OPT2020: 12th Annual Workshop on Optimization for Machine Learning

Catastrophic Fisher Explosion: Early Phase Fisher Matrix Impacts Generalization

Stanisław Jastrzębski STASZEK.JASTRZEBSKI@GMAIL.COM
New York University, USA

Devansh Arpit DEVANSHARPIT@GMAIL.COM
Salesforce Research, USA

Oliver Astrand OLIVER.ASTRAND@GMAIL.COM
New York University, USA

Giancarlo Kerg GIANCARLO.KERG@GMAIL.COM
Université de Montréal, Canada

Huan Wang HUAN.WANG@SALESFORCE.COM

Caiming Xiong CXIONG@SALESFORCE.COM

Richard Socher RSOCHER@SALESFORCE.COM
Salesforce Research, USA

Kyunghyun Cho[*] KYUNGHYUN.CHO@NYU.EDU

Krzysztof Geras[*] K.J.GERAS@NYU.EDU
New York University, USA

Abstract

The early phase of training has been shown to be important in two ways for deep neural networks. First, the degree of regularization in this phase significantly impacts the final generalization. Second, it is accompanied by a rapid change in the local loss curvature influenced by regularization choices. Connecting these two findings, we show that the trace of the Fisher Information Matrix (FIM) plays a crucial role in the implicit regularization effect of SGD. Specifically, smaller values of the trace of FIM during the early phase through implicit or explicit regularization significantly improves generalization. On the other hand, the absence of regularization leads to an increase of the trace of FIM during early phase and degrades generalization. We refer as catastrophic Fisher explosion. Finally, to gain intuition behind the regularization effect of penalizing the trace of FIM, we show that it limits memorization by reducing the learning speed on noisy labels.

1. Introduction

Implicit regularization in gradient-based training of deep neural networks (DNNs) remains relatively poorly understood despite being considered a critical component in their empirical success [15, 21, 28]. Recent work suggests that the early phase of training of DNNs might hold the key to understanding these implicit regularization effects. [1, 8, 17, 23] show that by introducing regularization later, a drop in performance due to lack of regularization in this phase is hard to re-

[*] Equal contribution

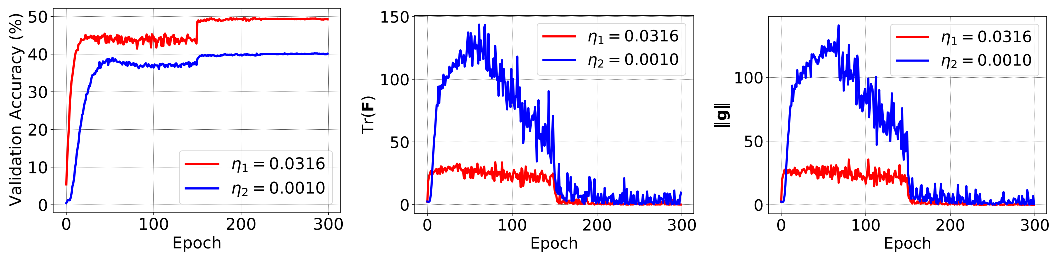

Figure 1: The catastrophic Fisher explosion phenomenon demonstrated for Wide ResNet trained using stochastic gradient descent on the TinyImageNet dataset. Training is done with either a learning rate optimized using grid search ($\eta_1 = 0.0316$, red), or a small learning rate ($\eta_2 = 0.001$, blue). Training with η_2 leads to large overfitting (left) and a sharp increase in $\mathrm{Tr}(\mathbf{F})$ (middle). $\mathrm{Tr}(\mathbf{F})$ is closely related to the gradient norm (right).

cover from, while on the other hand, removing regularization after the early phase has a relatively small effect on the final performance. Other works show that the early phase of training also has a dramatic effect on the trajectory in terms of properties such as the local curvature of the loss surface or the gradient norm [7, 14]. These observations lead to a question: what is the mechanism by which regularization in the early phase impacts the optimization trajectory and generalization? We investigate this question mainly through the lens of the Fisher Information Matrix (FIM), a matrix that can be seen as approximating the local curvature of the loss surface in DNNs [20, 25]. Fisher Information Matrix can be also used to define complexity measures such as the Fisher-Rao norm [16, 19].

Our main contribution is to show that the implicit regularization effect of using a large learning rate or a small batch size can be modeled as an implicit penalization of the trace of the FIM ($\mathrm{Tr}(\mathbf{F})$) from the very beginning of training. We show evidence that explicitly regularizing $\mathrm{Tr}(\mathbf{F})$ (which we call Fisher penalty) significantly improves generalization in scenarios when using a sub-optimal learning rate. On the other hand, growth of $\mathrm{Tr}(\mathbf{F})$ early in training, which may occur in practice when using a relatively small learning rate, coincides with poor generalization. We call this phenomenon the catastrophic Fisher explosion. Figure 1 illustrates this effect on the TinyImageNet dataset [18]. Our second contribution is to provide an intuition behind the regularization effect of $\mathrm{Tr}(\mathbf{F})$. We show that penalizing $\mathrm{Tr}(\mathbf{F})$ discourages memorizing noisy labels.

2. Implicit and explicit regularization of the FIM

Fisher Information Matrix Consider a probabilistic classification model $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$, where $\boldsymbol{\theta}$ denotes its parameters. Let $\ell(\boldsymbol{x}, y; \boldsymbol{\theta})$ be the cross-entropy loss function calculated for input $\boldsymbol{x}$ and label y. Let $g(\boldsymbol{x}, y; \boldsymbol{\theta}) = \frac{\partial}{\partial \theta}\ell(\boldsymbol{x}, y; \boldsymbol{\theta})$ denote the gradient computed for an example $(\boldsymbol{x}, y)$. The central object that we study is the Fisher Information Matrix $\mathbf{F}$ defined as

$$\mathbf{F}(\boldsymbol{\theta}) = \mathbb{E}_{x\sim\mathcal{X},\hat{y}\sim p_\theta(y|\boldsymbol{x})}[g(\boldsymbol{x}, \hat{y})g(\boldsymbol{x}, \hat{y})^T]$$

, where the expectation is often approximated using the empirical distribution $\hat{\mathcal{X}}$ induced by the training set. We denote its trace by $\mathrm{Tr}(\mathbf{F})$. Later, we also look into the Hessian $\mathbf{H}(\boldsymbol{\theta}) = \frac{\partial^2}{\partial \boldsymbol{\theta}^2}\ell(\boldsymbol{x}, y; \boldsymbol{\theta})$. We denote its trace by $\mathrm{Tr}(\mathbf{H})$. The FIM can be seen as an approximation to the Hessian [20]. In

Table 1: Using a 10-30x smaller learning rate (Baseline) results in up to 9% degradation in test accuracy on popular image classification benchmarks (c.f. to optimal η^*). Adding Fisher penalty (FP) substantially improves generalization and closes the gap to η^*. We do not use data augmentation with CIFAR-10 and CIFAR-100 to ensure that using a small learning rate does not lead to under-fitting.

