Supplementary Material to "NKFAC: A Fast and Stable KFAC Optimizer for Deep Neural Networks"

Ying Sun[1], Hongwei Yong[1], and Lei Zhang[1,2]✉

[1] The Hong Kong Polytechnic University, Hong Hum, Kowloon, Hong Kong, China.
`csysun@comp.polyu.edu.hk, hongwei.yong@polyu.edu.hk, cslzhang@comp.polyu.edu.hk`
[2] OPPO Research Institute, Shenzhen, Guangdong, China

Abstract. In this supplementary file, we first state the detailed AdaNKFAC algorithm. Then we show the running examples about the Newton's iteration and the slow change of Fisher information statistics. Next, we report the related hyper-parameters of different optimizers on CIFAR100/10 and ImageNet in Sections 4.2 and 4.3. Finally, we show the ablation studies results about the stepsize of the hyper-parameters, the Newton's iteration and the implementations to better study the properties of our proposed NKFAC.

1 AdaNKFAC Algorithm

The detailed AdaNKFAC algorithm, which is the combination of AdamW and our proposed NKFAC, is stated below.

Algorithm 1 AdaNKFAC

```
Inputs: $\mathbf{W}_0, \mathbf{L}_0{=}\epsilon\mathbf{I}_{C_{out}}, \mathbf{R}_0{=}\epsilon\mathbf{I}_{C_{in}}, \widehat{\mathbf{L}}_0{=}\mathbf{0}_{C_{out}}, \widehat{\mathbf{R}}_0{=}\mathbf{0}_{C_{in}}, \mathbf{M}_0{=}\mathbf{0}_{C_{out}\times C_{in}}, \mathbf{V}_0{=}\mathbf{0}_{C_{out}\times C_{in}}$;
float $\tau, \eta, \beta_1, \beta_2, \varepsilon$; integers $T_{stat}, T_{inv}, C, K$;
Outputs: $\mathbf{W}_T$.
 1: for $t = 0, 1, \dots, T-1$ do
 2:     compute the gradient $\mathbf{G}_t$;
 3:     save $\mathbf{X}_t{=}[x_{ti}]_{i=1}^n$ and $\mathbf{\Delta}_t{=}[\delta_{ti}]_{i=1}^n$;
 4:     $(\widehat{\mathbf{L}}_{t+1}, \widehat{\mathbf{R}}_{t+1}) =$ Alg.2$(\mathbf{L}_t, \mathbf{R}_t, \widehat{\mathbf{L}}_t, \widehat{\mathbf{R}}_t, \mathbf{X}_t, \mathbf{\Delta}_t, \eta, \alpha, T_{stat}, T_{inv}, C, K)$;
 5:     $\widehat{\mathbf{G}}_{t+1} = \widehat{\mathbf{L}}_{t+1}\mathbf{G}_t\widehat{\mathbf{R}}_{t+1}$;
 6:     $\widetilde{\mathbf{G}}_{t+1} = \widehat{\mathbf{G}}_{t+1}\frac{\|\mathbf{G}_t\|}{\|\widehat{\mathbf{G}}_{t+1}\|}$;
 7:     $\boldsymbol{M}_{t+1} = \beta_1\boldsymbol{M}_t + (1-\beta_1)\widetilde{\mathbf{G}}_{t+1}$;
 8:     $\boldsymbol{V}_{t+1} = \beta_2\boldsymbol{V}_t + (1-\beta_2)\widetilde{\mathbf{G}}_{t+1} \odot \widetilde{\mathbf{G}}_{t+1}$;
 9:     $\widetilde{\boldsymbol{M}}_{t+1} = \frac{\boldsymbol{M}_{t+1}}{1-\beta_1^t}, \widetilde{\boldsymbol{V}}_{t+1} = \frac{\boldsymbol{V}_{t+1}}{1-\beta_2^t}$;
10:     $\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \tau\frac{\widetilde{\boldsymbol{M}}_t}{\sqrt{\widetilde{\boldsymbol{V}}_t+\varepsilon}}$.
11: end for
```

2 Running Examples of Newton's Iteration

Here we give some simple examples to test the performance of the above Newton's iteration using random generated 1024×1024 matrices. We normalize the matrices to have norm 1 and add perturbations with different norm $0.1, 0.05, 0.01$ to them. Figure 1 shows the loss curve of different perturbation cases with respect to iteration. Figure 1 also shows that the fast convergence of Newton's method relies heavily on the request of a good initialized point.

