Failures of Gradient-Based Deep Learning

Shai Shalev-Shwartz [1] **Ohad Shamir** [2] **Shaked Shammah** [1]

Abstract

In recent years, Deep Learning has become the go-to solution for a broad range of applications, often outperforming state-of-the-art. However, it is important, for both theoreticians and practitioners, to gain a deeper understanding of the difficulties and limitations associated with common approaches and algorithms. We describe four types of simple problems, for which the gradient-based algorithms commonly used in deep learning either fail or suffer from significant difficulties. We illustrate the failures through practical experiments, and provide theoretical insights explaining their source, and how they might be remedied.

1. Introduction

The success stories of deep learning form an ever lengthening list of practical breakthroughs and state-of-the-art performances, ranging the fields of computer vision (Krizhevsky et al., 2012; He et al., 2016; Schroff et al., 2015; Taigman et al., 2014), audio and natural language processing and generation (Collobert & Weston, 2008; Hinton et al., 2012; Graves et al., 2013; van den Oord et al., 2016), as well as robotics (Mnih et al., 2015; Schulman et al., 2015), to name just a few. The list of success stories can be matched and surpassed by a list of practical "tips and tricks", from different optimization algorithms, parameter tuning methods (Sutskever et al., 2013; Kingma & Ba, 2014), initialization schemes (Glorot & Bengio, 2010), architecture designs (Szegedy et al., 2016), loss functions, data augmentation (Krizhevsky et al., 2012) and so on.

The current theoretical understanding of deep learning is far from being sufficient for a rigorous analysis of the difficulties faced by practitioners. Progress must be made from both parties: from a practitioner's perspective, emphasizing the difficulties provides practical insights to the theoretician, which in turn, supplies theoretical insights and guarantees, further strengthening and sharpening practical intuitions and wisdom. In particular, understanding *failures* of existing algorithms is as important as understanding where they succeed.

Our goal in this paper is to present and discuss families of simple problems for which commonly used methods do not show as exceptional a performance as one might expect. We use empirical results and insights as a ground on which to build a theoretical analysis, characterising the sources of failure. Those understandings are aligned, and sometimes lead to, different approaches, either for an architecture, loss function, or an optimization scheme, and explain their superiority when applied to members of those families. Interestingly, the sources for failure in our experiment do not seem to relate to stationary point issues such as spurious local minima or a plethora of saddle points, a topic of much recent interest (e.g. (Dauphin et al., 2014; Choromanska et al., 2015)), but rather to more subtle issues, having to do with informativeness of the gradients, signal-to-noise ratios, conditioning etc. The code for running all our experiments is available online[1]. In this version, due to the lack of space, we focus on two families of failures, and briefly describe two others in Section 4. We refer the reader to (Shalev-Shwartz et al., 2017) for an extended version of this paper.

We start off in Section 2 by discussing a class of simple learning problems for which the gradient information, central to deep learning algorithms, provably carries negligible information on the target function which we attempt to learn. This result is a property of the learning problems themselves, and holds for any specific network architecture one may choose for tackling the learning problem, implying that no gradient-based method is likely to succeed. Our analysis relies on tools and insights from the Statistical Queries literature, and underscores one of the main deficiencies of Deep Learning: its reliance on local properties of the loss function, with the objective being of a global nature.

[1]School of Computer Science and Engineering, The Hebrew University [2]Weizmann Institute of Science. Correspondence to: Shai Shalev-Shwartz <shais@cs.huji.ac.il>, Ohad Shamir <ohad.shamir@weizmann.ac.il>, Shaked Shammah <shaked.shammah@mail.huji.ac.il>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

[1] `https://github.com/shakedshammah/failures_of_DL`. See command lines in Appendix D.

Next, in Section 3, we tackle the ongoing dispute between two common approaches to learning. Most, if not all, learning and optimization problems can be viewed as some structured set of sub-problems. The first approach, which we refer to as the "end-to-end" approach, will tend to solve all of the sub-problems together in one shot, by optimizing a single primary objective. The second approach, which we refer to as the "decomposition" one, will tend to handle these sub-problems separately, solving each one by defining and optimizing additional objectives, and not rely solely on the primary objective. The benefits of the end-to-end approach, both in terms of requiring a smaller amount of labeling and prior knowledge, and perhaps enabling more expressive architectures, cannot be ignored. On the other hand, intuitively and empirically, the extra supervision injected through decomposition is helpful in the optimization process. We experiment with a simple problem in which application of the two approaches is possible, and the distinction between them is clear and intuitive. We observe that an end-to-end approach can be much slower than a decomposition method, to the extent that, as the scale of the problem grows, no progress is observed. We analyze this gap by showing, theoretically and empirically, that the gradients are much more noisy and less informative with the end-to-end approach, as opposed to the decomposition approach, explaining the disparity in practical performance.

