# Deep Relaxation: partial differential equations for optimizing deep neural networks

**Pratik Chaudhari** [*1] **Adam Oberman** [*2] **Stanley Osher** [3] **Stefano Soatto** [1] **Guillaume Carlier** [4]

## Abstract

This paper establishes a connection between non-convex optimization and nonlinear partial differential equations (PDEs). We interpret empirically successful relaxation techniques motivated from statistical physics for training deep neural networks as solutions of a viscous Hamilton-Jacobi (HJ) PDE. The underlying stochastic control interpretation allows us to prove that these techniques perform better than stochastic gradient descent (SGD). Moreover, we derive this PDE from a stochastic homogenization problem which proves connections to algorithms for distributed training of deep networks like Elastic-SGD. Our analysis provides insight into the geometry of the energy landscape and suggests new algorithms based on the non-viscous Hamilton-Jacobi PDE that can effectively tackle the high dimensionality of modern neural networks.

## 1. Introduction

Deep neural networks have achieved remarkable success in a number of applied domains from visual recognition and speech to natural language processing and robotics (LeCun et al., 2015). Despite many attempts, an understanding of the roots of this success remains elusive. A deep network is trained by minimizing a non-convex loss function of its parameters, typically using stochastic gradient descent (SGD), with a variety of regularization techniques.

If the parameters (weights) of a neural network are given by $x \in \mathbb{R}^N$, training involves solving an optimization problem

$$x^* = \arg\min_x \ f(x), \tag{1}$$

[*]Equal contribution [1]Computer Science Department, UCLA. [2]Department of Mathematics and Statistics, McGill University. [3]Department of Mathematics, UCLA. [4]CEREMADE, Université Paris IX Dauphine.

*Proceedings of the 34th International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. 

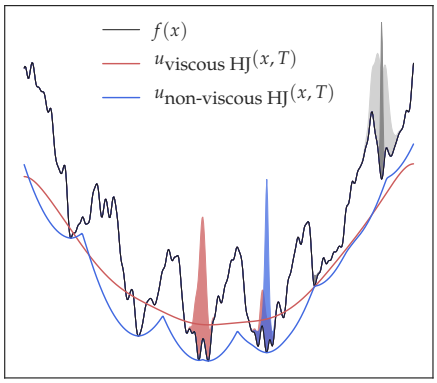



Figure 1: Smoothing of a loss function $f(x)$ by the viscous Hamilton-Jacobi equation (red) and the non-viscous Hamilton Jacobi equation (blue). An initial density (light gray) when evolved using SGD dynamics gets stuck in a local minimum (dark gray). If the viscous HJ equation is used to smooth the loss resulting in $u_{\text{viscous HJ}}(x,T)$, the terminal density is concentrated around the global minimum. Smoothing by the non-viscous HJ equation results in $u_{\text{non-viscous HJ}}(x,T)$ and has non-convex regions which slows down SGD resulting in a slightly sub-optimal solution. However, the critical points for the non-viscous HJ smoothing are unchanged in both value and location and $u_{\text{non-viscous HJ}}(x,T)$ can be interpreted as a generalized convex envelope of $f(x)$. This figure was produced using monotone finite differences (Oberman, 2006) with the Fokker-Planck equation (FP) to solve for the respective densities.

where $f(x)$ is the sum of the loss function, e.g., the cross-entropy loss, and possibly, a regularizer. This problem is challenging for deep networks because $f(x)$ is non-convex and $x$ is high-dimensional. Motivated from the statistical physics literature that studies the energy landscapes of discrete perceptrons (Baldassi et al., 2015), the authors in Chaudhari et al. (2016) optimize a modified loss function called "local entropy" defined as

$$f_\gamma(x) = -\log\left(G_\gamma * e^{-f(x)}\right), \tag{2}$$

where $G_\gamma(x) = (2\pi\gamma)^{-N/2}\exp\left(-\frac{|x|^2}{2\gamma}\right)$ is the Gaussian (heat) kernel. The modified loss $f_\gamma(x)$ is a smoother version of $f(x)$ obtained by convolution with a Gaussian and has been shown to perform well in practice. Moreover, it

can be shown analytically that local entropy for discrete perceptrons results in solutions that have low generalization error (Baldassi et al., 2016b). Such solutions belong to dense clusters, i.e., a large fraction of their neighboring configurations are solutions themselves. Similarly, for neural networks with continuous weights, these regions correspond to flat minima (Chaudhari et al., 2016).

Our first result in Section 3 shows that local entropy $f_\gamma(x) := u(x,\gamma)$ is the solution of the viscous Hamilton-Jacobi PDE

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2 + \frac{1}{2}\Delta u, \qquad \text{(viscous HJ)}$$

for $0 < t \le \gamma$ with the initial condition $u(x,0) = f(x)$. The Laplacian of $u(x,t)$ is defined as $\Delta u = \sum_{i=1}^n \frac{\partial^2}{\partial x_i^2} u$. In other words, $f_\gamma(x)$ that was designed in the physics literature to bias optimization towards flat regions is simply a way of smoothing the original loss using a PDE; this is surprising. We further exploit this connection in Section 5 to analyze the underlying stochastic optimal control problem (Fleming & Rishel, 2012) and show that minimizing local entropy leads to an improvement in the original loss $f(x)$ as compared to SGD.