Setting	η^*	Baseline	GP$_x$	GP	FP	GP$_r$
WResNet/TinyImageNet (aug.)	54.67%	52.57%	52.79%	56.44%	**56.73%**	55.41%
DenseNet/C100(w/o aug.)	66.09%	58.51%	62.12%	64.42%	**66.41%**	66.39%
VGG11/C100 (w/o aug.)	45.86%	36.86%	45.26%	47.35%	**49.87%**	48.26%
WResNet/C100 (w/o aug.)	53.96%	46.38%	**58.68%**	57.68%	57.05%	58.15%
SimpleCNN/C10(w/o aug.)	76.94%	71.32%	75.68%	75.73%	79.66%	**79.76%**

particular, as $p(y|\boldsymbol{x};\theta) \rightarrow \hat{p}(y|\boldsymbol{x})$, where $\hat{p}(y|\boldsymbol{x})$ is the empirical label distribution, the FIM converges to the Hessian. [25] showed on image classifications tasks that $\mathrm{Tr}(\mathbf{H}) \approx \mathrm{Tr}(\mathbf{F})$ along the optimization trajectory, which we also evidence in Appendix D.

Fisher Penalty Our main contribution is to propose and investigate a specific mechanism by which using a large learning rate or a small batch size implicitly influences final generalization. Our first insight is to shift the focus from studying the Hessian, to studying properties of the FIM. Concretely, we hypothesize that using a large learning rate or a small batch size improves generalization by implicitly penalizing $\mathrm{Tr}(\mathbf{F})$ from the very beginning of training. In order to study the effect of implicit regularization of $\mathrm{Tr}(\mathbf{F})$, we introduce a regularizer, which we refer to as Fisher penalty, explicitly penalizing $\mathrm{Tr}(\mathbf{F})$. We derive this regularizer in the following way. First, we note that $\mathrm{Tr}(\mathbf{F})$ can be written as $\mathrm{Tr}(\mathbf{F}) = \mathbb{E}_{x\sim\mathcal{X},\hat{y}\sim p_\theta(y|\boldsymbol{x})}\left[\|\frac{\partial}{\partial\theta}\ell(\boldsymbol{x},\hat{y})\|_2^2\right]$.

To regularize $\mathrm{Tr}(\mathbf{F})$, we add the following term to the loss function:

$$\ell'(\boldsymbol{x}_{1:B}, y_{1:B}; \boldsymbol{\theta}) = \frac{1}{B}\sum_{i=1}^{B}\ell(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}) + \alpha \left\| \frac{1}{B}\sum_{i=1}^{B} g(\boldsymbol{x}_i, \hat{y}_i) \right\|^2$$

, where $(\boldsymbol{x}_{1:B}, y_{1:B})$ is a mini-batch, $\hat{y}_i$ is sampled from $p_{\boldsymbol{\theta}}(y|\boldsymbol{x}_i)$, and α is a hyperparameter. We refer to this regularizer as Fisher penalty. The formulation is based on the empirical observation that $\left\|\frac{1}{B}\sum_{i=1}^{B} g(\boldsymbol{x}_i, \hat{y}_i)\right\|^2$ and $\mathrm{Tr}(\mathbf{F})$ correlate well during training. Crucially, this allows us to reduce the added computational cost of Fisher penalty to that of a single additional backpropagation call [6]. Finally, we compute the gradient of the second term only every 10 optimization steps, and in a given iteration use the most recently computed gradient. We discuss these approximations in detail in Appendix C.

3. Fisher Penalty

Experimental setting We run experiments using Wide ResNet [27] (depth 44 and width 3, with or without BN layers), SimpleCNN (without BN layers), DenseNet (L=40, K=12) [12] and VGG-11 [24]. We train these models on either the CIFAR-10 or the CIFAR-100 datasets. Due to larger computational cost, we replace ImageNet with the TinyImageNet dataset [18] in these experiments.

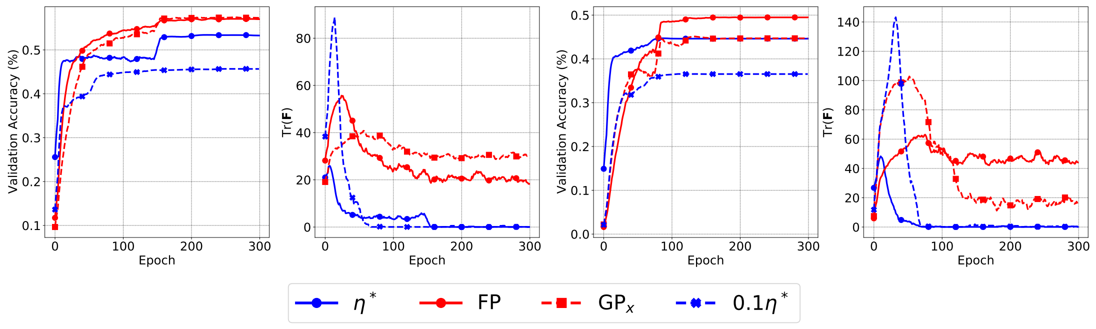

Figure 2: Training with FP or GP$_\text{x}$ improves generalization and limits early peak of $\mathrm{Tr}(\mathbf{F})$. Each subfigure shows validation accuracy (left) and $\mathrm{Tr}(\mathbf{F})$ (right) for training with η^* or a small learning rate (blue) and for training with either GP$_\text{x}$ or FP (red).

To investigate if $\mathrm{Tr}(\mathbf{F})$ and final generalization are related, we apply Fisher penalty in two settings. First, we use a learning rate 10-30x smaller than the optimal one, which both incur up to 9% degradation in test accuracy and results in large value of $\mathrm{Tr}(\mathbf{F})$. We also remove data augmentation from the CIFAR-10 and the CIFAR-100 datasets to ensure that training with small learning rate does not result in underfitting. In the second setting, we add Fisher penalty in training with an optimized learning rate using grid search (η^*) and train with data augmentation.

