Some SGD-based Bayesian ensembling approaches

Apostolos Psaros

1 Xie et al. (2013): Horizontal and vertical ensemble with deep representation for classification

One of the ensembling techniques they propose is horizontal voting; they obtain the weights at the end of each epoch and use them to make predictions. The weights collected can be seen as slices of the training trajectory. An ensemble is obtained by averaging these predictions; in a sense they use this average to stabilize their prediction because of the validation set error oscillating. They mention that this oscillation is due to the small number of training data. The first and last epochs (i.e., the range of training epochs) they use for collecting weights are hyperparameters. They seem to use a constant learning rate More ensembling techniques are reviewed in Huang et al. (2017).

2 Smith (2017): Cyclical learning rates for training neural networks

A minimum and maximum bound is set and the learning rate cyclically varies between them. They note that the function according to which the rate changes does not affect significantly the performance and thus select a triangular window as shown in Fig. 1. The reason for doing it is to avoid tuning the learning rate and reduce the number of training epochs; i.e., for optimization purposes.

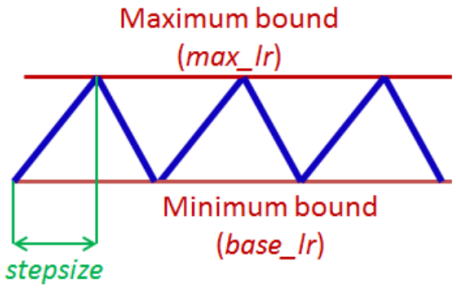

Fig. 1. Figure 2 from Smith (2017): Triangular learning rate policy. The blue lines represent learning rate values changing between bounds. The input parameter stepsize is the number of iterations in half a cycle.

After experimenting they propose to select the stepsize as $2-10$ times the number of iterations in an epoch. For selecting the bounds the propose to run the optimization algorithm for a few epochs with linearly increasing learning rate. By doing so the lower bound can be selected as the rate at which we see an increase to the accuracy (I guess based on the validation set) and the upper bound when it starts being ragged or decreasing. See Fig. 2 for an example; based on this they select the lower bound as 0.001 and the upper bound as 0.006.

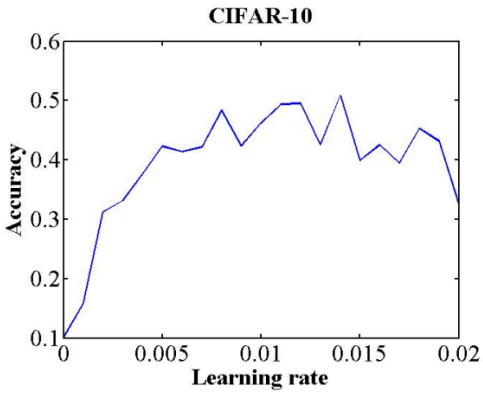

Fig. 2. Figure 3 from Smith (2017): Classification accuracy as a function of increasing learning rate for 8 epochs.

Finally, they note that cyclic learning rate can be combined with adaptive learning rates (e.g., AdaGrad, RMSProp, etc)

3 Loshchilov and Hutter (2016): SGDR: Stochastic gradient descent with warm restarts

First, they review how warm restarts have been used in gradient-free and gradient-based optimization. In this regard, for stochastic gradient descent they simulate a warm restart by increasing the learning rate while the last weight is used as the initial solution. They use a cosine annealing schedule that has a range of learning rates and the number of epochs until the next restart as hyperparameters. It seems that the difference with the cyclical schedule of Smith (2017) is that with cosine annealing there is also a jump in the learning rate for simulating a restart. See also Gotmare et al. (2018) for an empirical investigation of the type of optima obtained with SGDR and Brownlee (2019) for an implementation.