Our second result proves that Elastic-SGD (Zhang et al., 2015) which is a popular algorithm for distributed optimization of deep networks is equivalent to local entropy under certain conditions. The former solves

$$\min_{x, x^1, \ldots, x^n} \sum_{a=1}^n f(x^a) + \frac{1}{2\gamma}|x^a - x|^2 \qquad (3)$$

where $x^1, \ldots, x^n$ are $n$ copies of the parameters that are distributed among $n$ workers and $x$ is a reference copy, akin to a master. To show this, we start with the continuous-time SGD dynamics given by

$$dx(t) = -\nabla f(x)\, dt + dW(t) \qquad \text{(SGD)}$$

for $t \ge 0$ where $W(t) \in \mathbb{R}^N$ is the $N$-dimensional Wiener process. The algorithm named Entropy-SGD introduced by Chaudhari et al. (2016) to minimize $f_\gamma(x)$ is a Markov chain Monte Carlo (MCMC) algorithm that can be written in continuous-time as the following system of stochastic differential equations (SDEs)

$$dy(s) = -\frac{1}{\varepsilon}\left(\frac{y-x}{\gamma} + \nabla f(y)\right) ds + \frac{1}{\sqrt{\varepsilon}}\, dW(s) \qquad (4)$$

$$dx(s) = -\gamma^{-1}(x-y)\, ds, \qquad (5)$$

where $y(s) \in \mathbb{R}^N$ and $\varepsilon$ is a time scale. As $\varepsilon \to 0$, the $y(s)$ dynamics becomes very fast and using the technique of homogenization of SDEs (Pavliotis & Stuart, 2008), we show that the gradient descent dynamics for (7) minimizes $f_\gamma(x)$ and is equivalent to the above SDEs if ergodicity holds, i.e., when $\nabla^2 f(x) + \gamma^{-1} I \succ 0$. While Elastic-SGD averages the $x^a$ variables "spatially" across workers, Entropy-SGD averages the $y(s)$ variable in time. This connection is novel and powerful because (i) it leads to a deeper understanding of the loss that Elastic-SGD minimizes, and (ii) it shows the equivalence of a completely distributed algorithm to a seemingly-different non-distributed algorithm. This connection, under some approximations, has also been suggested by Baldassi et al. (2016a) using replica theory.

Our next result discusses applications of new PDEs for non-convex optimization. We observe that our results are independent of the viscosity of the PDE (coefficient of the Laplacian in (viscous HJ)) and exploit this to study the non-viscous HJ equation

$$\frac{\partial u}{\partial t} = -\frac{1}{2}|\nabla u|^2. \qquad \text{(non-viscous HJ)}$$

This effectively sets the thermal noise in the MCMC updates of Chaudhari et al. (2016) to zero and results in a much cleaner set of update equations using the Hopf-Lax formula (HL). We can further simplify these updates and show that smoothing using the HJ equation is equivalent to the proximal point iteration

$$x_{k+1} = \text{prox}_{tf}(x_k). \qquad (6)$$

Note that computing the proximal operator for a general non-convex function $f(x)$ is challenging, the MCMC updates of Entropy-SGD or the non-viscous HJ equation updates (6) can indeed be interpreted as computing the proximal operator approximately using (stochastic) gradient descent. We demonstrate that this new algorithm is faster than Entropy-SGD in practice for deep networks.

## 2. Background

For a prototypical $K$-class classification problem on a dataset $\{(\xi_i, y_i)\}_{i=1}^M$ where $\xi_i$ are samples and $y_i \in \{1, \ldots, K\}$ are ground truth labels, we wish to minimize

$$f(x) := \frac{1}{M}\sum_{i=1}^M f_i(x),$$

where $f_i(x)$ is the loss on the $i^{\text{th}}$ sample. For instance, the cross-entropy loss is given by

$$f_i(x) = -\sum_{k=1}^K \mathbf{1}_{\{y_i = k\}} \log \widehat{y}_k(\xi_i; x),$$

where $\widehat{y}(\xi_i; x) \in \mathbb{R}^K$ is the softmax output of the network for sample $\xi_i$ and weights $x \in \mathbb{R}^N$. Let us emphasize that **the loss function $f(x)$ in deep learning is a non-convex function of its argument $x$.**

### 2.1. Stochastic gradient descent (SGD)

Computing the entire gradient $\nabla f(x)$ is prohibitive for large $M$. Stochastic gradient descent (Robbins & Monro, 1951) avoids this and performs the updates

$$x_{k+1} = x_k - \eta_k \, \nabla f_{i_k}(x_k), \tag{7}$$

where $\eta_k > 0$ for $k \in \mathbb{N}$ is the learning rate and the example $i_k \in \{1, \dots, M\}$ is sampled uniformly randomly. We will overload the notation to denote a mini-batch of $m$ examples by $i_k$ itself. The gradient obtained by back-propagation on such a mini-batch is a stochastic variable, denoted by $\nabla f_{\text{mb}}(x)$, and it is a common to assume that it is unbiased with bounded variance, i.e.,

$$\mathbb{E}\left[\nabla f_{\text{mb}}(x)\right] = \nabla f(x),$$
$$\mathbb{E}\left[\left|\nabla f_{\text{mb}}(x) - \nabla f(x)\right|^2\right] \leq \beta_{\text{mb}}^{-1},$$

for all $x \in \mathbb{R}^N$ for some $\beta_{\text{mb}}^{-1} \geq 0$. The discrete-time dynamics in (7) can then be modeled by the SDE (8)

$$dx(t) = -\nabla f(x(t)) \, dt + dW(t). \tag{8}$$

If the initial condition $x(0)$ is sampled from some density $\rho_0(x)$, the Fokker-Planck equation (Risken, 1984) gives the evolution of the unnormalized density $\rho(x,t)$ for SGD:

$$\frac{\partial}{\partial t} \rho(x,t) = \nabla \cdot \Big( \nabla f(x) \, \rho(x,t) \Big) + \frac{1}{2} \Delta \, \rho(x,t). \tag{FP}$$

This is the equation used to plot the dark grey density in Fig. 1. With mild assumptions on $f(x)$, and even if $f$ is non-convex, $\rho(x,t)$ converges to the unique stationary solution of (FP) as $t \to \infty$ (Pavliotis, 2014, Section 4.5). This limit is the Gibbs distribution

$$\rho^\infty(x) = Z^{-1} e^{-f(x)}, \tag{9}$$

where $Z$ is a normalizing constant. If there is a multiplier of $\beta$ in the exponent (also known as the inverse temperature in physics), we can see from (9) that as $\beta \to \infty$, the Gibbs distribution concentrates on the global minimizers of $f(x)$.

