TỐI ƯU HOÁ VÀ ỨNG DỤNG

Đỗ Văn Nam *, Trần Nhật Nam *, Lê Đình Bảo Long *, Trần Thành Luân*

* University of Information Technology, Ho Chi Minh City, Vietnam
† Vietnam National University, Ho Chi Minh City, Vietnam
Email:{19521866, 19521782, 19521872, 19521810}@gm.uit.edu.vn

***Abstract*—Hiện nay, lĩnh vực công nghệ thông tin đang dần phát triển, đặc biệt là về mảng trí tuệ nhân tạo hay cụ thể là Deeplearning. Ứng dụng của Deeplearning thì ở khắp mọi nơi như là ô tô tự lái, alphago,...Ngoài tầm quan trọng của những kiến trúc Deeplearning thì thuật toán tối ưu (Optimizer) là cơ sở để xây dựng một mô hình neural network với mục đích "học" được các fearture dữ liệu đầu và, từ đó có thể tìm một cặp weights và bias phù hợp để tối ưu hoá mô hình. Nhưng vấn đề là học như thế nào? Cụ thể là weight và bias được tìm như thế nào. Đâu phải chỉ cần random (weight, bias) một số lần hữu hạn và hi vọng ở một bước nào đó để ta có thể nhận biết lời giải. Rõ ràng là không khả thi và lãng phí tài nguyên. Vì thế chúng ta cần phải tìm ra một thuật toán nào đó để cải thiện weight với bias theo từng bước, và đó là lý do khiến các thuật toán Optimizer ra đời.**

***Từ khóa*—**Optimizer, Deeplearning, weight, bias, Trí tuệ nhân tạo.

CONTENTS

I. GIỚI THIỆU

Chọn trình tối ưu hóa được coi là một trong những quyết định thiết kế quan trọng nhất trong học sâu và nó không phải là một việc dễ dàng. Các tài liệu đang phát triển hiện nay liệt kê hàng trăm phương pháp tối ưu hóa. Trong bài báo cáo này, chúng tôi đề xuất ra 3 trong số các thuật toán tối ưu mới trong kết quả của bài báo cáo "Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers". Chúng tôi sẽ tìm hiểu lý thuyết và cách hoạt động của 3 thuật toán: SAM (SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLYIMPROVING GENERALIZATION)[1], ADAMP[2] (Slowing Down the Slowdown for Momentum Optimizers on Scale-invariant Weights), EXPECTIGRAD[3](Fast Stochastic Optimization with Robust Convergence Properties). Từ đó, chúng tôi sẽ tiến hành sử dụng bộ dữ liệu MNIST(Nhận dạng chữ viết tay)[4] kết hợp với mạng tích chập(CNN) của Deeplearning để tiến hành thực nghiệm, sau đó so sánh, đánh giá kết quả. Từ đó đưa ra nhận xét, so sánh hiệu quả của các thuật toán tối ưu với các trường hợp khác nhau.

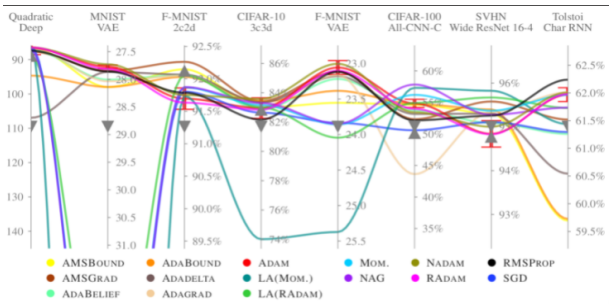

Fig. 1: Kết quả của một số thuật toán tối ưu

Ở các mục tiếp theo sẽ là từng bước trong quá trình mà chúng tôi thực hiện. Các phương pháp tiếp cận sẽ được giới thiệu trong mục II của bài báo cáo này. Mục III sẽ đề cập đến các thực nghiệm và thảo luận kèm theo cách giải quyết bài toán trên. Cuối cùng, là mục IV kết luận và hướng phát triển.

