EE763 Project - Paper Review

Karan Chadha
Roll No: 140070014

April 5, 2018

The topic explored in this project is the study of the mathematics underlying regularization in deep neural networks and how stochastic gradient descent performs variational inference. I have primarily referred to the following papers for this project:

1. Entropy-SGD: biasing gradient descent into wide valleys
   P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, R. Zecchina Proc. of the International Conference on Learning and Representations(ICLR).

2. Parle: distributed training of deep neural network
   P. Chaudhari, C. Baldassi, R. Zecchina, S. Soatto, A. Talwalkar, A. Oberman preprint arXiv:1707.00424

3. Deep Relaxations: partial differential equations for optimizing deep neural network
   P. Chaudhari, A. Oberman, S. Osher, S. Soatto, G. Carlier, Presented at SIAM Conference on Analysis of Partial Differential Equations.

4. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks
   P. Chaudhari, S. Soatto, Short version presented at the Workshop on Advances in Approximate Bayesian Inference, NIPS.

The structure of this project report is as follows:

- Motivation
- Mathematics of Stochastic Gradient Descent
- Entropy-SGD: biasing gradient descent into wide valleys
- Parle: A distributed training of deep neural network
- Deep Relaxations: partial differential equations for optimizing deep neural network
- Stochastic gradient descent performs variational inference
- Conclusive remarks

Motivation

Deep neural networks (DNNs) are parametric models that perform sequential operations called "layers" on input data. Each layer consists of a linear transformation followed by a pointwise non-linear transform (e.g. sigmoid, ReLU, etc.). It is aptly represented by the following expression:

$$y(x;\xi) = \sigma'(x^l\sigma(x^{l-1}\sigma(x^{l-2}\dots\sigma(x^1\xi))\dots)),$$

where ξ represents the input data, y represents the output, x represents the parameters (weights) for the linear transform, σ represents the non-linear transform and σ' represents the non-linear transform of the last layer. Training a deep neural network is usually done using the backpropogation algorithm, wherein one tries to find the parameters to minimize a suitably chosen loss function ($f(x)$). For theoretical analysis, we will consider training of deep neural networks as a generic non-convex optimization problem given by

$$x^* = \arg\min_x f(x)$$

DNNs have shown dramatic improvement in classification performance in various fields such as image processing, speech processing, natural language processing, reinforcement learning and so on. However, very few theoretical results relating to them are known. This report summarizes few papers which take a step in the direction of theoretical analysis of neural networks. Generically, there are three main factors in deep learning, namely the architectures, regularization techniques and optimization algorithms. The analysis here is based on including the regularization term in the optimization algorithm itself and it holds across all architectures.

An important goal in training deep neural networks is to have low generalization error. Generalization error is the difference between the empirical and the expected error. Various techniques are used to reduce this including using adaptive and annealed learning rates, momentum, as well as architectural modifications like dropout, batch-normalization, weight scaling etc. Empirically, it has been shown that stochastic gradient descent, when used to train deep neural networks, converges to regions where there are densely concentrated minima (although they may be shallower) rather than isolated deep minima. It has also been theoretically shown for shallow networks with discrete weights that their energy landscape is characterized by an exponential number of isolated minima and few very dense regions with lots of local minima close to each other. The regions densely populated with local minima are more robust to perturbations of both weights and data which result due to techniques of obtaining better generalization described above. They are expected to give better generalization error.

The first part of this report gives a new algorithm which looks for "wide" valleys rather than just deep ones and gives an outline of the proof that it performs better than the usual stochastic gradient descent. The second part relates to showing how stochastic gradient descent is essentially performing variational inference and how out of equilibrium behaviour is exhibited and is useful for better generalization.