The purpose of this work is primarily better convergence rate for optimization and better overall performance. However, during the time of this work Huang et al. (2017) published their snapshot ensembling results. Thus, Loshchilov and Hutter (2016) used the weights before the last M restarts and corroborated that the corresponding "free" ensemble leads to diverse predictions and improve performance. They left warm restarts for AdaDelta and Adam algorithms for future work.

4 Huang et al. (2017): Snapshot ensembles: Train 1, get M for free

The purpose of this paper is ensembling without incurring any additional costs as compared to standard training. Instead of training M different networks they attempt to let the SGD algorithm converge to M different local optima. After converging to a local minumum they increase the learning rate abruptly, letting the algorithm escape the current local minimum and then decrease the rate for convergence at a different local minimum. Before every restart they take a snapshot. The learning rate schedule is very similar to that of Loshchilov and Hutter (2016) and is changed at each iteration (not epoch). Note that a larger restart learning rate leads to a stronger perturbation of the weights and possibly to increased model diversity. The budget (e.g., 300 epochs) is divided into M cycles (e.g., $M = 6$) and a snapshot is collected at the end of each cycle. See Fig. 3 for an illustration.

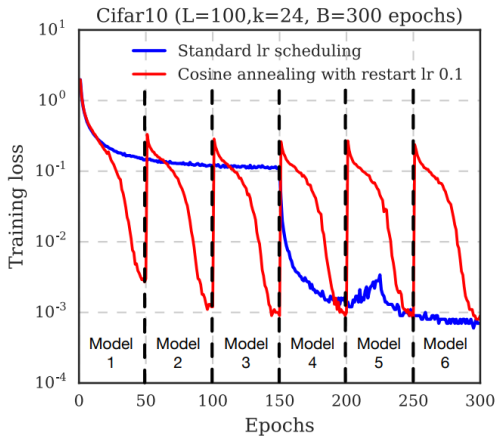

Fig. 3. Figure 2 from Huang et al. (2017): Training loss of 100-layer DenseNet on CIFAR10 using standard learning rate (blue) and M = 6 cosine annealing cycles (red). The intermediate models, denoted by the dotted lines, form an ensemble at the end of training.

With experiments they show the differences between snapshot ensembles (as described above), no-cycle snapshot ensembles (standard learning schedule), and single-cycle ensembles (no warm restarts, i.e., traditional ensemble with $1/M$ of the training time for each training). In this regard, no-cycle ensembles provide less model diversity and thus, the ensemble accuracy is lower as compared to snapshot ensembles. Further, single-cycle ensembles do not take advantage of warm restarts and thus, their ensemble accuracy is also lower as compared to snapshot ensembles.

Next, the local minima obtained are visualized for determining whether they correspond to the same basin of attraction or a different one. Specifically, the loss along a line connecting two minima, given by $\mathcal{L}(\lambda \boldsymbol{w}_1 + (1-\lambda)\boldsymbol{w}_2$ where λ is a mixing coefficient, can be plotted as in Fig. 4. As noted in the paper, two models that converge to a similar minimum (same basin of attraction) will have a smooth interpolation, whereas models that converge to different minima will likely have a spike in error for an intermediate value of λ. Note that this is also consistent with the results of Fort et al. (2020) where a line connecting two different minima is shown to pass from a region of high loss.

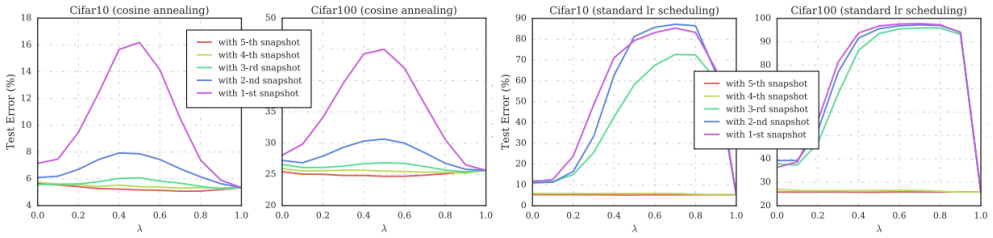

Fig. 4. Figure 5 from Huang et al. (2017): Interpolations in parameter space between the final model (sixth snapshot) and all intermediate snapshots. $\lambda = 0$ represents an intermediate snapshot model, while $\lambda = 1$ represents the final model. Left: A Snapshot Ensemble, with cosine annealing cycles ($a_0 = 0.2$ every B/M = 50 epochs). Right: A NoCycle Snapshot Ensemble, (two learning rate drops, snapshots every 50 epochs).