II. PHƯƠNG PHÁP

A. SAM

1) Đặt vấn đề: Trong các mô hình được phổ biến hiện nay, giá trị của tổn thất trong lúc đào tạo làm cho mô hình không đảm bảo về khả năng tổng quát hóa của mô hình. Thật vậy, chỉ tối ưu hóa giá trị tổn thất đào tạo, như thường được thực hiện, có thể dễ dàng dẫn đến chất lượng mô hình dưới mức tối ưu. Được thúc đẩy bởi sự kết nối giữa hình học của cảnh quan mất mát và tổng quát hóa[5]. Mô hình Sharpness-Aware Minimization (SAM) khá mới và hiệu quả để thay thế đồng thời cũng giảm thiểu giá trị tổn thất và độ chính xác. Đặc biệt, mô hình SAM sẽ tìm kiếm các thông số nằm trong các vùng lân cận có mức suy hao thấp đồng đều; công thức này dẫn đến một bài toán tối ưu hóa từ tối thiểu đến tối đa mà trên đó quá trình giảm độ dốc có thể được thực hiện một cách hiệu quả. Chúng tôi đưa ra các kết quả thực nghiệm cho thấy SAM cải thiện khả năng tổng quát hóa mô hình trên bộ dữ liệu MNIST, mang lại hiệu suất khá tốt. Ngoài ra, chúng tôi thấy rằng SAM cung cấp khả năng gắn nhãn nhiễu ngang bằng với khả năng được cung cấp bởi các quy trình hiện đại hướng tới mục tiêu cụ thể đến việc học với nhãn nhiễu.

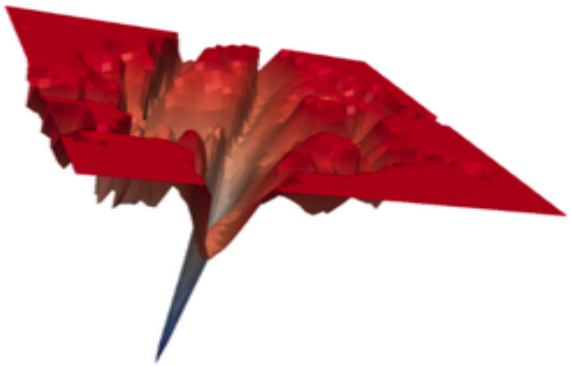

Fig. 2: Example loss landscapes at convergence for ResNet models trained without SAM

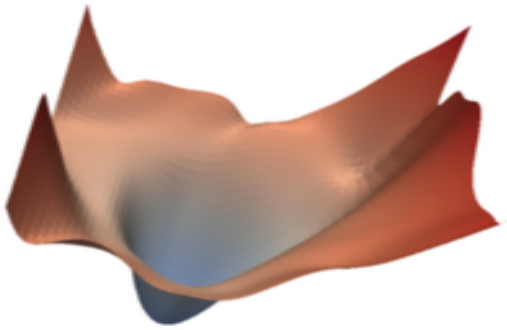

Fig. 3: Example loss landscapes at convergence for ResNet models trained with SAM

2) Công dụng: Sharpness-Aware Minimization (SAM) là một thuật toán tối ưu hóa để giảm thiểu cả sự mất mát và độ sắc nét của sự mất mát. Nó tìm thấy các thông số nằm trong vùng lân cận của tổn thất thấp. Các tác giả nhận thấy rằng điều này cải thiện khả năng tổng quát hóa mô hình.

3) Cách thức hoạt động: Đầu tiên, khai triển Taylor bậc nhất được sử dụng để tìm điểm xung quanh w, điểm này tối đa hóa sự mất mát. Sau đó, bài toán được chuyển thành bài toán quy chuẩn kép cổ điển, có một giải pháp cụ thể e_w. Sau khi đặt e_w trong biểu thức công thức SAM, chúng ta nhận được một bài toán tối ưu hóa tiêu chuẩn (bài toán tối ưu hóa với hàm chi phí L(e_w)) có thể được giải như bình thường bằng cách sử dụng gradient descent. Vì e_w chứa hàm mất mát gốc L gradient, L(e_w) chứa Ma trận Hessian của L. Tính Hessian khi w có hàng trăm triệu thành phần là một nhiệm vụ nặng về tính toán và bộ nhớ. Nhưng may mắn thay, biểu thức này bao gồm phép nhân Hessian trong một vectơ, cho phép tính toán giá trị gradient L(e_w) mà không cần tính Hessian. Cuối cùng, thuật toán có thể được chạy tương tự như Gradient Descent với các công cụ phân biệt tự động, chẳng hạn như TensorFlow hoặc PyTorch.

