Poor starting points in machine learning

Mark Tygert

Facebook Artificial Intelligence Research
1 Facebook Way, Menlo Park, CA 94025

tygert@aya.yale.edu

January 13, 2016

Abstract

Poor (even random) starting points for learning/training/optimization are common in machine learning. In many settings, the method of Robbins and Monro (online stochastic gradient descent) is known to be optimal for good starting points, but may not be optimal for poor starting points — indeed, for poor starting points Nesterov acceleration can help during the initial iterations, even though Nesterov methods not designed for stochastic approximation could hurt during later iterations. A good option is to roll off Nesterov acceleration for later iterations. The common practice of training with nontrivial minibatches enhances the advantage of Nesterov acceleration.

1 Introduction

The scheme of Robbins and Monro (online stochastic gradient descent) has long been known to be optimal for stochastic approximation/optimization . . . provided that the starting point for the iterations is good. Yet poor starting points are common in machine learning, so higher-order methods (such as Nesterov acceleration) may help during the early iterations (even though higher-order methods generally hurt during later iterations), as explained below. Below, we elaborate on Section 2 of Sutskever et al. (2013), giving some more rigorous mathematical details, somewhat similar to those given by Défossez and Bach (2015) and Flammarion and Bach (2015). In particular, we discuss the key role of the noise level of estimates of the objective function, and stress that a large minibatch size makes higher-order methods more effective for more iterations. Our presentation is purely expository; we disavow any claim we might make to originality — in fact, our observations should be obvious to experts in stochastic approximation. We opt for concision over full mathematical rigor, and assume that the reader is familiar with the subjects reviewed by Bach (2013).

The remainder of the present paper has the following structure: Section 2 sets mathematical notation for the stochastic approximation/optimization considered in the sequel. Section 3 elaborates a principle for accelerating the optimization. Section 4 discusses minibatching. Section 5 supplements the numerical examples of Bach (2013) and Sutskever et al. (2013). Section 6 reiterates the above, concluding the paper.

2 A mathematical formulation

Machine learning commonly involves a cost/loss function c and a single random sample X of test data; we then want to select parameters θ minimizing the expected value

$$e(\theta) = \mathbf{E}[c(\theta, X)] \tag{1}$$

(e is known as the "error" or "risk" — specifically, the "test" error, as opposed to "training" error). The cost c is generally a nonnegative scalar-valued function, whereas θ and X can be vectors of real or complex numbers. In supervised learning, X is a vector which contains both "input" or "feature" entries and corresponding "output" or "target" entries — for example, for a regression the input entries can be regressors and the outputs can be regressands; for a classification the outputs can be the labels of the correct classes. For training, we have no access to X (nor to e) directly, but can only access independent and identically distributed (i.i.d.) samples from the probability distribution p from which X arises. Of course, $e(\theta)$ depends only on p, not on any particular sample drawn from p.

To finish setting our notational conventions, we define the deviation or dispersion

$$d(\theta, X) = c(\theta, X) - e(\theta), \tag{2}$$

yielding the decomposition

$$c(\theta, X) = e(\theta) + d(\theta, X). \tag{3}$$

Combining (2) and (1) yields that

$$\mathbf{E}[d(\theta, X)] = 0 \tag{4}$$

for all θ, that is, the mean of $d(\theta, X)$ is 0 (whereas the mean of $c(\theta, X)$ is $e(\theta)$).