Overall, it appears that with cyclical learning rate most snapshots do not belong to the same basin, whereas with standard learning schedule the snapshots belong to the same basin.

Finally, model diversity is measured via pairwise correlation of outputs between any two snapshots. A higher number corresponds to less diversity; i.e., the predictive functions do not disagree much. As shown in Fig. 5, the correlation between models corresponding to adjacent snapshots is around 0.93 and corresponding to other snapshots around 0.9 (left). All models also have comparable accuracy. Therefore, they are good candidates for ensembling because we are looking for diverse and accurate predictions. On the other hand, although snapshots obtained with standard learning rate have small correlation in the beginning they correspond to small accuracy. And when the accuracy increases towards the fourth snapshot the correlation of the prediction is high. Therefore, these are not good candidates for ensembling.

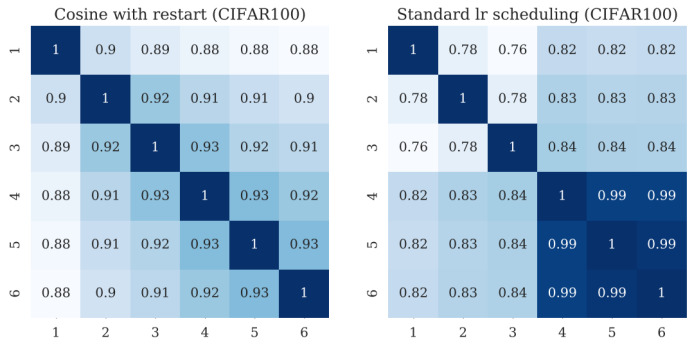

Fig. 5. Figure 6 from Huang et al. (2017): Pairwise correlation of softmax outputs between any two snapshots for DenseNet-100. Left:A Snapshot Ensemble, with cosine annealing cycles (restart with $a_0 = 0.2$ every 50 epochs). Right: A NoCycle Snapshot Ensemble, (two learning rate drops, snapshots every 50 epochs).

My comment: It seems that we cannot obtain uncertainty in the estimates with snapshot ensembles; it is only for increasing accuracy. This is because we cannot obtain more than say $6 - 10$ samples while retaining the same computational cost as standard training. This is the price we pay for visiting different basins of attraction. We need some iterations with decreasing learning rate after moving to a new basin before we are able to collect a sample.

5 Garipov et al. (2018): Loss surfaces, mode connectivity and fast ensembling of DNNs

This paper can be divided into three parts: connecting two modes with a path, error along an obtained path and ensembling.

5.1 Mode connectivity

Suppose we have two optima obtained by independently training a NN. We are interested in a path in the weight space connecting these two optima along which the loss is low on average. For doing so they consider two families of parametric curves, i.e., a polygonal chain (e.g., with two line segments and a bend in the middle) and a Bezier curve that defines a smooth path (e.g., a quadratic Bezier curve). These curves have some parameters that define them and we can solve for the parameters that make the average loss along the path minimum. See Fig. 6 for an illustration. In the left the

loss on the plane defined by three different optima is plotted, in the middle the loss on the plane[1] defined by the two optima and (as it appears) the middle point of the Bezier curve, and in the right the loss on the plane defined by the two optima and middle point of the polygonal chain. In all cases, connecting linearly two optima passes from a high-loss region. Finally, although this approach relates to connecting two optima obtained by independently training the same network with the same optimizer two times, see also Gotmare et al. (2018) for mode connectivity among pairs of networks trained with different optimizers, batch sizes, learning rate schedules etc.