Fisher penalty penalizes the gradient norm computed using labels sampled from $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$. We hypothesize that a similar, but weaker, effect can be introduced by other gradient norm regularizers. First, we compare FP to penalizing the input gradient norm $\|\boldsymbol{g}_x\| = \frac{\partial}{\partial \boldsymbol{x}}\ell(\boldsymbol{x}, y)$, which we denote by GP$_\text{x}$ [6, 22, 26]. We also note that regularizing GP$_\text{x}$ is related to regularizing the Jacobian input-output of the network [3, 11]. We also experiment with penalizing the vanilla mini-batch gradient [9], which we denote by GP. Finally, we experiment with penalizing the mini-batch gradient computed with random labels $\|\boldsymbol{g_r}\| = \frac{\partial}{\partial \boldsymbol{x}}\ell(\boldsymbol{x}, \hat{y})$ where $\hat{y}$ is sampled from a uniform distribution over the label set (GP$_\text{r}$). We are not aware of any prior work using GP or GP$_\text{r}$ in supervised training, with the exception of [2] where the authors penalized ℓ_1 norm of gradients to compress the network towards the end of training.

We tune the hyperparameters on the validation set. More specifically for α, we test 10 different values spaced uniformly between $10^{-1} \times v$ to $10^1 \times v$ on a logarithmic scale with $v \in \mathbb{R}_+$. For TinyImageNet we test 5 alternatives instead. To pick the optimal learning rate, we evaluate 5 values spaced equally on a logarithmic scale. We include the remaining experimental details in the Appendix E.1.

Fisher Penalty improves generalization Table 1 summarizes the results of the main experiment. First, we observe that a suboptimal learning rate (10-30x lower than the optimal) leads to dramatic overfitting. We observe a degradation of up to 9% in test accuracy, while achieving perfect training accuracy (see Table 5 in the Appendix).

Fisher penalty closes the gap in test accuracy between the small and optimal learning rate, and even achieves better performance than the optimal learning rate. A similar performance was observed when minimizing $\|g_r\|$. We will come back to this observation in the next section.

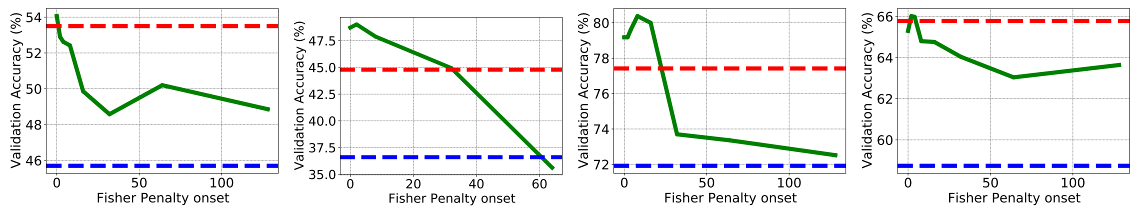

Figure 3: **Left**: Wide ResNet CIFAR-100 (w/o aug.), **Middle left**: VGG-11 CIFAR-100 (w/o aug.), **Middle right**: Simple CNN CIFAR-10 (w/o aug.), **Right**: DenseNet CIFAR-100 (w/o aug.). Each subplot summarizes an experiment in which we apply Fisher Penalty starting from a certain epoch (x axis) and measure the final test accuracy (y axis). Fisher Penalty has to be applied from the beginning of training to close the generalization gap to the optimal learning rate (c.f. the red horizontal line to the blue horizontal line).

GP and GP_x reduce the early value of $\text{Tr}(\mathbf{F})$ (see Table 3 in the Appendix). They, however, generally perform worse than $\text{Tr}(\mathbf{F})$ or GP_r and do not fully close the gap between small and optimal learning rate. We hypothesize they improve generalization by a similar but less direct mechanism than $\text{Tr}(\mathbf{F})$ and GP_r.

In the second experimental setting, we apply FP to a network trained with the optimal learning rate η^*. According to Table 6 (Appendix), FP improves generalization in 4 out of 5 settings. The gap between the baseline and FP is small in 3 out of 5 settings (below 1%), which is natural given that we already regularize training implicitly by using the optimal η and data augmentation.

Geometry and generalization in the early phase of training Here, we investigate the temporal aspect of Fisher Penalty on CIFAR-10 and CIFAR-100. In particular, we study whether early penalization of $\text{Tr}(\mathbf{F})$ matters for final generalization. First, we observe that all gradient-norm regularizers reduce the early value of $\text{Tr}(\mathbf{F})$ closer to $\text{Tr}(\mathbf{F})$ achieved when trained with the optimal learning rate η^*. We show this effect with Wide ResNet and VGG-11 on CIFAR-100 in Figure 2, and for other experimental settings in the Appendix. We also tabulate the maximum achieved values of $\text{Tr}(\mathbf{F})$ over the optimization trajectory in Appendix A.1.

To test the importance of explicitly penalizing $\text{Tr}(\mathbf{F})$ early in training, we start applying it after a certain number of epoch $E \in \{1, 2, 4, 8, 16, 32, 64, 128\}$. We use the best hyperparameter set from the previous experiments. Figure 3 summarizes the results. For both datasets, we observe a consistent pattern. When FP is applied starting from a later epoch, final generalization is significantly worse, and the generalization gap arising from a suboptimal learning rate is not closed.

3.1. Fisher Penalty Reduces Memorization

It is not self-evident how regularizing $\text{Tr}(\mathbf{F})$ influences generalization. In this section, we provide evidence that regularizing $\text{Tr}(\mathbf{F})$ slows down learning on data with noisy labels. We expect FP to reduce memorization. When the predictive distribution $p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$ and the true label distribution $p^*(y|\boldsymbol{x})$ are both uniform, $\text{Tr}(\mathbf{F})$ of the specific example $\boldsymbol{x}$ is equivalent to the squared loss gradient norm of the sample example. The proposed Fisher penalty thus minimizes the contribution of the loss gradient from the training examples whose labels were sampled uniformly. In other words,

Table 2: Fisher Penalty (FP) and GP_r both reduce memorization competitively to mixup. We measure test accuracy at the best validation point in training with either 25% or 50% examples with noisy labels in the CIFAR-100 dataset.

Noise	Setting	Baseline	Mixup	GP_x	FP	GP_r
25%	VGG-11/C100	41.74%	52.31%	45.94%	**60.18%**	58.46%
	ResNet-52/C100	53.30%	**61.61%**	52.70%	58.31%	57.60%
50%	VGG-11/C100	30.05%	39.15%	34.26%	**51.33%**	50.33%
	ResNet-52/C100	43.35%	**51.71%**	42.99%	47.99%	50.08%

the Fisher penalty implicitly suppresses learning noisy examples, under the assumption that clean examples' label distributions are not uniform.