**Remark 1 (Metastability).** While the distribution $\rho^\infty$ is still the unique stationary solution of (FP), convergence to it can take an exponentially long time for a non-convex $f(x)$. Such a dynamics exhibit metastability, i.e., there can be multiple quasi-stationary measures on large time scales. For example, for a one-dimensional double-well potential, Kramer's law of diffusion states that this time is inversely proportional to the spectral gap, i.e., the smallest non-zero eigenvalue of the Hessian $\nabla^2 f(x)$, and proportional to $\exp\left(\beta \left(\sup_x f(x) - \inf_x f(x)\right)\right)$. This also holds in high dimensions (Glasstone et al., 1941; Bovier & den Hollander, 2006). Simulated annealing (Kushner, 1987; Chiang et al., 1987) is a popular technique to accelerate convergence by modulating the inverse temperature $\beta$.

## 3. PDE interpretation of local entropy

It is a classical result (Evans, 1998) that for the heat equation $u_t = \frac{1}{2}\Delta u$ with the initial condition $u(x,0) = f(x)$, the solution $u(x,t)$ is given by

$$u(x,t) = G_t * f(x). \tag{10}$$

In other words, the heat equation results in a convolution of the original function $f(x)$ by a Gaussian kernel whose variance increases with time $t$. The Hopf-Cole transformation (Evans, 1998, Section 4.4.1) is another classical tool that relates the solutions of the heat equation to those of the Hamilton-Jacobi equation (viscous HJ). We restate this connection in the lemma below to get our first result.

**Lemma 2 (Hopf-Cole transformation).** *The local entropy $f_\gamma(x)$ defined by (2) is the solution at time $t = \gamma$ of the initial value problem for the viscous Hamilton-Jacobi equation (viscous HJ) with initial condition $u(x,0) = f(x)$.*

*Proof.* Define $u(x,t) = -\log v(x,t)$. From (2), $v = e^{-u}$ solves the heat equation $v_t = \frac{1}{2}\Delta v$ with initial data $v(x,0) = \exp(-f(x))$. Taking partial derivatives gives

$$v_t = -v \, u_t, \quad \nabla v = -v \, \nabla u, \quad \Delta v = -v \, \Delta u + v \, |\nabla u|^2,$$

and one obtains (viscous HJ) by combining these expressions. ∎

Taking the gradient of (2) gives the following lemma.

**Lemma 3 (Gradient of local entropy).** *The gradient of local entropy $\nabla f_\gamma(x) = \nabla u(x,\gamma)$ is given by*

$$\nabla u(x,t) = \int_{\mathbb{R}^n} \frac{x-y}{\gamma} \, \rho_1^\infty(dy;\, x) \tag{11}$$

*where*

$$\rho_1^\infty(y;\, x) = Z_1^{-1} \exp\left( -f(y) - \frac{1}{2t} |x-y|^2 \right) \tag{12}$$

*and $Z_1$ is a normalization constant.*

**Remark 4 (Alternative gradient).** The convolution in (2) in two different ways can be written using either $G_\gamma$ as the kernel or $\exp(-f(x))$ as the kernel. The former gives Lemma 3 while the latter gives

$$\nabla u(x,t) = \int_{\mathbb{R}^n} \nabla f(x-y) \, \rho_2^\infty(dy;\, x) \tag{13}$$

where

$$\rho_2^\infty(y;x) = Z_2^{-1} \exp\left( -f(x-y) - \frac{1}{2t} |y|^2 \right). \tag{14}$$

We will use this observation in two ways, (i) to show that local entropy is a more powerful way of smoothing than gradient averaging, and (ii) to derive a simpler update rule for the Entropy-SGD algorithm in Section 6.

### 3.1. Non-viscous Hamilton-Jacobi equation

In addition to the connection with (viscous HJ) provided by Lemma 2, we can also explore the non-viscous Hamilton-Jacobi equation. This corresponds to setting the viscosity, i.e., the cefficient of the Laplacian in (viscous HJ), to zero. This leads a simpler explicit formula for the gradient; the resultant deterministic dynamic is indeed a special case of the stochastic dynamics for the viscous-HJ equation.

In the following lemma, we apply the well-known Hopf-Lax formula (Evans, 1998) for the solution of the HJ equation. It is also called the inf-convolution of the functions $f(x)$ and $\frac{1}{2t}|x|^2$ (Cannarsa & Sinestrari, 2004) and is closely related to the proximal operator (Moreau, 1965; Rockafellar, 1976).

**Lemma 5 (Gradient of Hopf-Lax).** *If $u(x,t)$ is the viscosity solution of (HJ) with $u(x,0) = f(x)$,*

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

*If the proximal operator*

$$y^* := \text{prox}_{tf}(x) = \arg\min_y \left\{ f(y) + \frac{1}{2t}\,|x-y|^2 \right\} \tag{15}$$

*is a singleton, $\nabla_x\, u(x,t)$ exists, and*

$$\nabla_x\, u(x,t) = \frac{x-y^*}{t} = \nabla f(y^*), \tag{16}$$

The proof is a direct application of Danskin's theorem (Bertsekas et al., 2003, Prop. 4.5.1) which allows us to differentiate through the inf operation in (HL).

