SCALE-INVARIANT LEARNING AND CONVOLUTIONAL NETWORKS

Mark Tygert, Arthur Szlam, Soumith Chintala, Marc'Aurelio Ranzato,
Yuandong Tian & Wojciech Zaremba
Facebook Artificial Intelligence Research
{tygert,aszlam,soumith,ranzato,yuandong,wojciech}@fb.com

ABSTRACT

The conventional classification schemes — notably multinomial logistic regression — used in conjunction with convolutional networks (convnets) are classical in statistics, designed without consideration for the usual coupling with convnets, stochastic gradient descent, and backpropagation. In the specific application to supervised learning for convnets, a simple scale-invariant classification stage turns out to be more robust than multinomial logistic regression, appears to result in slightly lower errors on several standard test sets, has similar computational costs, and features precise control over the actual rate of learning. "Scale-invariant" means that multiplying the input values by any nonzero real number leaves the output unchanged.

1 INTRODUCTION

Classification of a vector of real numbers (called "feature activations") into one of several discrete categories is well established and well studied, with generally satisfactory solutions such as the ubiquitous multinomial logistic regression reviewed, for example, by Hastie et al. (2009). However, the canonical classification may not couple well with generation of the feature activations via convolutional networks (convnets) trained using stochastic gradient descent, as discussed, for example, by LeCun et al. (1998). Fitting (also known as learning or training) the combination of the convnet and the classification stage by minimizing the cost/loss/objective function associated with the classification suggests designing a stage specifically for use in such joint fitting/learning/training. In particular, many convnets are "equivariant" with respect to scalar multiplication — multiplying the input values by any real number multiplies the output by the same factor; the present paper leverages this equivariance via a "scale-invariant" classification stage — a stage for which multiplying the input values by any nonzero real number leaves the output unchanged. The scale-invariant classification stage turns out to be more robust to outliers (including obviously mislabeled data), fits/learns/trains precisely at the rate that the user specifies, and apparently results in slightly lower errors on several standard test sets when used in conjunction with some typical convnets for generating the feature activations. The computational costs are comparable to those of multinomial logistic regression. Similar classification has been introduced earlier in other contexts by Hill & Doucet (2007), Lange & Wu (2008), Wu & Lange (2010), Saberian & Vasconcelos (2011), Mroueh et al. (2012), Wu & Wu (2012), and others. Complementary normalization includes the work of Carandini & Heeger (2012), Ioffe & Szegedy (2015), and the associated references. The key to effective learning is rescaling, as described in Section 3 below (see especially the last paragraph there).

The remainder of the present paper has the following structure: Section 2 sets the notation. Section 3 introduces the scale-invariant classification stage. Section 4 analyzes its robustness. Section 5 illustrates the performance of the classification on several standard data sets. The two appendices, A and B, provide more detailed derivations.

2 NOTATIONAL CONVENTIONS

All numbers used in the classification stage will be real valued (though the numbers used for generating the inputs to the stage may in general be complex valued). We follow the recommendations

of Magnus & Neudecker (2007): all vectors are column vectors (aside from gradients of a scalar with respect to a column vector, which are row vectors), and we use $\|v\|$ to denote the Euclidean norm of a vector v; that is, $\|v\|$ is the square root of the sum of the squares of the entries of v. We use $\|A\|$ to denote the spectral norm of a matrix A; that is, $\|A\|$ is the greatest singular value of A, which is also the maximum of $\|Av\|$ over every vector v such that $\|v\| = 1$. The terminology "Frobenius norm" of A refers to the square root of the sum of the squares of the entries of A. The spectral norm of a vector viewed as a matrix having only one column or one row is the same as the Euclidean norm of the vector; the Euclidean norm of a matrix viewed as a vector is the same as the Frobenius norm of the matrix.

3 A SCALE-INVARIANT CLASSIFICATION STAGE

We study a linear classification stage that assigns one of k classes to each real-valued vector x of feature activations (together with a measure of confidence in its classification), with the assignment being independent of the Euclidean norm of x; the Euclidean norm of x is its "scale." We associate to the k classes target vectors t_1, t_2, ..., t_k that are the vertices of either a standard simplex or a regular simplex embedded in a Euclidean space of dimension $m \geq k$ — the dimension of the embedding space being strictly greater than the minimum $(k - 1)$ required to contain the simplex will give extra space to help facilitate learning; Hill & Doucet (2007), Lange & Wu (2008), Wu & Lange (2010), Saberian & Vasconcelos (2011), Mroueh et al. (2012), and Wu & Wu (2012) (amongst others) discuss these simplices and their applications to classification. For the standard simplex, the targets are just the standard basis vectors, each of which consists of zeros for all but one entry. For both the regular and standard simplices,

$$\|t_1\| = \|t_2\| = \ldots = \|t_k\| = 1. \tag{1}$$

Given an input vector x of feature activations, we identify the target vector t_j that is nearest in the Euclidean distance to

$$z = \frac{y}{\|y\|}, \tag{2}$$

where

$$y = Ax \tag{3}$$

for an $m \times n$ matrix A determined via learning as discussed shortly. The index j such that $\|z - t_j\|$ is minimal is the index of the class to which we assign x. The classification is known as "linear" or "multi-linear" due to (3). The index j to which we assign x is clearly independent of the Euclidean norm of x due to (2), and the assignment is "scale-invariant" even if we rescale A by a nonzero scalar multiple.

