Computational Analysis of the Bidirectional Activation-based Learning in Autoencoder Task

Peter Csiba and Igor Farkaš
Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava
Mlynská dolina, 84248 Bratislava, Slovak Republic
Email: farkas@fmph.uniba.sk

***Abstract*—We use computational simulations to analyse the behavior of the recently proposed Bidirectional Activation-based Learning algorithm (BAL) which was inspired by the Generalized Recirculation algorithm (GeneRec). Both algorithms avoid biologically implausible backpropagation of the error signal, and instead use propagation of neuron activations, which drive the weight updates, using only local variables. We take a closer look at the 4-2-4 autoencoder task for which, despite the task simplicity, reliable convergence could not be achieved by either of the two models. We propose the learning mode with two, significantly different, learning rates (BAL2) that leads to considerably more successful task learning. We also analyze various factors, related to hidden activations, that contribute to further increase of the learning success. In addition, we test BAL2 also on the large scale database of handwritten digits, in which it yields relatively good performance.**

I. Introduction

The well-known error backpropagation (BP) learning [1] is known to be biologically implausible due to the mechanism of error propagation and the nonlocal learning rule. As a remedy, O'Reilly [2] designed the Generalized Recirculation (GeneRec) algorithm that avoids the computation of error derivatives, but can compute the error gradient, using only local variables (unlike BP). GeneRec was designed as an extension of Hinton and McClelland's model [3] based on recirculation between two layers of units (visible and hidden) with symmetric weights, which was restricted to autoassociation. To make it work, Hinton and McClelland used a four-stage activation update process. Unlike the recirculation algorithm, GeneRec is applied to a three-layer network using bidirectional interaction (only) between two layers of units (hidden and output) in a two-phase activation update process, and can be trained to learn arbitrary input–output mapping.

Recently, we proposed a bidirectional activation-based learning (BAL), which is based on the GeneRec model, but unlike it, is completely symmetrical regarding the activation propagation and the weight update rules [4]. Our motivation for designing the BAL model, that also uses only local variables in learning rule, was to implement it in our robotic mirror neuron system model, that is assumed to require the bidirectional mapping between high-level sensory and motor representations [5]. The behavior of BAL was tested in three simulation experiments, of which the simplest task (4-2-4 autoencoder) did not result in a reliable convergence of the algorithm peaking at around 65% success rate (as opposed to 90% for GeneRec and 100% for error backpropagation). Hence, we looked at this phenomenon trying to understand the problem. Here we also test the models on a large scale database of hand-written digits (MNIST).

This paper is organized as follows. In Section 2, we describe relevant models and the methods used. In Section 3 we present results on both data sets, with a special focus on the autoencoder task. Section 4 concludes the paper.

II. Models and methods

A. Generalized Recirculation

The Generalized Recirculation algorithm (GeneRec) applies to a three-layer network with full connectivity between layers whose activation rules are described in Table I.

TABLE I. Equilibrium network variables in GeneRec.

Layer	Phase	Net Input	Activation
Input (s)	$-$	-	s_i = stimulus input
Hidden (h)	$-$	$\eta_j^- = \sum_i w_{ij}^{\mathrm{IH}} s_i + \sum_k w_{kj}^{\mathrm{OH}} o_k^-$	$h_j^- = \sigma(\eta_j^-)$
	+	$\eta_j^+ = \sum_i w_{ij}^{\mathrm{IH}} s_i + \sum_k w_{kj}^{\mathrm{OH}} o_k^+$	$h_j^+ = \sigma(\eta_j^+)$
Output (o)	$-$	$\eta_k^- = \sum_j w_{jk}^{\mathrm{HO}} h_j$	$o_k^- = \sigma(\eta_k^-)$
	+	-	o_k^+ = target output

GeneRec uses *plus* and *minus* phases during activation propagation and is able to learn, as error backpropagation, arbitrary input–output mappings [2]. The activation flow is depicted in Figure 1. The model has reciprocal connectivity between hidden and output layer. Hence it uses three weight matrices W^{IH}, W^{HO} and W^{OH} for the input–hidden, hidden–output and output–hidden weights, respectively. The activation flow starts in minus phase, when the stimulus s_i is presented. Note that the net input term at the hidden layer includes the input from both visible layers before applying the sigmoid activation function $\sigma(\eta) = 1/(1+\exp(-\eta))$. Output units produce activations o_k^- in minus phase but can also be clamped to target activations o_k^+ at the onset of plus phase. Input units can only deliver stimuli s_i at the onset of minus phase. The model requires the computation of *equilibrium* activation states which is achieved using an iterative method

$$a_m(t+1) = \sigma(\sum_n w_{mn} a_n(t)) \qquad (1)$$

where $a_m(t)$ is the activation of m-th unit at discrete time t. The equation (1) is iterated while $|a_m(t+1) - a_m(t)| > \epsilon$ for some unit m (hidden or output) and chosen $\epsilon > 0$.