**Remark 6 (Dynamics for HJ).** The previous lemma gives that $p^* := \nabla_x\, u(x,t)$ is the solution of

$$p = \nabla f(x - t\ p),$$

and it exists if $\nabla^2\ f(y) + t^{-1}\ I \succ 0$ near $y = x - tp^*$. It can be obtained by a fixed point iteration

$$p^{k+1} = \nabla f(x - tp^k) \tag{17}$$

which converges if $t\left|\nabla^2 f(y)\right| < 1$ near $y = x - tp^*$.

**Remark 7 (Proximal point iteration).** We can write the discrete-time gradient descent dynamics using $\nabla_x\ u(x,t)$ from (16) as $x_{k+1} = x_k - \eta\ t^{-1}\ \left(x_k - \text{prox}_{tf}(x_k)\right)$ where $\eta > 0$ is the step-size. For $\eta = t$, we therefore have the proximal point iteration given by

$$x_{k+1} = \text{prox}_{tf}(x_k).$$

Let us compare this to the standard proximal gradient descent update $x_{k+1} = \text{prox}_{th}(x_k - t\nabla g(x_k))$ where the loss function $f(x)$ is a sum of two terms: typically, a convex function $g(x)$ and possibly non-differentiable regularizer $h(x)$. Our update corresponds to setting $g(x) = 0$.

For a high-dimensional, non-convex function $f(x)$, computing the proximal operator in (15) is hard. Nevertheless, the fixed point iteration (17) allows us to compute the proximal operator approximately. Note that the convergence rate of approximate proximal gradient descent matches that of gradient descent under certain technical conditions (Schmidt et al., 2011).

**Remark 8 (Implicit gradient descent).** The proximal operator is equivalent to implicit gradient descent, also known as backward Euler iteration. While gradient descent updates $x_{k+1} = x_k - \eta\ \nabla f(x_k)$, the non-viscous HJ equation via (6) leads to

$$x_{k+1} = x_k - \eta\ \nabla f(x_{k+1}).$$

We note that the above backwards Euler method may have many solutions for $x_{k+1}$. Ideally, we wish to choose the one that gives the minimum in (HL) for $x = x_k$.

## 4. Derivation via homogenization

This section employs homogenization of SDEs (Pavliotis & Stuart, 2008, Chap. 10, 17), which is a technique used to analyze dynamical systems with multiple time-scales that have a few fast variables that may be coupled with other variables which evolve slowly. Computing averages over the fast variables allows us to obtain averaged equations for the slow variables in the limit that the times scales separate. This will allow us to give a rigorous mathematical treatment to the development of Chaudhari et al. (2016) and Lemma 3. We can also show that an algorithm called Elastic-SGD (Zhang et al., 2015) that was designed for distributed training of deep networks is equivalent to local entropy under certain conditions.

Consider the following system of SDEs

$$\begin{aligned} dx(s) &= h(x,\ y)\ ds \\ dy(s) &= \frac{1}{\varepsilon}\ g(x,\ y)\ ds + \frac{1}{\sqrt{\varepsilon}}\ dW(s); \end{aligned} \tag{18}$$

where $h, g$ are sufficiently smooth functions, $W(s) \in \mathbb{R}^n$ is the standard Wiener process. The parameter $\varepsilon > 0$ is the homogenization parameter and introduces a fast time-scale for the dynamics of $y(s)$. If the unique invariant measure $\rho^\infty(y;\ x)$ exists for a fixed $x$ and is ergodic, in the limit $\varepsilon \to 0$, the dynamics of $x(s)$ in (18) converges in distribution to

$$dX(s) = \overline{h}(X)\ ds$$

where the homogenized vector field for $X$ is defined as

$$\overline{h}(X) = \int h(X,y)\, \rho^{\infty}(dy;\, X);$$

in other words, it is the average against the invariant measure.

**Theorem 9 (Entropy-SGD via homogenization).** *Consider the gradient*

$$\nabla f_\gamma(x) = \gamma^{-1} \int_{\mathbb{R}^N} (x-y)\, \rho_1^{\infty}(dy;\, x)$$

*from Lemma 3. The gradient descent dynamics for $x(s)$ using this gradient is the homogenized version of*

$$\begin{aligned} dx(s) &= -\gamma^{-1}\,(x-y)\,ds \\ dy(s) &= -\frac{1}{\varepsilon}\left[\nabla f(y) + \frac{1}{\gamma}\,(y-x)\right] ds + \frac{1}{\sqrt{\varepsilon}}\, dW(s). \end{aligned} \tag{19}$$

*if $f(y) + \frac{1}{2\gamma}|x-y|^2$ is strictly convex, i.e., if $\nabla^2 f(y) + \gamma^{-1} I \succ 0$ for all $y \in \mathbb{R}^N$, the invariant measure of $y(s)$ is ergodic.*

The proof of this theorem is immediate from our definition of homogenization above.

**Remark 10 (Time average vs. spatial average).** Since the above analysis assumes that $\rho^{\infty}(y;\, x)$ is ergodic, we can also write the homogenized vector field equivalently as

$$\overline{h}(X) = \lim_{T\to\infty} \frac{1}{T} \int_0^T h(x,\, y(s))\, ds.$$

**Remark 11 (Alternative SDE).** Using Remark 4, we can also write the Entropy-SGD dynamics as the system

$$\begin{aligned} dx(s) &= -\gamma^{-1}\,\nabla f(x-y)\,ds \\ dy(s) &= \frac{1}{\varepsilon}\Big(\nabla f(x-y) - \gamma^{-1}\, y\Big)\, ds + \frac{1}{\sqrt{\varepsilon}}\, dW(s); \end{aligned} \tag{20}$$

this is equivalent to the system in (19).