Mathematics of Stochastic Gradient Descent

First order methods are the default choice of training deep neural networks since x is of very high dimension. Also, computing the entire gradient ($\frac{1}{|\xi|}\sum_{i=1}|\xi|\nabla f_i(x)$) at each iteration is not feasible due to a large dataset and hence gradient is calculated using randomly chosen minibatches. The update equation is given by:

$$x_{k+1} = x_k - \eta_k \nabla f_{mb}(x),$$

where η_k represents the learning rate and $\nabla f_{mb}(x)$ the minibatch gradient. If the loss function is convex and the gradient is uniformly lipschitz, stochastic gradient descent is provably convergent. Stochastic gradient descent can also be interpreted as a continuous time stochastic differential equation as follows:

$$dx(t) = -\nabla f(x(t))dt + \beta^{-1/2}dW(t),$$

where $W(t)$ represents the Brownian motion and β, the variance of the noise. Furthermore, $\rho(x,t)$, the probability density of $x(t)$ at time t, satisfies the Fokker-Planck equation, along with the initial condition $\rho(x,0) = \rho_0(x)$, which represents the probability density of the inital distributions. Under mild assumptions on f, even when it is non-convex, $\rho(x,t)$ converges to the unique stationary distribution of the Fokker-Planck equation as $t \to \infty$. The stationery distribution is given by the Gibbs distribution

$$\rho^{\infty}(x;\beta) = \frac{1}{Z(\beta)}e^{-\beta f(x)},$$

where $Z(\beta)$ is a normalizing constant. As $\beta \to \infty$, the Gibbs distribution concentrates on the global minimizers of $f(x)$. In practice momentum is often used to accelerate the convergence, which corresponds to

$$\begin{aligned} dx(t) &= p(t)dt \\ dp(t) &= -(\nabla f(x(t)) + cp(t))dt + \sqrt{c\beta^{-1}}dW(t). \end{aligned}$$

In the overdamped limit ($c \to \infty$), SGD can be recovered, and in the underdamped limit ($c \to 0$), Hamiltonian dynamics.

Entropy-SGD: biasing GD into wide valleys

This paper proposes a new optimization algorithm called Entropy-SGD for training deep neural networks that is motivated by the local geometry of the energy landscape. Local minima that generalize well and are found by running SGD are located in wide valleys of the energy landscape rather than in sharp isolated minima. Instead of minimizing $f(x)$, it is proposed to maximize

$$F(x, \gamma) = log \int_{x' \in \mathbb{R}^n} exp(-f(x') - \frac{\gamma}{2}||x - x'||_2^2)dx'.$$

It can also be represented as a convolution with the Gaussian kernel as follows:

$$F(x, \gamma) = log(exp(-f(x)) * G_\gamma),$$

where G_γ represents a gaussian kernel of variance γ^{-1}.

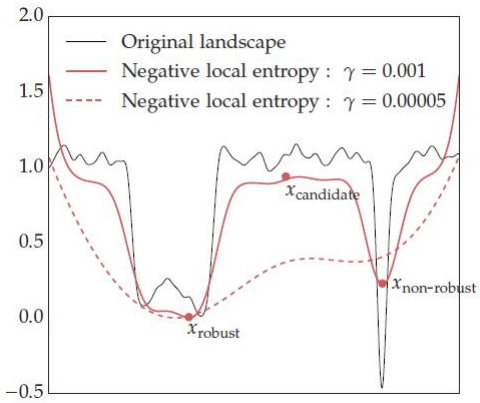

Figure 1: Example loss function showing robus minima in a wide valley and the non-robust minima in a sharp but deeper valley. It also has plots of the negative local entropy for two different values of γ. Source: *Entropy-SGD: biasing GD into wide valleys, P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, R. Zecchina Proc. of the International Conference on Learning and Representations(ICLR)*

As discussed, the parameters that lie in the wider local minima x_{robust} in Fig 1, should have better generalization than the ones that are simply at the global minimum ($x_{non-robust}$). Also, note that around x_{robust}, the local entropy is generally larger than that around $x_{non-robust}$. This is because, depending on the value of γ, local entropy can be thought of as averaging over a small interval around the point being considered. This is because, after a certain point, the quadratic distance term in the local entropy dominated the loss function term and the arguments inside the integral are too small to make a difference.

Fig 1 also shows the negative local entropies for two different values of gamma. We note that, the negative local entropy, in both the cases has minimum near x_robust rather than near the sharp minimum of the original loss function $x_{non-robust}$. When γ is small, there is smoothening (averaging) over a larger interval, since the quadratic distance term becomes dominant at farther distances as compared to when γ is bigger. For very small values of γ, the averaging is over the whole space and the

local entropy function is almost uniform. As γincreases, the local entropy landscape gets closer to the original loss function landscape and they become the same as $\gamma \to \infty$.