3 Accelerated learning

For definiteness, we limit consideration to algorithms which may access only values of $c(\theta, x)$ and its first-order derivatives/gradient with respect to θ. We assume that the cost c is calibrated (shifted by an appropriate constant) such that the absolute magnitude of c is a good absolute gauge of the quality of θ (lower cost is better, of course). As reviewed, for example, by Bach (2013), the algorithm of Robbins and Monro (online stochastic gradient descent) minimizes the test error e in (1) using a number of samples (from the probability distribution p) that is optimal — optimal to within a constant factor which depends on the starting point for θ. Despite this kind of optimality, if e happens to start much larger than $d = c - e$, then a higher-order method (such as the Nesterov acceleration discussed by Bach (2013) and others) can dramatically reduce the constant. We say that θ is "poor" to mean that $e(\theta)$ is much larger than $d(\theta, X)$. If θ is poor, that is,

$$e(\theta) \gg |d(\theta, X)|, \tag{5}$$

then (3) yields that

$$c(\theta, X) = e(\theta) + d(\theta, X) \approx e(\theta) \tag{6}$$

is effectively deterministic, and so Nesterov and other methods from Bach (2013) could accelerate the optimization until the updated θ is no longer poor.

Clearly, θ being poor, that is, $e(\theta)$ being much larger than $d(\theta, X)$, simply means that $c(\theta, x) = e(\theta) + d(\theta, x)$ depends mostly just on the probability distribution p underlying X, but not so much on any particular sample x from p. (For supervised learning, the probability distribution p is joint over the inputs and outputs, encoding the relation between the inputs and their respective outputs.) If θ is poor, that is, $e(\theta)$ is much larger than $d(\theta, X)$, so that $c(\theta, X) \approx e(\theta)$ is mostly deterministic, then Nesterov methods can be helpful.

Of course, optimization performance may suffer from applying a higher-order method to a rough, random objective (say, to a random variable $c(\theta, X)$ estimating $e(\theta)$ when the standard deviation of $c(\theta, X)$ is high relative to $e(\theta)$). For instance, Nesterov methods essentially sum across several iterations; if the values being summed were stochastically independent, then the summation would actually increase the variance. *In accordance with the optimality of the method of Robbins and Monro (online stochastic gradient descent), Nesterov acceleration can be effective only when the estimates of the objective function are dominantly deterministic, with the objective function being the test error e from (1).*

In practice, we can apply higher-order methods during the initial iterations when the starting point is poor, gradually transitioning to the original method of Robbins and Monro (stochastic gradient descent) as the test error approaches its lower limit, that is, as stochastic variations in estimates of the objective function and its derivatives become important.

4 Minibatches

Some practical considerations detailed, for example, by LeCun et al. (1998) are the following:

Many modern microprocessor architectures can leverage the parallelism inherent in batch processing. With such parallel (and partly parallel) processors, minimizing the number of samples drawn from the probability distribution underlying the data may not be the most efficient possibility. Rather than updating the parameters being learned for random individual samples as in online stochastic gradient descent, the common current practice is to draw all at once a number of samples — a collection of samples known as a "minibatch" — and then update the parameters simultaneously for all these samples before proceeding to the next minibatch. With sufficient parallelism, processing an entire minibatch may take little longer than processing a single sample.

Averaging the estimates of the objective function (that is, of the test error e) and its derivatives over all samples in a minibatch yields estimates with smaller standard deviations, effectively making more parameter values be "poor," so that Nesterov acceleration is more effective for more iterations. Moreover, minibatches provide (essentially for free) estimates of the standard deviations of the estimates of e. The number of initial iterations for which Nesterov methods can accelerate the optimization can be made arbitrarily large by setting the size of the minibatches arbitrarily large. That said, smaller minibatch sizes typically require fewer samples in total to approach full convergence (albeit typically with more iterations, processing one minibatch per iteration). Also, after sufficiently many iterations, stochastic variations in estimates of the objective function and its derivatives can become important, and continuing to use Nesterov acceleration in these later iterations would be counterproductive. Again, we recommend applying higher-order methods during the initial iterations when the starting point is poor, noting that larger minibatches make the higher-order methods advantageous for more iterations before the point requiring turning off the Nesterov acceleration.