### 4.1. Elastic-SGD as local entropy

Consider the loss function for Elastic-SGD (3), the continuous-time stochastic gradient descent dynamics for this can be given by:

$$\begin{aligned} dx^a(s) &= -\frac{1}{\varepsilon}\left[\nabla f(x^a)\, ds - \frac{(x^a - x)}{\gamma}\right] ds + \frac{1}{\sqrt{\varepsilon}} dW^a(s) \\ dx(s) &= -\gamma^{-1} \sum_{a=1}^{n} (x - x^a)\, ds; \end{aligned}$$

for $a \le n$. We now scale the time by $\varepsilon$ for the dynamics of the workers $x^a$ and by $n$ for the dynamics of $x(s)$. If each of them has an ergodic invariant measure $\rho^{\infty}(x^a;\, x)$, the dynamics of $x(s)$ only sees the homogenized vector field given by

$$\begin{aligned} \overline{h}(X) &= \frac{1}{n\,\gamma} \int \left(\sum_{a=1}^{n} (x^a - X)\right) \prod_{a=1}^{n} \rho^{\infty}(dx^a;\, X) \\ &= -\gamma^{-1}\, X + \frac{1}{n\,\gamma} \lim_{T\to\infty} \frac{1}{T} \int_0^T \sum_{a=1}^{n} x^a(s)\, ds \\ &= -\gamma^{-1} X + \gamma^{-1} \lim_{T\to\infty} \frac{1}{T} \int_0^T x^1(s)\, ds \\ &= \gamma^{-1} \int (x^1 - X)\, \rho^{\infty}(dx^1;\, X), \end{aligned}$$

which is exactly the homogenized dynamics of the Entropy-SGD algorithm from Theorem 9. In the above derivation, the first step is written using the average over the product measure of the workers conditional upon a fixed $X$, the second step follows using ergodicity of each worker and the third step follows because the workers are identically distributed.

This proves that the Elastic-SGD algorithm is equivalent to minimizing local entropy under ergodicity conditions.

### 4.2. Heat equation vs. the viscous Hamilton-Jacobi equation

We defined local entropy as the convolution of the exponentiated loss function (2). Instead, one can also study

$$f_\gamma^2(x) = G_\gamma * f(x) \tag{21}$$

which is the solution of the heat equation (10). Since the gradient $\nabla f_\gamma^2(x)$ corresponds to averaging the original gradient $\nabla f(x)$ over Gaussian perturbations of variance $\gamma$, we can write the gradient descent dynamics for $f_\gamma^2(x)$ as

$$\begin{aligned} dx(s) &= -\nabla f(x-y)\, ds \\ dy(s) &= -\frac{1}{\varepsilon\gamma}\, y\, ds + \frac{1}{\sqrt{\varepsilon}}\, dW(s). \end{aligned} \tag{22}$$

Note that the dynamics for $y(s)$ does not depend on $x(s)$ at all, this is in contrast to the system (20) for Entropy-SGD. This extra gradient term $\nabla f(x-y)$ in (20) is exactly the difference between the smoothing performed by local entropy and the smoothing performed by heat equation. The latter is common in the deep learning literature under different forms, e.g., Gulcehre et al. (2016) and Mobahi (2016). The former version however has much better empirical performance (cf. experiments in Section 7).

## 5. Stochastic control interpretation

The interpretation of local entropy $f_\gamma(x)$ as the solution of a viscous Hamiltonian-Jacobi equation provided

by Lemma 2 allows us to interpret gradient descent as an optimal control problem. While the interpretation does not have immediate algorithmic implications, we can prove that the expected value of a minimum is improved by using local entropy as compared to SGD.

Consider the following controlled SDE

$$dx(s) = -\nabla f(x)\, ds - \alpha(s)\, ds + dW(s) \qquad \text{(CSGD)}$$

where $t \le s \le T$, $x(t) = x$ and $\alpha(\cdot)$ is the control. Let us define a stochastic optimal control problem that minimizes the functional for a given solution $x(\cdot)$ of (CSGD) corresponding to some control $\alpha(\cdot)$,

$$\mathscr{C}(x(\cdot),\, \alpha(\cdot)) = \mathbb{E}\left[V(x(T)) + \frac{1}{2}\int_0^T |\alpha(s)|^2\, ds\right]. \quad (23)$$

Here the terminal cost is a given function $V : \mathbb{R}^N \to \mathbb{R}$ and we use the prototypical quadratic running cost. If the value function

$$u(x,t) = \min_{\alpha(\cdot)} \mathscr{C}(x(\cdot),\, \alpha(\cdot)).$$

is the minimum expected cost over admissible controls and over paths which start at $x(t) = x$, it is known that $u(x,t)$ is the unique viscosity solution of a Hamilton-Jacobi-Bellman (HJB) PDE (Fleming & Rishel, 2012)

$$-u_t(x,t) = -\nabla f(x)\cdot\nabla u(x,t) - \frac{1}{2}|\nabla u|^2 + \frac{1}{2}\Delta u \qquad \text{(HJB)}$$

for $t \le s \le T$ along with the terminal condition $u(x,T) = V(x)$. Note that (HJB) is written as a backward-time equation. We make note that in this case, the optimal control is equal to the gradient of the solution

$$\alpha^*(x,t) = \nabla u(x,t). \qquad (24)$$

### 5.1. Improvement in the value function

We first show that the value function obtained by the dynamics (CSGD) using the optimal control (24) improves as compared to SGD.