Comparing local entropy with classical entropy as a loss function, we notice that classical entropy just tries to find the widest region irrespective of the value of the loss function there. Hence, local entropy serves as a middle ground between the original loss function and classical entropy.

Algorithm

Result: Minimizer of the local entropy function
Inputs: current weights x, Langevin iterations L;
Hyperparameters: scope γ, learning rate η, SGLD stepsize η';
$x', \mu' \leftarrow x$;
while $l \leq L$ **do**
  $\Xi^l \leftarrow$ Sample minibatch;
  $dx' \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_{x'} f(x'; \xi_{l_i}) - \gamma(x - x')$;
  $x' \leftarrow x' - \eta' dx' + \sqrt{\eta'}\epsilon N(0, \mathbf{I})$ $\mu \leftarrow (1-\alpha)\mu + \alpha x'$;
end
$x \leftarrow x - \eta\gamma(x - \mu)$;

Algorithm 1: Entropy SGD

The algorithm applies SGD two times, one is for the final update of the parameters and the other is for the estimate of the gradient of the local entropy function using Langevin dynamics. The need for using Langevin dynamics is explained below. The gradient of local entropy over a randomly sampled mini-batch of m samples denoted by $\xi_{l_i} \in \Xi^l$ for $i \leq m$ is easy to derive and is given by

$$-\nabla F(x, \gamma; \Xi^l) = \gamma(x - \mathbb{E}_{\Xi^l}[x]);$$

the expectation above is over a Gibbs distribution and is hard to compute. Hence, we approximate it using MCMC techniques. In particular, we use stochastic gradient Langevin dynamics which is an MCMC algorithm for drawing samples from a Bayesian posterior and scaled well using mini-batches.

The Langevin dynamics is breifly described as follows. Suppose we want the uodates for a parameter vector x with a prior distribution $p(x)$ and a probability of generating a data item ξ_i given by $p(\xi_i|x)$. Then the update equation is given by

$$\Delta x_t = \frac{\eta}{2}(\nabla logp(x_t) + \frac{N}{m}\sum_{k=1} m\nabla logp(\xi_k|x_t)) + \sqrt{\eta_t}\epsilon_t,$$

where ϵ_t is the noise added at time t. Intuitively, the expectation term in the gradient is the avaerage over a locally focused Gibbs distribution and it pushes the iterates into wider valleys. Unrolling the SGLD steps into a single update, the algorithm looks like that of averaged SGD in the literature, which intuitively explains the improved generalization performance. Increasing γ as training progresses has the effect of exploring the energy landscape on finer scales. Since this increase might conflict the annealing of the learning rate in the final update equation, while updating $\eta\gamma$ is replaced by just η and the values are set accordingly. It is suggested to set the value of γ to be such that the magnitude of the local entropy gradient is comparable to that of original function SGD.

Theoretical results showing higher level of smoothness of the local entropy function over the original functions have been proved. Along with this, it has also been proved that if both algorithms converge after T passes over the samples, Entropy SGD generalizes better than SGD.

Parle: A distributed training of deep neural network

A new algorithm called Parle for parallel training of deep networks that converges 2-4 ×faster than a data-parallel implementation of SGD, while achieving significantly improved error rates (tested empirically). Parallel and distributed training of deep learning is paramount to tackle the problems that exist in the current world. The world has a lot of data for some problems but not enough computational ability to learn from it. However, such implementations have reached a roadblock: to minimize communication costs, one needs to increase the batch size in SGD which leads to degradation of the generalization performance. The purpose of this paper is to provide an algorithm which requires very infrequent communication with the parameter server and instead performs more computations on each client.