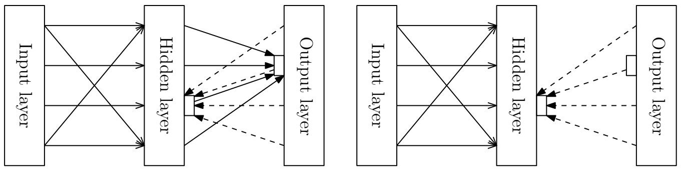

Fig. 1. Depicting the minus (left) and plus (right) phases of GeneRec defined in Table I. Taken from [6].

The GeneRec learning rule derived for all three weight matrices has the form

$$\Delta w_{ij} = \lambda a_i^- (a_j^+ - a_j^-), \tag{2}$$

where a_i^- denotes the presynaptic and a_j^- the postsynaptic unit activation in minus phase, a_j^+ is the postsynaptic activation in plus phase and λ denotes the learning rate. For updating the concrete weight matrices, the activation values are set correspondingly. For example, for updating W^{HO}, we set $a_i^- = h_i^-$, $a_j^- = o_j^-$ and $a_j^+ = o_k^+$.

It was proven [2] that GeneRec converges if the learning rule (2) is a valid approximation to the error derivate and the weights are symmetric, i.e. $W^{\mathrm{HO}} = (W^{\mathrm{OH}})^\top$. O'Reilly [2] proposed two more modifications, the *midpoint* learning rule

$$\Delta w_{ij} = \lambda \frac{1}{2}(a_i^- + a_i^+)(a_j^+ - a_j^-) \tag{3}$$

and the *symmetric* learning rule

$$\Delta w_{ij} = \lambda (a_j^+ a_i^- - a_j^- a_i^+ - 2a_j^- a_i^-) \tag{4}$$

which aims to preserve the weight symmetry. By combining (3) and (4), O'Reilly got the rule

$$\Delta w_{ij} = \lambda (a_i^+ a_j^+ - a_i^- a_j^-) \tag{5}$$

which is formally equivalent to the *Contrastive Hebbian learning* (CHL), earlier introduced in the context of Hopfield networks [7].

B. Bidirectional activation-based learning algorithm

The design of Bidirectional Activation-based Learning algorithm (BAL, [4]) was motivated by the biological plausibility of GeneRec. BAL inherits the learning rule (2) of GeneRec and also the two activation phases. But unlike GeneRec, BAL aims to learn bidirectional mapping between inputs and outputs and for this purpose it uses four weight matrices (W^{IH}, W^{HO}, W^{OH} and W^{HI}). The design of BAL is completely symmetric as shown in Table II.

TABLE II. ACTIVATION PHASES AND STATES IN BAL.

Layer	Phase	Net Input	Activation
x	F	-	x_i^{F} = forward stimulus
h	F	$\eta_j^{\mathrm{F}} = \sum_i w_{ij}^{IH} x_i^F$	$h_j^{\mathrm{F}} = \sigma(\eta_j^{\mathrm{F}})$
y	F	$\eta_k^{\mathrm{F}} = \sum_j w_{jk}^{HO} h_j^F$	$y_k^{\mathrm{F}} = \sigma(\eta_k^{\mathrm{F}})$
y	B	-	y_k^{B} = backward stimulus
h	B	$\eta_j^{\mathrm{B}} = \sum_k w_{kj}^{OH} y_k^{\mathrm{B}}$	$h_j^{\mathrm{B}} = \sigma(\eta_j^{\mathrm{B}})$
x	B	$\eta_i^{\mathrm{B}} = \sum_j w_{ji}^{HI} h_j^{\mathrm{B}}$	$x_i^{\mathrm{B}} = \sigma(\eta_i^{\mathrm{B}})$

Therefore, we avoid input-output notation of layers as used in GeneRec, because in our case not only the output can be evoked by input presentation, but also vice versa. Hence, instead of minus and plus phases, we rather use *forward* and *backward* phases. This brings us to a different notation where a^{F} denotes forward activations (pass) and a^{B} denotes backward activations. Layers **x** and **y** are *visible* and layer **h** is hidden. During the forward pass, the **x** units are clamped to $\mathbf{x}^{\mathrm{F}}$ and we get the activations $\mathbf{x}^{\mathrm{F}} \rightarrow \mathbf{h}^{\mathrm{F}} \rightarrow \mathbf{y}^{\mathrm{F}}$. During the backward pass, the **y** units are clamped to $\mathbf{y}^{\mathrm{B}}$ and we get the activations $\mathbf{y}^{\mathrm{B}} \rightarrow \mathbf{h}^{\mathrm{B}} \rightarrow \mathbf{x}^{\mathrm{B}}$.