**Theorem 12.** *Let $x_{\text{csgd}}(s)$ and $x_{\text{sgd}}(s)$ be solutions of (CSGD) and (8), respectively, with the same initial data $x_{\text{csgd}}(0) = x_{\text{sgd}}(0) = x_0$. For a fixed time $t \ge 0$, we have*

$$\mathbb{E}\left[V(x_{\text{csgd}}(t))\right] \le \mathbb{E}\left[V(x_{\text{sgd}}(t))\right] - \frac{1}{2}\,\mathbb{E}\left[\int_0^t \left|\alpha^*(x_{\text{csgd}}(s),s)\right|^2 ds\right].$$

*Proof.* The value function $v(x,t) = \mathbb{E}[V(x(t))]$ for SGD where the expectation is taken over paths of (8), satisfies

$$v_t = -\nabla f(x)\cdot\nabla v + \frac{1}{2}\Delta v.$$

Since $u(x,t)$ is the solution of (HJB), we have

$$u_t \le -\nabla f(x)\cdot\nabla u + \frac{1}{2}\Delta u - \frac{1}{2}|\nabla u|^2.$$

Note that $u(x,T) = v(x,T) = V(x)$. We now use the comparison principle (Evans, 1998) to conclude

$$u(x,s) \le v(x,s), \quad \text{for all } t \le s \le T; \qquad (25)$$

But since $u(x,t)$ is the value function of (23), we also have

$$u(x,t) = \mathbb{E}\left[V(x(t)) + \frac{1}{2}\int_0^t |\alpha^*|^2\, ds\right]$$

over paths of (CSGD) with $x(t) = x$, where $\alpha^*(\cdot)$ is the optimal control in (24). This completes the proof. ∎

**Remark 13.** The dynamics in (CSGD) has a control $\alpha(\cdot)$ in addition to the gradient term $\nabla f(x)$. This is therefore different from performing gradient descent on $f_\gamma(x)$ and instead amounts to minimizing $f(x) + f_\gamma(x)$, i.e., treating $f_\gamma(x)$ as a regularizer. The comparison principle in the proof of Theorem 12 requires the coefficient of $\nabla f(x)$ to be the same in (8) and (CSGD).

## 6. Algorithmic details

In this section, we compare and contrast the various algorithms in this paper and provide implementation details that are used for the empirical validation in Section 7.

### 6.1. Entropy-SGD

We perform an Euler-Maruyama discretization of the system in (19) and update the $y(s)$ dynamics $1/\varepsilon = L$ times before updating the $x(s)$ variable. This results in

$$\begin{aligned} y_{k+1} &= y_k - \eta'\left(\nabla f_{\text{mb}}(y_k) + \gamma^{-1}(y_k - x_k)\right) + \sqrt{\eta'\beta^{-1}}\, w_k \\ z_{k+1} &= \alpha\, z_k + (1-\alpha)\, y_{k+1} \\ x_{k+1} &= \begin{cases} x_k - \eta\,\gamma^{-1}(x_k - z_{k+1}) & \text{if } k/L \text{ is an integer,} \\ x_k & \text{else;} \end{cases} \end{aligned} \qquad \text{(Entropy-SGD)}$$

where $w_k$ are zero-mean, unit variance Gaussian random variables. We initialize $y_k = x_k$ every time $k/L$ is an integer.

In practice, we only have access to the noisy gradient for a deep network, which we have explicitly denoted as $\nabla f_{\text{mb}}$ above. The update in (Entropy-SGD) thus has two sources of noise, one from the mini-batches and an additional noise term $w_k$. For small values of $L$ we may be far from the ergodic limit, we therefore maintain the exponential average of the $y_k$ variables using $z_k$. This gives more weight to later iterations and is beneficial in practice. The above iterations are equivalent to the original MCMC algorithm of Chaudhari et al. (2016).

### 6.2. Non-viscous Hamilton-Jacobi (HJ)

Remark 6 and (17) result in an update equation for HJ of the form

$$\begin{aligned} p_{k+1} &= \alpha\ p_k + (1-\alpha)\ \nabla f_{\text{mb}}(x_k - \gamma\ p_k) \\ x_{k+1} &= \begin{cases} x_k - \eta\ p_k & \text{if } k/L \text{ is an integer,} \\ x_k & \text{else;} \end{cases} \end{aligned} \tag{HJ}$$

We initialize $p_k = 0$ every time $k/L$ is an integer. We have introduced a damping $\alpha > 0$ compared to the fixed point iteration (17) because the gradient $\nabla f_{\text{mb}}$ is stochastic.

**Remark 14.** The distinction between (Entropy-SGD) and (HJ) is subtle. We can see this by using Remark 4 to construct a system of updates (for $\beta^{-1} = 0$)

$$\begin{aligned} y_{k+1} &= \left(1 - \gamma^{-1}\ \eta'\right)\ y_k + \eta'\ \nabla f_{\text{mb}}(x_k - y_k) \\ x_{k+1} &= \begin{cases} x_k - \eta\ \nabla f_{\text{mb}}(x_k - y_k) & \text{if } k/L \text{ is an integer,} \\ x_k & \text{else.} \end{cases} \end{aligned}$$

It is easy to see that setting $y_k := \gamma\ p_k$ and $\alpha := (1 - \eta'/\gamma)$ gives the same system as (HJ) if $p_k$ is the fixed point in (17). Indeed, the non-viscous HJ equation is the limit of the viscous HJ equation as the viscosity goes to zero. It is however easy to see that (HJ) results in much simpler updates than (Entropy-SGD).

### 6.3. Heat equation

The gradient descent dynamics for the smoothing by the heat equation (21) is given by

$$x_{k+1} = x_k - \frac{\eta}{L} \sum_{i=1}^{L} \nabla f_{\text{mb}}(x_k + w^i) \tag{26}$$

where $w_i$ are Gaussian random variables with zero mean and variance $\gamma\ I$. We have implemented the convolution in (21) as an average over Gaussian perturbations of the parameters $x$.