The idea is to combine two known algorithms, namely Entropy SGD (discussed in the previous section) and Elastic SGD to provide an algorithm which has lesser communication overhead and performs SGD in parallel. Again the phenomenon of "flat minima" is leveraged to change the loss function. We know that training consists of solving

$$x^* = \arg\min_x f(x).$$

Let $x^1, x^2, \ldots, x^n$ denote the copies of the model by variables, also called replicas. They may reside on multiple computer nodes. To couple these variables we consider the Elastic-SGD(Zhang et al., 2015) loss function:

$$\arg\min_{x, x^1, \ldots, x^n} \sum_{a=1}^{n} f(x^a) + \frac{1}{2\kappa}||x^a - x||^2;$$

where the parameter $\kappa > 0$ couples any two replicas through a reference variable x. Each noide performs minimization of its own objective and comunicates the elastic gradient $\rho^-1(x_k^a - x_k)$ after each SGD update. The updates look like:

$$x_{k+1}^a = x_k^a - \eta[\nabla f(x_k^a) + \frac{1}{\rho}(x_k^a - x_k)] \quad \forall a \leq n$$

$$x_{k+1} = x_k - \eta(x_k - \frac{1}{n}\sum_{a=1}^{n} x^a).$$

Although implemented parallely, this algorithm has large communication bottlenecks. Also, the variable x here can be seen as updating itself as the average of the clients (nodes). This resembles the ideas of Entropy SGD, where in we did update parameters with some sort of average. This resemblance is not a coincidence. It is shown in the *DeepRelaxation* paper which will be summarized in the next section that, if $\nabla^2 f(y) + \gamma^{-1}\mathbf{I} > 0$, then the parameter updates have an ergodic steady state distribution whereby the temporal averages (Entropy SGD update) are equivalent to the spatial averages (Elastic SGD update).

The combined "Parle" updates are given by:

$$
\begin{aligned}
y^a_{k+1} &= y^a_k - \eta'[\nabla f(y^a_k) + \frac{1}{\gamma}(y^a_k)] \\
z^a_{k+1} &= \alpha z^a_k + (1-\alpha) y^a_k \\
x^a_{k+1} &= \begin{cases} x^a_k - \eta(x^a_k - z^a_{k+1}) - \frac{\eta}{\rho}(x^a_k - x_k) & \text{if } k/L \text{ is an integer} \\ x^a_k & \text{otherwise} \end{cases} \\
x_{k+1} &= \begin{cases} x_k - \frac{\eta''\eta}{\rho}(x_k - \frac{1}{n}\sum_{a=1}^n x^a) & \text{if } k/L \text{ is an integer} \\ x_k & \text{otherwise} \end{cases}
\end{aligned}
$$

Note that y^a_k is reset to x^a_k whenever k/L is an integer. Also, all updates are similar to the Entropy-SGD and Elastic SGD updates, except for the fact that x^a_k is updated using the gradient of both the local entropy function and the elastic term.

By adjusting the hyperparameters we can shift the overhead from communication to decentralized nodes and vice versa easily. On high computing devices like GPUs, it is recommended to keep the communication overhead low whereas in CPUs and mobile devices one can leverage the fact that communication is easier.

Deep Relaxations: partial differential equations for optimizing deep neural network

In this paper, a connection has been established between deep neural networks and non-linear partial differential equations. Theoretical analysis of well known recent advancements in the field of deep learning (Entropy SGD and Elastic SGD) has been provided. The tools used are non-linear partial differential equations, stochastic differential equations and their homogenization techniques and some results from stochastic control. Using these, the equivalence of Elastic SGD and Entropy SGD has been derived and it has also been proved that Entropy SGD performs better than SGD on the original loss function for training deep neural networks.

First, it is proved that the local entropy loss function $f_\gamma(x)$ is the solution of a non-linear partial differential equation, which is the viscous Hamilton-Jacobi PDE given by,

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2 + \frac{1}{2\beta}\Delta u, \qquad \text{for } < t \leq \gamma$$

with $u(x,0) = f(x)$, where Δ represents the Laplacian. It is shown that $f_\gamma(x) = u(x,\gamma)$. Next, it has also been proved that

$$u(x,t) = \inf_y \{f(y) + \frac{1}{2t}|x-y|^2\},$$

and if $y^* = prox_{tf}(x)$ is a singleton, then $\nabla_x u(x,t)$ exists and

$$\nabla_x u(x,t) = \frac{x - y^*}{t} = \nabla f(y^*),$$

and $prox_{tf}(x)$ is the proximal operator defined as $prox_{tf}(x) = \arg\min_y \{f(y) + \frac{1}{2t}|x - y|^2\}$.