To determine A, we first initialize all its entries to random numbers, then divide each entry by the Frobenius norm of A and multiply by the square root of the number of rows in A. We then conduct iterations of stochastic gradient descent as advocated by LeCun et al. (1998), updating A to $\tilde{A}$ on each iteration via

$$\tilde{A} = A - h \left(\frac{\partial c}{\partial A} \right)^{\top}, \tag{4}$$

where h is a positive real number (known as the "learning rate" or "step length") and c is the cost to be minimized that is associated with a vector chosen at random from among the input vectors and its associated vector x of feature activations,

$$c = \|z - t_j\|^2, \tag{5}$$

where t_j is the target for the correct class associated with x, and z is the vector-valued function of x specified in (2) and (3). Appendix A calculates $\partial c / \partial A$ explicitly.

As elaborated by LeCun et al. (1998), usually we combine stochastic gradient descent with back-propagation to update the entries of x associated with the chosen input, which requires propagating the gradient $\partial c / \partial x$ back into the network generating the feature activations that are the entries of x for the chosen input sample. We use the same learning rate from the classification stage throughout the network generating the feature activations. Fortunately, the Euclidean norm of the gradient $\partial c / \partial x$ is bounded independent of the scaling of A:

$$\left\| \frac{\partial c}{\partial x} \right\| \leq \frac{4\|A\|}{\|Ax\|}; \tag{6}$$

please note that scaling the matrix A by any nonzero scalar multiple has no effect on the right-hand side of (6) — the gradient propagating in backpropagation is independent of the size of A. Appendix B gives a proof of (6).

Critically, after every update as in (4), we rescale the matrix A: we divide every entry by the Frobenius norm of A and multiply by the square root of the number of rows in A. We use the rescaled matrix for subsequent iterations of stochastic gradient descent. Rescaling A yields precisely the same vector z in (2) and cost c in (5); together with the scale-invariance of the right-hand side of (6), rescaling ensures that the stochastic gradient iterations are effective and numerically stable for any learning rate h.

4 ROBUSTNESS

Combining (1) and the fact that the Euclidean norm of z from (2) is 1 yields that the cost c from (5) satisfies

$$0 \leq c \leq 4. \tag{7}$$

As reviewed by Hastie et al. (2009), the cost associated with classification via multinomial logistic regression is

$$r = \ln \left(\sum_{\ell=1}^{m} \exp(y^{(\ell)}) \right) - y^{(j)}, \tag{8}$$

where j is the index among 1, 2, ..., k of the correct class, and $y^{(1)}$, $y^{(2)}$, ..., $y^{(m)}$ are the entries of the vector y from (3), with $m = k$ for multinomial logistic regression. Whereas the cost c is bounded as in (7), the cost r from (8) is bounded only for positive values of $y^{(j)}$, growing linearly for negative $y^{(j)}$. Thus, c is more robust than r to outliers; logistic regression is less robust to outliers (including obviously mislabeled inputs).

5 NUMERICAL EXPERIMENTS

The present section provides a brief empirical evaluation of rescaling in comparison with the usual multinomial logistic regression, performing the learning for both via stochastic gradient descent (the learning is end-to-end, training the entire network — including both the convolutional network and the classification stage — jointly, with the same learning rate everywhere). We renormalize the parameters in the classification stage after every minibatch of 100 samples when rescaling (not with the multinomial logistic regression), as detailed in the last paragraph of Section 3 and the penultimate paragraph of the present section. The rescaled approach appears to perform somewhat better than multinomial logistic regression in all but Figure 2 for the experiments detailed in the present section.

Following LeCun et al. (1998), the architectures for generating the feature activations are convolutional networks (convnets) consisting of series of stages, with each stage feeding its output into the next (except for the last, which feeds into the classification stage). Each stage 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 convnets, and we append an additional feature activation to those obtained from the convnets, namely the standard deviation of the set of values of the pixels in the image. For each data set, we use two network architectures, where the second is a somewhat smaller variant of the first. We consider three data sets whose training properties are reasonably straightforward to investigate, with each set consisting of $k = 10$ classes of images; the first two are the usual CIFAR-10 and MNIST of Krizhevsky (2009) and LeCun et al. (1998). The third is a subset of the 2012 ImageNet data 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. CIFAR-10 contains 50,000 images in its training set and 10,000 images in its testing set. MNIST contains 60,000 images in its training set and 10,000 images in its testing set. The images in the MNIST set

Table 1: CIFAR-10 and MNIST, larger (†=1 for grayscale MNIST; †=3 for full-color CIFAR-10)

Stage	Input images	Output images	Kernel size	Input image size	Output image size
first	†	16	3×3	32×32	15×15
second	16	128	2×2	15×15	7×7
third	128	1024	2×2	7×7	3×3

Table 2: CIFAR-10 and MNIST, smaller (†=1 for grayscale MNIST; †=3 for full-color CIFAR-10)

Stage	Input images	Output images	Kernel size	Input image size	Output image size
first	†	16	3×3	32×32	15×15
second	16	64	2×2	15×15	7×7
third	64	256	2×2	7×7	3×3

are grayscale. The images in both the CIFAR-10 and ImageNet sets 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.