5 Numerical experiments

The present section supplements the examples of Bach (2013) and Sutskever et al. (2013) with a few more experiments indicating that a higher-order method — namely, momentum, a form of Nesterov acceleration — can help a bit in training convolutional networks. All training reported is via the method of Robbins and Monro (stochastic gradient descent), with minibatches first of size 1 sample per iteration (as in the original, online algorithm) and then of size 100 samples per iteration (size 100 is among the most common choices used in practice on modern processors which can leverage the parallelism inherent in batch processing). Section 4 and the third-to-last paragraph of the present section describe the minibatching in more detail. Figures 1 and 3 report the accuracies for various configurations of "momentum" and "learning rates"; momentum appears to accelerate training somewhat, especially with the larger size of minibatches.

Each iteration subtracts from the parameters being learned a multiple of a stored vector, where the multiple is the "learning rate," and the stored vector is the estimated gradient plus the stored vector from the previous iteration, with the latter stored vector scaled by the amount of "momentum":

$$\theta^{(i)} = \theta^{(i-1)} - \alpha^{(i)}_{l.r.}\, v^{(i)}, \tag{7}$$

with

$$v^{(i)} = \beta^{(i)}_{mom.}\, v^{(i-1)} + \frac{1}{k}\sum_{j=1}^{k} \left(\frac{\partial c}{\partial \theta}\left(\theta, x^{(i,j)}\right)\right)\Bigg|_{\theta=\theta^{(i-1)}}, \tag{8}$$

where i is the index of the iteration, $\theta^{(i)}$ is the updated vector of parameters, $\theta^{(i-1)}$ is the previous vector of parameters, $v^{(i)}$ is the updated stored auxiliary vector, $v^{(i-1)}$ is the previous stored auxiliary vector, $\alpha^{(i)}_{l.r.}$ is the learning rate, $\beta^{(i)}_{mom.}$ is the amount of momentum, $x^{(i,j)}$ is the jth training sample in the ith minibatch, k is the size of the minibatch, and $\partial c/\partial\theta$ is the gradient of the cost c with respect to the parameters θ.

Following LeCun et al. (1998) exactly as done by Chintala et al. (2015), the architecture for generating the feature activations is a convolutional network (convnet) consisting of a series of stages, with each stage feeding its output into the next. The last stage has the form of a multinomial logistic regression, applying a linear transformation to its inputs, followed by the "softmax" detailed by LeCun et al. (1998), thus producing a probability distribution over the classes in the classification. The cost/loss is the negative of the natural logarithm of the probability so assigned to the correct class. Each stage before the last convolves each image from its input against several learned convolutional kernels, summing together the convolved images from all the inputs into several output images, then takes the absolute value of each pixel of each resulting image, and finally averages over each patch in a partition of each image into a grid of 2×2 patches. All convolutions are complex valued and produce pixels only where the original images cover all necessary inputs (that is, a convolution reduces each dimension of the image by one less than the size of the convolutional kernel).

We subtract the mean of the pixel values from each input image before processing with the convnet, and we append an additional feature activation feeding into the "softmax" to those obtained from the convnet, namely the standard deviation of the set of values of the pixels in the image.

The data is a subset of the 2012 ImageNet set of Russakovsky et al. (2015), retaining 10 classes of images, representing each class by 100 samples in a training set and 50 per class in a testing set. Restricting to this subset facilitated more extensive experimentation (optimizing hyperparameters more extensively, for example). The images are full color, with three color channels. We neither augmented the input data nor regularized the cost/loss functions. We used the Torch7 platform — http://torch.ch — for all computations.

Table 1 details the convnet architecture we tested. "Stage" specifies the positions of the indicated layers in the convnet. "Input channels" specifies the number of images input to the given stage for each sample from the data. "Output channels" specifies the number of images output from the given stage. Each input image is convolved against a separate, learned convolutional kernel for each output image (with the results of all these convolutions summed together for each output image). "Kernel size" specifies the size of the square grid of pixels used in the convolutions. "Input channel size" specifies the size of the square grid of pixels constituting each input image. "Output channel size" specifies the size of the square grid of pixels constituting each output image. The feature activations that the convnet produces feed into a linear transformation followed by a "softmax," as detailed by LeCun et al. (1998).

