using retrospection on F-MNIST and SVHN datasets

CIFAR-10. For experiments on CIFAR-10 (Krizhevsky, 2009), we use larger variants of ResNet including ResNet - 44, 56, 110 (He et al., 2016). Models in each experiment are trained for 200 epochs, using the training configuration (mini-batch, lr policy) detailed in (He et al., 2016). Here, we observe that using the retrospection loss in later stages of training results in best improvement in performance. Correspondingly, the retrospective loss is introduced after a warm-up of 150 epochs and the retrospective update frequency is one epoch. The parameter, K, is initialized at 4 and updated by 2% once every ten retrospective updates. Quantitative performance is reported in Table 1. For sake of completion, we also mention (in brackets) the error rates for the corresponding experiments as reported by authors in the original work (He et al., 2016).

Dataset	Model	Original	Retrospective
F-MNIST	LeNet	10.8	**9.4**
	ResNet-20	7.6	**6.8**
SVHN	VGG-11	5.54	**4.70**
	ResNet-18	4.42	**4.06**
CIFAR-10	ResNet-44	6.98 (7.17)	**6.55**
	ResNet-56	6.86 (6.97)	**6.52**
	ResNet-110	6.55 (6.61)	**6.29**

Table 1: Classification error using retrospection on F-MNIST, SVHN and CIFAR-10 dataset

4.2 IMAGE GENERATION

Next, we perform experiment with Generative Adversarial Networks (GAN) using Fashion-MNIST (F-MNIST) (Xiao et al., 2017) and CIFAR-10 (Krizhevsky, 2009) datasets. Our study considers

both unconditional (DCGAN, LSGAN) and conditional (ACGAN) variants of GANs. We adapt implementations from (dcg) for LSGAN (Mao et al., 2016), DCGAN (Radford et al., 2015) and from (acg) for ACGAN (Odena et al., 2016). For our experiments, we train the generator and discriminator for 100 epochs, with initial learning rate of 0.0002 on minibatches of size 64 using Adam optimizer. We report performance using Inception Score (Salimans et al., 2016), a standard metric for evaluating GANs. The inception score is calculated using implementation in (inc, 2018) with predictions for CIFAR-10 generated using network in (Szegedy et al., 2015) and features for F-MNIST using network in (Krizhevsky et al., 2012).

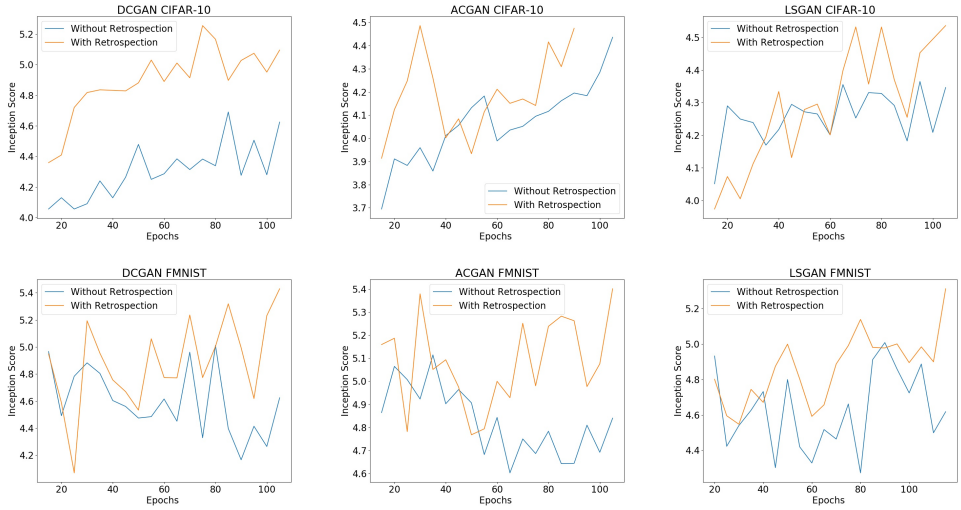

Figure 3: Inception Scores using retrospection on CIFAR-10 (Krizhevsky, 2009)(row 1) and F-MNIST (Xiao et al., 2017) (row 2) datasets using DCGAN (Radford et al., 2015) (col 1), ACGAN (Odena et al., 2016)(col 2), LSGAN (Mao et al., 2016)(col 3).