2. Parities and Linear-Periodic Functions

Most existing deep learning algorithms are gradient-based methods; namely, algorithms which optimize an objective through access to its gradient w.r.t. some weight vector $\mathbf{w}$, or estimates of the gradient. We consider a setting where the goal of this optimization process is to learn some underlying hypothesis class $\mathcal{H}$, of which one member, $h \in \mathcal{H}$, is responsible for labelling the data. This yields an optimization problem of the form

$$\min_{\mathbf{w}} F_h(\mathbf{w}).$$

The underlying assumption is that the gradient of the objective w.r.t. $\mathbf{w}$, $\nabla F_h(\mathbf{w})$, contains useful information regarding the target function h, and will help us make progress.

Below, we discuss a family of problems for which with high probability, at any fixed point, the gradient, $\nabla F_h(\mathbf{w})$, will be essentially the same regardless of the underlying target function h. Furthermore, we prove that this holds independently of the choice of architecture or parametrization, and using a deeper/wider network will not help. The family we study is that of compositions of linear and periodic functions, and we experiment with the classical problem of learning parities. Our empirical and theoretical study shows that indeed, if there's little information in the gradient, using it for learning cannot succeed.

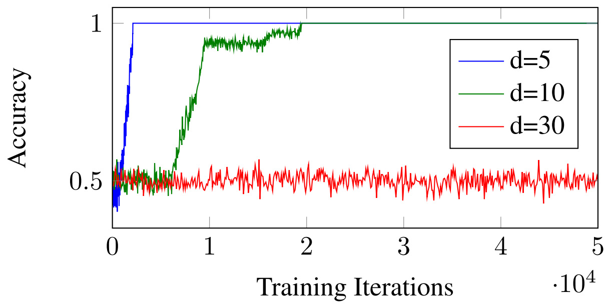


Figure 1. Parity Experiment: Accuracy as a function of the number of training iterations, for various input dimensions.

2.1. Experiment

We begin with the simple problem of learning random parities: After choosing some $\mathbf{v}^* \in \{0,1\}^d$ uniformly at random, our goal is to train a predictor mapping $\mathbf{x} \in \{0,1\}^d$ to $y = (-1)^{\langle \mathbf{x}, \mathbf{v}^* \rangle}$, where $\mathbf{x}$ is uniformly distributed. In words, y indicates whether the number of 1's in a certain subset of coordinates of $\mathbf{x}$ (indicated by $\mathbf{v}^*$) is odd or even.

For our experiments, we use the hinge loss, and a simple network architecture of one fully connected layer of width $10d > \frac{3d}{2}$ with ReLU activations, and a fully connected output layer with linear activation and a single unit. Note that this class realizes the parity function corresponding to any $\mathbf{v}^*$ (see Lemma 3 in the appendix).

Empirically, as the dimension d increases, so does the difficulty of learning, which can be measured in the accuracy we arrive at after a fixed number of training iterations, to the point where around $d = 30$, no advance beyond random performance is observed after reasonable time. Figure 1 illustrates the results.

2.2. Analysis

To formally explain the failure from a geometric perspective, consider the stochastic optimization problem associated with learning a target function h,

$$\min_{\mathbf{w}} F_h(\mathbf{w}) := \mathbb{E}_{\mathbf{x}} \left[\ell(p_{\mathbf{w}}(\mathbf{x}), h(\mathbf{x})) \right], \quad (1)$$

where ℓ is a loss function, $\mathbf{x}$ are the stochastic inputs (assumed to be vectors in Euclidean space), and $p_{\mathbf{w}}$ is some predictor parametrized by a parameter vector $\mathbf{w}$ (e.g. a neural network of a certain architecture). We will assume that F is differentiable. A key quantity we will be interested in studying is the *variance* of the gradient of F with respect to h, when h is drawn uniformly at random from a collection of candidate target functions $\mathcal{H}$:

$$\mathrm{Var}(\mathcal{H}, F, \mathbf{w}) = \mathop{\mathbb{E}}_{h} \left\| \nabla F_h(\mathbf{w}) - \mathop{\mathbb{E}}_{h'} \nabla F_{h'}(\mathbf{w}) \right\|^2 \qquad (2)$$

Intuitively, this measures the expected amount of "signal" about the underlying target function contained in the gradient. As we will see later, this variance correlates with the difficulty of solving (1) using gradient-based methods[2].

The following theorem bounds this variance term.

Theorem 1 *Suppose that*

- *$\mathcal{H}$ consists of real-valued functions h satisfying $\mathbb{E}_{\mathbf{x}}[h^2(\mathbf{x})] \le 1$, such that for any two distinct $h, h' \in \mathcal{H}$, $\mathbb{E}_{\mathbf{x}}[h(\mathbf{x})h'(\mathbf{x})] = 0$.*
- *$p_{\mathbf{w}}(\mathbf{x})$ is differentiable w.r.t. $\mathbf{w}$, and satisfies $\mathbb{E}_{\mathbf{x}}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x})\|^2\right] \le G(\mathbf{w})^2$ for some scalar $G(\mathbf{w})$.*
- *The loss function ℓ in (1) is either the square loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ or a classification loss of the form $\ell(\hat{y}, y) = r(\hat{y} \cdot y)$ for some 1-Lipschitz function r, and the target function h takes values in $\{\pm 1\}$.*

Then

$$Var(\mathcal{H}, F, \mathbf{w}) \le \frac{G(\mathbf{w})^2}{|\mathcal{H}|}.$$

The proof is given in Appendix B.1. The theorem implies that if we try to learn an unknown target function, possibly coming from a large collection of uncorrelated functions, then the sensitivity of the gradient to the target function at any point decreases linearly with $|\mathcal{H}|$.