Using these facts, the authors first claim that using standard properties of inf-convolution, one can show that if $f(x)$ is convex, then so is $u(x,t)$. Moreover, they also claim that for bounded $f(x)$, one can prove that for short times t, local minima of $u(x,t)$ persist. The authors then prove that the local entropy loss function is smoother than the original loss function. To quantify the amount of smothness, standard semiconcavity results have been used. The semiconcavity estimates give bounds on $sup_x u_{x_k x_k}(x,t)$ and the Laplacian $sup_x \Delta u(x,t)$. Using these estimates, the authors give bounds on the harmonic mean of the eigenvalue spectrum of the Hessian of a trained (converged) deep neural network. It was empirically observed earlier that the Hessian has most of it's eigenvalues close to 0. Since, we want to focus on the low magnitude eigenvalues, the maximum eigen value or the trace don't give much information about the same, whereas the Harmonic mean is more sensitive to the lowest values. The HM discounts the high eigenvalues and gives more information about the ones close to 0, which dictate the width of the minima. For instance,

$$\min_i x_i \leq HM(x) \leq n \min_i x_i.$$

An upper bound on the Harmonic mean has been provided which is indicative of the most eigenspectrum observed in practice.

Next, the iterative updates using the local entropy cost function is represented in the form of Stochastic Differential Equations. First, one computes the invariant measure (ρ) of the auxiliary stochastic differential equation(SDE)

$$dy(s) = -(\gamma^{-1} y(s) + \nabla f(x(t) - y(s)))ds + \beta^{-1/2} dW(s)$$

then update x(t) by averaging against this invariant measure

$$dx(t) = -\int \nabla f(x(t) - y)\rho(dy).$$

It is shown using homogenization of SDEs that the above equations recover the gradient of the local entropy function, $u(x,\gamma)$. Homogenization is a popular technique used to analyse dynamical systems when there are multiple timescales with coupled variables in each timescale. Informally, it helps in computing averages over the variables which vary at fast timescales which in turn gives us averaged equations in slow variables in the limit that the timescales separate. In particular, the authors prove that the above SDEs are equivalent to

$$dx(t) = -\nabla u(x(t), \gamma)dt.$$

The effectiveness of the algorithm may be partially explained by the fact that for small value of γ, that solutions of Fokker-Planck equation associated with the auxiliary SDE converge exponentially quickly to the invariant measure ρ. Tools from optimal transport have been used to support this claim. Thus while smoothening of the loss function using PDEs directly is not practical, due to the curse of dimensionality, the auxiliary SDE achieves the same with exponentially convergent dynamics.

Besides, again using homogenization it was proved that Elastic SGD is equivalent to Entropy SGD when one observes continuous time dynamics. For this, ergodicity results were used which are valid only when $\nabla^2 f(y) + \gamma^{-1}\mathbf{I} > 0$ is satisfied. It is uncommon in stochastic nonconvex optimization to obtain a proof of the superiority of an algorithm over SGD. Such a result was obtained in terms of the expected value of the loss function using techniques from stochastic control theory.

SGD performs variational inference

SGD is believed to perform implicit regularization when used to train deep neural networks. But the precise reason for that appears to be unknown. This paper proves that SGD performs variational inference, albiet on a possibly different loss function. More surprisingly, it is also shown that SGD doesn't even converge in the traditional sense, i.e. most likely trajectories for DNNs do not behave like Brownian motion around critical points.

The minibatch size is assumed to be b. the diffusion matrix $(D(x))$ is defined as

$$D(x) = (\frac{1}{N}\sum_{k=1}^{N} \nabla f_k(x) \nabla f_k(x)^T) - \nabla f(x) \nabla f(x)^T \geq 0.$$

It is shown that when sampling with replacement the variance of the minibatch gradient is given by $var(\nabla f_b(x)) = \frac{D(x)}{b}$. Note that $D(x)$ is independent of the learning rate and the minibatch size. It only depends on the weight, architecture, loss and dataset. They have defined isotropic diffussion to be the case when $D(x)$ is a scalar multiple of identity, independent of x, and non-isotropic diffusion, when D(x) is any other general function of x.

