Ensemble strategies III – Snapshots

Based on *"Snapshot Ensembles: Train 1, get M for free"* [1]

https://arxiv.org/pdf/1704.00109.pdf

This section III explores strategies on how to build and train ensembles of deep neural networks as fast as possible.

In particular, this works explores the possibility of **training one network entirely and get the rest of the networks of the ensemble from that training process, at any cost.**

1. Nature of Stochastic Gradient Descent

The shape of the loss function with respect to the values of the weights of the neural networks is known to be a non-convex problem. Therefore, there are many singularities on that surface that can affect the convergence of the optimization process, such as local minima and saddle points.

Stochastic gradient descent and its accelerated variants have become the default optimizer precisely because their ability to avoid and even run away from these singular points.

Not so bad local minima

The points of the authors [1] is that local minima can have indeed a lot of useful information we could use, instead of trying to avoid and only think a model is good when it has reached the most minimum local minima when it converges. This way, we should indeed change a little bit the way SGD behaves to actually go and explore those local minima we are now interested in. Figure 2 shows the different behavior between the standard SGD approach and the approach propose by the authors we will discuss now.

Scheduling the Learning Rate

By analyzing a simple non-convex 1D function as the one shown in Figure 1 we could realize two things about the learning rate and why the standard approach is at is.

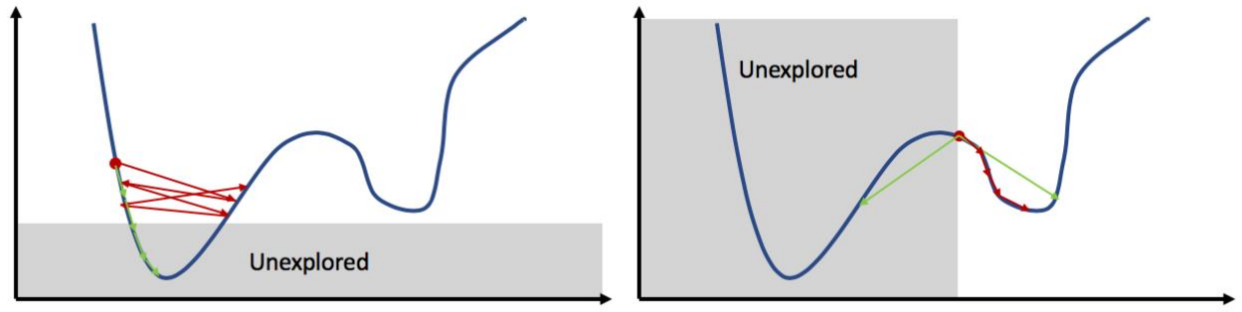


Figure 1. Consequences of too big and too small learning rates

On the image on the left, the red lines correspond to what actually the point will do by having that big learning rate corresponding to the arrows length. It can be seen how because of that length, there will be an area that the point will never reach, marked as unexplored. Precisely, the global minima we are interested in finding is in that unexplored area. Furthermore, the lines in green shows that if we had a smaller learning rate for that particular scenarios, the point would have indeed reached the desired location.

On the image on the right we have the opposite direction. Because of the learning rate being now small, the point 'thinks' that the direction to find the global minima is going right, since the gradient (derivative) is bigger in that direction. However, if it had a greater learning rate as shown in green lines, the point would have decided to explore the left side towards the global minima.

Combining these two observations, the standard way of scheduling a learning rate for an optimizer in deep learning consist on using a large value (typically 0.1) at the beginning of the training attempting to perform an exploration of the loss surface. Then, after some chosen number of iterations/epochs or after reaching some defined criteria, the learning rate is reduced to exploit the region found as the most suitable to have the global minima according to the previous learning rate.

Finally, the number of those local minima increases exponentially with the number of parameters, of which neural networks these days have in the order of millions. This encourage to find a place for the so already mentioned **diversity** within the different local minima:

Two identical architectures with different initializations (or minibatch ordering) will converge to different solutions. They will reach different local minima.

Besides this, their error rates could be almost identical, meaning both local minima are almost as minimum as each other. However, the mistakes that both networks will make are different.

This diversity can be exploited to create a good ensemble of deep neural networks

---

Now it is more understandable the motivation to use a different learning rate as authors suggest in Figure 2 from their paper. While left picture shows how the regular SGD learning rate, right side of the picture shows how authors approach explores few local minima before reaching the end goal.