Before we make a more general statement, let us return to the case of parities, and study it through the lens of this framework. Suppose that our target function is some parity function chosen uniformly at random, i.e. a random element from the set of 2^d functions $\mathcal{H} = \{\mathbf{x} \mapsto (-1)^{\langle \mathbf{x}, \mathbf{v}^* \rangle} : \mathbf{v}^* \in \{0,1\}^d\}$. These are binary functions, which are easily seen to be mutually orthogonal: Indeed, for any $\mathbf{v}, \mathbf{v}'$,

$$\mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x}, \mathbf{v} \rangle}(-1)^{\langle \mathbf{x}, \mathbf{v}' \rangle}\right] = \mathop{\mathbb{E}}_{\mathbf{x}}\left[(-1)^{\langle \mathbf{x}, \mathbf{v}+\mathbf{v}' \rangle}\right]$$
$$= \prod_{i=1}^{d} \mathbb{E}\left[(-1)^{x_i(v_i+v_i')}\right] = \prod_{i=1}^{d} \frac{(-1)^{v_i+v_i'} + (-1)^{-(v_i+v_i')}}{2}$$

which is non-zero if and only if $\mathbf{v} = \mathbf{v}'$. Therefore, by Theorem 1, we get that $\mathrm{Var}(\mathcal{H}, F, \mathbf{w}) \le G(\mathbf{w})^2/2^d$ – that is, exponentially small in the dimension d. By Chebyshev's inequality, this implies that the gradient at any point $\mathbf{w}$ will be extremely concentrated around a fixed point independent of h.

This phenomenon of exponentially-small variance can also be observed for other distributions, and learning problems other than parities. Indeed, in (Shamir, 2016), it was shown that this also holds in a more general setup, when the output y corresponds to a linear function composed with a periodic one, and the input $\mathbf{x}$ is sampled from a smooth distribution:

Theorem 2 (Shamir 2016) *Let ψ be a fixed periodic function, and let $\mathcal{H} = \{\mathbf{x} \mapsto \psi(\mathbf{v}^{*\top}\mathbf{x}) : \|\mathbf{v}^*\| = r\}$ for some $r > 0$. Suppose $\mathbf{x} \in \mathbb{R}^d$ is sampled from an arbitrary mixture of distributions with the following property: The square root of the density function φ has a Fourier transform $\hat{\varphi}$ satisfying $\frac{\int_{\mathbf{x}:\|\mathbf{x}\|>r} \hat{\varphi}^2(\mathbf{x})d\mathbf{x}}{\int_{\mathbf{x}} \hat{\varphi}^2(\mathbf{x})d\mathbf{x}} \le \exp(-\Omega(r))$. Then if F denotes the objective function with respect to the squared loss,*

$$Var(\mathcal{H}, F, \mathbf{w}) \le O\left(\exp(-\Omega(d)) + \exp(-\Omega(r))\right).$$

The condition on the Fourier transform of the density is generally satisfied for smooth distributions (e.g. arbitrary Gaussians whose covariance matrices are positive definite, with all eigenvalues at least $\Omega(1/r)$). Thus, the bound is extremely small as long as the norm r and the dimension d are moderately large, and indicates that the gradients contains little signal on the underlying target function.

Based on these bounds, one can also formally prove that a gradient-based method, under a reasonable model, will fail in returning a reasonable predictor, unless the number of iterations is exponentially large in r and d [3] . This provides strong evidence that gradient-based methods indeed cannot learn random parities and linear-periodic functions. We emphasize that these results hold *regardless of which class of predictors we use* (e.g. they can be arbitrarily complex neural networks) – the problem lies in using a gradient-based method to train them. Also, we note that the difficulty lies in the random choice of $\mathbf{v}^*$, and the problem is not difficult if $\mathbf{v}^*$ is known and fixed in advance (for example, for a full parity $\mathbf{v}^* = (1, \ldots, 1)$, this problem is known to be solvable with an appropriate LSTM network (Hochreiter & Schmidhuber, 1997)).

Finally, we remark that the connection between parities, difficulty of learning and orthogonal functions is not new, and has already been made in the context of statistical query learning (Kearns, 1998; Blum et al., 1994). This refers to algorithms which are constrained to interact with

[2]This should not be confused with the variance of gradient estimates used by SGD, which we discuss in Section 3.