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_w L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\epsilon}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{w+\hat{\epsilon}(w)}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

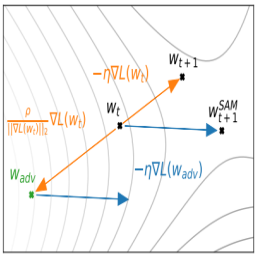

Fig. 4: Cách thức hoạt động của SAM

B. ADAMP

1) Đặt vấn đề: Các kỹ thuật chuẩn hóa được sử dụng rộng rãi trong các mạng sâu dẫn đến bất biến tỷ lệ cho trọng số. Sau khi xem xét cho thấy rằng sự ra đời của momentum trong trình tối ưu hóa gradient-descent (GD), khi được áp dụng trên các tham số bất biến tỷ lệ như vậy, làm giảm tỷ lệ học hiệu quả nhanh chóng hơn nhiều. Hiện tượng này vẫn chưa được nghiên cứu trong tài liệu, mặc dù nó có mặt ở khắp nơi. Chúng tôi nghi ngờ kết quả sự hội tụ sớm có thể đã tạo ra tính tối ưu phụ trong nhiều SGD và các mô hình do Adam đào tạo qua các tác vụ học máy. Đây là vấn đề vì kích thước bước tối ưu hóa hiệu quả tỷ lệ nghịch với định mức trọng số; sự phân rã sớm của kích thước bước hiệu quả có thể dẫn đến hiệu suất mô hình dưới mức tối ưu.

2) Cách thức hoạt động: Chúng tôi xác định hàm Rosenbrock 3D bằng cách thêm trục bán kính r dư thừa, đồng thời coi tọa độ ban đầu (x1, x2) là cực góc (ψ, ϕ) của tọa độ cầu, dẫn đến hàm eh$(r, \psi, \phi) = h(\psi, \phi)$. eh được tối ưu hóa trong không gian 3D với tọa độ Descartes. Trong Hình 3D Scale-Invariant Rosenbrock, chúng tôi so sánh các quỹ đạo cho bộ tối ưu hóa trên tọa độ cầu (ψ, ϕ) Chúng tôi so sánh momentum cơ sở GD và giải pháp dự báo của chúng tôi. Chúng tôi cũng kiểm tra tác động của phân rã trọng số (WD), vì việc lựa chọn cẩn thận WD là một cách khác để điều chỉnh tăng trưởng chuẩn. Chúng tôi quan sát thấy rằng xung lượng GD không hội tụ đủ đến mức tối ưu. Sự chậm lại được giải thích là do kích thước bước hiệu quả giảm trên S 2 do sự gia tăng trong định mức tham số $(r = 1, 0 \to 3, 2)$. Sự điều chỉnh cẩn thận của WD giải quyết một phần sự hội tụ chậm (quỹ đạo xung lượng GD + WD) bằng cách điều hòa tốc độ tăng trưởng chuẩn, nhưng WD

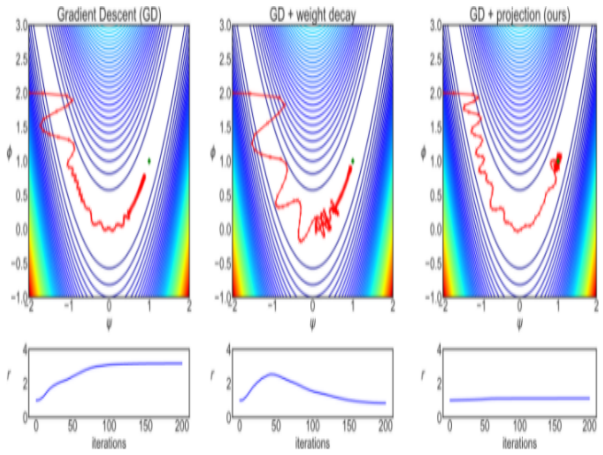

Fig. 5: 3D Scale-Invariant Rosenbrock

vẫn không thể loại trừ sự gia tăng ban đầu trong định mức trọng số($r = 1, 0 \rightarrow 2, 53$) . Trong thực tế, giải quyết vấn đề với WD thậm chí còn kém hấp dẫn hơn vì WD thường là một siêu thông số nhạy cảm (xem các thí nghiệm sau). Mặt khác, giải pháp dự báo của chúng tôi đã thành công trong việc điều chỉnh tăng trưởng định mức trọng số($r = 1, 0 \rightarrow 1, 12$) , đảm bảo kích thước bước hiệu quả không bị hạn chế và hội tụ nhanh hơn đến mức tối ưu.