Tables 1–4 display the specific configurations we used. "Stage" specifies the positions of the indicated layers in the convnet. "Input images" specifies the number of images input to the given stage for each sample from the data. "Output images" 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 image size" specifies the size of the square grid of pixels constituting each input image. "Output image size" specifies the size of the square grid of pixels constituting each output image. Tables 1 and 2 display the two configurations used for processing both CIFAR-10 and MNIST. Tables 3 and 4 display the two configurations used for processing the subset of ImageNet described above.

Figures 1–6 plot the accuracies attained by the different schemes for classification while varying h from (4) (h is the "learning rate," as well as the length of the learning step relative to the magnitude of the gradient) and varying the dimension m of the space containing the simplex targets; m is the number of rows in A from (3) and (4). In each figure, the top panel — that labeled "(a)" and "rescaled" — plots the error rates for classification using rescaling, with the targets being the vertices on the hypersphere of a regular simplex; the middle panel — that labeled "(b)" and "logistic" — plots the error rates for classification using multinomial logistic regression; the bottom panel — that labeled "(c)" and "best of both" — plots the error rates for the best-performing instance from the top panel (a) together with the best-performing instance from the middle panel (b). All error rates refer to performance on the test set. The label "epoch" for the horizontal axes refers, as usual, to the number of training sweeps through the data set, as reviewed in the coming paragraph.

As recommended by LeCun et al. (1998), we learn via (minibatched) stochastic gradient descent, with 100 samples per minibatch; rather than updating the parameters being learned for randomly selected individual images from the training set exactly as in Section 3, we instead 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." The horizontal axes in the figures count the number of epochs.

In the experiments of the present section, the accuracies attained using the scale-invariant classification stage are comparable to (if not better than) those attained using the usual multinomial logistic regression. The scale-invariant classification stage is stable for all values of h, that is, for all learning rates.

ACKNOWLEDGMENTS

We would like to thank Léon Bottou and Rob Fergus for critical contributions to this project.