[3]Formally, this requires an oracle-based model, where given a point $\mathbf{w}$, the algorithm receives the gradient at $\mathbf{w}$ up to some arbitrary error much smaller than machine precision. See (Shamir, 2016), Theorem 4, for details.

data by receiving estimates of the expected value of some query over the underlying distribution (e.g. the expected value of the first coordinate), and it is well-known that parities cannot be learned with such algorithms. Recently, (Feldman et al., 2015) have formally shown that gradient-based methods with an approximate gradient oracle can be implemented as a statistical query algorithm, which implies that gradient-based methods are indeed unlikely to solve learning problems which are known to be hard in the statistical queries framework, in particular parities. In the discussion on random parities above, we have simply made the connection between gradient-based methods and parities more explicit, by direct examination of gradients' variance w.r.t. the target function.

3. Decomposition vs. End-to-end

Many practical learning problems, and more generally, algorithmic problems, can be viewed as a structured composition of sub-problems. Applicable approaches for a solution can either be tackling the problem in an end-to-end manner, or by decomposition. Whereas for a traditional algorithmic solution, the "divide-and-conquer" strategy is an obvious choice, the ability of deep learning to utilize big data and expressive architectures has made "end-to-end training" an attractive alternative. Prior results of end-to-end (Mnih et al., 2015; Graves et al., 2013) and decomposition and added feedback (Gülçehre & Bengio, 2016; Hinton & Salakhutdinov, 2006; Szegedy et al., 2015; Caruana, 1998) approaches show success in both directions. Here, we try to address the following questions: What is the price of the rather appealing end-to-end approach? Is letting a network "learn by itself" such a bad idea? When is it necessary, or worth the effort, to "help" it?

There are various aspects which can be considered in this context. For example, (Shalev-Shwartz & Shashua, 2016) analyzed the difference between the approaches from the sample complexity point of view. Here, we focus on the optimization aspect, showing that an end-to-end approach might suffer from non-informative or noisy gradients, which may significantly affect the training time. Helping the SGD process by decomposing the problem leads to much faster training. We present a simple experiment, motivated by questions every practitioner must answer when facing a new, non trivial problem: What exactly is the required training data, what network architecture should be used, and what is the right distribution of development efforts. These are all correlated questions with no clear answer. Our experiments and analysis show that making the wrong choice can be expensive.

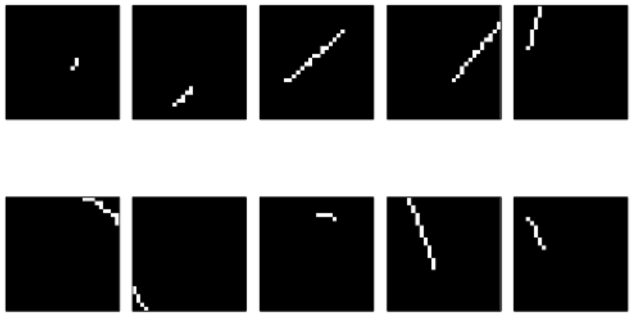


Figure 2. Section 3.1's experiment - examples of samples from X. The y values of the top and bottom rows are 1 and -1, respectively.

3.1. Experiment

Our experiment compares the two approaches in a computer vision setting, where convolutional neural networks (CNN) have become the most widely used and successful algorithmic architectures. We define a family of problems, parameterized by $k \in \mathbb{N}$, and show a gap (rapidly growing with k) between the performances of the end-to-end and decomposition approaches.

Let X denote the space of 28×28 binary images, with a distribution D defined by the following sampling procedure:

- Sample $\theta \sim U([0,\pi])$, $l \sim U([5, 28-5])$, $(x,y) \sim U([0,27])^2$.
- The image $\mathbf{x}_{\theta,l,(x,y)}$ associated with the above sample is set to 0 everywhere, except for a straight line of length l, centered at (x,y), and rotated at an angle θ. Note that as the images space is discrete, we round the values corresponding to the points on the lines to the closest integer coordinate.

Let us define an "intermediate" labeling function $y : X \to \{\pm 1\}$, denoting whether the line in a given image slopes upwards or downwards, formally:

$$y(\mathbf{x}_{\theta,l,(x,y)}) = \begin{cases} 1 & \text{if } \theta < \pi/2 \\ -1 & \text{otherwise} \end{cases} .$$

Figure 2 shows a few examples. We can now define the problem for each k. Each input instance is a tuple $\mathbf{x}_1^k := (\mathbf{x}_1, \ldots, \mathbf{x}_k)$ of k images sampled i.i.d. as above. The target output is the parity over the image labels $y(\mathbf{x}_1), \ldots, y(\mathbf{x}_k)$, namely $\tilde{y}(\mathbf{x}_1^k) = \prod_{j=1\ldots k} y(\mathbf{x}_j)$.

