# Multi-task modular backpropagation for feature-based pattern classification

Rohitash Chandra

Centre for Translational Data Science
The University of Sydney, NSW 2006, Australia.
rohitash.chandra@sydney.edu.au

**Abstract.** Modular knowledge development in neural networks have the potential to feature robust decision given sudden changes in the environment or the data during real-time implementation. It can also provide a means to address robustness in decision making given certain features of the data are missing post training stage. In this paper, we present a multi-task modular backpropagation algorithm that features developmental learning where the training takes into account several groups of features that constitute the overall task. The proposed algorithm employs multi-task learning where knowledge from previously trained neural network modules are used to guide knowledge developmental in future modules. The results show that it is possible to implement a modular network without losing training or generalization performance.

**Keywords:** Backpropagation, modular network design, multi-task learning, modular pattern classification.

## 1 Introduction

Embedded control systems guided with machine learning can encounter states that have disruptions in stream of data from sensors [8]. This can as a result provide poor decision making for actuators. Modular knowledge development in neural networks can provide features of robustness and scalability given the changes in the environment or the data [2]. Examples include block-based neural networks that have been implemented and reconfigured in digital hardware such as field programmable gate arrays [12]. Modular neural networks have been popular for hardware implementations [11]. Due to knowledge representation as modules, the system should be able to make decisions with some degree of uncertainty even if some of the modules are damaged or missing. Such motivations come from biological neural systems, i.e due to modular knowledge representation, one is able to see even if one eye is damaged [9].

Multi-task learning considers a shared knowledge representation that exploits knowledge learned from different but related tasks that can be helpful for each other during learning [4]. An example multi-task learning is a system that learns to recognise the face and facial expression recognition at the same time [17]. Transfer learning, on the other hand, considers a one-way transfer of knowledge from source tasks into target task [13] and recently have been popular for deep learning [16]. The main goal of transfer learning is to improve the the training or generalization performance of the target task. In some problems, more than one

source data would exist and hence the area is known as multi-source transfer learning [15]. In cases where the source task data has features that are not alighted directly or inconsistent with target task, the problem is knowledge as heterogeneous and multi-view transfer learning [14]. The notion of multi-task learning has been incorporated in the field of time series to address dynamic time series problems [5] and multi-step ahead prediction [6]. These methods incorporate elements of transfer and multi-task learning.

The motivations from related learning techniques is incorporated, in particular, multi-task learning for modular knowledge representation in neural networks. In this paper, a multi-task modular back-propagation algorithm is presented that takes into account feature groups that are partitioned from the data. The method produces a modular network that provides decision making for the partitioned feature groups.

The rest of the paper is organised as follows. Section 2 presents the proposed method and Section 3 presents experiments and results. Section 4 concludes the paper with a discussion of future work.

## 2 Multi-task modular backpropagation

As mentioned earlier, the proposed algorithm feature properties of transfer and multi-task learning. The the training takes into account several groups of overlapping features partitioned from the original dataset. The proposed multi-task modular backpropagation (MTMB) algorithm considers knowledge from smaller network modules to guide knowledge development for larger cascaded network modules. Although multi-task learning typically employs a set of datasets for the tasks, a single dataset is decomposed into overlapping feature groups which are referred as subtasks. Essentially, the network can be viewed as a cascaded network architecture that increases in size with the subtasks.

A feature group, $X_m$, is a subset of features. The overlapping subtasks $\Omega_1, \ldots, \Omega_m$ are defined as the union of selected feature groups $X_1, \ldots X_m$, for $m = 1, \ldots, M$, where $M$ is the total number of feature groups, so that;

$$\begin{aligned} \Omega_1 &= [X_1] \\ \Omega_2 &= [X_1, X_2] \\ \Omega_3 &= [X_1, X_2, X_3] \\ \Omega_M &= [X_1, X_2, ..., X_M] \end{aligned} \tag{1}$$

In the example in Figure 1, the first feature group is defined to be $X_1 = \{S1, S2\}$ and the second to be $X_2 = \{S3\}$. The overlapping subtasks are then $\Omega_1 = X_1$ and $\Omega_2 = \{X_1, X_2\}$. The input-hidden layer $\omega_m$ weights and the hidden-output layer $\upsilon_m$ weights are combined for the respective network module $\Phi_m$. Hence, the cascaded network module $\theta_m$ of subtask $m$ is constructed by

combining with current $\Phi_m$ and previous network module $\Phi_{m-1}$ as follows.

$$\begin{aligned}
\Phi_1 &= [\omega_1, \upsilon_1]; \;\; \theta_1 = (\Phi_1) \\
\Phi_2 &= [\omega_2, \upsilon_2]; \;\; \theta_2 = [\theta_1, \Phi_2] \\
&\vdots \\
\Phi_M &= [\omega_M, \upsilon_M]; \;\; \theta_M = [\theta_{M-1}, \Phi_M]
\end{aligned} \tag{2}$$

The list of network modules considered for training or optimisation is therefore $\mathbf{\Phi} = (\Phi_1, \ldots, \Phi_M)$.