### 6.4. Choosing hyper-parameters

As Fig. 1 shows, a larger $\gamma$ leads to a smoother loss (2). We can exploit this to introduce a technique called "scoping" which reduces the smoothing effect as training progresses. This enables quick progress with SGD in the beginning and preserves the locations of the minimizers towards the end as $\gamma \to 0$. We update $\gamma$ every time $k/L$ is an integer using

$$\gamma = \gamma_0\ (1 - 10^{-3})^{k/L};$$

we pick $\gamma_0 \in [10^4,\ 10]$ so as to obtain the best validation error. Other parameters in (Entropy-SGD), (HJ) and (26) are fixed to $\alpha = 0.75$ and $\eta' = 0.1$ while the learning rate schedule is the same as that of SGD with the number of epochs scaled by a factor of $L$.

## 7. Empirical validation

We now discuss experimental results on deep neural networks that demonstrate that the PDE methods considered in this article achieve good regularization, aid optimization and lead to improved classification on modern datasets.

### 7.1. Setup

We demonstrate experimental results for image classification on MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky, 2009) datasets. As is standard practice for these benchmark problems, to enable comparison of numerical values with previous literature, we use the test set for validation and report errors on it. We do not perform any preprocessing for the MNIST dataset. For CIFAR-10, we perform a global contrast normalization (Coates et al., 2010) followed by a ZCA whitening transform (Krizhevsky, 2009). We use the cross-entropy loss for all our experiments. We report mean and standard deviation of validation error over 6 independent runs for all algorithms. The mini-batch size is fixed to 128 for all experiments. We will use a well-tuned implementation of SGD as a baseline for comparison.

We run the following algorithms for each network:

- **Entropy-SGD:** described in Section 6.1,
- **HEAT:** smoothing by the heat equation (cf. Section 6.3),
- **HJ:** described in Section 6.2,
- **SGD:** described by (7),

**Remark 15 (Effective epochs).** As discussed in Section 6, $L$ is the number of gradient evaluations performed before each weight update. We use $L = 20$ for Entropy-SGD and $L = 5$ for the HJ and HEAT equation, $L$ is defined to be 1 for SGD. Note that each weight update of Entropy-SGD or HJ uses $L$ times more number of back-props (with different mini-batches) than SGD and is thus $L$ times slower. We therefore plot error curves against "effective epochs", i.e., the number of epochs multiplied by $L$, which is a direct measure of the wall-clock time.

### 7.2. MNIST

We use a standard LeNet for MNIST (LeCun et al., 1998) with batch-normalization (Ioffe & Szegedy, 2015) and dropout of probability 0.25 after every convolutional layer. This has two convolutional layers with 20 and 50 channel outputs respectively, and a fully-connected layer with 500 hidden units before softmax.

Table 1: Summary: Validation error (%) @ Effective epochs

| Model | Entropy-SGD | HEAT | HJ | SGD |
|---|---|---|---|---|
| LeNet | $0.5 \pm 0.01$ @ 80 | $0.59 \pm 0.02$ @ 75 | $\mathbf{0.5 \pm 0.01}$ @ **70** | $0.5 \pm 0.02$ @ 67 |
| All-CNN | $7.96 \pm 0.05$ @ 160 | $9.04 \pm 0.04$ @ 150 | $\mathbf{7.89 \pm 0.07}$ @ **145** | $7.94 \pm 0.06$ @ 195 |

The results for LeNet are described in Fig. 2a and Table 1. The final validation error is very similar for all algorithms at 0.50% with the exception of the heat equation which only reaches 0.59%. This is consistent with Section 4.2 which suggests that the viscous or non-viscous HJ equations result in better smoothing than the heat equation.

### 7.3. CIFAR-10

We use the All-CNN-C network of Springenberg et al. (2014) with batch-normalization after every convolutional layer. We match the hyper-parameters of the original authors and set dropout to 0.5 with a weight decay of $10^{-3}$.

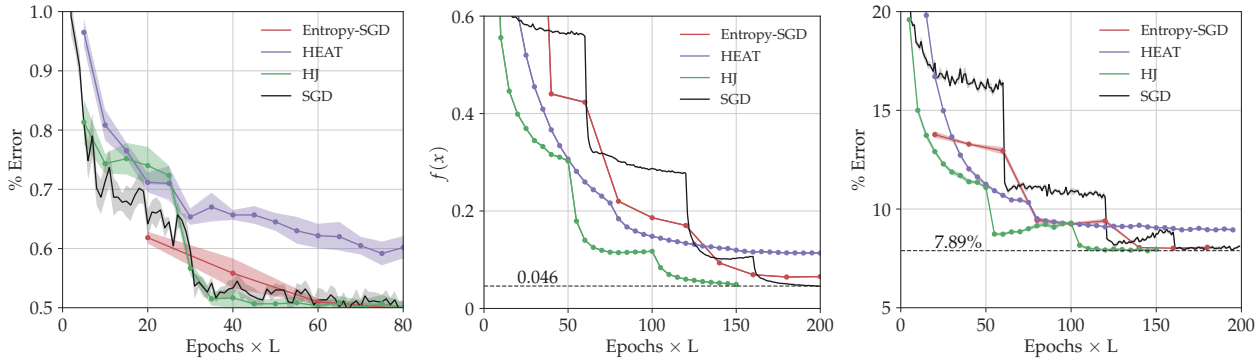
Figs. 2b and 2c show the training loss and validation error for the All-CNN network on the CIFAR-10 dataset. The Hamilton-Jacobi equation (HJ) obtains a validation error of 7.89% in 145 epochs and thus performs best among the algorithms tested here; it also has the lowest training cross-entropy loss of 0.046. Note that both HJ and Entropy-SGD converge faster than SGD. The heat equation again performs poorly on this dataset and has a much higher validation error than others (9.04%).