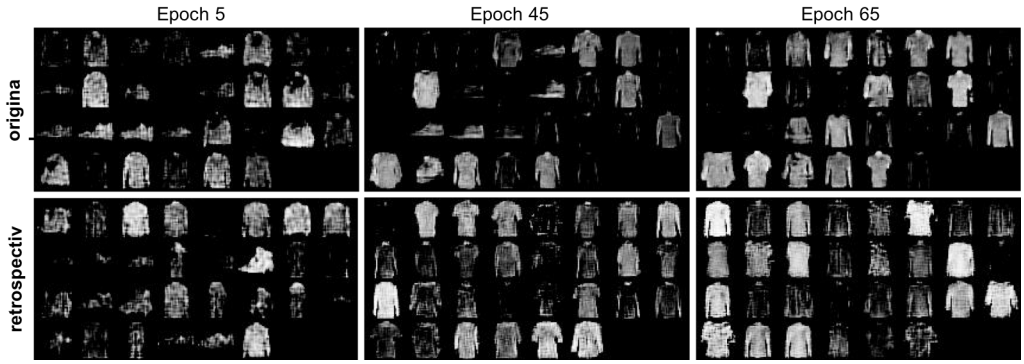

Figure 4: Images generated over training epochs when ACGAN (Odena et al., 2016) trained on FMNIST dataset: (a) without retrospection (row 1) (b) with retrospection (row 2)

For all experiments, the retrospection loss is initialized without any warm-up period (zero epochs). The loss is computed on outputs of the discriminator and is used to train the generator model. For DCGAN (Radford et al., 2015) and LSGAN (Mao et al., 2016) L_2-norm as choice of norm. The retrospective update happens six times in one epoch. The scaling parameter, K is initialized at 4 and is not changed during training. For ACGAN (Odena et al., 2016), which is conditional, the retrospective loss consists of both adversarial loss and class loss components. L_1-norm is used for class component and L_2-norm is used for adversarial component. Figure 3 presents comparative inception score plots when the various dataset-network pairs are trained with (without) the retrospection

loss. Additionally, Figure 4 presents images generated over epochs when training ACGAN (Odena et al., 2016), with and without retrospection, on F-MNIST (Xiao et al., 2017).

4.3 SPEECH RECOGNITION

We perform speech recognition experiments using the Google Commands (Warden, 2017) dataset. The dataset consists of 65,000 utterances, where each utterance is about one-second long and belongs to one out of 30 classes. The classes correspond to voice commands such as yes, no, down, left, as pronounced by a few thousand different speakers. We follow (Zhang et al., 2017) to preprocess the utterances where we first extract normalized spectrograms from the original waveforms at a sampling rate of 16 kHz and subsequently we zero-pad the spectrograms to equalize their sizes at 160 x 101.

For this experiment, we compare LeNet(Lecun et al., 2001) and VGG-11(Simonyan & Zisserman, 2014) architecture, each of which is composed of two convolutional and two fully-connected layers. We train each model for 30 epochs with minibatches of 100 examples, using Adam as the optimizer. Training starts with a learning rate of $3\text{x}10^{-}3$ and is divided by 10 every 10 epochs. The retrospective loss is introduced after a warm-up period of eight epochs, since we find it speeds up initial convergence. The retrospection update frequency is half epoch. The loss scaling margin, K, is initialized at 4, and is increased by 1% at each retrospective update. Results in Table 2 highlight that training using the retrospection loss decreases error rate for both LeNet (Lecun et al., 2001) and VGG-11 (Simonyan & Zisserman, 2014) on both validation and testing sets.

Model	Validation Set		Testing Set	
LeNet	original	9.8	original	10.3
	retrospective	**9.6**	retrospective	**9.9**
VGG-11	original	5.2	original	5.0
	retrospective	**4.4**	retrospective	**4.2**

Table 2: Classification error using retrospection on the Google Commands dataset

4.4 TEXT CLASSIFICATION

We perform text classification experiments on the task of emotion detection in dyadic conversations. We baseline our experiments against DialogueRNN (Majumder et al., 2019), a recent state-of-the-art work, which is composed of an attentive network consisting of three Gated Recurrent Units(GRU). We perform experiments using AVEC (Schuller et al., 2012) and IEMOCAP (Busso et al., 2008) datasets. While the datasets are multi-modal (image and text), following (Majumder et al., 2019), we restrict scope of our experiments to using text. To feed into the network, the text data is preprocessed to obtain n-gram features as detailed in (Majumder et al., 2019). We follow the same train-test split and training configurations as in the original work. Performance comparison is reported against $BiDialogueRNN_+Att$, the best performing variant from the original work.