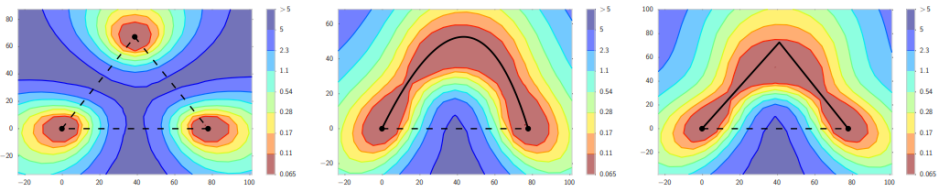

Fig. 6. Figure 1 from Garipov et al. (2018): The $l2$-regularized cross-entropy train loss surface of a ResNet-164 on CIFAR-100, as a function of network weights in a two-dimensional subspace. In each panel, the horizontal axis is fixed and is attached to the optima of two independently trained networks. The vertical axis changes between panels as we change planes (defined in the main text). Left: Three optima for independently trained networks. Middle and Right: A quadratic Bezier curve, and a polygonal chain with one bend, connecting the lower two optima on the left panel along a path of near-constant loss. Notice that in each panel a direct linear path between each mode would incur high loss.

5.2 Loss along the obtained path

In Fig. 4 we showed that connecting with a line segment two local minima that lie in the same basin results in a training loss along the path that does not have "barriers". On the other hand, a line segment connecting two local minima from different basins typically have a high-loss barrier between them. As a result, it is interesting to plot

[1]Suppose we have 3 weights w_1, w_2, w_3. We set $u = (w_2 - w_1), v = (w_3 - w_1) - \langle w_3 - w_1, w_2 - w_1 \rangle / \|w_2 - w_1\|^2 \cdot (w_2 - w_1)$. Then the normalized vectors $\hat{u} = u/\|u\|, \hat{v} = v/\|v\|$ form an orthonormal basis in the plane containing w_1, w_2, w_3. To visualize the loss in this plane, we define a Cartesian grid in the basis $\hat{u}, \hat{v}$ and evaluate the networks corresponding to each of the points in the grid. A point P with coordinates (x, y) in the plane would then be given by $P = w_1 + x\hat{u} + y\hat{v}$.

the loss along the paths obtained as described above. In this regard, it is shown in Fig. 7 (left and middle) that the training loss and the test error remain almost constant along the obtained paths but there is a high-loss region if a line segment is used. Most importantly, as shown in Fig. 7 (right) by ensembling the network corresponding to the left end (t=0) and a network along the path (at some t) reduces the test error. This means that along the path the predictions are diverse, otherwise the obtained ensemble would not reduce the test error compared the single network.

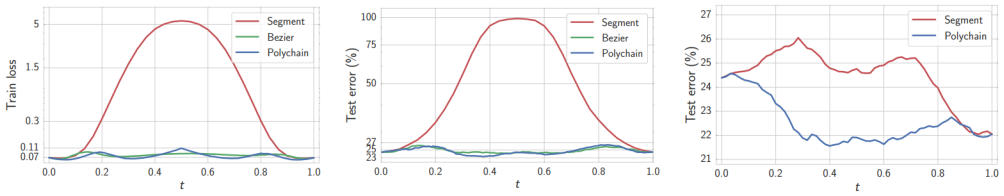

Fig. 7. Figure 2 from Garipov et al. (2018): The $l2$-regularized cross-entropy train loss (left) and test error (middle) as a function of the point on the curves $\phi_\theta(t)$ found by the proposed method (ResNet-164 on CIFAR-100). Right: Error of the two-network ensemble consisting of the endpoint $\phi_\theta(0)$ of the curve and the point $\phi_\theta(t)$ on the curve (CIFAR-100, ResNet-164). "Segment" is a line segment connecting two modes found by SGD. "Polychain" is a polygonal chain connecting the same endpoints.