It is assumed that the steady state distribution of parameters, given by the Fokker-Planck equations exists and is unique. The potential is defined using the steady state distribution as $\Phi(x)$ satisfying

$$\rho^{ss}(x) = \frac{1}{Z(\beta)} e^{-\beta \Phi(x)}.$$

The above is also a steady state solution of

$$dx(t) = \beta^{-1} \nabla \cdot D(x) dt - D(x) \nabla \Phi(x) dt + \sqrt{2\beta^{-1} D(x)} dW(t).$$

Since we want $\nabla f(x)$ instead of the term above multiplying dt, we get a remainder term on putting $\nabla f(x)$ in the above equation as

$$j(x) = -\nabla f(x) + D(x) \nabla \Phi(x) - \beta^{-1} \nabla \cdot D(x)$$

Another assumption is made on $j(x)$ which draws inspiration from physics. $j(x)$ is represents force when the FP equations model energy exchange with the environment. Since force is conservative, $\nabla \cdot j(x) = 0$. It is then shown that the following functional decreases along the trajectories of the Fokker-Planck equation and converges to its minimum at steady state:

$$\begin{aligned} F(\rho) &= \beta^{-1} KL(\rho || \rho^{ss}) \\ &= \mathbb{E}_{x \in \rho}[\Phi(x)] - \beta^{-1} H(\rho) + constant \end{aligned}$$

This shows, SGD implicitly minimizes a combination of two terms, an "energetic" term and an "entropic" term. Also note that the energetic term has potential $\Phi(x)$ instead of $f(x)$. The relation between $\Phi(x)$ and $f(x)$ is tackled next. It is shown that $\Phi(x) = f(x)$ if and only if the diffusion matrix is isotropic i.e. $D(x) = cI_{d \times d}$.

When $D(x)$ is non-isotropic, $j = 0$. Hence, a deterministic component always prevails in the FP equation, although it doesn't affect the functional $F(\rho)$. This deterministic component is given by

$$\dot{x} = j(x).$$

The condition $\nabla \cdot j(x) = 0$ implies that most likely trajectories of SGD traverse in closed trajectories in weight space.

Next a few remarks are stated and justified which have practical implications. The effect of two important parameters, the learning rate and the mini-batch size completely determines the strength of the entropic regularization term. If $\beta^{-1} = \frac{\eta}{2b} \to 0$, the effect of the entropic regularization goes to 0. Hence, it should not be very small. The following must be noted:

- Learning rate should be scaled linearly with batch-size to generalize well. This is straightforward from the previous observation. It is also empirically supported that for good generalization, one needs small minibatches or minibatch size generated via such linear scaling.

- Sampling with replacement is better than without replacement. The diffusion matrix for sampling without replacement is very similar to that of with replacement. But, the inverse temperature for sampling without replacement is given by
$$\beta^{-1} = \frac{\eta}{2b}(1 - \frac{b}{N}).$$
Hence, it decreases faster than the inverse temperature for sampling with replacement and leads to worse regularization.

Finally, the paper characterises the out of equilibrium behaviour of SGD on DNNs. It is shown that continuous time original loss function SDE is equivalent to the one with the potential function only if the following is satisfied:

$$\nabla f(x) = (D(x) + Q(x))\nabla\Phi(x) - \beta^{-1}\nabla \cdot (D(x) + Q(x)),$$

where $Q(x)$ is an antisymmeteric matrix which along with the potential function can be explicitly calculated in terms of the graient and the diffusion matrix. Some remarks that follow are that the out of equilibrium behaviour is exhibited even when D is constant and it increases with β^{-1}. But these are the exact conditions which yield good generalization. This suggests out of equilibrium behaviour may be good for SGD to achieve good generalization error.

To conclude, local entropy function does not rely on non-isotropic gradient noise to attain out of equilibrium (of original loss function) behaviour, but instead it gets it explicitly by construction (convolution with gaussian kernel). This suggests actively inducing out of equilibrium behaviour yields good generalization in practice.

Conclusive Remarks

This line of work is to provide theoretical contributions to undertand the working of deep neural networks. It is mainly motivated by the observation that the shallow but wide valleys obtain better generalization than sharp but deep ones. New algorithms have been formulated, empirically tested and then analytically shown to perform better than stochastic gradient descent in terms of generalization error. Parallel implementations have also been suggested to cater to the new needs of Big Data. Finally, analysis on why out of equilibrium behaviour is good for generalization is given.