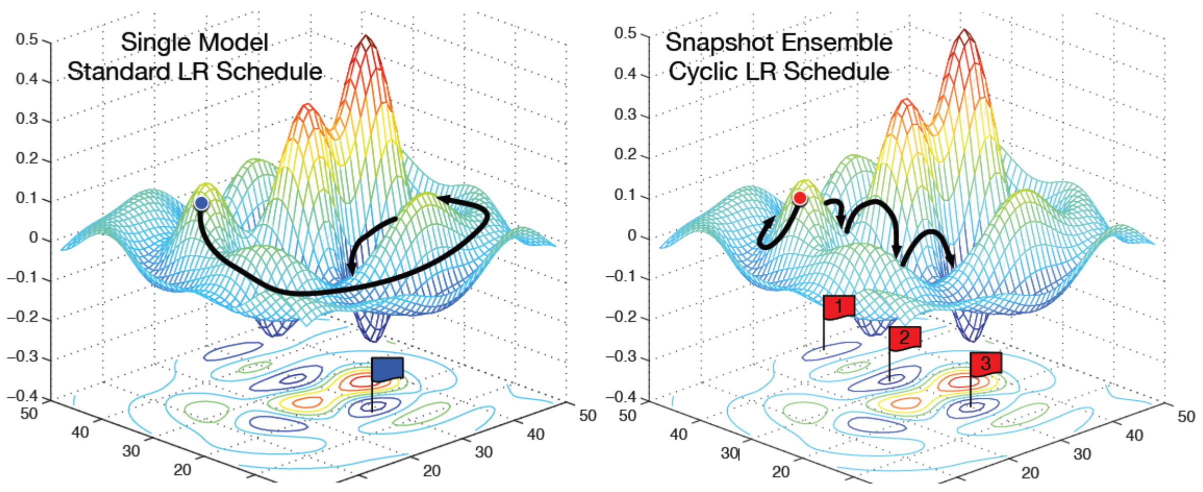


Figure 2. Standard SGD approach vs Snapshot SGD approach

2. Snapshots

The term snapshot in this paper comes from the fact of **'taking a picture of the network weights value'** when the optimization has reached a local minimum (each of the flags on Figure 2).

This way:

While doing a single network training, several snapshots yields to populate the ensemble

How to set the learning rate to perform this way?

As we want to get into local minima and scape from them during the training, it is possible to make use of the two phenomena shown in Figure 1.

Based on those, authors propose a learning rate based in [2], following a **cyclic annealing schedule** as shown in Figure 3. This figure also illustrates how each member of the ensemble is being incorporate to the whole before resetting the value of the learning rate at every cycle.

The learning rate is lowered at a very fast pace, encouraging the model to converge fast towards a close local minimum. The snapshot of the weights is then taken, and the first network of the ensemble is obtained.

To continue de optimization, the learning rate is restarted to be large enough to escape from that local minimum and look for the next one.

Formally, this learning rate have the shape as:

$$\alpha(t) = f\big(mod(t-1, \lceil T/M \rceil)\big) \qquad \text{(Eq 1)}$$

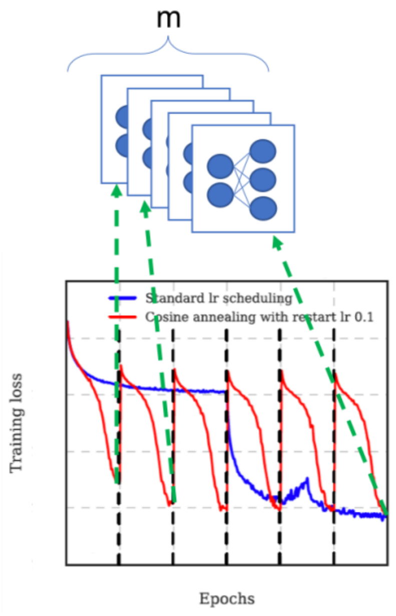


Figure 3. Cyclic annealing schedule

Where t correspond with the current iteration index, T is the total iterations during the entire training process and f is a monotonic decreasing function.

Authors chose for f a shifted cosine function proposed by [2], leading to the following function for the learning rate:

$$\alpha(t) = \frac{\alpha_0}{2}\left(\cos\left(\frac{\pi\, mod(t-1, \lceil T/M \rceil)}{\lceil T/M \rceil}\right) + 1\right) \qquad \text{(Eq 2)}$$

Following Eq2, the learning rate will vary from its initial value α_0 towards ≈ 0 every cycle.

3. Results

Authors compare the approach with variations of it and classic benchmarks.