Many architectures of DNN can be used for predicting $\tilde{y}(\mathbf{x}_1^k)$ given $\mathbf{x}_1^k$. A natural "high-level" choice can be:

- Feed each of the images, separately, to a single CNN (of some standard specific architecture, for example, LeNet-like), denoted $N^{(1)}_{\mathbf{w}_1}$ and parameterized by its weights vector $\mathbf{w}_1$, outputting a single scalar, which can be regarded as a "score".

- Concatenate the "scores" of a tuple's entries, transform them to the range $[0, 1]$ using a sigmoid function, and feed the resulting vector into another network, $N^{(2)}_{\mathbf{w}_2}$, of a similar architecture to the one defined in Section 2, outputting a single "tuple-score", which can then be thresholded for obtaining the binary prediction.

Let the whole architecture be denoted $N_{\mathbf{w}}$. Assuming that $N^{(1)}$ is expressive enough to provide, at least, a weak learner for y (a reasonable assumption), and that $N^{(2)}$ can express the relevant parity function (see Lemma 3 in the appendix), we obtain that this architecture has the potential for good performance.

The final piece of the experimental setting is the choice of a loss function. Clearly, the primary loss which we'd like to minimize is the expected zero-one loss over the prediction, $N_{\mathbf{w}}(\mathbf{x}_1^k)$, and the label, $\tilde{y}(\mathbf{x}_1^k)$, namely:

$$\tilde{L}_{0-1}(\mathbf{w}) := \mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left[N_{\mathbf{w}}(\mathbf{x}_1^k) \neq \tilde{y}(\mathbf{x}_1^k)\right]$$

A "secondary" loss which can be used in the decomposition approach is the zero-one loss over the prediction of $N^{(1)}_{\mathbf{w}_1}(\mathbf{x}_1^k)$ and the respective $y(\mathbf{x}_1^k)$ value:

$$L_{0-1}(\mathbf{w}_1) := \mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left[N^{(1)}_{\mathbf{w}_1}(\mathbf{x}_1^k) \neq y(\mathbf{x}_1^k)\right]$$

Let $\tilde{L}, L$ be some differentiable surrogates for $\tilde{L}_{0-1}, L_{0-1}$. A classical end-to-end approach will be to minimize $\tilde{L}$, and only it; this is our "primary" objective. We have no explicit desire for $N^{(1)}$ to output any specific value, and hence L is, a priori, irrelevant. A decomposition approach would be to minimize both losses, under the assumption that L can "direct" $\mathbf{w}_1$ towards an "area" in which we know that the resulting outputs of $N^{(1)}$ can be separated by $N^{(2)}$. Note that using L is only possible when the y values are known to us.

Empirically, when comparing performances based on the "primary" objective, we see that the end-to-end approach is significantly inferior to the decomposition approach (see Figure 3). Using decomposition, we quickly arrive at a good solution, regardless of the tuple's length, k (as long as k is in the range where perfect input to $N^{(2)}$ is solvable by SGD, as described in Section 2). However, using the end-to-end approach works only for $k = 1, 2$, and completely fails already when $k = 3$ (or larger). This may be somewhat surprising, as the end-to-end approach optimizes exactly the primary objective, composed of two subproblems each of which is easily solved on its own, and with no additional irrelevant objectives.

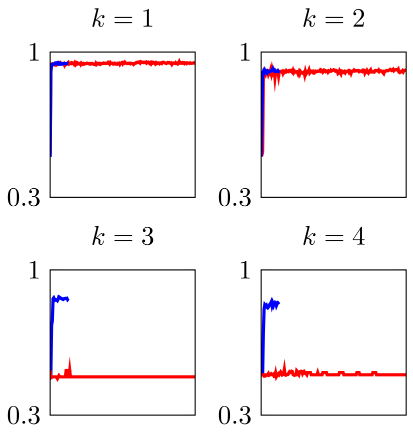


Figure 3. Performance comparison, Section 3.1's experiment. The red and blue curves correspond to the end-to-end and decomposition approaches, respectively. The plots show the zero-one accuracy with respect to the primary objective, over a held out test set, as a function of training iterations. We have trained the end-to-end network for 20000 SGD iterations, and the decomposition networks for only 2500 iterations.

3.2. Analysis

We study the experiment from two directions: Theoretically, by analyzing the gradient variance (as in Section 2), for a somewhat idealized version of the experiment, and empirically, by estimating a signal-to-noise ratio (SNR) measure of the stochastic gradients used by the algorithm. Both approaches point to a similar issue: With the end-to-end approach, the gradients do not seem to be sufficiently informative for the optimization process to succeed.

Before continuing, we note that a conceptually similar experiment to ours has been reported in (Gülçehre & Bengio, 2016) (also involving a composition of an image recognition task and a simple Boolean formula, and with qualitatively similar results). However, that experiment came without a formal analysis, and the failure was attributed to local minima. In contrast, our analysis indicates that the problem is not due to local-minima (or saddle points), but from the gradients being non-informative and noisy.