## 8. Discussion

Our results apply nonlinear PDEs, stochastic optimal control and stochastic homogenization to the analysis of empirically successful algorithms for deep networks.

The first practical implication is that by solving the stochastic control problem corresponding to the viscous Hamilton-Jacobi PDE, we achieve a smaller expected loss as compared to SGD. The improvement is an interpretable quantity, it is the integral of the control along the optimal path. The second is the equivalence of seemingly disparate methods such as Elastic-SGD and local entropy. This can help the development of distributed algorithms. Furthermore, our analysis provides insight into the choice of hyper-parameters for these methods which is largely a black-art in machine learning. For instance, the parameter $\gamma$ in local entropy and Elastic-SGD is equivalent to the time for the PDE flow. The third is that our analysis suggests more efficient algorithms, motivated by the fixed-point iteration for the gradient of the non-viscous HJ equation. We thus simplify Entropy-SGD and improve its performance in practice and disentangle key parts of the original algorithm such as scoping and thermal noise.

Conceptually, while simulated annealing and related methods work by modulating the level of noise in the dynamics, our algorithm works by modulating the smoothness of the underlying loss function. Moreover, while most algorithms used in deep learning derive their motivation from the literature on convex optimization, the algorithms we have presented here are specialized to non-convex loss functions and have been shown to perform well on these problems, both in theory and in practice.




(a) LeNet: Validation error (b) All-CNN: Training loss (c) All-CNN: Validation error

Figure 2: Performance on MNIST and CIFAR-10

## References


Baldassi, C., Borgs, C., Chayes, J., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *PNAS*, 113(48):E7655–E7662, 2016a.

Baldassi, C., Ingrosso, A., Lucibello, C., Saglietti, L., and Zecchina, R. Local entropy as a measure for sampling solutions in constraint satisfaction problems. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(2):023301, 2016b.

Baldassi, Carlo, Ingrosso, Alessandro, Lucibello, Carlo, Saglietti, Lucibello, and Zecchina, Riccardo. Subdominant dense clusters allow for simple learning and high computational performance in neural networks with discrete synapses. *Physical review letters*, 115(12):128101, 2015.

Bertsekas, Dimitri, Nedi, Angelia, Ozdaglar, Asuman, et al. Convex analysis and optimization. 2003.

Bovier, A. and den Hollander, F. Metastability: A potential theoretic approach. In *International Congress of Mathematicians*, volume 3, pp. 499–518, 2006.

Cannarsa, Piermarco and Sinestrari, Carlo. *Semiconcave functions, Hamilton-Jacobi equations, and optimal control*, volume 58. Springer Science & Business Media, 2004.

Chaudhari, Pratik, Choromanska, Anna, Soatto, Stefano, LeCun, Yann, Baldassi, Carlo, Borgs, Christian, Chayes, Jennifer, Sagun, Levent, and Zecchina, Riccardo. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *arXiv:1611.01838*, 2016.

Chiang, Tzuu-Shuh, Hwang, Chii-Ruey, and Sheu, Shuenn. Diffusion for global optimization in $\mathbb{R}^n$. *SIAM Journal on Control and Optimization*, 25(3):737–753, 1987.

Coates, Adam, Lee, Honglak, and Ng, Andrew Y. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001(48109):2, 2010.

Evans, Lawrence C. *Partial differential equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 1998. ISBN 0-8218-0772-2.

Fleming, Wendell H and Rishel, Raymond W. *Deterministic and stochastic optimal control*, volume 1. Springer Science & Business Media, 2012.

Glasstone, Samuel, Eyring, Henry, and Laidler, Keith J. *The theory of rate processes*. McGraw-Hill, 1941.

Gulcehre, Caglar, Moczulski, Marcin, Denil, Misha, and Bengio, Yoshua. Noisy activation functions. In *ICML*, 2016.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. Master's thesis, Computer Science, University of Toronto, 2009.

Kushner, Harold. Asymptotic global behavior for stochastic approximation and diffusions with slowly decreasing noise effects: global minimization via Monte Carlo. *SIAM Journal on Applied Mathematics*, 47(1):169–185, 1987.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

LeCun, Yann, Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.

Mobahi, Hossein. Training Recurrent Neural Networks by Diffusion. *arXiv:1601.04114*, 2016.

Moreau, Jean-Jacques. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965.

Oberman, Adam M. Convergent difference schemes for degenerate elliptic and parabolic equations: Hamilton-Jacobi equations and free boundary problems. *SIAM J. Numer. Anal.*, 44 (2):879–895 (electronic), 2006. ISSN 0036-1429.

Pavliotis, Grigorios A. *Stochastic processes and applications*. Springer, 2014.

Pavliotis, Grigorios A and Stuart, Andrew. *Multiscale methods: averaging and homogenization*. Springer Science & Business Media, 2008.

Risken, Hannes. Fokker-planck equation. In *The Fokker-Planck Equation*, pp. 63–95. Springer, 1984.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Rockafellar, R Tyrrell. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

Schmidt, Mark, Roux, Nicolas Le, and Bach, Francis. Convergence rates of inexact proximal-gradient methods for convex optimization. In *NIPS*, 2011.

Springenberg, J., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for simplicity: The all convolutional net. *arXiv:1412.6806*, 2014.

Zhang, S., Choromanska, A., and LeCun, Y. Deep learning with elastic averaging SGD. In *NIPS*, 2015.