To study whether the above happens in practice, we compare FP to GP_x, GP_r, and mixup [29]. For gradient norm based regularizers, we evaluate 6 different hyperparameter values spaced uniformly on a logarithmic scale, and for mixup we evaluate $\beta \in \{0.2, 0.4, 0.8, 1.6, 3.2, 6.4\}$. We experiment with the Wide ResNet and VGG-11 models. We describe remaining experimental details in the Appendix E.2.

Results We begin by studying the learning dynamics on data with noisy labels through the lens of training accuracy and mini-batch gradient norm. We show the results for VGG-11 and ResNet-50 in Figure 5 and Figure 6 in the Appendix. We observe that FP limits the ability of the model to memorize data more strongly than it limits its ability to learn from clean data. We can further confirm our interpretation of the effect $\mathrm{Tr}(\mathbf{F})$ has on training by studying the gradient norms. As visible in Figure 5, the gradient norm on examples with noisy labels is larger than on clean examples, and the ratio is closer to 1 when large regularization is applied.

We report test accuracy (at the best validation point) in Table 2. We observe that $\mathrm{Tr}(\mathbf{F})$ reduces memorization competitively to mixup. Furthermore, FP performs similarly to GP_r, which agrees with our interpretation of why FP limits learning on examples with noisy labels.

4. Conclusion

We proposed and investigated a hypothesis that SGD influences generalization by implicitly penalizing the trace of the Fisher Information Matrix ($\mathrm{Tr}(\mathbf{F})$) from the very beginning of training. We show that (1) the value of early $\mathrm{Tr}(\mathbf{F})$ correlates with final generalization, and (2) explicitly regularizing $\mathrm{Tr}(\mathbf{F})$ can substantially improve generalization.

To gain further insight into the mechanism by which penalizing $\mathrm{Tr}(\mathbf{F})$ improves generalization, we investigated training on noisy data. We found that penalizing $\mathrm{Tr}(\mathbf{F})$ reduces memorization by penalizing examples with noisy labels more strongly than clean ones.

References

[1] Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2019.

[2] Milad Alizadeh, Arash Behboodi, Mart van Baalen, Christos Louizos, Tijmen Blankevoort, and Max Welling. Gradient ℓ_1 regularization for quantization robustness. In *International Conference on Learning Representations*, 2020.

[3] Alvin Chan, Yi Tay, Yew-Soon Ong, and Jie Fu. Jacobian adversarially regularized networks for robustness. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hke0V1rKPS.

[4] François Chollet and others. *Keras*. GitHub, 2015.

[5] Soham De and Samuel L. Smith. Batch normalization biases deep residual networks towards shallow paths, 2020.

[6] H. Drucker and Y. Le Cun. Improving generalization performance using double backpropagation. *IEEE Transactionsf on Neural Networks*, 1992.

[7] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations*, 2020.

[8] Aditya Sharad Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Advances in Neural Information Processing Systems 32*. 2019.

[9] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30*. 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *CoRR*, abs/1908.02729, 2019. URL http://arxiv.org/abs/1908.02729.

[12] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[13] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 1990.

[14] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho*, and Krzysztof Geras*. The break-even point on optimization trajectories of deep neural networks. In *International Conference on Learning Representations*, 2020.

[15] Yiding Jiang, Behnam Neyshabur, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Fantastic Generalization Measures and Where to Find Them. In *International Conference on Learning Representations*, 2020.

[16] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 1032–1041. PMLR, 2019. URL http://proceedings.mlr.press/v89/karakida19a.html.

[17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *5th International Conference on Learning Representations, ICLR*, 2017.

[18] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. 2015.

[19] Tengyuan Liang, Tomaso A. Poggio, Alexander Rakhlin, and James Stokes. Fisher-rao metric, geometry, and complexity of neural networks. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, volume 89 of *Proceedings of Machine Learning Research*, pages 888–896. PMLR, 2019. URL http://proceedings.mlr.press/v89/liang19a.html.

[20] James Martens. New insights and perspectives on the natural gradient method, 2020.

[21] Behnam Neyshabur. Implicit regularization in deep learning. 2017.

[22] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

[23] Levent Sagun, Utku Evci, V. Ugur Guney, Yann Dauphin, and Leon Bottou. Empirical analysis of the hessian of over-parametrized neural networks, 2018.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[25] Valentin Thomas, Fabian Pedregosa, Bart Merriënboer, Pierre-Antoine Manzagol, Yoshua Bengio, and Nicolas Le Roux. On the interplay between noise and curvature and its effect on optimization and generalization. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

[26] Dániel Varga, Adrián Csiszárik, and Zsolt Zombori. Gradient regularization improves accuracy of discriminative models, 2018.

[27] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

[28] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2016.

[29] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

Table 3: The maximum value of $\mathrm{Tr}(\mathbf{F})$ along the optimization trajectory for experiments on CIFAR-10 or CIFAR-100 included in Table 1.

Setting	η^*	Baseline	$\mathrm{GP_x}$	GP	FP	$\mathrm{GP_r}$
DenseNet/C100 (w/o aug.)	24.68	98.17	83.64	64.33	66.24	73.66
VGG11/C100 (w/o aug.)	50.88	148.19	102.95	58.53	64.93	62.96
WResNet/C100 (w/o aug.)	26.21	91.39	41.43	40.94	56.53	39.31
SCNN/C10 (w/o aug.)	24.21	52.05	47.96	25.03	19.63	25.35

Table 4: Time per epoch (in seconds) for experiments in Table 1.

Setting	η^*	Baseline	$\mathrm{GP_x}$	GP	FP	$\mathrm{GP_r}$
WResNet/TinyImageNet (aug.)	214.45	142.69	233.14	143.78	208.62	371.74
DenseNet/C100 (w/o aug.)	78.88	57.40	77.89	78.66	97.25	75.96
VGG11/C100 (w/o aug.)	30.50	35.27	31.54	32.52	43.41	42.40
WResNet/C100 (w/o aug.)	49.64	47.99	71.33	61.36	76.93	53.25
SCNN/C10 (w/o aug.)	18.64	19.51	26.09	19.91	21.21	20.55

Apppendix

Appendix A. Additional results

A.1. Fisher Penalty

We first show additional metrics for experiments summarized in Table 1. In Table 5 we show the final training accuracy. Table 3 confirms that generally all gradient norm regularizers reduce the maximum value of $\mathrm{Tr}(\mathbf{F})$ (we measure $\mathrm{Tr}(\mathbf{F})$ starting from after one epoch of training because $\mathrm{Tr}(\mathbf{F})$ explodes in networks with batch normalization layers at initialization). Finally, Table 4 confirms that the regularizers incurred a relatively small additional computational cost.