3 Running Examples about the Change of Statistics

Although we have clearly explained in the paper why the Fisher information statistics $\mathbf{L}_t$ and $\mathbf{R}_t$ do not vary much, to have a more intuitive explanation, we draw the changing curves of these two statistics during the training process for ResNet18 on CIFAR100. Here in the Fig. 2, the plot value represents the norm of the difference of the statistics between an interval of T_{stat} (takes 20 here) and is divided by the current statistics norm. We take the average of the results of four experiments to plot the figure. We can see from Fig. 2 that, except for the very beginning stage, all the values are small, which verifies the slow change of our Fisher information statistics.

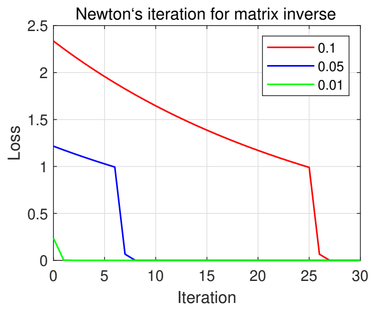

Fig. 1. Loss curves of Algorithm 1.

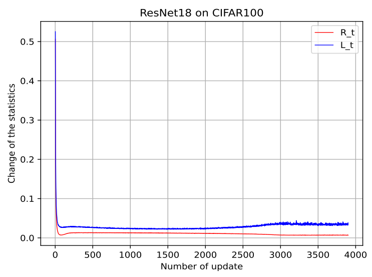

Fig. 2. Change of $\mathbf{L}_t$ and $\mathbf{R}_t$ in training ResNet18 on CIFAR100.

4 Hyper-parameters Settings

In this section, we report the initial learning rate and weight decay of different optimizers on CIFAR100/10 in Table 1 and on ImageNet in Table 2. These hyper-parameters are adopted in the experiments in Sections 4.2 and 4.3.

Table 1. Settings of learning rate (LR) and weight decay (WD) for different optimizers on CIFAR100/10.

Optimizer	SGDM	AdamW	RAdam	Adabelief	KFAC	SKFAC	NKFAC
LR	0.1	0.001	0.001	0.001	0.0005	0.0005	0.05
WD	0.0005	0.5	0.5	0.5	0.1	0.1	0.001

Table 2. Settings of learning rate (LR) and weight decay (WD) for different optimizers on ImageNet.

Optimizer	SGDM	AdamW	RAdam	Adabelief	KFAC	SKFAC	NKFAC
LR	0.1	0.001	0.001	0.001	0.0005	0.0005	0.05
WD	0.0001	0.1	0.1	0.5	0.02	0.02	0.0002

Table 3. Testing accuracy (Acc.) of NKFAC with different dampening parameter η for ResNet50 on CIFAR100.

η	0.1	0.01	0.001	0.0001
Acc. (%)	$81.74 \pm .27$	$\mathbf{81.78} \pm .06$	$80.78 \pm .13$	$79.69 \pm .05$

Table 4. Testing accuracy (Acc.) of NKFAC with different statistics momentum α for ResNet50 on CIFAR100.

α	0.9	0.95	0.995	0.9995
Acc. (%)	$81.82 \pm .15$	$81.78 \pm .06$	$81.84 \pm .12$	$\mathbf{81.95} \pm .16$

5 Ablation Studies On Hyperparameters

In this section, we report some ablation studies on NKFAC with respect to the hyper-parameters on CIFAR100[1]. Our experiments are accomplished on 8 Geforce RTX 2080Ti GPUs on the same server. Besides the parameter we focus on, other hyper-parameters are identical to those in Section 4.2. Here, we mainly focus on the dampening parameter η, statistics momentum α, the number of Newton's iterations K, and the updating intervals T_{stat} and T_{inv}.