Algorithm 2: AdamP

Require: Learning rate $\eta > 0$, momentum $0 < \beta_1, \beta_2 < 1$, thresholds $\delta, \varepsilon > 0$.

while $\boldsymbol{w}_t$ not converged **do**
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\nabla_{\boldsymbol{w}} f_t(\boldsymbol{w}_t)$
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)(\nabla_{\boldsymbol{w}} f_t(\boldsymbol{w}_t))^2$
  $\boldsymbol{p}_t \leftarrow \boldsymbol{m}_t / (\sqrt{\boldsymbol{v}_t} + \varepsilon)$
  if $\boldsymbol{w}_t \cdot \nabla_{\boldsymbol{w}} f(\boldsymbol{w}_t) < \delta$ **then**
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\, \Pi_{\boldsymbol{w}_t}(\boldsymbol{p}_t)$
  else
    $\boldsymbol{w}_{t+1} \leftarrow \boldsymbol{w}_t - \eta\, \boldsymbol{p}_t$
  end if
end while

Fig. 6: Cách thức hoạt động của ADAMP

C. EXPECTIGRAD

1) Đặt vấn đề: Nhiều phương pháp gradient thích ứng phổ biến như Adam và RMSProp dựa vào đường trung bình động hàm mũ (EMA) để chuẩn hóa kích thước các bước của chúng. Trong khi EMA làm cho các phương pháp này có độ phản hồi cao với thông tin gradient mới, nghiên cứu gần đây đã chỉ ra rằng nó cũng gây ra sự phân kỳ đối với ít nhất một vấn đề tối ưu hóa lồi. Chúng tôi đề xuất một phương pháp mới có tên là Expectigrad, phương pháp này điều chỉnh các bước theo giá trị trung bình không trọng số của mỗi thành phần của tất cả các gradient lịch sử và tính toán một số hạng xung lượng được hiệu chỉnh chệch cho nhau giữa tử số và mẫu số. Chúng tôi chứng minh rằng Expectigrad không thể phân kỳ đối với mọi trường hợp của bài toán tối ưu hóa được biết là nguyên nhân khiến Adam phân kỳ. Chúng tôi cũng thiết lập một điều đáng tiếc bị ràng buộc trong cài đặt không lồi ngẫu nhiên chung cho thấy Expectigrad ít bị ảnh hưởng bởi phương sai gradient hơn so với các phương pháp hiện có. Thử nghiệm Expectigrad trên một số tác vụ học máy chiều cao, chúng tôi thấy nó thường thực hiện tốt các phương pháp hiện đại với ít điều chỉnh siêu thông số.

2) Thuật toán: Chúng tôi cho rằng cách tiếp cận hạn chế các phương pháp dựa trên EMA như Adam trong nỗ lực buộc hội tụ là phản tác dụng. Làm như vậy không chỉ có nguy cơ ảnh hưởng đến hiệu suất mà còn dẫn đến việc thiếu hiểu biết về lý thuyết. Mặc dù việc hạn chế sự thích ứng của đường EMA có thể ngăn cản sự phân kỳ trong Vấn đề Reddi, nhưng bản thân nó không giải quyết được nguyên nhân gốc rễ của sự phân kỳ. Thay vào đó, chúng tôi quan tâm đến các thuật toán tự động mạnh mẽ đến độ dốc hiếm, có cường độ cao là đặc trưng của Bài toán Reddi. Chúng tôi bắt đầu tìm kiếm của mình bằng cách xem xét quy tắc cập nhật sau đây, trong đó các kích thước được chuẩn hóa bằng trung bình cộng của tất cả các mẫu gradient cho đến bước thời gian tại t. Ở đây, $\alpha > 0$ là tỷ lệ học tập và $\epsilon > 0$ là hằng số dương ngăn phép chia cho 0. Để cải thiện hiệu suất của thuật toán, chúng tôi sẽ thực hiện một số sửa đổi Đầu tiên, hãy lưu ý rằng tóm tắt rõ ràng là không cần thiết. Nếu chúng ta lưu trữ tổng st tại mỗi thời điểm t, thì chúng ta có thể tính 3 Preprint một cách hiệu quả từ tổng trước đó:$(st \leftarrow s(t-1) + gt)$ Thứ hai, hãy quan sát rằng việc chia cho t để tính giá trị trung bình có thể là vấn đề khi các gradient rất thưa thớt thường là trường hợp khi sử dụng kích hoạt Đơn vị tuyến tính chỉnh lưu (ReLU)[6]. Hãy xem xét điều gì sẽ xảy ra khi một thành phần của gradient bằng 0 trong một thời gian rất dài: khi t tăng lên vô điều kiện, tỷ số st/t tiến tới 0. Khi một thành phần khác không đột ngột gặp phải, thuật toán sẽ thực hiện một bước lớn và có khả năng gây thảm hoạ dọc theo thứ nguyên đó. Chúng ta có thể giảm thiểu điều này bằng cách đưa vào một biến phụ nt để đếm số lượng các gradient khác không đã xảy ra. Lưu ý rằng chúng ta có thể cập nhật chính xác $(nt - 1)$ bằng cách chỉ cần thêm dấu phần tử của gradient bình phương.