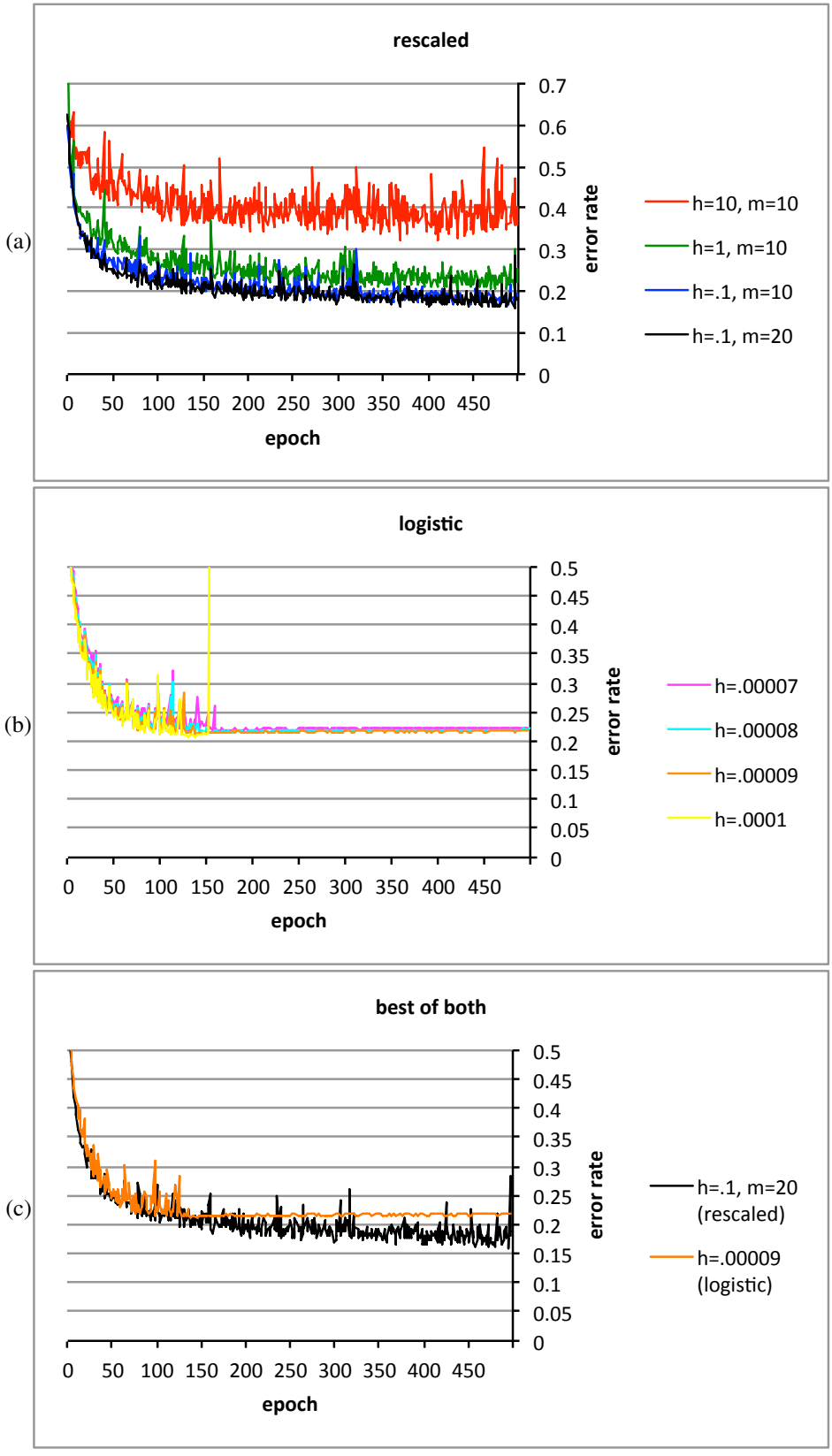

Figure 1: CIFAR-10 — larger architecture

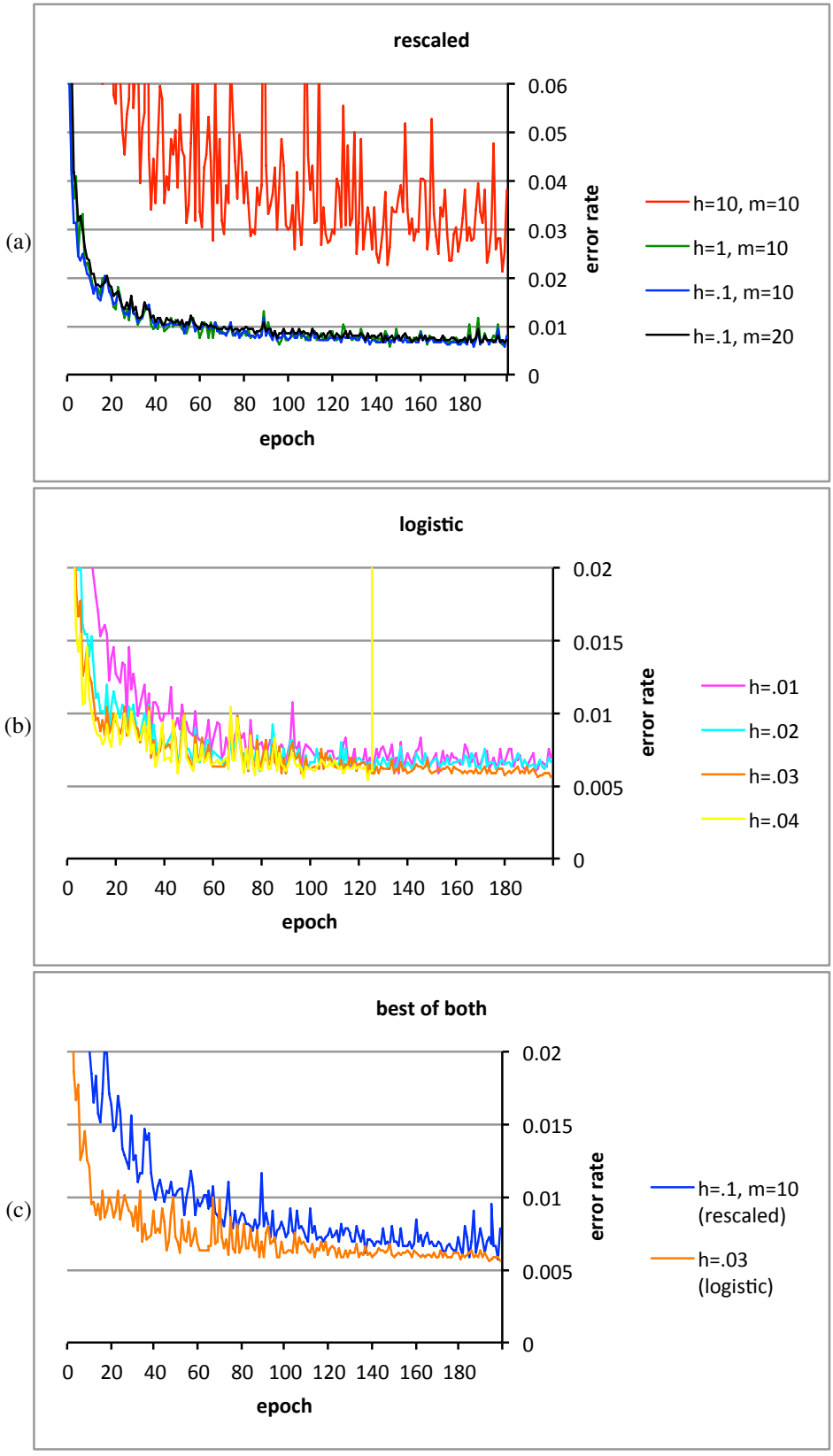

Figure 2: MNIST — larger architecture

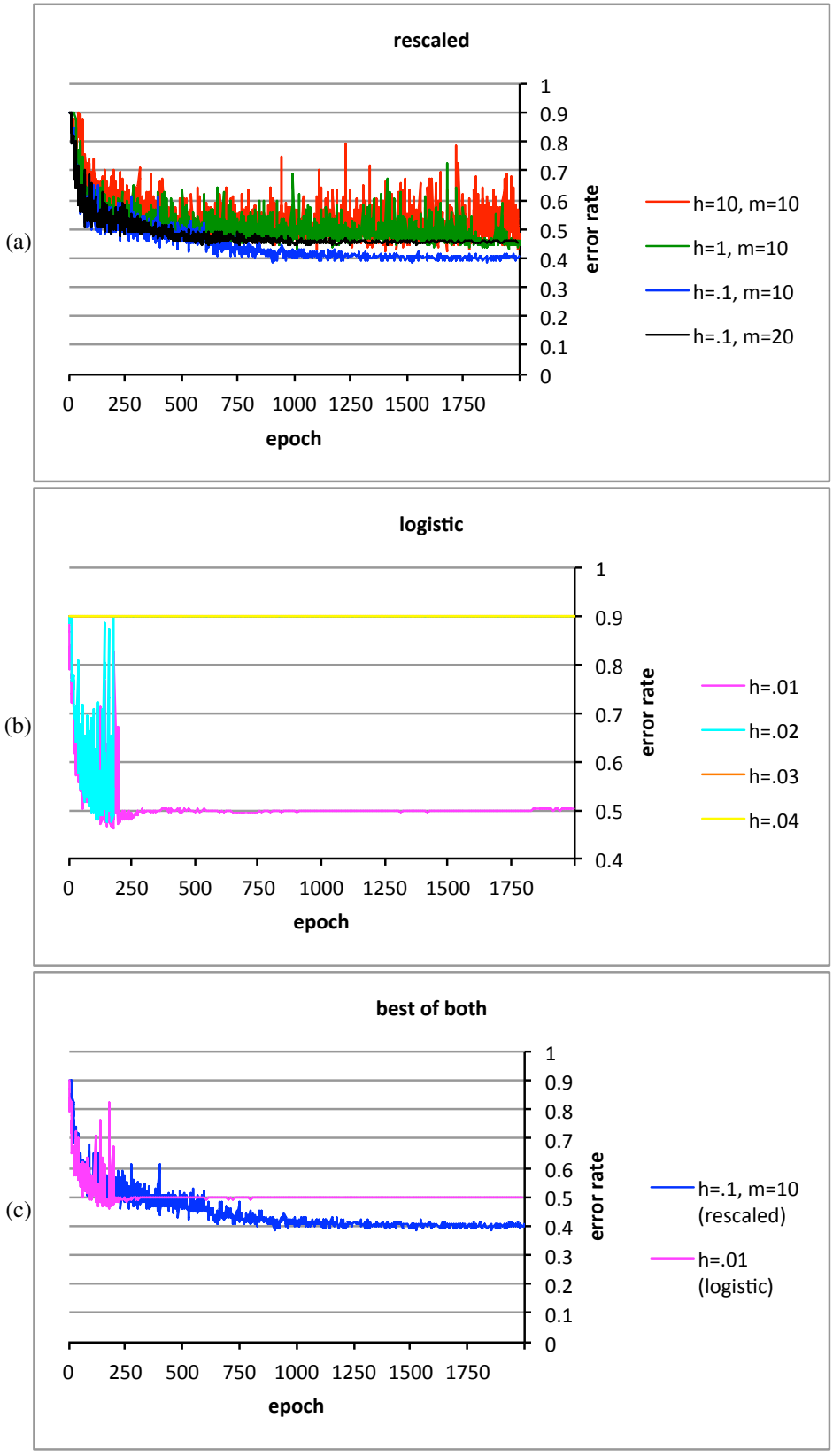

Figure 3: ImageNet subset — larger architecture

Table 3: ImageNet subset, larger (full color, with 3 color channels)

Stage	Input images	Output images	Kernel size	Input image size	Output image 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	1024	3×3	14×14	6×6

Table 4: ImageNet subset, smaller (the smaller number is italicized in this table)

Stage	Input images	Output images	Kernel size	Input image size	Output image 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

REFERENCES

Carandini, Matteo and Heeger, David J. Normalization as a canonical neural computation. *Nature Reviews Neurosci.*, 13(1):51–52, 2012.

Hastie, Trevor, Tibshirani, Robert, and Friedman, Jerome. *Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009.

Hill, Simon I. and Doucet, Arnaud. A framework for kernel-based multi-category classification. *J. Artificial Intel. Research*, 30:525–564, 2007.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: accelerating deep network training by reducing internal covariate shift. Technical Report 1502.03167, arXiv, 2015.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical Report Master's Thesis, University of Toronto Department of Computer Science, 2009.

Lange, Kenneth and Wu, Tong Tong. An MM algorithm for multicategory vertex discriminant analysis. *J. Comput. Graph. Statist.*, 17(3):527–544, 2008.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

Magnus, Jan R. and Neudecker, Heinz. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley and Sons, 3rd edition, 2007.