5.3 Ensembling

The consequence of Fig. 7 (right) is that by ensembling a model obtained by a local minimum (t=0) with a model along the polygonal chain curve (e.g., t=0.4) reduces the test error even if t is small. Therefore, it may be possible to reduce the test error without training a second model and without obtaining a low-loss path. Inspired by Smith (2017) and Huang et al. (2017) the method they propose is cyclical learning rate, but with higher frequency cycles (e.g., 2-4 epochs). The rationale for this choice is that even by slightly perturbing the weights the obtained ensemble can reduce the test error. It seems that with this method the basin in which the local minimum found is explored.

On a practical note, they use a piecewise linear cyclical learning rate schedule and the local minimum is obtained with say 80% of the computational budget and with a standard learning rate. After this pre-training phase, the rest of the computational budget is used for perturbing the weights with cyclical learning rate.

6 Izmailov et al. (2019a): Averaging weights leads to wider optima and better generalization

In this paper they average the weights instead of the models produced by the weights. As a result, at test time the computational cost is not higher than a standard NN. The weights to be averaged are obtained by SGD with constant or cyclical learning rate. The authors show that doing so leads to better generalization as compared to standard SGD and to comparable generalization with the ensembling method (FGE) of Garipov et al. (2018).

As shown by Mandt et al. (2017), under simplifying assumptions running SGD with constant learning rate is equivalent to sampling from a Gaussian distribution centered at the minimum of the loss. Whether this distribution is the posterior depends on the learning rate schedule. In this regard, Izmailov et al. (2019a) construe the SGD samples as being constrained to the surface of the sphere (because of high dimensionality) and averaging the samples allows us to go inside the sphere to find higher density solutions. In other words, they do not use the fact that the SGD iterates come from the posterior neither they try to sample the posterior. Instead, they use the above as the intuition behind averaging. An additional intuitive explanation for averaging weights is given in Figs. 8 and 9. It seems that the points of the SGD trajectory are in the periphery of a set of high-performing networks.

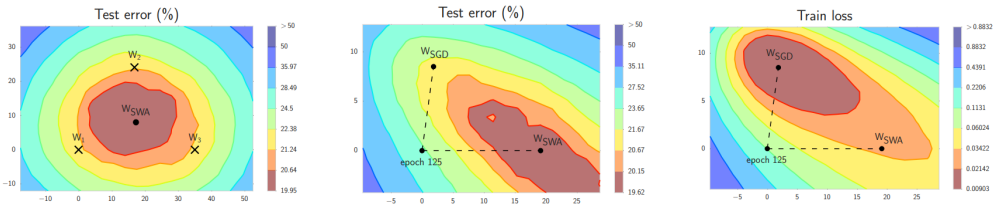

Fig. 8. Figure 1 from Izmailov et al. (2019a): Illustrations of SWA and SGD with a Preactivation ResNet-164 on CIFAR-1001. Left: test error surface for three FGE samples and the corresponding SWA solution (averaging in weight space). Middle and Right: test error and train loss surfaces showing the weights proposed by SGD (at convergence) and SWA, starting from the same initialization of SGD after 125 training epochs.

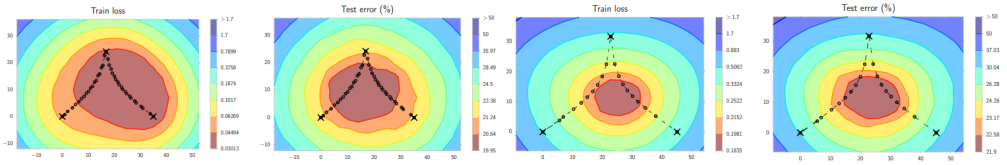

Fig. 9. Figure 3 from Izmailov et al. (2019a): The l2-regularized cross-entropy train loss and test error surfaces of a Preactivation ResNet-164 on CIFAR100 in the plane containing the first, middle and last points (indicated by black crosses) in the trajectories with (left two) cyclical and (right two) constant learning rate schedules.