Algorithm 1 Expectigrad

Select learning rate $\alpha > 0$ ▷ Default: $\alpha = 10^{-3}$
Select momentum constant $\beta \in [0, 1)$ ▷ Default: $\beta = 0.9$
Select denominator constant $\epsilon > 0$ ▷ Default: $\epsilon = 10^{-8}$
Initialize parameters $\mathbf{x}_0 \in \mathbb{R}^d$ arbitrarily
Initialize running sum $\mathbf{s}_0 \leftarrow 0 \cdot \mathbf{x}_0$
Initialize running counter $\mathbf{n}_0 \leftarrow 0 \cdot \mathbf{x}_0$
Initialize momentum $\mathbf{m}_0 \leftarrow 0 \cdot \mathbf{x}_0$
for $t = 1, 2, \ldots, T$ **do**
  Compute gradient estimate: $\mathbf{g}_t \leftarrow \nabla l(\mathbf{x}_{t-1}, \boldsymbol{\xi}_t)$ with $\boldsymbol{\xi}_t \sim \mathbb{P}(\Xi)$
  Update running sum: $\mathbf{s}_t \leftarrow \mathbf{s}_{t-1} + \mathbf{g}_t^2$
  Update running counter: $\mathbf{n}_t \leftarrow \mathbf{n}_{t-1} + \text{sign}(\mathbf{g}_t^2)$
  Update momentum: $\mathbf{m}_t \leftarrow \beta \mathbf{m}_{t-1} + (1-\beta)\frac{\mathbf{g}_t}{\epsilon + \sqrt{\mathbf{s}_t / \mathbf{n}_t}}$ ▷ Define $\frac{0}{0} = 0$
  Update parameters: $\mathbf{x}_t \leftarrow \mathbf{x}_{t-1} - \frac{\alpha}{1-\beta^t}\mathbf{m}_t$
end for
return $\mathbf{x}_T$

Fig. 7: EXPECTIGRAD Algorithm

III. KẾT QUẢ

A. Bộ dữ liệu

Ở đề tài này, chúng tôi sử dụng bộ dữ liệu chữ viết tay - MNIST. Bộ dữ liệu này chứa 60.000 hình ảnh đào tạo và 10.000 hình ảnh thử nghiệm. Một nửa tập huấn luyện và một nửa tập thử nghiệm được lấy từ tập dữ liệu huấn luyện của NIST, trong khi nửa tập huấn luyện còn lại và nửa tập kiểm tra còn lại được lấy từ tập dữ liệu thử nghiệm của NIST.

B. Mô hình

Để tiến hành so sánh, thực nghiệm một cách chính xác, chúng tôi sử dụng chung một kiến trúc Deeplearning - CNN cơ bản để huấn

luyện mô hình.

TABLE I: Mô hình Deeplearning được sử dụng

Layer(type)	Output Shape	Param
dense	None, 784	615440
dense_1	None, 10	7850

- Khởi tạo tham số Learning rate = 0.001.
- Hàm loss được sử dụng là CategoricalCrossentropy().
- Độ đo dùng để đánh giá Accuracy.
- Tiến hành huấn luyện trên tập test với các thông số
+batch_size=128
+epochs=100

C. Đánh giá mô hình

TABLE II: Kết quả độ đo Accuracy epochs=30

	SAM	ADAMP	EXPECTIGRAD
Epochs=10	0.41	0.11	0.89
Epochs=20	0.56	0.20	0.90
Epochs=30	0.63	0.32	0.90
Epochs=40	0.69	0.43	0.90
Epochs=50	0.74	0.50	0.90
Epochs=60	0.78	0.54	0.90
Epochs=70	0.79	0.58	0.90
Epochs=80	0.81	0.62	0.90
Epochs=90	0.81	0.64	0.90
Epochs=100	0.81	0.69	0.90

*1) **SAM**:* Cả loss và accuracy trên tập train và tập val đều có xu hướng là hình vòng cung. Kết quả accuracy thì có xu hướng tiến về 0.8. Kết quả của loss có xu hướng tiến về 0.5.