For experiments on IEMOCAP, models in each experiment are trained for 60 epochs on cross-entropy objective with F1-Score and accuracy as performance metrics. For retrospection, a warm-up of zero epochs is used. On AVEC, models in each experiment are trained for 100 epochs using MSE loss with MSE and pear-score(r) as the performance metrics. Here, introducing the retrospection loss after a warm-up of seventy five epochs produces best performance. For experiments on both IEMOCAP and AVEC, the retrospective update frequency is one epoch. The loss scaling margin, K, is set to 4 at initialization and is updated by 2% at each retrospective update. Experiments are conducted using the official code repository (Co, 2019). Results in Table 3 show that using the retrospection loss when training DialogueRNN improves performance on both IECOMAP and AVEC datasets.

Dataset	IEMOCAP		AVEC	
	Accuracy	F1-Score	MSE	Pear-Score (r)
original	62.66	62.75	0.179	0.318
retrospective	**64.60**	**64.75**	**0.177**	**0.332**

Table 3: Performance on using retrospection on task of dyadic emotion recognition on DialogueRNN

4.5 FEW-SHOT CLASSIFICATION

We conduct experiments on the task of few shot classification using the CUB-200 (Wah et al., 2011) dataset. The CUB-200 dataset consists of 11,788 images from 200 bird species. In few-shot learning,

the ability of a model is measured by its performance on n-shot, k-way tasks where the model is given a query sample belonging to a new, previously unseen class and a support set, S, consisting of n examples each from k different unseen classes. The model then has to determine which of the support set classes the query sample belongs to. We restrict the scope of our experiments to the 5-way 5-shot setting and baseline against closerlook (Chen et al., 2019), a recent state-of-the-art work, and protonet (Snell et al., 2017) another popular work from the domain. Our experiments follow from (Chen et al., 2019) and implementations use code in (Chen, 2019). We conduct experiments with backbones of varying depths - Conv4, Conv6 and ResNet34, as presented in (Chen et al., 2019).

For our experiments, each model is trained on protonet (Snell et al., 2017) for 400 epochs and on closerlook (Chen et al., 2019) for 200 epochs. For Conv4 and Conv6 configurations on both closerlook and protonet, retrospection is introduced without any warm-up period (zero epochs). For ResNet34, a warm-up period of 280 epochs for protonet and 150 epochs for closerlook is used. For all experiments, the retrospective update frequency is one epoch each. The scaling parameter, K, is initialized at 4 and increased by 2% at each retrospective update. For closerlook, we report comparative performance with baseline++, the best performing variant. Results in Table 4 highlight that training with the retrospective loss results in improved classification accuracy for all backbones configurations on both closerlook and protonet. [1]

Model	protonet		closerlook	
	original	retrospective	original	retrospective
Conv4	75.26 $\pm$1.05	**77.42 $\pm$ 1.25**	79.03 $\pm$0.63	**79.95 $\pm$ 0.75**
Conv6	80.71 $\pm$1.55	**81.78 $\pm$ 1.40**	81.05 $\pm$0.55	**81.35 $\pm$ 0.30**
ResNet34	88.75 $\pm$1.01	**89.99 $\pm$ 1.13**	82.23 $\pm$0.59	**83.11 $\pm$ 0.55**

Table 4: Classification performance using retrospection for few-shot classification on CUB dataset

5 Analysis

In this section, we presents ablation studies to analyse the impact of different hyperparameters - batch size, optimizer, retrospective update frequency (F) and the scaling parameter κ. The studies are conducted on the task of image classification on F-MNIST (Xiao et al., 2017) dataset using LeNet (Lecun et al., 2001) architecture. The default training configurations are used from Section 4.1. In all the studies, networks trained for each configuration are initialized with the same weights to ensure consistent comparison.

Impact of Batch Size We perform experiments to analyse the invariance of the proposed retrospection loss to batch size. For this study, we consider batch sizes - 32, 64, 128. Results presented in Figure 5 highlight that the retrospection loss results in improved training performance, which is achieved much faster, across all the different batch sizes.