Next, the authors show that the optimum found by averaging (SWA) corresponds to a wider solution. As shown in Fig. 10, moving away in random directions from SWA leads to less increase in train and test loss as compared to SGD.

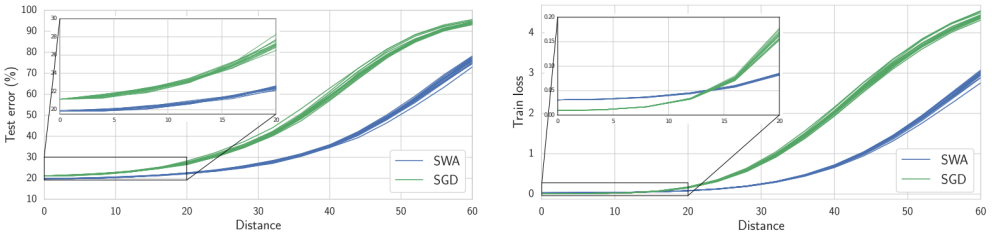

Fig. 10. Figure 4 from Izmailov et al. (2019a): The l2-regularized cross-entropy train loss and test error surfaces of a Preactivation ResNet-164 on CIFAR100 in the plane containing the first, middle and last points (indicated by black crosses) in the trajectories with (left two) cyclical and (right two) constant learning rate schedules.

This is also shown in Fig. 11. In the direction connecting SWA and SGD, the SGD optimum is close to a steep ascent in train and test loss. Although the training loss of SGD is smaller than SWA, SWA leads to better generalization.

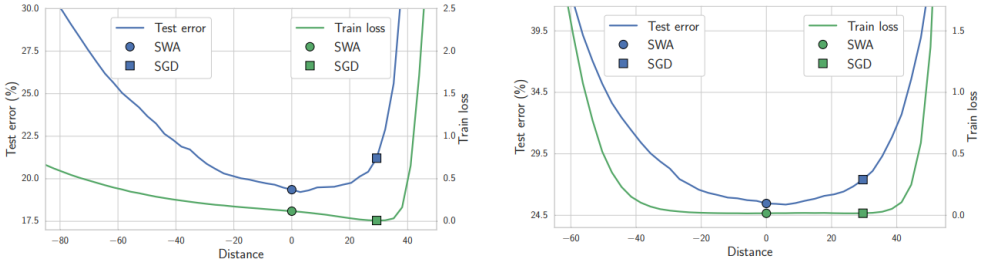

Fig. 11. Figure 5 from Izmailov et al. (2019a): l2-regularized cross-entropy train loss and test error as a function of a point on the line connecting SWA and SGD solutions on CIFAR-100. Left: Preactivation ResNet-164. Right: VGG-16.

My comment: What is the difference with Polyak averaging (Polyak and Juditsky, 1992)? The authors claim that Polyak averaging is not typically used for training NNs and that practitioners instead sometimes use an exponentially decaying running average of the weights found by SGD with a decaying learning rate, which smooths the trajectory of SGD but performs comparably.

7 Maddox et al. (2019): A simple baseline for Bayesian uncertainty in deep learning

In this paper a Gaussian distribution is fitted to the iterates of SGD. This technique is called SWAG. Specifically, the SWA approach of Izmailov et al. (2019a) is used for the mean of the distribution and two different approaches for estimating the covariance matrix are used. The first one is to fit a diagonal covariance matrix; a running average of the weights and of the squares of the weights is maintained (updated once per epoch) and the covariance matrix is computed at the end of training. The second approach is to estimate the covariance as the sum of the diagonal covariance and a low-rank matrix given as

$$\Sigma = \frac{1}{M-1}\sum_{i=1}^{M}(w_i - \bar{w}_i)(w_i - \bar{w}_i)^T \tag{1}$$

where $\bar{w}$ is the running average of the weights. Note that M defines the rank of the matrix and can be regarded as a hyperparameter of the method. After fitting a covariance matrix it is easy to draw samples of the weights.