The size of the feature group and feature space needs to be defined experimentally or can be dependent on the problem given the contribution of the features. It is important to have the most contributing features in the first feature space in order to make use of the full potential of the algorithm. There is a need for appropriate feature selection algorithms as a pre-processing stage in finding the value of the features. Since feature selection is beyond the scope, the decomposition for feature groups would be arbitrary done as the goal is to develop an algorithm that provides robust decision making given some of the feature groups are missing.

The number of input $i$, hidden $h$ and output $o$ neurons in a cascaded network architecture defines a knowledge module $\theta_n = f(i_n, h_n, o)$. The number of output neurons $o$ is same for all the respective modules and $f(.)$ represents the modules of the cascaded network. Hence, $n$ modules are defined by concatenation of knowledge modules which can also be viewed as network ensembles. Note that the input size for a given module is given by the size of the corresponding subtask, $i_n = s(\Omega_n)$ where $s$ represents the size. The number of hidden neurons for different modules can vary. In the proposed modular network architecture, we consider that the hidden neurons used in a given module is computed simply by considering the number of hidden neurons used in the previous module, $h_n = h_{n-1} + \bar{h}$ where $\bar{h}$ refers to the number of hidden neurons in the base knowledge module $\theta_1$.

Algorithm 1 gives an overview of the multi-task modular backpropagation algorithm that employs gradient descent. The training of the network modules are implemented in a incremental mode where each module is trained for a short period of time given by a fixed or adaptive depth $d$ which is predetermined experimentally. The procedure repeats in a round-robin fashion until the termination condition is met which could be given by the number of epochs or the minimal training performance. This incremental training strategy can be viewed as developmental learning [7,10] while the transfer of knowledge from the smaller modules to larger ones could be seen as heterogeneous transfer learning [15]. Figure 1 shows the stage of the network with module 1 and module 2 after knowledge has been incorporated. The incorporation of knowledge considers weights and bias in the respective layers as shown in the figure. The implementation of Algorithm 1 in Python is given online [1] .

[1] https://github.com/rohitash-chandra/modular-backpropagation-classification

**Alg. 1** Multi-task modular backpropagation

**Step 1:** Partition $n$ subtasks $\Omega_n$ from data.
**Step 2:** Define cascaded neural network of modules: $\theta_n = f(i_n, h_n, o)$
**while** until termination **do**
  **for** each Module $\theta_n$ **do**
    **for** Depth $d$ **do**
      i. Forwardpropagate($\theta_n$, $\Omega_n$)
      ii. Backpropagate using Gradient-descent($\theta_n$, $\Omega_n$)
      iii. a) Calculate gradients
      b) Weight updates
    **end for**
    transfer-knowledge($\theta_n$, $\theta_{n+1}$)
  **end for**
**end while**

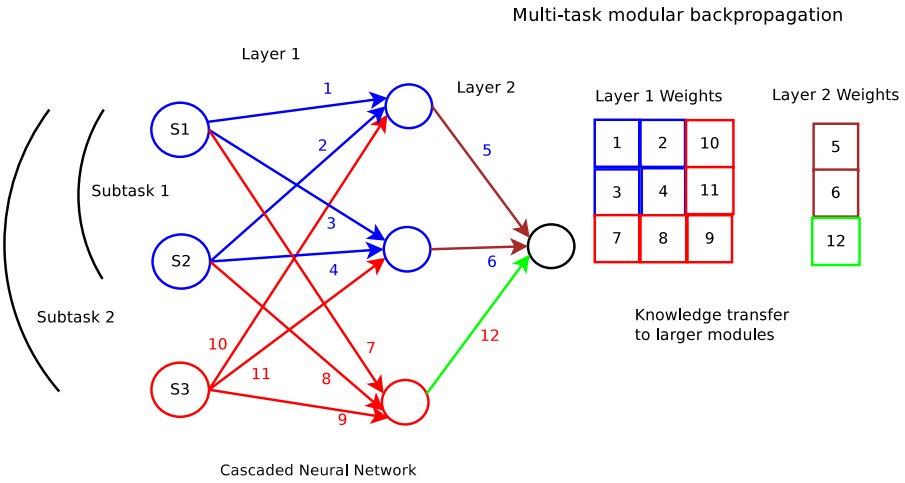


**Fig. 1.** Knowledge transfer through multi-task modular backpropagation. Note that the modules are represented by different colours. The neural network is implemented as a cascaded network architecture defined by the subtasks.