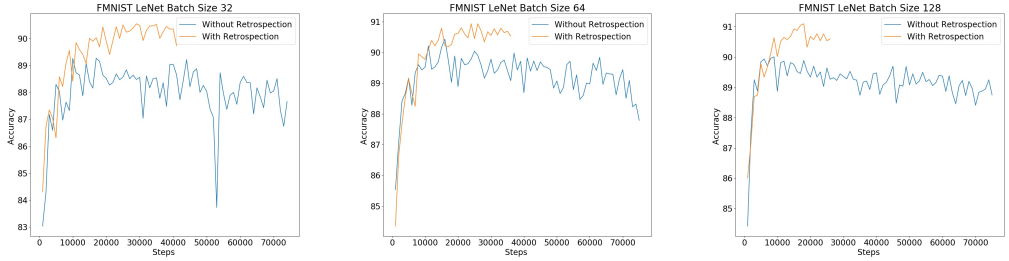

Figure 5: Classification performance using retrospection on LeNet(Lecun et al., 2001) across different batch sizes on FMNIST (Xiao et al., 2017)

[1] Results in some experiments on the original configuration do not match values (are higher or lower) reported in (Chen et al., 2019) even after using official code and same training config. However, we ensure consistency of comparison by using the same initializations for original and retrospective configurations

Impact of Optimizer We perform experiments to analyse the invariance of the proposed retrospection loss to choice of optimizer. For this study, we use Adam (Kingma & Ba, 2014) and SGD optimizers. The classification performance when using Adam and SGD (momentum=0.5) are reported in Figure 6 (Row 2). The observed results highlight that the retrospective loss results in improved training performance across different optimizers.

Choice of Retrospective Update Frequency, F. We study the impact of different update frequencies (F) for the retrospective loss. We experiment with 150, 200, 250 steps. Results are presented in Figure 6 (Row 1) with the best performance achieved using $F = 150$ steps. All configurations of the retrospection loss outperforms the configuration (in blue) trained without it. While experiments in the current work used randomized search to estimate update frequencies, retrospective mining can be an interesting future direction.

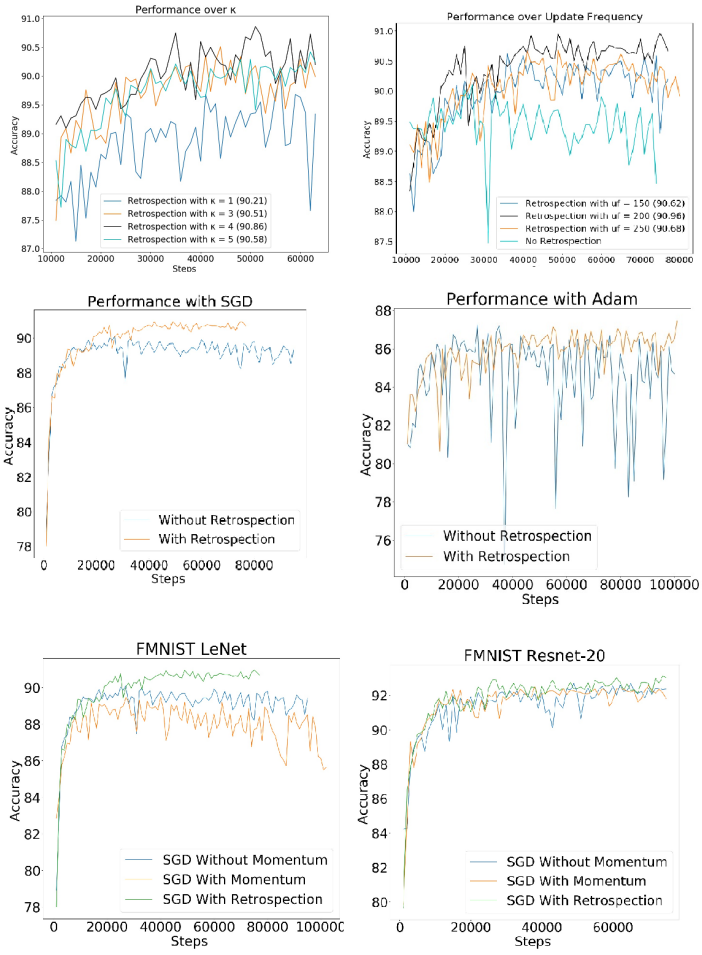

Figure 6: Ablation studies of classification performance on the F-MNIST dataset: *(Row 1)* Varying loss scaling parameter, κ *(left)*, and retrospective update frequency *(right)*; *(Row 2)* Using retrospection on LeNet (Lecun et al., 2001) and SGD vs Adam optimizers; *(Row 3)* Using SGD, SGD +momentum, SGD + retrospection for LeNet, ResNet-20 architectures

Choice of scaling margin, κ We conduct experiments using different initial values of the loss scaling margin, κ. For this analysis, the value of κ remains unchanged during the training. Results are presented in Figure 6 (Row 1) with best performance achieved with $\kappa = 4$. All configurations produce better performance than with $\kappa = 1$.