The mechanism of weights update partially matches that of GeneRec. Each weight in BAL network (i.e. belonging to one of the four weight matrices) is updated using the same learning mechanism, according to which the weight difference is proportional to the product of the presynaptic (sending) unit activation a_p and the difference of postsynaptic (receiving) unit activations a_q, corresponding to two activation phases (F and B, in particular order). Namely, weights in **x**-to-**y** direction (belonging to **h** and **y** units) are updated as

$$\Delta w_{pq}^{\mathrm{F}} = \lambda\ a_p^{\mathrm{F}} (a_q^{\mathrm{B}} - a_q^{\mathrm{F}}), \tag{6}$$

where, as in the GeneRec algorithm, a_p^{F} denotes the presynaptic activity, a_q^{F} is the postsynaptic activity, and a_q^{B} denotes the postsynaptic activity from the opposite phase (**y**-to-**h**). Analogically, the weights in **y**-to-**x** direction (belonging to **h** and **x** units) are updated as

$$\Delta w_{pq}^{\mathrm{B}} = \lambda\ a_p^{\mathrm{B}} (a_q^{\mathrm{F}} - a_q^{\mathrm{B}}) \tag{7}$$

All units have trainable thresholds (biases) that are updated in the same way as regular weights (being fed with a constant input 1).

C. BAL with two learning rates

In this paper, we propose and analyze the *two learning rates* version of the model (BAL2, [8]), that uses two separate learning rates for different layers, *lambda hidden* (λ_{H}), for weight matrices W^{IH} and W^{OH} and *lambda visible* (λ_{V}), for weights W^{HI} and W^{HO}. Both λ_{H} and λ_{V} are held constant during learning. Their names are derived from the layers on which the error term $(a_j^+ - a_j^-)$ is computed.

Our simulations show that setting $\lambda_{\mathrm{H}} \ll \lambda_{\mathrm{V}}$ leads to significantly better performance in comparison to the standard BAL model. The intuition behind it as follows: because $\lambda_{\mathrm{H}} \ll 1$, W^{IH} and W^{OH} are updated only minimally and also activations h^{F} and h^{B} change only a little and hence $|h^{\mathrm{F}} - h^{\mathrm{B}}|$ converges to zero more slowly. Thus the error terms $(y_j^{\mathrm{B}} - y_j^{\mathrm{F}})$ and $(x_j^{\mathrm{F}} - x_j^{\mathrm{B}})$ from the BAL learning rule for W^{HI} and W^{HO} have a longer lasting impact on the weight change with *non-constant hidden activations*. The importance of suitable hidden activations was further confirmed by the candidate selection experiment (described below).

Most of the previous work regarding different learning rates is based on *dynamic learning rate* (DLR) model [9]. The aim of DLR is to compute the *best* learning rate in terms of successful convergence and avoidance of local minima [10]. There have been several approaches proposed how to achieve this. Most of them have individual learning rates for each weight in the network which could change in time. Some approaches precompute learning rates [11] while others adapt learning rates dynamically through the training process [12], [13], [14]. According to [15], picking a different learning rate for each weight can improve convergence. Our simpler approach only uses two learning rates, so BAL2 model is in a sense probably unique in terms of the number of learning rates.

D. Candidate selection

We introduced the candidate selection approach to test if some particular network *features*, related to the hidden layer activations, have an impact on the learning success of BAL2. The only difference between standard BAL or BAL2, and the candidate selection is that before the training phase, a number of networks are randomly generated from which a *best candidate network* is selected, based on the considered feature.

We proposed and investigated several features: (a) the average distance between all hidden activations, $dist_{\mathrm{H}}$, (b) the average distance between corresponding forward and backward hidden activations, (c) the average distance between corresponding forward and backward visible activations, (d) the average weight of all 4 weight matrices, (e) convexity of hidden activations, and (f) the maximal difference between activations in the last two iterations. The impact of all these features was tested using a fitted linear regression model. The training dataset was created by generating standard BAL networks, measuring feature values before the training phase and adding the success rate label equal to $1 - bitSucc^F$ after the training phase.