Figure 4 is a counterpart of Figure 2 for the other two models on the CIFAR-10 and the CIFAR-100 datasets.

Table 5: The final epoch training accuracy for experiments shown in Table 1. Experiments with small learning rate reach no lower accuracy than experiments corresponding to a large learning rate η^*.

Setting	η^*	Baseline	$\mathrm{GP_x}$	GP	FP	$\mathrm{GP_r}$
WResNet/TinyImageNet (aug.)	99.84%	99.96%	99.97%	93.84%	81.05%	86.46%
DenseNet/C100 (w/o aug)	99.98%	99.97%	99.96%	99.91%	99.91%	99.39%
VGG11/C100 (w/o aug)	99.98%	99.98%	99.85%	99.62%	97.73%	86.32%
WResNet/C100 (w/o aug)	99.98%	99.98%	99.97%	99.96%	99.99%	99.94%
SCNN/C10 (w/o aug)	100.00%	100.00%	97.79%	100.00%	93.80%	94.64%

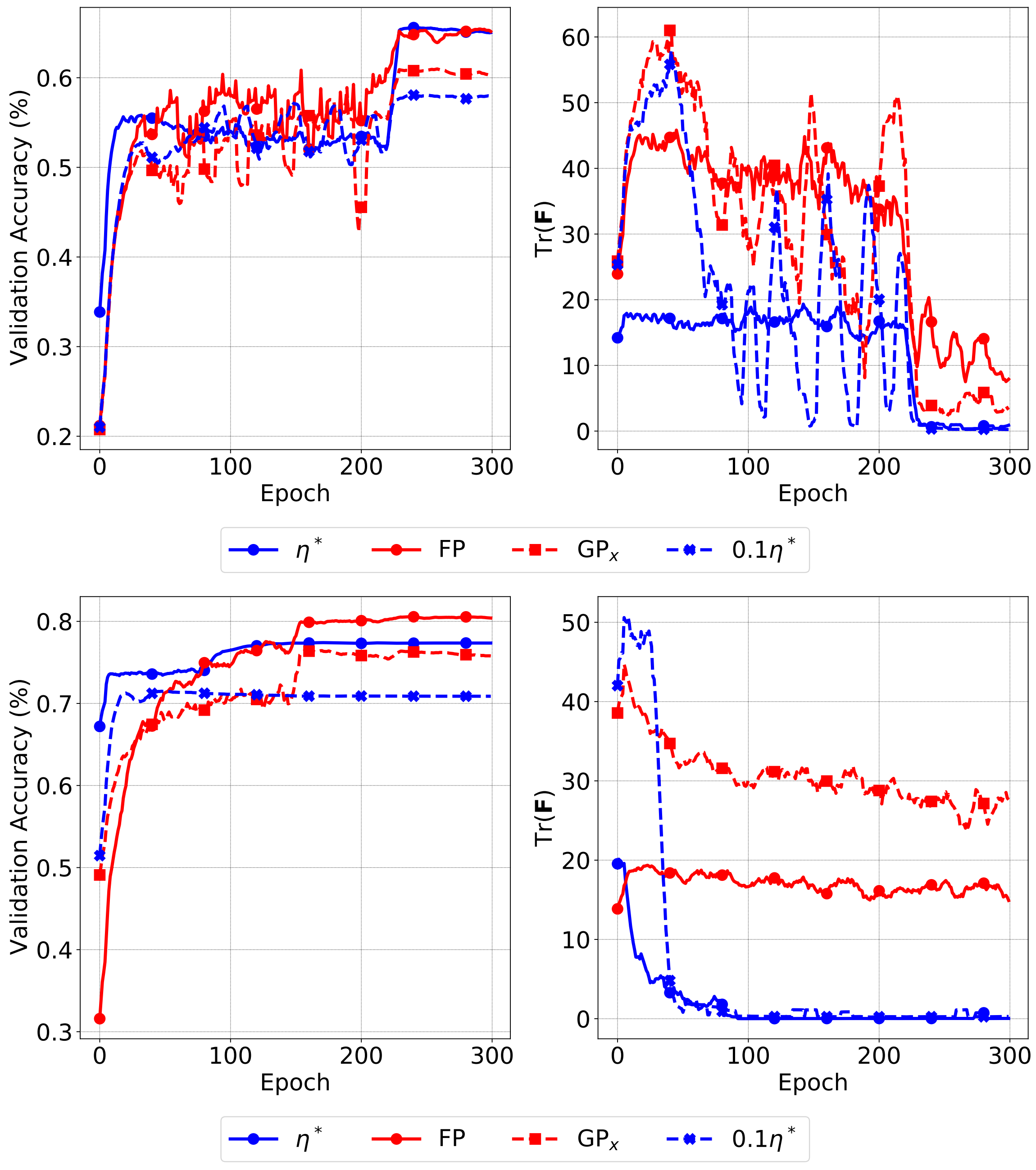

Figure 4: Same as Figure 2, but for DenseNet on CIFAR-100 (top), and SimpleCNN on CIFAR-10 (below). Curves were smoothed for visual clarity.

Table 6: Fisher penalty (FP) improves generalization in 4 out of 5 settings when applied with the optimal learning rate η^* and trained using standard data augmentation. In 3 out of 5 settings the difference between FP and η^* is small (below 1%), which is expected given that FP is aimed at reproducing the regularization effect of large η.