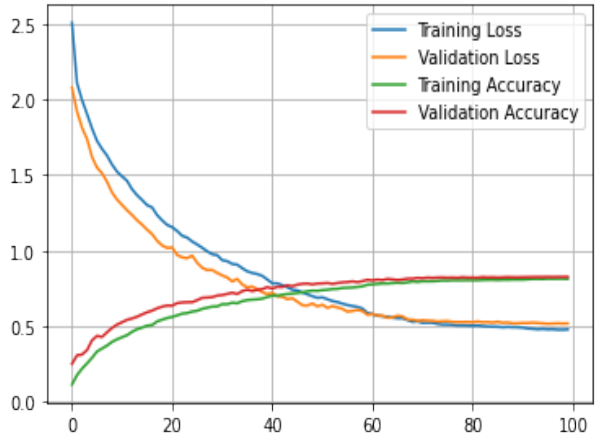

Fig. 8: Đồ thị học sử dụng SAM Optimizer

Lợi ích của SAM ít nhất một phần đến từ các hiệu ứng điều chỉnh của nó, và do đó, nó được kỳ vọng sẽ mang lại lợi nhuận giảm dần khi được kết hợp với các phương pháp chính quy khác như MixUp và làm mịn nhãn. Thuật toán SAM dường như cải thiện một cách có ý nghĩa khả năng tổng quát hóa ngay cả trên các mô hình được coi là đã bao gồm các sơ đồ chính quy được điều chỉnh tốt. Việc hiểu thêm về khả năng kết hợp của SAM sẽ được dành cho quá trình thử nghiệm trong tương lai.

*2) **ADAMP**:* Accuracy trên tập train và tập val có xu hướng đỗ dốc nhanh ở giai đoạn đầu(2-3 epoch đầu tiên) nhưng sau đó bắt đầu tăng dần và tăng khá ổn định. Loss trên tập train và test có xu hướng ổn định(tăng giảm nhẹ).

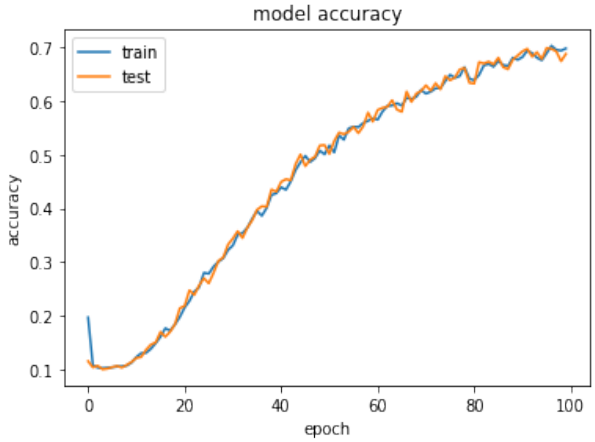

Fig. 9: Accuracy của ADAMP

Fig. 10: Loss của ADAMP

Điều đó có thể thấy rằng thao tác này ngăn chặn việc cập nhật không cần thiết dọc theo hướng xuyên tâm chỉ làm tăng định mức trọng lượng mà không góp phần giảm thiểu tổn thất, đảm bảo kích thước bước hiệu quả không bị hạn chế và hội tụ nhanh hơn đến mức tối ưu.. Và phương pháp của chúng tôi cũng hoạt động trên các bất biến tỷ lệ không bắt nguồn từ việc chuẩn hóa thống kê. Trong tập hợp các thử nghiệm ở trên, chúng tôi cho thấy rằng các sửa đổi được đề xuất là AdamP mang lại hiệu suất nhất quán khá cao. AdamP kết quả đã ngăn chặn thành công sự tăng trưởng định mức trọng lượng và đào tạo một mô hình với tốc độ không bị cản trở.

*3) **EXPECTIGRAD**:* Dựa vào hai biểu đồ bên dưới ta có thể thấy thuật toán này cho kết quả nhanh với hiệu suất rất cao, điều đó dẫn đến việc thuật toán bão hoà nhanh ngay ở các epochs đầu tiên.