Mroueh, Youssef, Poggio, Tomaso, Rosasco, Lorenzo, and Slotine, Jean-Jeacques. Multiclass learning with simplex coding. In *Advances in Neural Information Processing Systems*, volume 25, pp. 2789–2797. Curran Associates, 2012.

Russakovsky, Olga, Deng, Jia, Su, Hao, Kruse, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet large scale visual recognition challenge. Technical Report 1409.0575v3, arXiv, 2015.

Saberian, Mohammad J. and Vasconcelos, Nuno. Multiclass boosting: theory and algorithms. In *Advances in Neural Information Processing Systems*, volume 24, pp. 2124–2132. Curran Associates, 2011.

Wu, Tong Tong and Lange, Kenneth. Multicategory vertex discriminant analysis for high-dimensional data. *Annals Appl. Statist.*, 4(4):1698–1721, 2010.

Wu, Tong Tong and Wu, Yichao. Nonlinear vertex discriminant analysis with reproducing kernels. *Statist. Anal. Data Mining*, 5(2):167–176, 2012.

A CALCULATION OF THE UPDATE IN (4)

In these appendices, we leverage matrix Calculus as elaborated by Magnus & Neudecker (2007). The chain rule specifies that the gradient of c with respect to A is

$$\frac{\partial c}{\partial A} = \frac{\partial c}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial A} \tag{9}$$

The gradient of c from (5) with respect to z is

$$\frac{\partial c}{\partial z} = 2(z - t_j)^\top. \tag{10}$$

The Jacobian of z from (2) with respect to y is

$$\frac{\partial z}{\partial y} = \frac{I - zz^\top}{\|y\|}, \tag{11}$$

where I denotes the identity matrix and $zz^\top$ is the orthogonal projection onto z — due to (2), $\|z\| = 1$. The derivative of y from (3) with respect to A is

$$\frac{\partial y}{\partial A} = x \otimes I, \tag{12}$$

where I is the identity matrix and $\otimes$ is the tensor product; $x \otimes I$ is a vector of the same height as x, with each entry being the identity matrix I times the corresponding entry of x. Combining (3) and (9)–(12) yields

$$\frac{\partial c}{\partial A} = \frac{2x(z^\top t_j\, z - t_j)^\top}{\|Ax\|}, \tag{13}$$

where z is defined in (2).

B PROOF OF THE BOUND IN (6)

In these appendices, we leverage matrix Calculus as elaborated by Magnus & Neudecker (2007). To prove (6), we calculate $\partial c/\partial x$ using the chain rule

$$\frac{\partial c}{\partial x} = \frac{\partial c}{\partial z}\frac{\partial z}{\partial y}\frac{\partial y}{\partial x}. \tag{14}$$

The gradient of c from (5) with respect to z is

$$\frac{\partial c}{\partial z} = 2(z - t_j)^\top. \tag{15}$$

The Jacobian of z from (2) with respect to y is

$$\frac{\partial z}{\partial y} = \frac{I - zz^\top}{\|y\|}, \tag{16}$$

where I denotes the identity matrix and $zz^\top$ is the orthogonal projection onto z — due to (2), $\|z\| = 1$. Of course, the Jacobian of y from (3) with respect to x is

$$\frac{\partial y}{\partial x} = A. \tag{17}$$

The fact that $I - zz^\top$ is an orthogonal projection yields that

$$\|I - zz^\top\| \leq 1. \tag{18}$$

Combining (1)–(3) and (14)–(18) yields (6), as desired.