Setting	η^*	FP
DenseNet/C100 (aug.)	**74.41±0.47%**	74.19±0.51%
VGG11/C100 (aug.)	59.82±1.23%	**65.08±0.53%**
WResNet/C100 (aug.)	69.48±0.30%	**71.53±1.22%**
SimpleCNN/C10 (aug.)	87.16±0.16%	**87.52±0.50%**
WResNet/TinyImageNet (aug.)	54.70±0.04%	**60.00±0.07%**

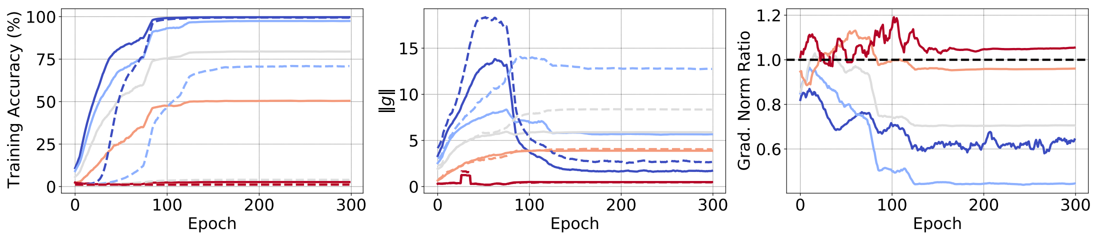

Figure 5: Fisher penalty slows down training on data with noisy labels more strongly than it slows down training on clean data for VGG-11 on CIFAR-100. This likely happens because FP penalizes more strongly gradient norm on data with noisy labels. Left plot shows the training accuracy on examples with clean/noisy labels (solid/dashed line). Middle plot shows the gradient norm evaluated on examples with clean/noisy labels (solid/dashed). Right plot shows the ratio of gradient norm on clean to noisy data. Red to blue color represents the regularization coefficient (from 10^{-2} to 10^{1}).

A.2. Fisher Penalty Reduces Memorization

In this section, we describe additional experimental results for Section 3.1. Figure 6 is the same as Figure 5, but for ResNet-50.

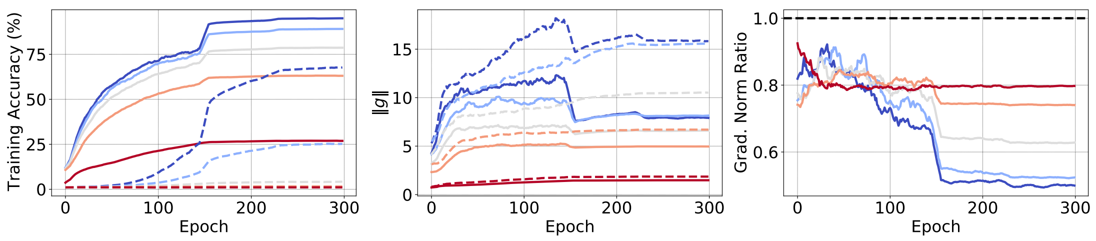

Figure 6: Same as Figure 5, but for ResNet-50.

Appendix B. Computation of $\mathrm{Tr}(\mathbf{H})$

We computed $\mathrm{Tr}(\mathbf{H})$ in our experiments using the Hutchinson's estimator [13],

$$\begin{aligned}
Tr(\mathbf{H}) &= Tr(\mathbf{H}\cdot\mathbf{I}) \\
&= Tr(\mathbf{H}\cdot\mathbb{E}[\mathbf{z}\mathbf{z}^T]) \\
&= \mathbb{E}[Tr(\mathbf{H}\cdot\mathbf{z}\mathbf{z}^T)] \\
&= \mathbb{E}[\mathbf{z}^T\mathbf{H}\cdot\mathbf{z}] \\
&\approx \frac{1}{M}\sum_{i=1}^{M}\mathbf{z}_i^T\mathbf{H}\cdot\mathbf{z}_i \\
&= \frac{1}{M}\sum_{i=1}^{M}\mathbf{z}_i^T\frac{\partial}{\partial\theta}\left(\frac{\partial\ell}{\partial\theta^T}\right)\cdot\mathbf{z}_i \\
&= \frac{1}{M}\sum_{i=1}^{M}\mathbf{z}_i^T\frac{\partial}{\partial\theta}\left(\frac{\partial\ell}{\partial\theta}^T\mathbf{z}_i\right),
\end{aligned}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{z}$ is a multi-variate standard Gaussian random variable, and $\mathbf{z}_i$'s are i.i.d. instances of $\mathbf{z}$. The larger the value of M, the more accurate the approximation is. We used $M = 30$. To make the above computation efficient, note that the gradient $\frac{\partial\ell}{\partial\theta}$ only needs to be computed once and it can be re-used in the summation over the M samples.

Appendix C. Approximations in Fisher penalty

In this section, we describe the approximations made in Fisher penalty in detail. Recall, that $\mathrm{Tr}(\mathbf{F})$ can be expressed as

$$\mathrm{Tr}(\mathbf{F}) = \mathbb{E}_{x\sim\mathcal{X},\hat{y}\sim p_\theta(y|\boldsymbol{x})}\left[\|\frac{\partial}{\partial\theta}\ell(\boldsymbol{x},\hat{y})\|_2^2\right]. \tag{1}$$

In the preliminary experiments, we found empirically that we can use the norm of the expected gradient rather than the expected norm of the gradient, which is a more direct expression of $\mathrm{Tr}(\mathbf{F})$:

$$\begin{aligned}
\nabla\mathbb{E}_{x\sim\mathcal{X},\hat{y}\sim p_\theta(y|\boldsymbol{x})}\left[\left\|\frac{\partial}{\partial\theta}\ell(\boldsymbol{x},\hat{y})\right\|_2^2\right] &\approx \frac{1}{N}\sum_{n=1}^{N}\frac{1}{M}\sum_{m=1}^{M}\nabla\left\|\frac{\partial}{\partial\theta}\ell(\boldsymbol{x}_n,\hat{y}_{nm})\right\|_2^2 \\
&\geq \nabla\left\|\frac{1}{NM}\sum_{n=1}^{N}\sum_{m=1}^{M}\frac{\partial}{\partial\theta}\ell(\boldsymbol{x}_n,\hat{y}_{nm})\right\|_2^2,
\end{aligned}$$

where N and M are the minibatch size and the number of samples from $p_\theta(y|\boldsymbol{x}_n)$, respectively. This greatly improves the computational efficiency. With $N = B$ and $M = 1$, we end up with the following learning objective function:

$$\ell'(\boldsymbol{x}_{1:B}, y_{1:B};\boldsymbol{\theta}) = \frac{1}{B}\sum_{i=1}^{B}\ell(\boldsymbol{x}_i, y_i;\boldsymbol{\theta}) + \alpha\left\|\frac{1}{B}\sum_{i=1}^{B}g(\boldsymbol{x}_i,\hat{y}_i)\right\|^2. \tag{2}$$