For the minibatch size 100, rather than updating the parameters being learned for randomly selected individual images from the training set as in online stochastic gradient descent, we instead do the following: we randomly permute the training set and partition this permuted set of images into subsets of 100, updating the parameters simultaneously for all 100 images constituting each of the subsets (known as "minibatches"), processing the series of minibatches in series. Each sweep through the entire training set is known as an "epoch." LeCun et al. (1998), among others, made the above terminology the standard for training convnets. The horizontal axes in the figures count the number of epochs.

Figures 1 and 2 present the results for minibatches of size 100 samples per iteration; Figure 3 presents the results for minibatches of size 1 sample per iteration. "Average precision" is the fraction of all classifications which are correct, choosing only one class for each input sample image from the test set. "Error on the test set" is the average over all samples in the test set of the negative of the natural logarithm of the probability assigned to the correct class (assigned by the "softmax"). "Coefficient of variation" is an estimate of the standard deviation of $c(\theta, X)$ (from (1)) divided by the mean of $c(\theta, X)$, that is, an estimate of the standard deviation of $d(\theta, X)$ (from (2)) divided by $e(\theta)$ (from (1) and (2)). Rolling off momentum as the coefficient of variation increases can be a good idea.

Please beware that these experiments are far from definitive, and even here the gains from using momentum seem to be marginal. Even so, as Section 4 discusses, minibatching effectively reduces the standard deviations of estimates of the objective function e, making Nesterov acceleration more effective for more iterations.

6 Conclusion

Though in many settings the method of Robbins and Monro (stochastic gradient descent) is optimal for good starting points, higher-order methods (such as momentum and Nesterov acceleration) can help during early iterations of the optimization when the parameters being optimized are poor in the sense discussed above. The opportunity for accelerating the optimization is clear theoretically and apparently observable via numerical experiments. Minibatching makes the higher-order methods advantageous for more iterations. That said, higher order and higher accuracy need not be the same, as higher order guarantees only that accuracy increase at a faster rate — higher order guarantees higher accuracy only after enough iterations.

Acknowledgements

We would like to thank Francis Bach, Léon Bottou, Yann Dauphin, Piotr Dollár, Yann LeCun, Marc'Aurelio Ranzato, Arthur Szlam, and Rachel Ward.

A A simple analytical example

This appendix works through a very simple one-dimensional example.

Consider the cost c as a function of a real scalar parameter θ and sample of data x, defined via

$$c(\theta, x) = (\theta - x)^2. \tag{9}$$

Suppose that X is the (Rademacher) random variable taking the value 1 with probability 1/2 and the value -1 with probability 1/2. Then, the coefficient of variation of $c(\theta, X)$, that is, the standard deviation $\sigma(\theta)$ of $c(\theta, X)$ divided by the mean $e(\theta)$ of $c(\theta, X)$, is

$$\frac{\sigma(\theta)}{e(\theta)} = \frac{2|\theta|}{\theta^2 + 1} : \tag{10}$$

(proof) X taking the value 1 with probability 1/2 and the value -1 with probability 1/2 yields that

$$|X| = 1 \tag{11}$$

and that the expected value of X is 0,

$$\mathbf{E}[X] = 0. \tag{12}$$

Combining (9) and (11) then yields

$$c(\theta, X) = \theta^2 + 1 - 2\theta X. \tag{13}$$

Moreover, together with the definitions (1) and (2), combining (13) and (12) yields

$$e(\theta) = \mathbf{E}[c(\theta, X)] = \theta^2 + 1 \tag{14}$$

and

$$d(\theta, X) = c(\theta, X) - e(\theta) = -2\theta X. \tag{15}$$

The mean of $c(\theta, X)$ is this $e(\theta)$ in (14). The standard deviation of $c(\theta, X)$ is the same as the standard deviation of $d(\theta, X)$ in (15), which is just

$$\sigma(\theta) = 2|\theta|, \tag{16}$$

due to (15) combined with X taking the value 1 with probability 1/2 and the value -1 with probability 1/2. Combining (14) and (16) yields (10), as desired. ■