According to Gur-Ari et al. (2018), quickly during training the parameters on the trajectory are in a region where the spectrum of the Hessian separates into two

components: a bulk component with many small eigenvalues and a top component of much larger positive eigenvalues. The large positive eigenvalues correspond to sharp directions. As a result, the tangent space (I guess this means where the gradient lives) has two orthogonal components, the bulk subspace and the top subspace. In other words, training happens in a tiny subspace and a large number of directions are left unchanged.

For example, the SWAG technique of Maddox et al. (2019) fits a distribution over this low-dimensional subspace spanned by the first principal components of the SGD iterates. Next, they claim that this fitted Gaussian distribution captures the geometry of the training loss (i.e., the posterior). They show that through Fig. 12: in the left the loss is plotted along the first principal components of the covariance matrix, in the middle the training loss is plotted around the SWA estimate and along the first two principal components and in the right along the third and the fourth. However, according to Gur-Ari et al. (2018): "First, the gradient of the loss during training quickly moves to lie within the top subspace of the Hessian. Within this subspace the gradient seems to have no special properties; its direction appears random with respect to the eigenvector basis". Therefore, is the result of Fig. 12 just random? In any case, the technique may still be accurate and an efficient Bayesian approximation. Note that the paper of Maddox et al. (2019) has been criticized for this reason by the reviewers of NIPS and a meta-reviewer was used who finally approved it. See more info at `http://media.nips.cc/nipsbooks/nipspapers/paper_files/nips32/reviews/7203.html`. Finally, since there is a large number of directions that are practically not modified during training, the fitted distribution is expected to under-estimate the variances along these directions. However, this seems to not have significant practical implications for the approximation accuracy.

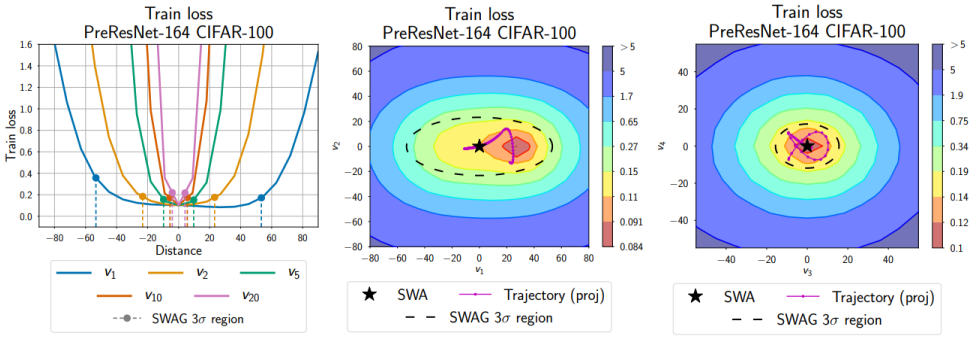

Fig. 12. Figure 1 from Maddox et al. (2019): Left: Posterior joint density cross-sections along the rays corresponding to different eigenvectors of SWAG covariance matrix. Middle: Posterior joint density surface in the plane spanned by eigenvectors of SWAG covariance matrix corresponding to the first and second largest eigenvalues and (Right:) the third and fourth largest eigenvalues. All plots are produced using PreResNet-164 on CIFAR-100. The SWAG distribution projected onto these directions fits the geometry of the posterior density remarkably well (see discussion).