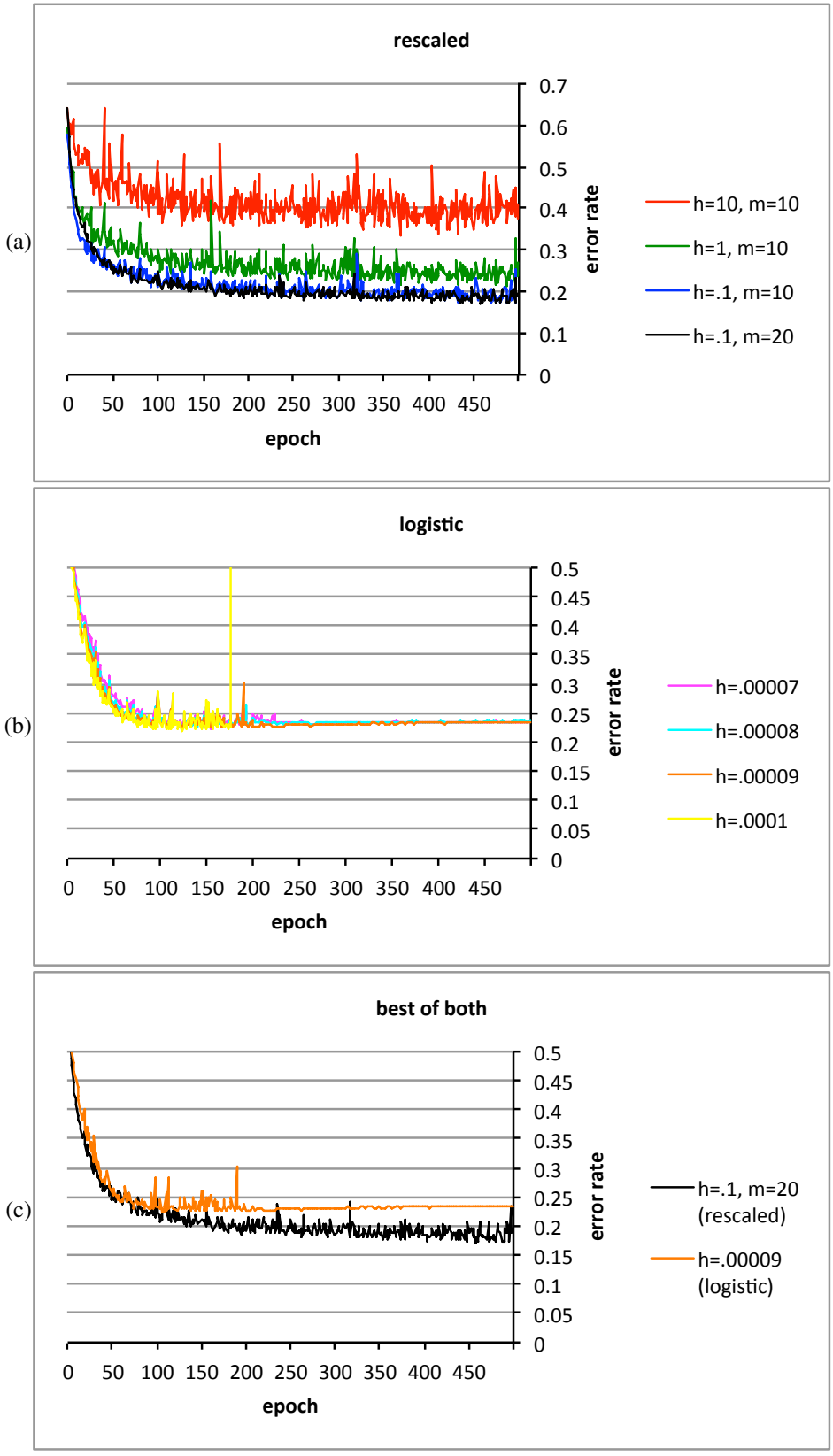

Figure 4: CIFAR-10 — smaller architecture

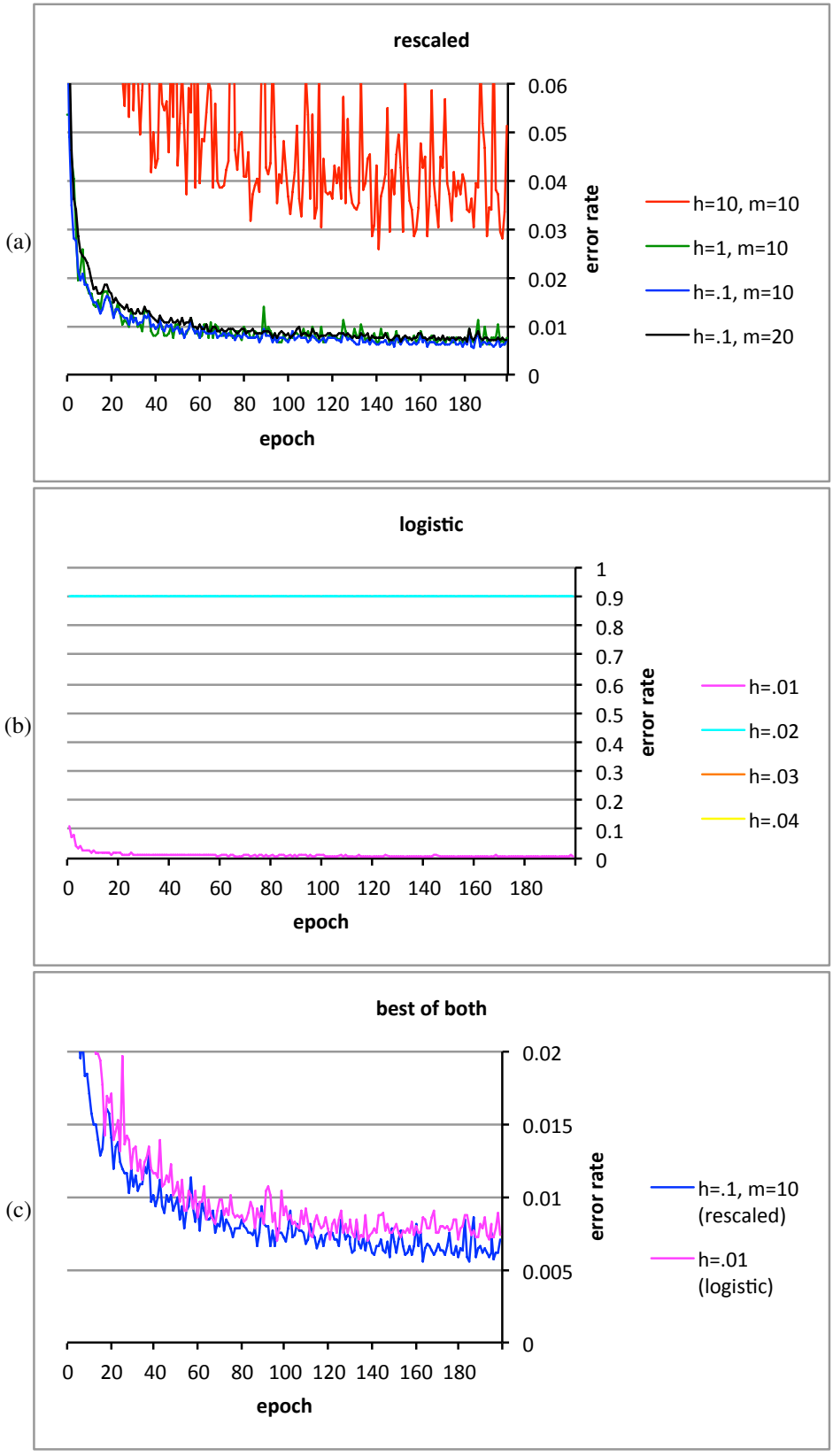