Cụ thể ở bộ dữ liệu này thì thuật toán bão hoà ở epochs=20.

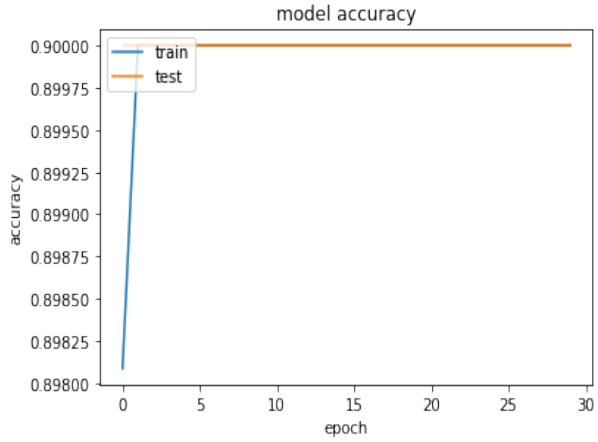

Fig. 11: Accuracy của Expectigrad

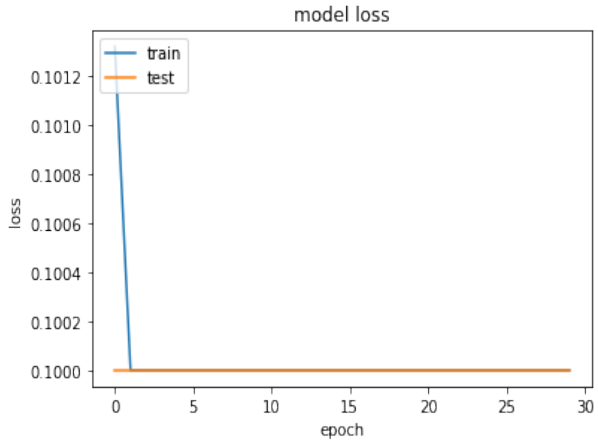

Fig. 12: Loss của Expectigrad

Expectigrad là một phương pháp gradient thích ứng chuẩn hóa các bước bằng một số học, nghĩa là thay vì EMA và cùng tính toán động lượng giữa tử số và mẫu số của nó. Chúng tôi đã chứng minh rằng Expectigrad hội tụ cho các chức năng trơn tru (có thể không lồi) bao gồm Reddi Sự cố trong đó các phương pháp dựa trên EMA như Adam, RMSProp và ADADELTA không thành công. Expectigrad thực hiện một cách cạnh tranh trên nhiều loại kiến trúc mạng nơ-ron kích thước cao, thường làm tốt hơn các phương pháp hiện đại. Các thí nghiệm dịch máy của chúng tôi cho thấy thực tế các vấn đề mà Adam phân kỳ nhưng Expectigrad đạt được hiệu suất rất mạnh, mang lại tiềm năng khẳng định về những dự đoán lý thuyết của chúng tôi. Cuối cùng, hiệu suất nhanh của Expectigrad từ chối bất kỳ quan điểm rằng các phương pháp thích ứng phải dựa vào EMA để phản hồi với thông tin gradient gần đây.

IV. Kết luận

Sau khi tiến hành tìm hiểu và thực nghiệm ba thuật toán tối ưu mới thì chúng tôi nhận ra tầm quan trọng của các thuật toán trong việc tối ưu hoá giá trị tổn thất khi đào tạo, cũng như tìm ra được những ưu, nhược điểm của các thuật toán rồi từ đó ta có thể linh hoạt trong việc áp dụng các thuật toán tối ưu vào những mô hình Deeplearning khác nhau

References

[1] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[2] Byeongho Heo, Sanghyuk Chun, Seong Joon Oh, Dongyoon Han, Sangdoo Yun, Gyuwan Kim, Youngjung Uh, and Jung-Woo Ha. Adamp: Slowing down the slowdown for momentum optimizers on scale-invariant weights. *arXiv preprint arXiv:2006.08217*, 2020.

[3] Brett Daley and Christopher Amato. Expectigrad: Fast stochastic optimization with robust convergence properties. *arXiv preprint arXiv:2010.01356*, 2020.

[4] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[5] Liu Guohua and Wang Shuyu. Methods and applications of arch sam optimization. , page 04, 1994.

[6] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.