Using linear regression, we found that the most important feature was $dist_{\mathrm{H}}$. This discovery actually led to the introduction of the BAL2 model. Another important feature turned out to be the convexity (the boolean value) which reveals whether the four hidden activations (points in 2D space, in case of two hidden units) form a convex polygon, i.e. whether each of those is linearly separable from the others, to be correctly associated with an output unit. This is a necessary condition for perfect success rate.

III. Simulations

Following [4], we measure two properties of a model, the success rate and the convergence time.

Success rate quantifies output accuracy. Before comparing the computed outputs on both visible layers to targets, the activations y^F and x^B are classified using a threshold 0.5. We define two measures of success: *Bit success (bitSucc)* quantifies the fraction of correctly predicted bits, whereas *pattern success (patSucc)* quantifies the fraction of correctly predicted patterns (i.e. all bits in a pattern have to be correctly classified) evaluated for both directions.

Convergence time denotes the number of epochs before we stop training. In case of autoencoder, we use two options: the preset number of epochs or when successful learning was achieved ($patSucc^F = patSucc^B = 1$). In case of the digits dataset we decided to stop the training if $patSucc^F$ did not increase during 3 consecutive epochs.

Datasets. As mentioned in the introduction, we test BAL2 on two different tasks. The first one is the 4-2-4 autoencoder task which, despite its simplicity, resisted reliable convergence in case of both GeneRec and BAL, so the primary motivation was to discover the reasons behind it. The second is the classification task of the well-known large-scale database of handwritten digits [16], first analysed in [17]. We chose this dataset, because it is big and complex enough, well-known such that BAL2 performance can be compared with other models. In all simulations, following [2], we initialize the network weights in both tasks to small values from the normal distribution $\mathcal{N}(0; 1/\sqrt{n_I + 1})$, where n_I denotes the input data dimension.

A. The 4-2-4 Autoencoder

The encoder task only contains four four–dimensional patterns with one-hot numerical representation (i.e. one unit with value 1, all others 0). We use two units in the hidden layer yielding the 4-2-4 architecture (with 3 hidden units, BAL learns the task reliably). In the case of BAL, only 60–65% *patSucc* was achieved for a range of learning rates (with a sharp drop in performance towards 0 for λ>2). This behavior served as a motivation to look for improvement. As mentioned in Section 2, the solution to the problem of convergence can be accessed using two different learning rates. Hence, we systematically analyse the performance of BAL2 model depending on parameters λ_{H} and λ_{V}, using 2D plots. In 2D plot, we display the mean performance of 500 networks for each pair (λ_{V}, λ_{H}). The networks were trained while *patSucc*$^{\mathrm{F}}$ < 1 or $epoch < Epoch_{\max}$ = 100,000.

In Table III we can see the comparison of the most important models. We achieved an improvement of *patSucc*$^{\mathrm{F}}$ from 62.7% to 93.1% by using BAL2. This result was further improved to 99.86% by setting networks with candidate selection (i.e. almost guaranteed convergence). This confirmed the finding that hidden distance and convexity of hidden representations are important features of BAL. Of the GeneRec-based models, the original GeneRec version and the midpoint version show good performance ($> 90\%$), whereas the symmetric and CHL version work inferior to BAL. Note that BAL-based models require correct mapping in both directions (not required in original autoencoder), which makes the task more difficult.

TABLE III. Comparison of different models on the 4-2-4 encoder task. Results for BP through CHL are taken from [2].

Algorithm	λ_{H}	λ_{V}	*patSucc*$^{\mathrm{F}}$	Epochs
BP	2.4	2.4	100%	60
GR	0.6	0.6	90%	418
GR Sym	1.4	1.4	56%	88
GR Mid	2.4	2.4	92%	60
CHL	1.2	1.2	56%	77
BAL	0.9	0.9	62.7%	5136
BAL2	0.0002	500	93.1%	5845
BAL2 Can	0.0002	500	99.86%	150.4

If we want to compare execution time based on epochs in Table III, then we must be aware of that GeneRec epochs take longer than those of other models. This is because of the recirculation step, for which a number of iterations is needed