The secant method (a higher-order method) consists of the iterations updating $\theta^{(i-2)}$ and $\theta^{(i-1)}$ to $\theta^{(i)}$ via

$$\theta^{(i)} = \theta^{(i-1)} - \frac{\partial c}{\partial \theta}\left(\theta^{(i-1)}, x^{(i-1)}\right) \cdot \left(\theta^{(i-1)} - \theta^{(i-2)}\right) \Big/ \left(\frac{\partial c}{\partial \theta}\left(\theta^{(i-1)}, x^{(i-1)}\right) - \frac{\partial c}{\partial \theta}\left(\theta^{(i-2)}, x^{(i-2)}\right)\right), \tag{17}$$

where $x^{(i-2)}$, $x^{(i-1)}$, and $x^{(i)}$ are independent realizations (random samples) of X and the derivative of $c(\theta, x)$ from (9) with respect to θ is

$$\frac{\partial c}{\partial \theta} = 2(\theta - x); \tag{18}$$

combining (17) and (18) and simplifying yields

$$\theta^{(i)} = \left(\theta^{(i-1)} x^{(i-2)} - \theta^{(i-2)} x^{(i-1)}\right) \Big/ \left(\theta^{(i-1)} - x^{(i-1)} - \theta^{(i-2)} + x^{(i-2)}\right). \tag{19}$$

In the limit that both $x^{(i-1)}$ and $x^{(i-2)}$ be 0, (19) shows that $\theta^{(i)}$ is also 0 (provided that $\theta^{(i-1)}$ and $\theta^{(i-2)}$ are distinct), that is, the secant method finds the optimal value for θ in a single iteration in such a limit. In fact, (14) makes clear that the optimal value for θ is 0, while combining (19) and (11) yields that

$$|\theta^{(i)}| \le \left(|\theta^{(i-1)}| + |\theta^{(i-2)}|\right) \Big/ \left(|\theta^{(i-1)} - \theta^{(i-2)}| - 2\right), \tag{20}$$

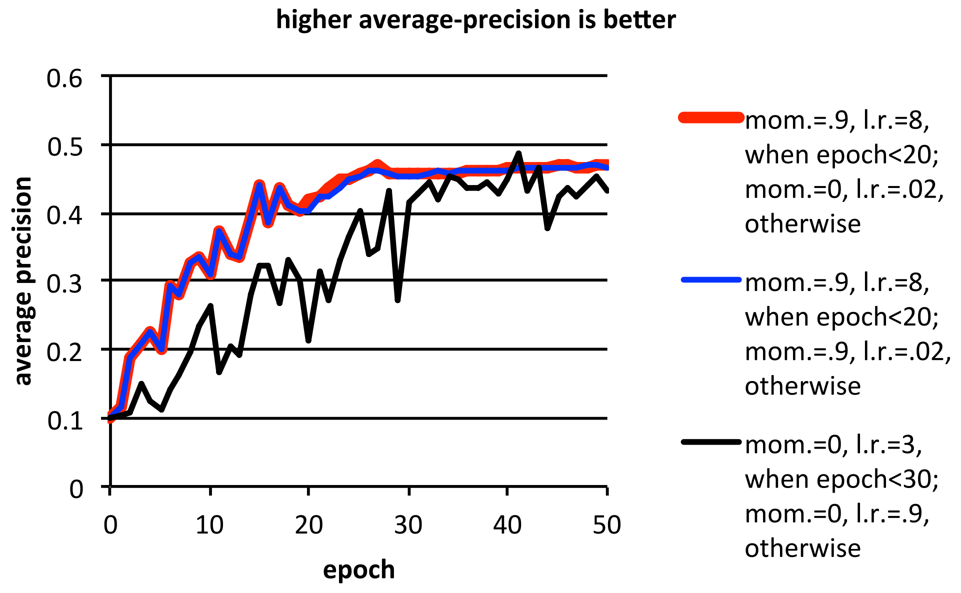
so that $|\theta^{(i)}|$ is likely to be reasonably small (around 1 or so) even if $|\theta^{(i-1)}|$ or $|\theta^{(i-2)}|$ (or both) are very large and random and $|X| = 1$.