6 Conclusion and Future Work

In this work, we introduced a retrospective loss that utilizes parameter states from previous training steps to condition weight updates and guide the network towards convergence. We conduct experiments across multiple tasks, input types and architectures to empirically validate the effectiveness of the proposed loss. We perform ablation studies to analyze its behaviour. As an interesting future direction to explore the connection between retrospection and momentum, we conducted preliminary experiments on image classification to evaluate the impact of the retrospective loss on optimization. We contrast performance from three different configurations on image classification: (a) trained without retrospective loss (SGD); (b) trained without retrospective loss (SGD + momentum); and (c) with retrospective loss (SGD). Results in Figure 6 (Row 3) highlight that introducing retrospection improves performance (blue vs green); moreover, using the retrospective loss improves convergence even when SGD is optimized without momentum.

REFERENCES

Acgan-pytorch. URL https://github.com/eriklindernoren/PyTorch-GAN.

Dcgan-pytorch. URL https://github.com/pytorch/examples/tree/master/dcgan.

Inception score code, 2018. URL https://github.com/sbarratt/inception-score-pytorch.

Carlos Busso, Murtaza Bulut, Chi-Chun Lee, Abe Kazemzadeh, Emily Mower, Samuel Kim, Jeannette N Chang, Sungbok Lee, and Shrikanth S Narayanan. Iemocap: Interactive emotional dyadic motion capture database. *Language resources and evaluation*, 42(4):335, 2008.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010.

Wei-Yu Chen. https://github.com/wyharveychen/closerlookfewshot. *URL*, 2019.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *CoRR*, abs/1904.04232, 2019. URL http://arxiv.org/abs/1904.04232.

Sentic-Emotion Co. https://github.com/senticnet/conv-emotion. *URL*, 2019.

Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, Bryan Catanzaro, and William J. Dally. Dsd: Dense-sparse-dense training for deep neural networks. In *ICLR*, 2016.

Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems 23*, pp. 2613–2621. Curran Associates, Inc., 2010. URL http://papers.nips.cc/paper/3964-double-q-learning.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Phuongtrang Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29:82–97, 11 2012. doi: 10.1109/MSP.2012.2205597.

Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pp. 84–92. Springer, 2015.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, and Bo Li. Orthogonal weight normalization: Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *CoRR*, abs/1709.06079, 2017. URL http://arxiv.org/abs/1709.06079.

Chi Jin, Praneeth Netrapalli, and Michael I. Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet (eds.), *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pp. 1042–1085. PMLR, 06–09 Jul 2018. URL http://proceedings.mlr.press/v75/jin18a.html.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. *Gradient-based learning applied to document recognition*, pp. 306–351. IEEE Press, 2001.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander Gelbukh, and Erik Cambria. Dialoguernn: An attentive rnn for emotion detection in conversations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6818–6825, 2019.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks, 2016.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pp. 3111–3119, USA, 2013. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999792.2999959.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Yuval Netzer, Tiejie Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Lam M. Nguyen, Jie Liu, Katya Scheinberg, and Martin Takác. SARAH: A novel method for machine learning problems using stochastic recursive gradient. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pp. 2613–2621, 2017.

Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 5115–5124, USA, 2017. Curran Associates Inc. ISBN 978-1-5108-6096-4. URL http://dl.acm.org/citation.cfm?id=3295222.3295264.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans, 2016.

Nhan H. Pham, Lam M. Nguyen, Dzung T. Phan, and Quoc Tran-Dinh. Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization, 2019.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Björn Schuller, Michel Valster, Florian Eyben, Roddy Cowie, and Maja Pantic. Avec 2012: the continuous audio/visual emotion challenge. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pp. 449–456. ACM, 2012.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pp. 1889–1897. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045319.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 4077–4087, 2017.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–1139–III–1147. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3043064.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, 2011.

Yong Wang, Xiao-Ming Wu, Qimai Li, Jiatao Gu, Wangmeng Xiang, Lei Zhang, and Victor O. K. Li. Large margin few-shot learning. *CoRR*, abs/1807.02872, 2018. URL http://arxiv.org/abs/1807.02872.

Pete Warden. https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html. *Google AI Blog*, 1:URL, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *ArXiv*, abs/1708.04896, 2017.