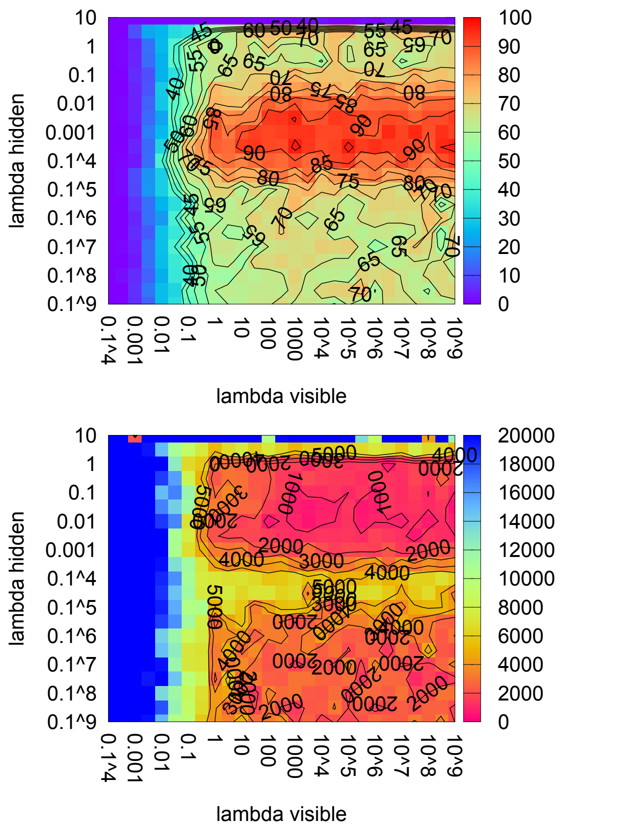

Fig. 2. BAL2 success rate (top) and convergence time (bottom) needed for successful networks on the autoencoder task. The best network achieved 96.5% with $\lambda_H = 0.0003$ and $\lambda_V = 1000$.

for activations to settle (up to 30). Thus the 418 epochs of GeneRec roughly correspond to 5845 epochs of BAL2 in terms of execution time.

In Figure 2 we compare the success rate for a range of λ_{V} and λ_{H}. It is interesting that the subspace with best achieving networks is organized around the horizontal midline $[(10, 0.001), (10^9, 0.001)]$. That means that the performance mainly depends on λ_{H} and for a fairly long range of λ_{V}.

In the corresponding convergence plot, the elongated *peak* occurs around the midline $[(0.01, 0.0001), (10^9, 0.0001)]$, whose position is unclear (with respect to the plot of success rate). Maybe it is related to $Epoch_{\max}$ and the fact that we calculated epochs only from successful networks. Therefore, successful networks having $\lambda_H < 10^{-6}$ tend to converge faster using $\lambda_{\mathrm{V}}>1$, because otherwise they would fail to converge due to $\lambda_H \cdot Epoch_{\max} < 1$.

There is a small inconsistency between Table III, with 93.12% success rate for BAL2, and Figure 2 where it was 96.5%. This is because in the first case, the average performance of 10000 networks was used. This result is more likely to mirror the reality, because in the second case only 200 networks were used for a particular (λ_V, λ_H) pair. As there were about 50 candidates for best success rate, it was likely that some of them performed better than average.

In Figure 3 and 4 we show forward hidden activations of example networks of BAL and BAL2 learning, respectively.

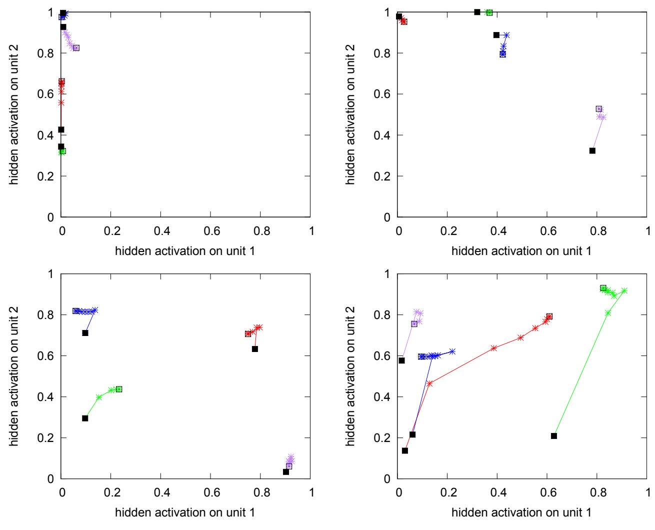

Fig. 3. BAL hidden forward activations corresponding to four input pattern in the autoencoder task during learning. The top row shows examples of unsuccessful networks and the bottom row shows successful ones. Only the first $\approx$ 100 epochs led to changes in hidden unit activations.