## 3 Simulation and Analysis

This section presents performance evaluation of the proposed multi-task modular backpropgation method for feature-based and modular pattern classification. We compare the results with canonical backpropagation that features gradient descent for pattern benchmark classification tasks.

### 3.1 Experimental Design

Twelve problems from the University of California, Irvine machine learning data repository [1] were selected and the data was partitioned into 4 groups that contained overlapping features given the respective subtasks. The size $\delta$ of each subtask ($\Omega_m$) is given by the proportion $\beta$ considering the total number of features for the given problem $p$.

$$\beta = [0.25, 0.5, 0.75, 1] \tag{3}$$

$$\delta_n = \beta_n * p \tag{4}$$

Table 1 gives details of the respective problems used with details about the number of features, number of classes, and number of instances. The termination condition is also given. The number of hidden neurons for foundation module that corresponds to $\Omega_1$ is given by $\hat{h}$ in Table 1. The rest of hidden neurons in the respective network modules are incremented by $h_n = \hat{h} + h_{n-1} + \epsilon$, given that $h_1 = \hat{h}$. $\epsilon = 2$ is used in all the experiments.

Note that the two major strategies in multi-task modular back-propagation algorithm deal with number of iterations used for each module while they are evolved or trained in a round-robin fashion. The performance is evaluated where adaptive and fixed depth of search strategies are used. The fixed depth of $f = 5$ is used. The adaptive depth $\alpha$ considers a set of values where $\alpha = [10, 7, 4, 1]$. The motivation behind the adaptive depth is to use more training time for the subtasks that represent the building blocks of knowledge. The termination condition is defined by maximum epochs or when minimum training performance (97 %) has been reached.

**Table 1.** Configuration

| Dataset | Features | Classes | Instances | $\hat{h}$ | Max. Time |
|---|---|---|---|---|---|
| Iris | 4 | 3 | 150 | 6 | 500 |
| Wine | 13 | 3 | 178 | 6 | 500 |
| Cancer | 9 | 2 | 699 | 6 | 1000 |
| Heart | 13 | 2 | 270 | 16 | 2000 |
| Credit | 15 | 2 | 690 | 20 | 3000 |
| Baloon | 4 | 2 | 20 | 5 | 500 |
| Tic-Tac-Toe | 9 | 2 | 269 | 30 | 2000 |
| Ionosphere | 34 | 2 | 351 | 8 | 500 |
| Zoo | 17 | 7 | 101 | 6 | 300 |
| Lenses | 4 | 3 | 24 | 5 | 500 |
| Balance | 4 | 3 | 625 | 8 | 200 |
| Robot (Four) | 4 | 4 | 5456 | 14 | 2000 |
| Robot(TwentyFour) | 24 | 4 | 5456 | 14 | 2000 |

Table 2 show the classification performance results that mean (Train and Test) and standard deviation (std) for 30 independent experimental runs. The results for MTMB for the instances of fixed and adaptive depth are further compared with canonical backpropagation that employs gradient descent (BP-GD). Note that the results of the different feature space for the respective problem has been shown ($\Omega_n$) in Table 2. The results compare the fixed and adaptive depth

in terms of of training and generalisation performance. The two strategies have similar performance for most of the problems. The major difference is shown for the Balloon problem where the fixed depth gives a much better training and test performance when compared to adaptive depth strategy. In the Heart problem, the fixed depth $\Omega_4$ gives a much better training performance, howsoever, both methods give a similar test performance. The Tic-Tac-Toe problem also gets better training and test performance by fixed depth strategy. The Lenses problem gets better training performance by fixed depth strategy, howsoever, both methods have the same test performance. In general, the results show that the fixed depth strategy provides better performance when compared to the adaptive depth. Next, the results of the proposed algorithm are compared with BP-GD. $\Omega_4$ is selected specifically for comparison since it uses all the features. The fixed depth strategy ($\Omega_4$) gives similar or better training performance when compared to BP-GD in most of the problems except for the Tic-Tac-Toe problem. In the test performance, it is observed that fixed depth strategy gives better performance for most of the problems, except for the Tic-Tac-Toe problem. Similarly, the adaptive depth strategy ($\Omega_4$) also gives better test performance than BP-GD in general.