8 Izmailov et al. (2019b): Subspace inference for Bayesian deep learning

Since training happens in a tiny subspace, the authors of this paper attempt to perform approximate inference in a subspace. For example, Li et al. (2018) attempts to perform training in a randomly oriented small subspace and define a size above which the accuracy difference is not significant. Thus, the first thing Izmailov et al. (2019b) try is to perform inference in a random subspace centered at the SWA estimate (this is an affine space). So they select a small number of random dimensions, create an affine space centered at the SWA estimate and perform inference in this subspace; i.e., obtain a distribution over this low-dimensional subspace. So the obtained weight samples after inference will belong to this subspace only and follow the distribution obtained via inference.

Another approach is to perform inference on the subspace defined by the first PCA components of the SGD iterates. This is similar to the technique of Maddox et al. (2019) but instead of fitting a Gaussian distribution they perform inference in the subspace. Further, they perform inference in the curve subspace discussed in Garipov et al. (2018). Note, however, that not only one has to construct this curve subspace (by

finding a connecting path) but then also perform inference.

Finally, note that because the subspaces discussed above are very low-dimensional and there are much more training datapoints the posterior of this inference concentrates and thus leads to overconfident uncertainty estimates. For this reason, they use a tempered posterior, i.e., raise the likelihood to a power. This power is a hyperparameter to be determined through cross-validation.

References

Brownlee, J. (2019). *Snapshot Ensemble Deep Learning Neural Network in Python (https://machinelearningmastery.com/snapshot-ensemble-deep-learning-neural-network/)*.

Fort, S., Hu, H., and Lakshminarayanan, B. (2020). "Deep Ensembles: A Loss Landscape Perspective". *arXiv:1912.02757 [cs, stat]*. arXiv: 1912.02757 [cs, stat].

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. (2018). "Loss Surfaces, Mode Connectivity, and Fast Ensembling of Dnns". *Advances in Neural Information Processing Systems*, pp. 8789–8798.

Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. (2018). "Using Mode Connectivity for Loss Landscape Analysis". *arXiv preprint arXiv:1806.06977*.

Gur-Ari, G., Roberts, D. A., and Dyer, E. (2018). "Gradient Descent Happens in a Tiny Subspace". *arXiv preprint arXiv:1812.04754*.

Huang, G. et al. (2017). "Snapshot Ensembles: Train 1, Get M for Free". *arXiv:1704.00109 [cs]*. arXiv: 1704.00109 [cs].

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. (2019a). "Averaging Weights Leads to Wider Optima and Better Generalization". *arXiv:1803.05407 [cs, stat]*. arXiv: 1803.05407 [cs, stat].

Izmailov, P. et al. (2019b). "Subspace Inference for Bayesian Deep Learning". *arXiv Preprint arXiv:1907.07504*.

Li, C., Farkhoor, H., Liu, R., and Yosinski, J. (2018). "Measuring the Intrinsic Dimension of Objective Landscapes". *arXiv:1804.08838 [cs, stat]*. arXiv: 1804.08838 [cs, stat].

Loshchilov, I. and Hutter, F. (2016). "Sgdr: Stochastic Gradient Descent with Warm Restarts". *arXiv preprint arXiv:1608.03983*.

Maddox, W. J., Izmailov, P., Garipov, T., Vetrov, D. P., and Wilson, A. G. (2019). "A Simple Baseline for Bayesian Uncertainty in Deep Learning". *Advances in Neural Information Processing Systems*, pp. 13153–13164.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). "Stochastic Gradient Descent as Approximate Bayesian Inference". *The Journal of Machine Learning Research* 18, pp. 4873–4907.

Polyak, B. T. and Juditsky, A. B. (1992). "Acceleration of Stochastic Approximation by Averaging". *SIAM journal on control and optimization* 30, pp. 838–855.

Smith, L. N. (2017). "Cyclical Learning Rates for Training Neural Networks". *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, pp. 464–472.

Xie, J., Xu, B., and Chuang, Z. (2013). "Horizontal and Vertical Ensemble with Deep Representation for Classification". *arXiv preprint arXiv:1306.2759*.