Figure 5: MNIST — smaller architecture

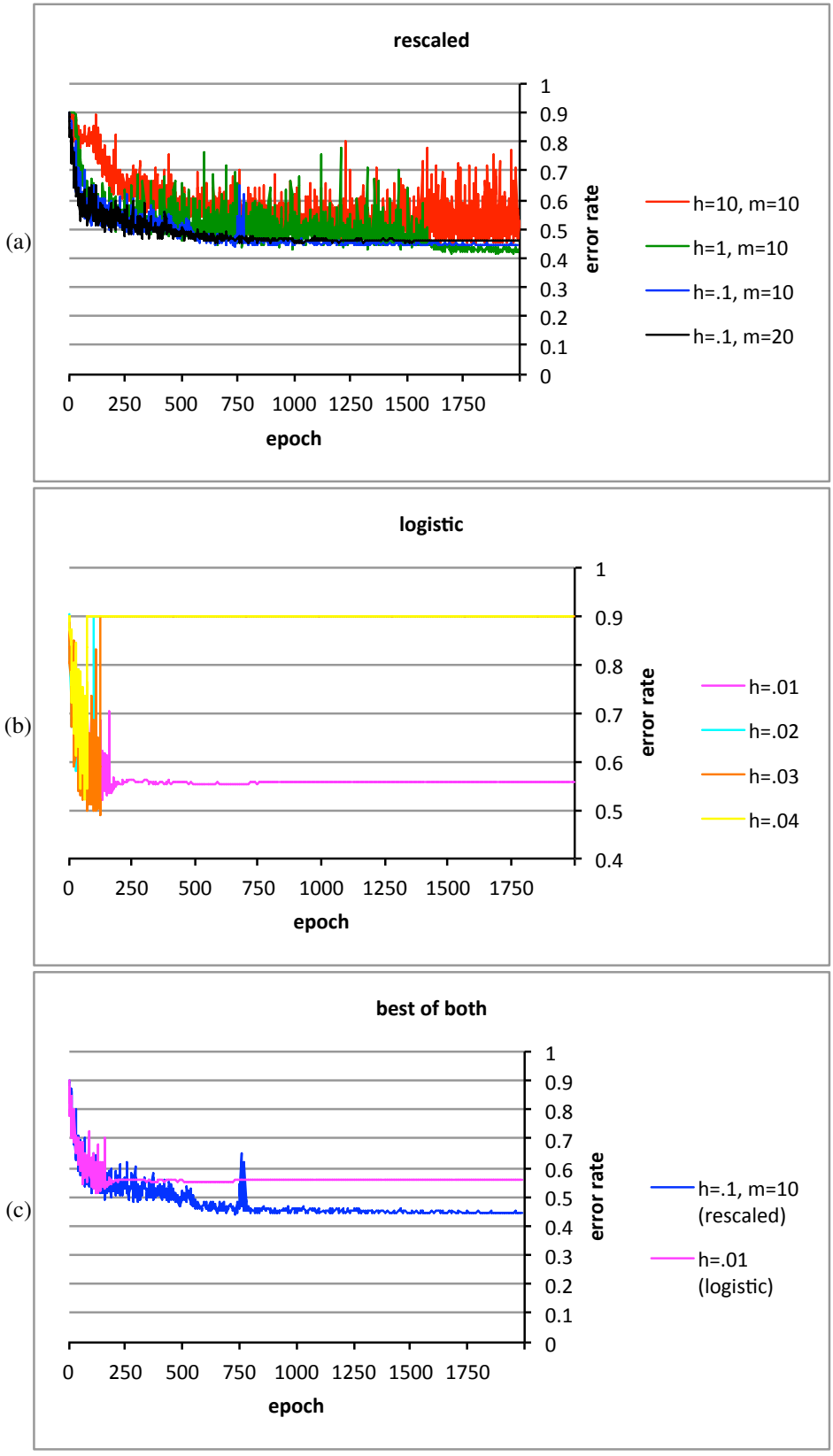

Figure 6: ImageNet subset — smaller architecture