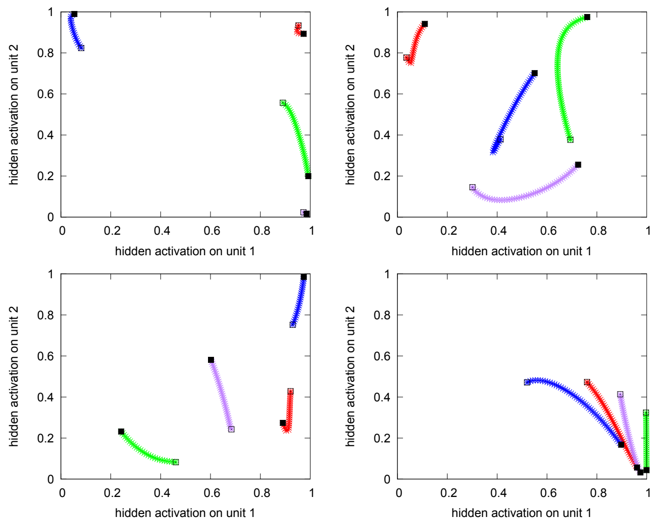

Fig. 4. BAL2 hidden forward activations corresponding to four patterns in the autoencoder task during learning. The top row shows examples of unsuccessful networks and the bottom row shows successful networks. Only the first $\approx$ 10000 epochs led to changes in hidden unit activations.

Each color represents the forward hidden representation of one of the four inputs in the *4-2-4 encoder* task. The plotted activations start at *epoch* $= 0$ depicted with black squares and continue as shown. The main difference between BAL2 and BAL is the speed of activation change. For BAL, we use $\lambda = 0.6$, which is several orders of magnitude different from both λ_{H} and λ_{H}. Another observation is that after some initial steps BAL tends to stop the activation change, due to settling $\|\mathbf{h}^{\mathrm{F}} - \mathbf{h}^{\mathrm{B}}\| \approx 0$.

Another source of the error that we investigated (and explained in Section II-D) could be *non-convex* hidden activation initializations. In the beginning, the weight matrices are initialized at random and that leads to random hidden activations. And if the hidden activations are also non-convex

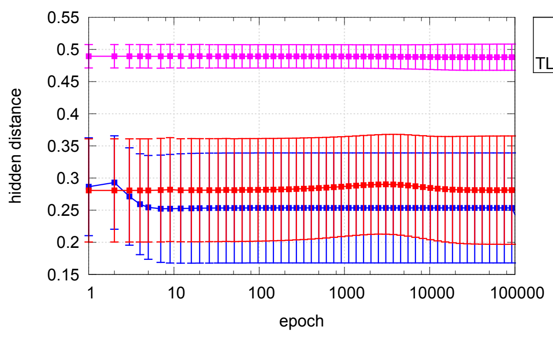

Fig. 5. Comparison of $dist_{\mathrm{H}}$ evolution for the 4-2-4 autoencoder task in three models: BAL, BAL2 (TLR) and BAL2 with candidate selection.

in the end, then it is impossible to perfectly classify on the hidden–to–visible layer due the linear separability theorem. Therefore if the network had non-convex hidden activations in the beginning, then it must *escape* the non-convex hidden-state organization for successful convergence. Figure 5 reveals that candidate selection indeed picks networks with greater $dist_{\mathrm{H}}$. We can observe that $dist_{\mathrm{H}}$ stagnates through the training phase. This confirms the importance of proper weight initialization.

As with generalization of BAL to BAL2, we tried the generalization of GeneRec using two different learning rates. The results in Figure 6 show that the highest success rate of around 90% is obtained for $\lambda \approx 0.1$ to 1, with no increase in the success rate compared to λ_{V}=λ_{H}, i.e. the original GeneRec.

The results suggest a hypothesis why BAL2 outperforms BAL on the 4-2-4 autoencoder task. The reason is that the hidden activations settle before W^{HO} and W^{HI} manage to tune to optimal values. This is explained by the fact that forward and backward hidden activations become identical too early. Moreover, the weight initialization can help this. In summary, the first issue is solved by setting $\lambda_H \ll \lambda_V$ what increases the number of epochs to the training phase. The second issue is solved by candidate selection which prevents initializing hidden activations too close to each other.

B. Handwritten digits

We performed tests with BAL2 on a high-dimensional graphical task using *handwritten digits* dataset and compared it to other known models. The MNIST database consists of 42000 samples of 28×28 grayscale images of single digits. As in the previous simulations, we trained the networks for range of λ_{H} and λ_{V} values to find the best parameters. Then we analysed the network with the best parameters.

Before the training we split the dataset to the training set with 38000 samples and test set with 4000 samples. Then we trained the networks on the train set and evaluated them on the test set. We set $Epoch_{\max} = 20$ and the training was stopped if *patSucc*$^{\mathrm{F}}$ was not increased for 3 successive epochs. The network architecture 784–300–10 was chosen, since results for BP with such architectures exist. Note that for the final classification we chose the unit with the maximal activation (analogically to softmax).