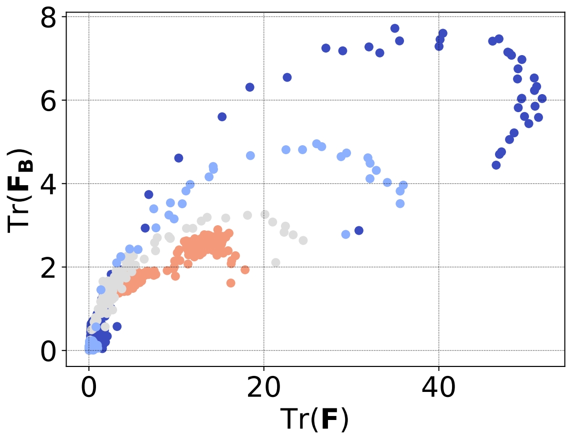

Figure 7: Correlation between $\mathrm{Tr}(\mathbf{F})$ and $\mathrm{Tr}(\mathbf{F}^B)$ for SimpleCNN trained on the CIFAR-10 dataset. Blue to red color denotes learning rates from 10^{-3} to 10^{-1}. The value of $\mathrm{Tr}(\mathbf{F})$ and $\mathrm{Tr}(\mathbf{F}^B)$ correlate strongly for the most of the training trajectory. Using large learning rate reduces both $\mathrm{Tr}(\mathbf{F})$ and $\mathrm{Tr}(\mathbf{F}^B)$.

We found empirically that $\left\|\frac{1}{B}\sum_{i=1}^{B} g(\boldsymbol{x}_i, \hat{y}_i)\right\|^2$, which we denote by $\mathrm{Tr}(\mathbf{F}^B)$, and $\mathrm{Tr}(\mathbf{F})$ correlate well during training. To demonstrate this, we train SimpleCNN on the CIFAR-10 dataset with 5 different learning rates (from 10^{-3} to 10^{-1}). The outcome is shown in Figure 7. We see that for most of the training, with the exception of the final phase, $\mathrm{Tr}(\mathbf{F}^B)$ and $\mathrm{Tr}(\mathbf{F})$ correlate extremely well. Equally importantly, we find that using a large learning affects both $\mathrm{Tr}(\mathbf{F}^B)$ and $\mathrm{Tr}(\mathbf{F})$, which further suggests the two are closely connected.

We also update the gradient of $\mathrm{Tr}(\mathbf{F}^B)$ only every 10 optimization steps. We found empirically it does not affect generalization performance nor the ability to regularize $\mathrm{Tr}(\mathbf{F})$ in our setting. However, we acknowledge that it is plausible that this choice would have to be reconsidered in training with very large learning rates or with larger models.

Figure 8 compares learning curves of training with FP recomputed every optimization step, or every 10 optimization steps. For each, we tune the hyperparameter α, checking 10 values equally spaced between 10^{-2} and 10^{0} on a logarithmic scale. We observe that for the optimal value of α, both validation accuracy and $\mathrm{Tr}(\mathbf{F})$ are similar between the two runs. Both experiments achieve approximately 80% test accuracy.

Appendix D. $\mathrm{Tr}(\mathbf{H})$ and $\mathrm{Tr}(\mathbf{F})$ correlate strongly

We demonstrate a strong correlation between $\mathrm{Tr}(\mathbf{H})$ and $\mathrm{Tr}(\mathbf{F})$ for DenseNet, ResNet-56 and SimpleCNN in Figure 9. We calculate $\mathrm{Tr}(\mathbf{F})$ using a mini-batch. We see that $\mathrm{Tr}(\mathbf{F})$ has a smaller magnitude (because we use the mini-batch gradient which has lower variance), but correlates extremely well with $\mathrm{Tr}(\mathbf{H})$.

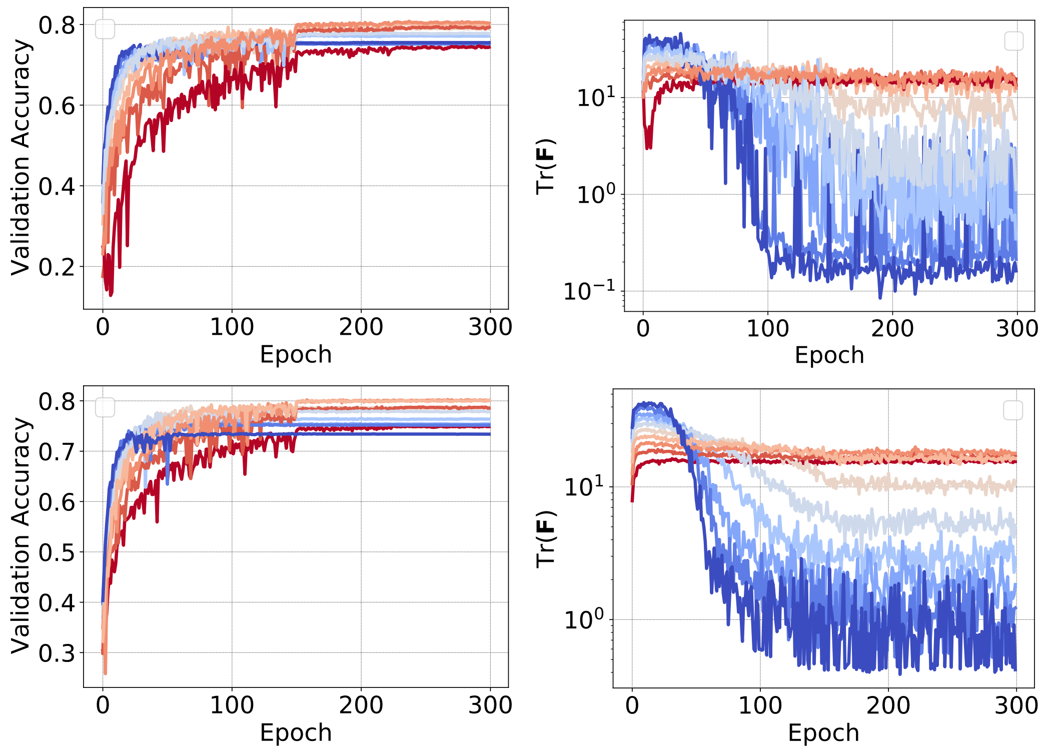

Figure 8: A comparison between the effect of recomputing Fisher penalty gradient every 10 iterations (left) or every iteration (right), with respect to validation accuracy and $\mathrm{Tr}(\mathbf{F})$. We denote by f the frequency with which we update the gradient (top: $f = 10$, bottom: $f = 1$). Both experiments result in approximately 80% test accuracy of the best configuration.