We begin with a theoretical result, which considers our experimental setup under two simplifying assumptions: First, the input is assumed to be standard Gaussian, and second, we assume the labels are generated by a target function of the form $h_{\mathbf{u}}(\mathbf{x}_1^k) = \prod_{l=1}^{k} \text{sign}(\mathbf{u}^\top \mathbf{x}_l)$. The first assumption is merely to simplify the analysis (similar results can be shown more generally, but the argument becomes more involved). The second assumption is equivalent to assuming that the labels $y(\mathbf{x})$ of individual images can be realized by a linear predictor, which is roughly the case for simple image labelling task such as ours.

Theorem 3 *Let $\mathbf{x}_1^k$ denote a k-tuple $(\mathbf{x}_1, \dots, \mathbf{x}_k)$ of input instances, and assume that each $\mathbf{x}_l$ is i.i.d. standard Gaussian in $\mathbb{R}^d$. Define*

$$h_{\mathbf{u}}(\mathbf{x}_1^k) = \prod_{l=1}^{k} sign(\mathbf{u}^\top \mathbf{x}_l),$$

and the objective (w.r.t. some predictor $p_{\mathbf{w}}$ parameterized by $\mathbf{w}$)

$$F(\mathbf{w}) = \mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left[\ell(p_{\mathbf{w}}(\mathbf{x}_1^k), h_{\mathbf{u}}(\mathbf{x}_1^k)\right].$$

Where the loss function ℓ is either the square loss $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ or a classification loss of the form $\ell(\hat{y}, y) = r(\hat{y} \cdot y)$ for some 1-Lipschitz function r.

Fix some $\mathbf{w}$, and suppose that $p_{\mathbf{w}}(\mathbf{x})$ is differentiable w.r.t. $\mathbf{w}$ and satisfies $\mathbb{E}_{\mathbf{x}_1^k}\left[\|\frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x}_1^k\|^2\right] \le G(\mathbf{w})^2$. Then if $\mathcal{H} = \{h_{\mathbf{u}} : \mathbf{u} \in \mathbb{R}^d, \|\mathbf{u}\| = 1\}$, then

$$Var(\mathcal{H}, F, \mathbf{w}) \;\le\; G(\mathbf{w})^2 \cdot O\left(\sqrt{\frac{k\log(d)}{d}}\right)^k.$$

The proof is given in Appendix B.2. The theorem shows that the "signal" regarding $h_{\mathbf{u}}$ (or, if applying to our experiment, the signal for learning $N^{(1)}$, had y been drawn uniformly at random from some set of functions over X) decreases exponentially with k. This is similar to the parity result in Section 2, but with an important difference: Whereas the base of the exponent there was $1/2$, here it is the much smaller quantity $k\log(d)/\sqrt{d}$ (e.g. in our experiment, we have $k \le 4$ and $d = 28^2$). This indicates that already for very small values of k, the information contained in the gradients about $\mathbf{u}$ can become extremely small, and prevent gradient-based methods from succeeding, fully according with our experiment.

To complement this analysis (which applies to an idealized version of our experiment), we consider a related "signal-to-noise" (SNR) quantity, which can be empirically estimated in our actual experiment. To motivate it, note that a key quantity used in the proof of Theorem 3, for estimating the amount of signal carried by the gradient, is the squared norm of the correlation between the gradient of the predictor $p_{\mathbf{w}}$, $g(\mathbf{x}_1^k) := \frac{\partial}{\partial \mathbf{w}} p_{\mathbf{w}}(\mathbf{x}_1^k)$ and the target function $h_{\mathbf{u}}$, which we denote by $\text{Sig}_{\mathbf{u}}$:

$$\text{Sig}_{\mathbf{u}} := \left\|\mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left[h_{\mathbf{u}}(\mathbf{x}_1^k) g(\mathbf{x}_1^k)\right]\right\|^2.$$

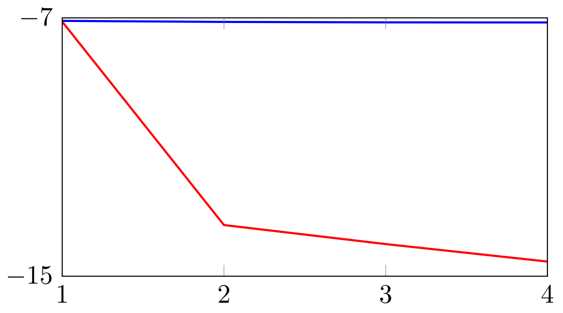


Figure 4. Section 3.1's experiment: comparing the SNR for the end-to-end approach (red) and the decomposition approach (blue), as a function of k, in $\log_e$ scale.