The experiments where conducted on different architectures:

- ResNets
- Wide ResNets
- DenseNets

On different datasets:

- CIFAR 10 / 100
- ImageNet
- Tiny ImageNet
- SVHN

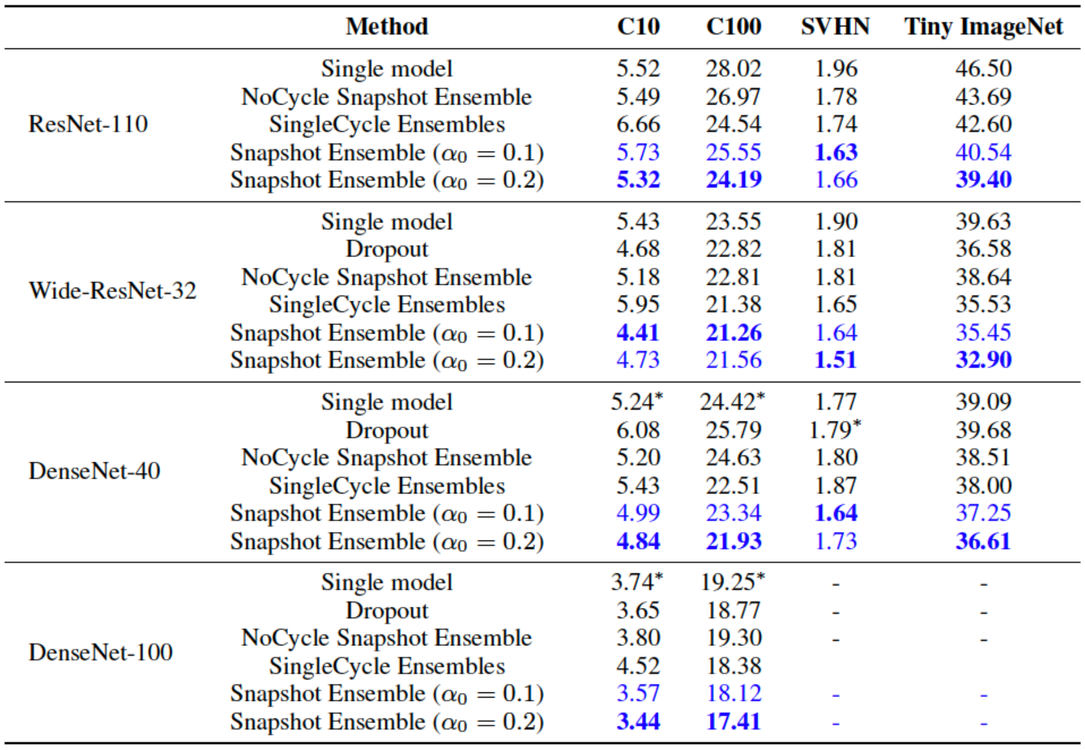
To serve as baselines:

- Single model trained on a standard learning rate scheduling – **Single Model**
- The explained snapshot ensembling algorithm – **Snapshot Ensemble**
- A snapshot ensembling without the cyclic schedule – **NoCycle Snapshot Ensemble**
- A snapshot that restart entirely after every cycle – **SingleCycle Ensemble**

NoCycle is intended to prove the importance of having the cyclic annealing for the learning rate

SingleCycle is intended to prove the importance of this approach against traditional ensemble

Table 1. Results for Snapshot Ensembling experiments



	Method	C10	C100	SVHN	Tiny ImageNet
ResNet-110	Single model	5.52	28.02	1.96	46.50
	NoCycle Snapshot Ensemble	5.49	26.97	1.78	43.69
	SingleCycle Ensembles	6.66	24.54	1.74	42.60
	Snapshot Ensemble ($\alpha_0 = 0.1$)	5.73	25.55	**1.63**	40.54
	Snapshot Ensemble ($\alpha_0 = 0.2$)	**5.32**	**24.19**	1.66	**39.40**
Wide-ResNet-32	Single model	5.43	23.55	1.90	39.63
	Dropout	4.68	22.82	1.81	36.58
	NoCycle Snapshot Ensemble	5.18	22.81	1.81	38.64
	SingleCycle Ensembles	5.95	21.38	1.65	35.53
	Snapshot Ensemble ($\alpha_0 = 0.1$)	**4.41**	**21.26**	1.64	35.45
	Snapshot Ensemble ($\alpha_0 = 0.2$)	4.73	21.56	**1.51**	**32.90**
DenseNet-40	Single model	5.24*	24.42*	1.77	39.09
	Dropout	6.08	25.79	1.79*	39.68
	NoCycle Snapshot Ensemble	5.20	24.63	1.80	38.51
	SingleCycle Ensembles	5.43	22.51	1.87	38.00
	Snapshot Ensemble ($\alpha_0 = 0.1$)	4.99	23.34	**1.64**	37.25
	Snapshot Ensemble ($\alpha_0 = 0.2$)	**4.84**	**21.93**	1.73	**36.61**
DenseNet-100	Single model	3.74*	19.25*	-	-
	Dropout	3.65	18.77	-	-
	NoCycle Snapshot Ensemble	3.80	19.30	-	-
	SingleCycle Ensembles	4.52	18.38		
	Snapshot Ensemble ($\alpha_0 = 0.1$)	3.57	18.12	-	-
	Snapshot Ensemble ($\alpha_0 = 0.2$)	**3.44**	**17.41**	-	-