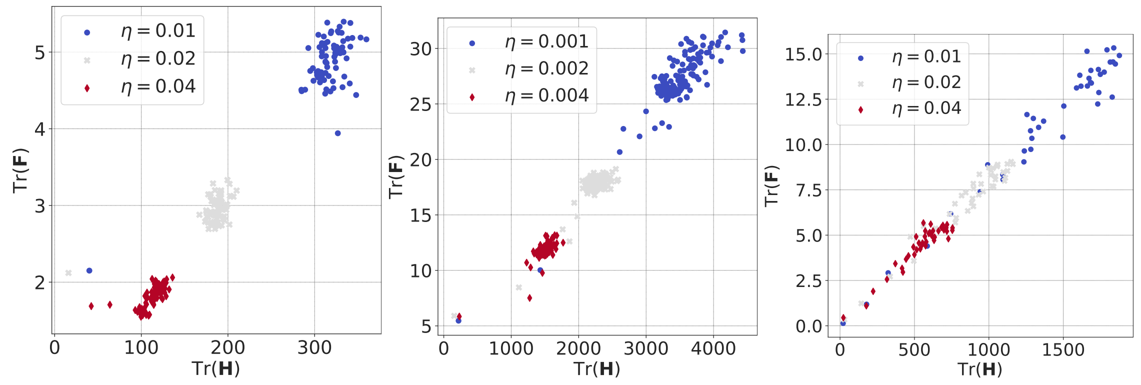

Figure 9: Correlation between $\mathrm{Tr}(\mathbf{F})$ and $\mathrm{Tr}(\mathbf{H})$. **Left**: DenseNet on CIFAR-100, **Middle**: SimpleCNN on CIFAR-10, **Right**: ResNet-56 on CIFAR-100

Appendix E. Additional Experimental Details

E.1. Fisher Penalty

Here, we describe the remaining details for the experiments in Section 3. We first describe how we tune hyperparameters in these experiments. In the remainder of this section, we describe each setting used in detail .

Tuning hyperparameters In all experiments, we refer to the optimal learning rate η^* as the learning rate optimized using grid search. In most experiments we check 5 different learning rate values uniformly spaced on a logarithmic scale, usually between 10^{-2} and 10^{0}. In some experiments we adapt the range to ensure that the range includes the optimal learning rate. We tune the learning rate only once for each configuration (i.e. we do not repeat it for different random seeds).

In the first setting, for most experiments involving gradient norm regularizers, we use $10\times$ smaller learning rate than η^*. For TinyImageNet, we use $30\times$ smaller learning rate than η^*. To pick the regularization coefficient α, we evaluate 10 different values uniformly spaced on a logarithmic scale between $10^{-1} \times v$ to $10^{1} \times v$ with $v \in \mathbb{R}_{+}$. We choose the best performing α according to best validation accuracy. We pick the value of v manually with the aim that the optimal α is included in this range. We generally found that $v = 0.01$ works well for GP, GP_r, and FP. For GP_x we found in some experiments that it is necessary to pick larger values of v.

Measuring $\text{Tr}(\mathbf{F})$ We measure $\text{Tr}(\mathbf{F})$ using the number of examples equal to the batch size used in training. For experiments with Batch Normalization layers, we use Batch Normalization in evaluation mode due to the practical reason that computing $\text{Tr}(\mathbf{F})$ uses batch size of 1, and hence $\text{Tr}(\mathbf{F})$ is not defined for a network with Batch Normalization layers in training mode.

DenseNet on the CIFAR-100 dataset We use the DenseNet (L=40, k=12) configuration from [12]. We largely follow the experimental setting in [12]. We use the standard data augmentation (where noted) and data normalization for CIFAR-100. We hold out random 5000 examples as the validation set. We train the model using SGD with momentum of 0.9, a batch size of 128, and weight decay of 0.0001. Following [12], we train for 300 epochs and decay the learning rate by a factor of 0.1 after epochs 150 and 225. To reduce variance, in testing we update Batch Normalization statistics using 100 batches from the training set.

Wide ResNet on the CIFAR-100 dataset We train Wide ResNet (depth 44 and width 3, without Batch Normalization layers). We largely follow experimental setting in [10].We use the standard data augmentation and data normalization for CIFAR-100. We hold out random 5000 examples as the validation set. We train the model using SGD with momentum of 0.9, a batch size of 128, weight decay of 0.0010. Following [10], we train for 300 epochs and decay the learning rate by a factor of 0.1 after epochs 150 and 225. We remove Batch Normalization layers. To ensure stable training we use the SkipInit initialization [5].

VGG-11 on the CIFAR-100 dataset We adapt the VGG-11 model [24] to CIFAR-100. We do not use dropout nor Batch Normalization layers. We hold out random 5000 examples as the validation set. We use the standard data augmentation (where noted) and data normalization for CIFAR-100. We train the model using SGD with momentum of 0.9, a batch size of 128, and weight decay of 0.0001. We train the model for 300 epochs, and decay the learning rate by a factor of 0.1 after every 40 epochs starting from epoch 80.

SimpleCNN on the CIFAR-10 dataset We also run experiments on the CNN example architecture from the Keras example repository [4][1], which we change slightly. Specifically, we remove dropout and reduce the size of the final fully-connected layer to 128. We train it for 300 epochs and decay the learning rate by a factor of 0.1 after the epochs 150 and 225. We train the model using SGD with momentum of 0.9, a batch size of 128.

Wide ResNet on the TinyImageNet dataset We train Wide ResNet (depth 44 and width 3, with Batch Normalization layers) on TinyImageNet Le and Yang [18]. TinyImageNet consists of subset of 100,000 examples from ImageNet that we downsized to 32×32 pixels. We train the model using SGD with momentum of 0.9, a batch size of 128, and weight decay of 0.0001. We train for 300 epochs and decay the learning rate by a factor of 0.1 after epochs 150 and 225. We train the model using SGD with momentum of 0.9, a batch size of 128. We do not use validation in TinyImageNet due to its larger size. To reduce variance, in testing we update Batch Normalization statistics using 100 batches from the training set.

E.2. Fisher Penalty Reduces Memorization

Here, we describe additional experimental details for Section 3.1. We use two configurations described in Section E.1: VGG-11 trained on CIFAR-100 dataset, and Wide ResNe trained on the CIFAR-100 dataset. We tune the regularization coefficient α in the range $\{0.01, 0.1, 0.31, 10\}$, with the exception of GP_x for which we use the range $\{10, 30, 100, 300, 1000\}$. We tuned mixup coefficient in the range $\{0.4, 0.8, 1.6, 3.2, 6.4\}$. We removed weight decay in these experiments.

1. Accessible at https://github.com/keras-team/keras/blob/master/examples/cifar10_cnn.py.