However, these iterations generally fail to converge to any value much smaller than unit magnitude: with probability 1/2, indeed, $x^{(i-2)} = x^{(i-1)} = 1$ or $x^{(i-2)} = x^{(i-1)} = -1$; hence (19) yields that $|\theta^{(i)}| = 1$ with probability at least 1/2, assuming that $\theta^{(i-2)}$ and $\theta^{(i-1)}$ are distinct (while in the degenerate case that $\theta^{(i-2)}$ and $\theta^{(i-1)}$ take exactly the same value, $\theta^{(i)}$ also takes that same value).

All in all, a sensible strategy is to start with a higher-order method (such as the secant method) when $|\theta|$ is large, and transition to the asymptotically optimal method of Robbins and Monro (stochastic gradient descent) as $|\theta|$ becomes of roughly unit magnitude. The transition can be based on $|\theta|$ or on estimates of the coefficient of variation — due to (10), the coefficient of variation is essentially inversely proportional to $|\theta|$ when $|\theta|$ is large.

Figure 1: Test accuracies for various settings of momentum (mom.) and learning rates (l.r.) [minibatch size 100]

(a)



(b)

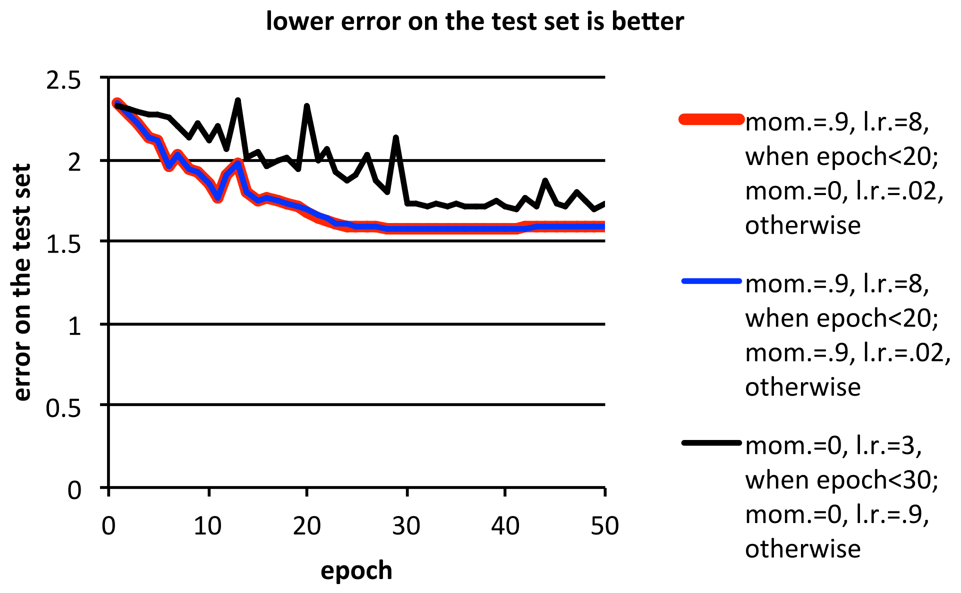


Figure 2: Estimated coefficients of variation (CVs) of the cost $c(\theta, X)$ in (1) and (3), using all samples in each minibatch to construct estimates of the CVs (the estimates of the CVs are themselves subject to stochastic variations; the distribution of the plotted points within an epoch provides an indication of the probability distribution of the estimates of the CVs) [minibatch size 100]