It is noticeable how snapshot ensemble outperforms the rest of the models in almost every scenario. The explanation can reside in the diversity achieved with snapshot ensembling approach.

Diversity of the ensembles

We can verify that the ensemble built is diverse if the different networks misclassifies different samples

We can look at the pairwise correlation of the probabilities (after softmax) of each member of the ensemble and compare that result between the cyclic approach and the standard approach for the learning rate schedule, as shown in Figure 4:

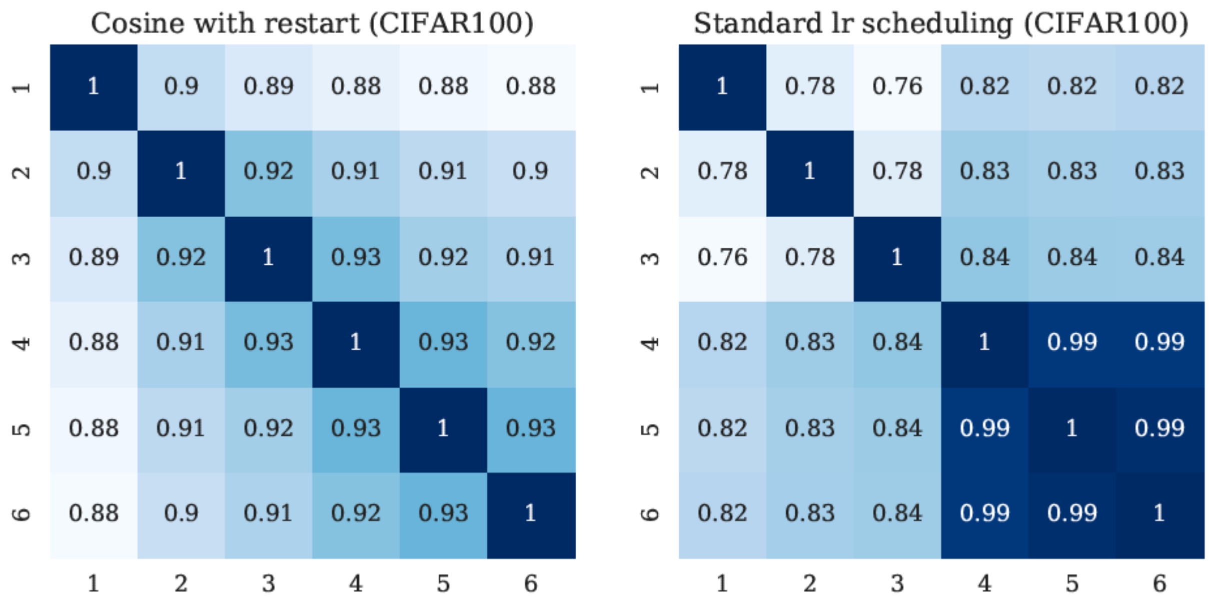


Figure 4. Pairwise correlation of probabilities between any two members of the ensemble

The right-hand side demonstrate how after the learning rate was reduced (for the last 3 snapshots) the correlation is big. This means that they are reaching the same local minima, and therefore the three snapshots are very alike, and there is no diversity among them. Furthermore, the first 3 that are not correlated have a poor performance, since the learning rate was high and therefore they didn’t converge to any ‘good’ local minima at all.

The left-hand side shows how by making the learning rate schedule cyclic, all of the snapshots achieve a good performance since the learning rate has been reduced for each of them. Furthermore, there are no so big correlations among them, meaning that there is diversity in their beliefs for every sample.

Bibliography

[1] G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcrof and K. Q. Weinberger, "Snapshot Ensembles: Train 1, get M for free," *ICLR,* 2017.

[2] I. Loshchilov and F. Hutter, "Stochastic gradient descent with restats," *Arxiv,* 2016.