We confirmed that BAL2 could learn the high-dimensional task quite successfully, as shown in Figure 7. The properties

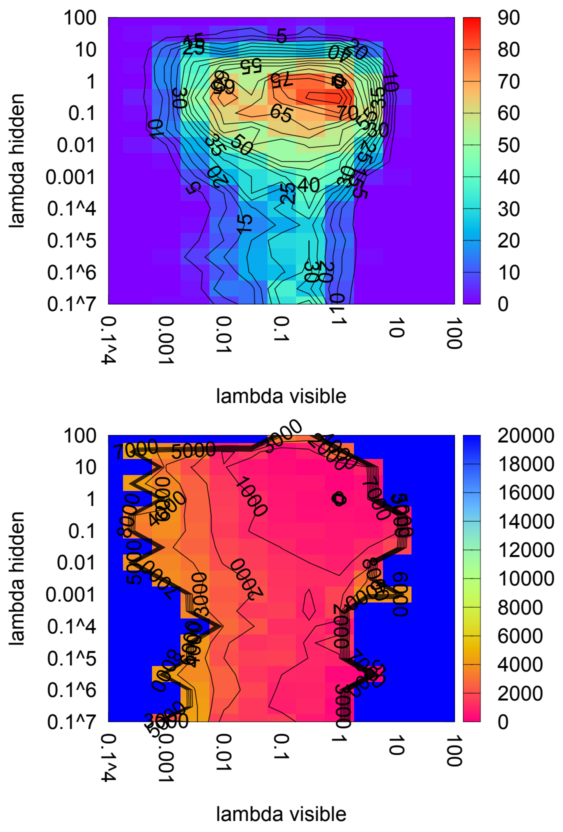

Fig. 6. GeneRec success rate and convergence time on the autoencoder task. The best result 83% was achieved with $\lambda_H = 0.3$ and $\lambda_V = 1$.

of the plot are similar to the autoencoder case, even though in the case of digits a wider plateau of best models spans several orders of magnitudes of both learning rates. However, it again holds that $\lambda_H \ll \lambda_V$ and the success space changes smoothly in parameters. The main difference is that the magnitude of values of both learning rates is smaller; concretely, $\lambda_{\mathrm{H}} < 10^{-6}$ and $\lambda_{\mathrm{V}} \approx 10^{-2} \sim 10^{2}$. This leads to a suggestion that for higher dimensional tasks, which usually also have more samples, lower values of learning rates should be chosen.

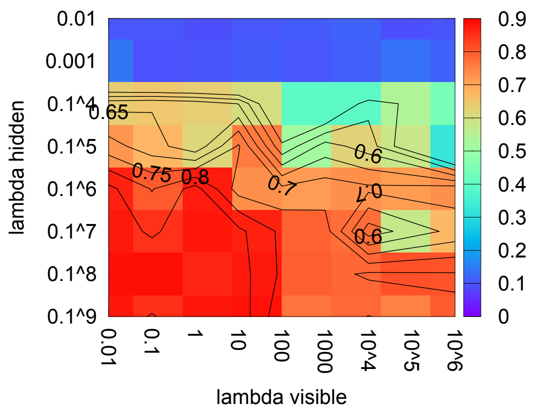

Fig. 7. BAL2 performance on the digits task. The highest *patSucc*$^{\mathrm{F}}$ = 88.47% was achieved with $\lambda_H = 10^{-8}$ and $\lambda_V = 0.1$.

For comparison of BAL2 on the *digits* task we chose neural network models with similar architecture from [16]. In Table IV we see that BAL2 is able to learn higher dimensional tasks, but still has a performance gap to fill. Note that it performs comparably to a linear classifier, i.e. a two-layer neural network. Surprisingly, BAL performs extremely bad, but we did not analyze reasons for this failure.

TABLE IV. COMPARISON OF DIFFERENT MODELS ON THE HANDWRITTEN DIGITS TASK. DATA FROM [17] AND [16].

Algorithm	λ_{H}	λ_{V}	*patSucc*$^{\mathrm{F}}$	Epochs
Linear classifier	–	–	88	–
BP 784–300–10	–	–	95.3	–
BAL 784–300–10	0.01	0.01	9.8	20
BAL2 784–300–10	10^{-8}	0.1	88.47	20
GR 784–300–50–10	0.03	0.03	43.22	50

To peek into BAL2 behavior after convergence, we reconstructed backward representations of inputs. Since in this task, multiple inputs of the same category are supposed to lead to the same (classified) output, we expected the backward images to be a blend of the category inputs. As we can see in Figure 8, BAL2 provides readable backward activations. The best shapes could be seen for digits ”0”, ”1” and ”8”. This intuitively proves that the model is capable of learning the bidirectional mapping (although for classification tasks the backward mapping is not required).