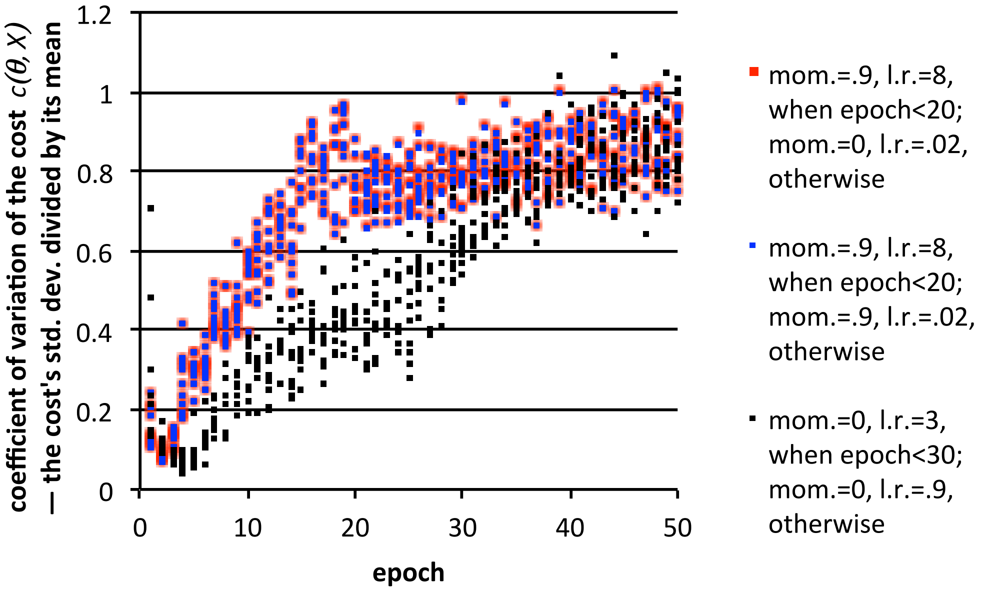
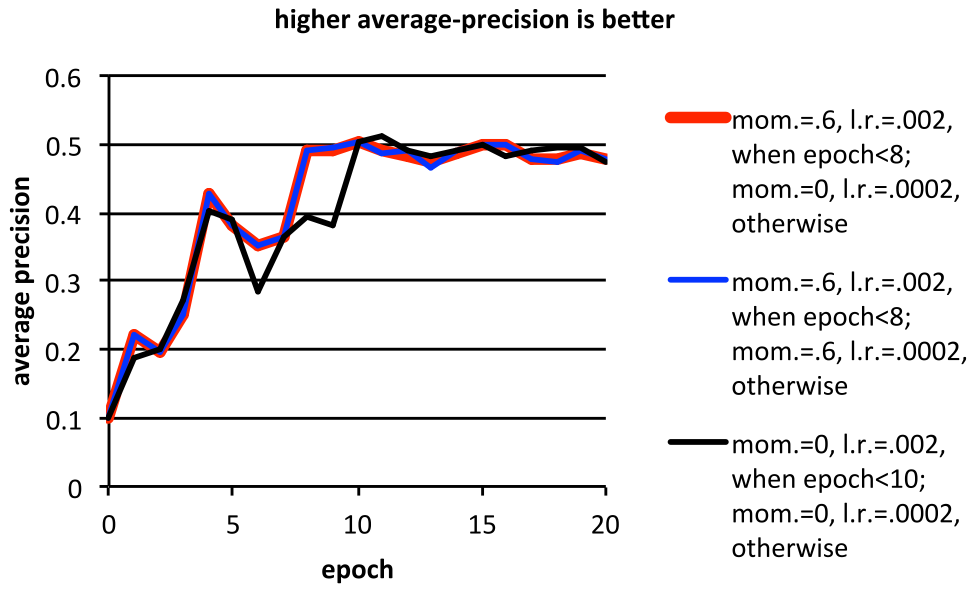