Table 3 shows the result with different dampening parameters η. As we mentioned before, the dampening parameter matters a lot in a second-order optimizer, and here $\eta = 0.01$ obtains the best result while $\eta = 0.001$ reduces the performance by 1.00%. Table 4 shows the result with different second-order statistics momentum α. Benefiting from the fact that the Fisher information matrix (FIM) changes little between iterations, NKFAC is stable related to α, and all the momentum parameters α achieve good results. To prevent a greater change of FIM on larger datasets, we finally take a moderate choice $\alpha = 0.95$.

Table 5 reports the testing results with different numbers of Newton's step K, where the time shown is the cost of inversion plus adaptive dampening, and is reported by taking the average. When K takes 2, 3 or 5, the accuracy becomes better, while more steps result in more time cost. As a balance, we finally take $K = 1$. Meanwhile, we report Table 6 about the testing accuracy of different updating interval of the second-order statistics T_{stat} and the inversion T_{inv}. Within the range listed in the table, NKFAC performs stable and satisfactory, so one can choose these intervals according to different needs.

6 Ablation Study on the stepsize of Newton's iteration

Recall that for a given real matrix $\mathbf{A}$, our **Algorithm 1** (i.e., the Newton's method for solving matrix inverse) obtains a sequence $\{\mathbf{B}_k\}$ that performs simply by

$$\mathbf{B}_{k+1} = (1 + \alpha_{k+1})\mathbf{B}_k - \alpha_{k+1}\mathbf{B}_k\mathbf{A}\mathbf{B}_k, \tag{1}$$

with $\alpha_{k+1} = 1$ when $\|\mathbf{I} - \mathbf{A}\mathbf{B}_0\| < 1$ and $\alpha_{k+1} = \frac{1}{\|\mathbf{A}\mathbf{B}_k\|}$ when $\|\mathbf{I} - \mathbf{A}\mathbf{B}_0\| > 1$. When choosing α_{k+1} in the latter case, the most natural idea is to use the

Table 5. Testing accuracy (Acc.) and time cost by computing inversion of NKFAC with different number of Newton's step K.

K	1	2	3	5	10
Acc. (%)	$81.78 \pm .06$	$\mathbf{81.93} \pm .21$	$81.91 \pm .11$	$81.92 \pm .18$	$81.79 \pm .16$
Time (s)	**132.00**	202.37	268.00	405.62	747.88

Table 6. Testing accuracy (Acc.) of NKFAC with different updating intervals T_{stat} and T_{inv} for ResNet50 on CIFAR100.

T_{stat}	20	50	100	200
T_{inv}	200	500	1000	2000
Acc. (%)	$81.78 \pm .06$	$81.74 \pm .18$	$\mathbf{81.90} \pm .26$	$81.80 \pm .13$

Table 7. Detection results of Faster-RCNN on COCO. Δ means the improvement of α_{k+1} compared with α^*_{k+1}.

Backbone, LR	Stepsize	AP	AP$_{.5}$	AP$_{.75}$	AP$_s$	AP$_m$	AP$_l$
ResNet50, 1×	α^*_{k+1}	38.7	59.5	42.0	22.9	42.3	50.3
	α_{k+1}	39.7	60.7	43.0	23.4	43.3	51.5
	Δ	↑1.0	↑1.2	↑1.0	↑0.5	↑1.0	↑1.2
ResNet101, 1×	α^*_{k+1}	39.1	59.5	42.6	22.6	42.9	51.2
	α_{k+1}	41.3	62.0	45.0	24.4	44.7	54.9
	Δ	↑2.2	↑ 2.5	↑2.4	↑1.8	↑1.8	↑3.7

Table 8. Detection and segmentation results of Mask-RCNN on COCO. Δ means the improvement of α_{k+1} compared with α^*_{k+1}.