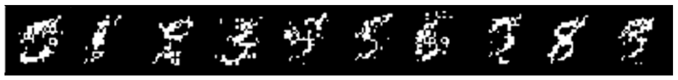

Fig. 8. Backward representations for the most successful BAL2 instance on the *digits* task.

IV. CONCLUSION

We proposed and analysed the BAL2, a modification of Bidirectional Activation-based Learning algorithm with two, but significantly different, learning rates, which increased the success rate in the 4–2–4 autoencoder task from 62.7% to 93.1%. Prior to this modification, we observed that original BAL converges rapidly to the state, when the backward and forward activations converge to the same values. This inspired our hypothesis to explain why BAL had convergence problems learning the autoencoder task. Our hypothesis was further confirmed by candidate selection approach, which selected the initialized networks with more distant hidden activations. This further increased the success rate from 93.1% to 99.84% and reduced an average number of epochs needed for convergence from 5845 to 150. In the second, classification task, we applied BAL2 on the handwritten digit recognition task using the 784–300–10 architecture. Although BAL2 still has a performance gap compared to backpropagation, it achieved a far better success rate (over 88%) than the original BAL.

The concept of (significantly) different learning rates turned out to be important in both tasks that we presented in this paper (autoencoding and classification). As such, it could be worth further investigation, with a goal to find out, whether it is useful in general, in task-dependent manner, or it results from the nature of activation-based learning.

ACKNOWLEDGMENT

This work was supported by grants 1/0898/14 and 1/0503/13 from Slovak Grant Agency for Science (VEGA).

REFERENCES

[1] D. Rumelhart, G. Hinton, and R. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[2] R. C. O’Reilly, “Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm,” *Neural Computation, MIT Press*, vol. 8, no. 5, pp. 895–938, 1996.

[3] G. Hinton and J. McClelland, “Learning representations by recirculation,” in *Neural Information Processing Systems*. American Institute of Physics, 1988, pp. 358–366.

[4] I. Farkaš and K. Rebrová, “Bidirectional activation-based neural network learning algorithm,” in *Artificial Neural Networks and Machine Learning (ICANN)*. Springer, 2013, pp. 154–161.

[5] K. Rebrová, M. Pecháč, and I. Farkaš, “Towards a robotic model of the mirror neuron system,” in *The 3rd Joint IEEE International Conference on Development and Learning and on Epigenetic Robotics*, 2013.

[6] T. Orrú, J. L. G. Rosa, and M. Andrade Netto, “Sabio: A biologically plausible connectionist approach to automatic text summarization,” *Applied Artificial Intelligence*, vol. 22, no. 9, pp. 896–920, 2008.

[7] J. Movellan, “Contrastive hebbian learning in the continuous hopfield model,” in *Proceedings of the Connectionist Models Summer School*, 1990, pp. 10–17.

[8] P. Csiba, “Analysis of the generalized recirculation-based learning algorithm in bidirectional neural network,” Master’s thesis, Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava, 2014.

[9] R. A. Jacobs, “Increased rates of convergence through learning rate adaptation,” *Neural Networks, Elsevier*, vol. 1, no. 4, pp. 295–307, 1988.

[10] L. Behera, S. Kumar, and A. Patnaik, “On adaptive learning rate that guarantees convergence in feedforward networks,” *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1116–1125, 2006.

[11] M. K. Weir, “A method for self-determination of adaptive learning rates in back propagation,” *Neural Networks, Elsevier*, vol. 4, no. 3, pp. 371–379, 1991.

[12] X.-H. Yu and G.-A. Chen, “Efficient backpropagation learning using optimal learning rate and momentum,” *Neural Networks*, vol. 10, no. 3, pp. 517–527, 1997.

[13] G. D. Magoulas, M. N. Vrahatis, and G. S. Androulakis, “Improving the convergence of the backpropagation algorithm using learning rate adaptation methods,” *Neural Computation*, vol. 11, no. 7, pp. 1769–1796, 1999.

[14] C.-C. Yu and B.-D. Liu, “A backpropagation algorithm with adaptive learning rate and momentum coefficient,” in *IJCNN. Proceedings of the International Joint Conference on Neural Networks*, vol. 2. IEEE, 2002, pp. 1218–1223.

[15] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, “Efficient backprop,” in *Neural Networks: Tricks of the Trade*. Springer, 1998, pp. 9–48.

[16] Y. A. LeCun, C. Corinna, and J. C. B. Christopher, “The MNIST database of handwritten digits,” http://yann.lecun.com/exdb/mnist/, 1998, [Online; accessed 28-April-2014].

[17] Y. A. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.