We will consider the ratio between this quantity and a "noise" term $\text{Noi}_{\mathbf{u}}$, i.e. the variance of this correlation over the samples:

$$\text{Noi}_{\mathbf{u}} := \mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left\|h_{\mathbf{u}}(\mathbf{x}_1^k) g(\mathbf{x}_1^k) - \mathop{\mathbb{E}}_{\mathbf{x}_1^k}\left[h_{\mathbf{u}}(\mathbf{x}_1^k) g(\mathbf{x}_1^k)\right]\right\|^2.$$

Since here the randomness is with respect to the data rather than the target function (as in Theorem 3), we can estimate this SNR ratio in our experiment. It is well-known (e.g. (Ghadimi & Lan, 2013)) that the amount of noise in the stochastic gradient estimates used by stochastic gradient descent crucially affects its convergence rate. Hence, smaller SNR should be correlated with worse performance.

We empirically estimated this SNR measure, $\text{Sig}_y/\text{Noi}_y$, for the gradients w.r.t. the weights of the last layer of $N^{(1)}$ (which potentially learns our intermediate labeling function y) at the initialization point in parameter space. The SNR estimate for various values of k are plotted in Figure 4. We indeed see that when $k \ge 3$, the SNR appears to approach extremely small values, where the estimator's noise, and the additional noise introduced by a finite floating point representation, can completely mask the signal, which can explain the failure in this case.

In Section A in the Appendix, we also present a second, more synthetic, experiment, which demonstrates a case where the decomposition approach directly decreases the stochastic noise in the SGD optimization process, hence benefiting the convergence rate.

4. Additional Failure Families - Brief Discussion

In an extended version of this paper, (Shalev-Shwartz et al., 2017), we broadly discuss two additional families of failures. Here, due to lack of space, we present them briefly.

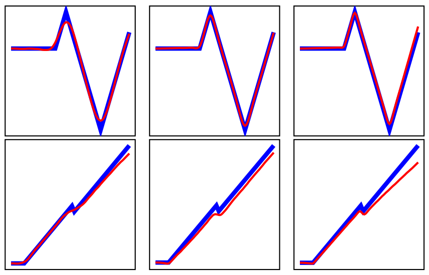
First, we demonstrate the importance of both the network's architecture and the optimization algorithm on the training time. While the choice of architecture is usually studied in the context of its expressive power, we show that even when two architectures have the same expressive power for a given task, there may be a tremendous difference in the ability to optimize them. We analyze the required runtime of gradient descent for the two architectures through the lens of the condition number of the problem. We further show that conditioning techniques can yield additional orders of magnitude speedups. The experimental setup for this problem is around a seemingly simple problem — encoding a piece-wise linear one-dimensional curve. A summary of experimental results, when training with different architectures and conditioning techniques, is found in Figure 5. Despite the simplicity of this problem, we show that following the common rule of "perhaps I should use a deeper/wider network"[4] does not significantly help here.

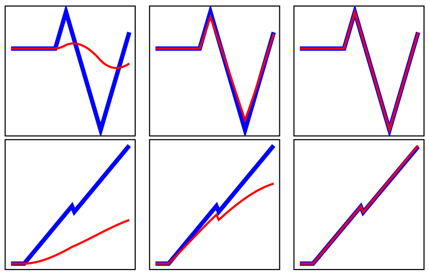
Finally, we consider deep learning's reliance on "vanilla" gradient information for the optimization process. We previously discussed the deficiency of using a local property of the objective in directing global optimization. We turn our focus to a simple case in which it is possible to solve the optimization problem based on local information, but not in the form of a gradient. We experiment with architectures that contain activation functions with flat regions, which leads to the well known vanishing gradient problem. Practitioners take great care when working with such activation functions, and many heuristic tricks are applied in order to initialize the network's weights in non-flat areas of its activations. Here, we show that by using a different update rule, we manage to solve the learning problem efficiently. Moreover, one can show convergence guarantees for a family of such functions. This provides a clean example where non-gradient-based optimization schemes can overcome the limitations of gradient-based learning.

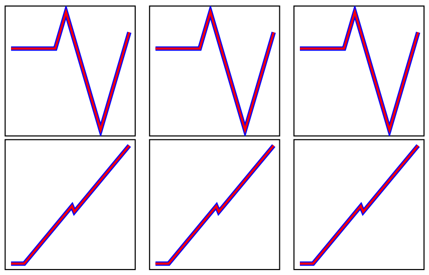
5. Summary

In this paper, we considered different families of problems, where standard gradient-based deep learning approaches appear to suffer from significant difficulties. Our analysis indicates that these difficulties are not necessarily related to stationary point issues such as spurious local minima or a plethora of saddle points, but rather more subtle issues: Insufficient information in the gradients about the underlying target function; low SNR; bad conditioning; or flatness in the activations (see Figure 6 for a graphical illustration). We consider it as a first step towards a better understanding of where standard deep learning methods might fail, as well as what approaches might overcome these failures.

[4]See http://joelgrus.com/2016/05/23/fizz-buzz-in-tensorflow/ for the inspiration behind this quote.


(a) Linear architecture.


(b) Convolutional architecture.


(c) Convolutional architecture with conditioning.