Backbone, LR	Algorithm	AP^b	AP$^b_{.5}$	AP$^b_{.75}$	AP^m	AP$^m_{.5}$	AP$^m_{.75}$
ResNet50, 1×	α^*_{k+1}	39.5	60.4	43.1	36.3	57.5	38.9
	α_{k+1}	40.1	60.9	43.9	36.6	57.9	39.2
	Δ	↑0.6	↑ 0.5	↑0.8	↑0.3	↑0.4	↑0.3
ResNet101, 1×	α^*_{k+1}	41.4	61.9	45.4	37.4	59.0	40.2
	α_{k+1}	41.8	62.1	45.9	37.8	59.2	40.3
	Δ	↑0.4	↑ 0.2	↑0.5	↑0.4	↑0.2	↑0.1

length that drops the deviation most along this descent direction as the stepsize, specifically, as in the below Proposition 1.

Proposition 1. *Denote the descent direction* $\mathbf{D}_k := \mathbf{B}_k(\mathbf{A}\mathbf{B}_k - \mathbf{I})$ *and the function* $f_k(\alpha) := \frac{1}{2}\|\mathbf{A}(\mathbf{B}_k - \alpha\mathbf{D}_k) - \mathbf{I}\|^2$ *for each iteration* k. *Then the optimization problem* $\alpha^*_{k+1} := \arg\min_\alpha f_k(\alpha)$ *takes the optimal* $\alpha^*_{k+1} = \frac{\langle \mathbf{A}\mathbf{B}_k - \mathbf{I}, \mathbf{A}\mathbf{B}_k(\mathbf{A}\mathbf{B}_k - \mathbf{I})\rangle}{\|\mathbf{A}\mathbf{B}_k(\mathbf{A}\mathbf{B}_k - \mathbf{I})\|^2}$ *in iteration* k.

However, our experiments show that this α^*_{k+1} may not work as good as $\alpha_{k+1} = \frac{1}{\|\mathbf{A}\mathbf{B}_k\|}$, which is chosen from our experience. Here we show some of the comparison results on detection and segmentation tasks in Table 7 and Table 8. In summary, the stepsize chosen in Newton's iteration indeed leave impact on the performance of NKFAC. Luckily, our stepsize $\alpha_{k+1} = \frac{1}{\|\mathbf{A}\mathbf{B}_k\|}$ that chosen in **Algorithm 1** performs good throughout our experiments.

7 Ablation Study on Implementations

With our experiment results in Section 4 in our paper, we are motivated to add our implementations to KFAC and SKFAC, denoted by KFAC* and SKFAC*, respectively. In this section, we compare the implemented KFAC*, SKFAC* with NKFAC on CIFAR100[1] and ImageNet[2] datasets.

Here, the learning rate and weight decay of KFAC*, SKFAC* and NKFAC are set to be the same, specifically, 0.05 and 0.001, respectively. We see NKFAC still shows its advantage of time cost from Table 9 and achieves higher generalization performances than SKFAC*, while stably reducing the inversion cost by 56% and 64%. KFAC* shows its efficiency benefited from our useful implementations and achieves the highest accuracy in ResNet50 while NKFAC is still the best in DenseNet121. Thus, as a balance, NKFAC can be a good choice in applications.

To clearly compare the time cost of inversion and adaptive dampening in each step, we plot Figure 3 to show the time cost proportion of each part in a

Table 9. Testing accuracies (%) and time cost by computing dampening and inversion (s) of different optimizers **with implementations** on CIFAR100.

	Accuracy			Time Cost		
Optimizer	KFAC*	SKFAC*	NKFAC	KFAC*	SKFAC*	NKFAC
DenseNet121	81.04 ± .22	80.66 ± .13	**81.13** ± .16	362.14	331.04	**132.00**
ResNet50	**81.89** ± .21	81.08 ± .18	81.78 ± .06	306.88	191.33	**134.21**

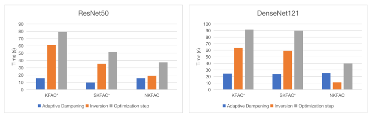

Fig. 3. Time cost of optimization steps for KFAC*, SKFAC* and NKFAC on CIFAR100.

single optimization step using the experiment results on CIFAR100. To eliminate randomness, the time reported in Figure 3 is the cumulative time of the first 50 epochs in the training process. SKFAC can also shorten the time in computing the adaptive dampening after dimension reduction, as shown in Figure 3, while NKFAC shows the least time in the optimization step for both the DNNs tested.

References

1. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
2. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)