Table 1: Architecture of the convolutional network

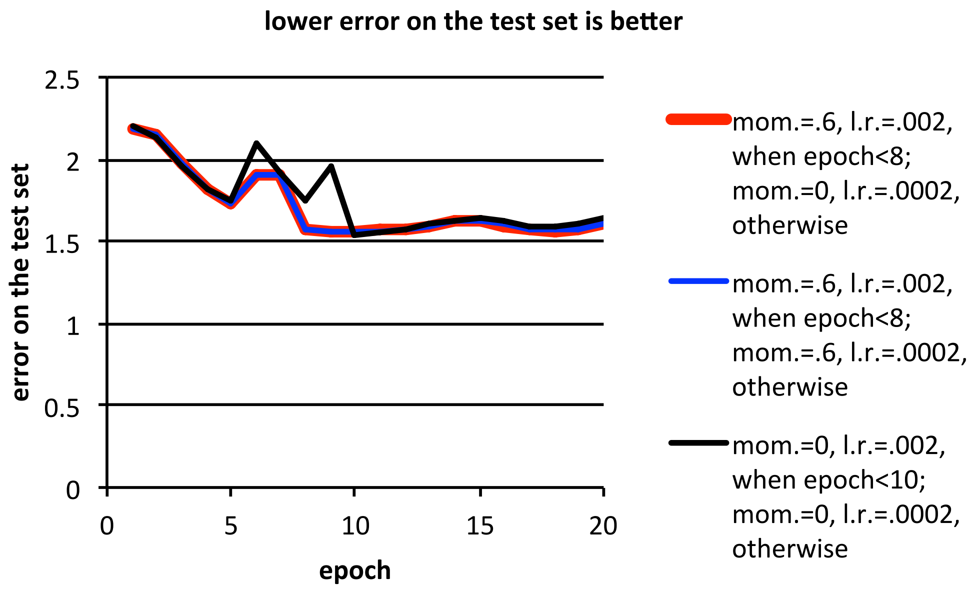
stage	input channels	output channels	kernel size	input channel size	output channel size
first	3	16	5×5	128×128	62×62
second	16	64	3×3	62×62	30×30
third	64	256	3×3	30×30	14×14
fourth	256	256	3×3	14×14	6×6

Figure 3: Test accuracies for various settings of momentum (mom.) and learning rates (l.r.) [minibatch size 1]

(a)



(b)



References

Bach, F. (2013). Stochastic gradient methods for machine learning. Technical report, INRIA-ENS, Paris, France. `http://www.di.ens.fr/~fbach/fbach_sgd_online.pdf`.

Chintala, S., Ranzato, M., Szlam, A., Tian, Y., Tygert, M., and Zaremba, W. (2015). Scale-invariant learning and convolutional networks. Technical Report 1506.08230, arXiv. `http://arxiv.org/abs/1506.08230`.

Défossez, A. and Bach, F. (2015). Averaged least-mean-squares: bias-variance trade-offs and optimal sampling distributions. In *Proc. 18th Internat. Conf. Artificial Intelligence and Statistics*, volume 38, pages 205–213. `http://jmlr.org/proceedings/papers/v38/defossez15.pdf`.

Flammarion, N. and Bach, F. (2015). From averaging to acceleration, there is only a step-size. Technical Report 1504.01577, arXiv. `http://arxiv.org/abs/1504.01577`.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324. `http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf`.

Russakovsky, O., Deng, J., Su, H., Kruse, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. Technical Report 1409.0575v3, arXiv. `http://arxiv.org/abs/1409.0575`.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *Proc. 30th International Conference on Machine Learning*, volume 28, pages 1139–1147. J. Machine Learning Research. `http://www.jmlr.org/proceedings/papers/v28/sutskever13.pdf`.