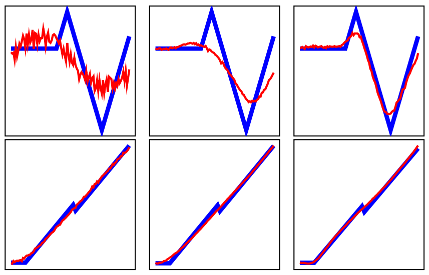
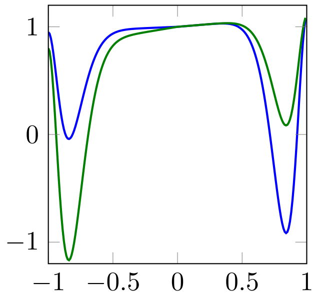
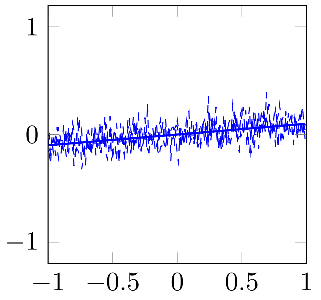

(d) Vanilla deep auto encoder.

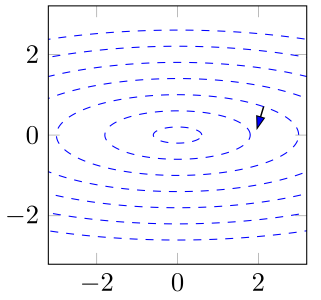
Figure 5. Examples for decoded outputs of several experiments, learning to encode PWL curves. In blue are the original curves. In red are the decoded curves. The plot shows the outputs for two curves, after 500, 10000, and 50000 iterations, from left to right. The convolutional architecture, with conditioning, clearly outperforms others, both in terms of convergence rate and final accuracy.



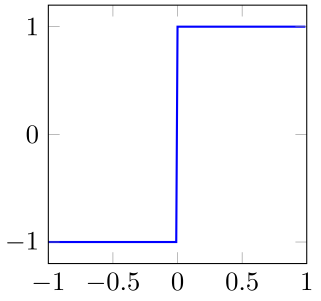
(a) Extremely small variance in the loss surface's gradient, w.r.t. different target functions, each with a very different optimum.



(b) Low SNR of gradient estimates. The dashed lines represent losses w.r.t. different samples, each implying a very different estimate than the average gradient.



(c) Bad conditioning - 2 dimensional example of a loss function's quiver. Following the gradient is far from being the best direction to go.



(d) Completely flat activation - no information in the gradient.

Figure 6. A graphical summary of limitations of gradient-based learning.

References

Blum, Avrim, Furst, Merrick, Jackson, Jeffrey, Kearns, Michael, Mansour, Yishay, and Rudich, Steven. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 253–262. ACM, 1994.

Caruana, Rich. Multitask learning. In *Learning to learn*, pp. 95–133. Springer, 1998.

Choromanska, Anna, Henaff, Mikael, Mathieu, Michael, Arous, Gérard Ben, and LeCun, Yann. The loss surfaces of multilayer networks. In *AISTATS*, 2015.

Collobert, Ronan and Weston, Jason. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167. ACM, 2008.

Dauphin, Yann N, Pascanu, Razvan, Gulcehre, Caglar, Cho, Kyunghyun, Ganguli, Surya, and Bengio, Yoshua. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Feldman, Vitaly, Guzman, Cristobal, and Vempala, Santosh. Statistical query algorithms for stochastic convex optimization. *arXiv preprint arXiv:1512.09170*, 2015.

Ghadimi, Saeed and Lan, Guanghui. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pp. 249–256, 2010.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

Gülçehre, Çalar and Bengio, Yoshua. Knowledge matters: Importance of prior information for optimization. *Journal of Machine Learning Research*, 17(8):1–32, 2016.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George E, Mohamed, Abdel-rahman, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Sainath, Tara N, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29 (6):82–97, 2012.

Hinton, Geoffrey E and Salakhutdinov, Ruslan R. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kearns, Michael. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Schroff, Florian, Kalenichenko, Dmitry, and Philbin, James. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael I, and Moritz, Philipp. Trust region policy optimization. In *ICML*, pp. 1889–1897, 2015.

Shalev-Shwartz, Shai and Shashua, Amnon. On the sample complexity of end-to-end training vs. semantic abstraction training. *arXiv preprint arXiv:1604.06915*, 2016.

Shalev-Shwartz, Shai, Shamir, Ohad, and Shammah, Shaked. Failures of deep learning. *arXiv preprint arXiv:1703.07950*, 2017.

Shamir, Ohad. Distribution-specific hardness of learning neural networks. *arXiv preprint arXiv:1609.01037*, 2016.

Sutskever, Ilya, Martens, James, Dahl, George E, and Hinton, Geoffrey E. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

Szegedy, Christian, Ioffe, Sergey, Vanhoucke, Vincent, and Alemi, Alex. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

Taigman, Yaniv, Yang, Ming, Ranzato, Marc'Aurelio, and Wolf, Lior. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *CoRR abs/1609.03499*, 2016.