### 3.2 Discussion

The results suggest that modularity enforced through backpropagation helped to retain the knowledge of the smaller network modules that are used as building blocks of knowledge for multi-task learning. Heterogeneous form of transfer learning is employed to transfer knowledge from small network modules to larger network modules which link with the respective subtasks. Hence, through transfer learning, the knowledge in foundational modules are projected through the learning of the entire network. This network architecture can be very useful in hardware implementations [11] given that real-time embedded systems may not have full information from sensors in some cases, the network will be able to provide a decision from knowledge in the foundational building blocks or modules. Multi-task modular back-propagation implements an ensemble of cascaded network modules that employ heterogeneous transfer learning for utilizing knowledge in smaller modules. This can also be viewed as a form of dynamic programming [3] as the problem is decomposed into modules and knowledge from the base network modules are used in larger modules through transfer learning. The proposed algorithm implements a neural network that would be operational with a degree of error even when some of the neurons or links in the larger modules are damaged. Moreover, it would be operational with a degree of error in decision making when selected subtasks are unavailable during an event.

## 4 Conclusions and Future Work

The paper presented a cascaded network architecture that employs multi-task modular backpropgation algorithm for feature-based pattern classification. The method incorporates heterogeneous transfer learning in order to utilize knowledge from smaller modules into larges ones. The goal of the method was to

**Table 2.** Results

| Problem | Domain | Adapt Depth MTMB | | | | Fixed Depth MTMB | | | | BP-GD |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\Omega_1$ | $\Omega_2$ | $\Omega_3$ | $\Omega_4$ | $\Omega_1$ | $\Omega_2$ | $\Omega_3$ | $\Omega_4$ | |
| Iris | Train | 55.73 | 60.52 | 74.79 | 80.18 | 52.70 | 59.27 | 89.76 | 95.09 | 93.30 |
| | (std) | 4.89 | 3.72 | 4.96 | 4.39 | 4.95 | 5.04 | 3.16 | 1.38 | 0.89 |
| | Test | 85.25 | 88.67 | 89.42 | 93.08 | 88.75 | 90.00 | 91.08 | 94.42 | 83.50 |
| | (std) | 5.38 | 5.07 | 3.14 | 4.46 | 5.31 | 4.38 | 4.36 | 2.56 | 2.47 |
| Wine | Train | 98.26 | 99.81 | 99.83 | 99.95 | 98.09 | 99.86 | 100.00 | 100.00 | 98.45 |
| | (std) | 0.91 | 0.42 | 0.55 | 0.26 | 1.25 | 0.34 | 0.00 | 0.00 | 0.87 |
| | Test | 99.75 | 100.00 | 100.00 | 100.00 | 99.75 | 99.83 | 100.00 | 100.00 | 83.25 |
| | (std) | 0.75 | 0.00 | 0.00 | 0.00 | 0.75 | 0.62 | 0.00 | 0.00 | 9.90 |
| Cancer | Train | 86.32 | 90.03 | 93.59 | 93.84 | 86.03 | 89.56 | 94.00 | 94.56 | 94.07 |
| | (std) | 0.75 | 0.90 | 0.65 | 0.81 | 0.53 | 0.85 | 0.48 | 0.73 | 0.54 |
| | Test | 97.73 | 97.49 | 98.37 | 98.44 | 97.46 | 97.67 | 98.22 | 98.41 | 96.24 |
| | (std) | 0.47 | 0.87 | 0.27 | 0.39 | 0.55 | 0.91 | 0.27 | 0.41 | 0.57 |
| Heart | Train | 49.61 | 56.76 | 68.39 | 77.03 | 47.89 | 55.21 | 71.99 | 88.20 | 90.92 |
| | (std) | 3.72 | 4.36 | 2.11 | 2.45 | 4.21 | 5.15 | 2.96 | 2.69 | 1.44 |
| | Test | 74.78 | 75.67 | 72.56 | 79.11 | 75.22 | 76.22 | 73.41 | 77.56 | 70.63 |
| | (std) | 1.84 | 1.63 | 2.65 | 2.19 | 1.70 | 1.74 | 1.88 | 2.42 | 3.14 |
| Credit | Train | 19.79 | 42.53 | 82.34 | 84.35 | 18.94 | 40.62 | 83.24 | 87.44 | 89.64 |
| | (std) | 2.91 | 4.46 | 1.86 | 2.26 | 4.15 | 5.18 | 1.97 | 1.39 | 1.07 |
| | Test | 56.97 | 61.82 | 82.22 | 82.64 | 55.23 | 62.40 | 82.71 | 82.06 | 78.79 |
| | (std) | 3.32 | 2.96 | 1.73 | 1.43 | 4.59 | 2.76 | 1.67 | 1.48 | 1.56 |
| Balloon | Train | 7.14 | 15.00 | 34.76 | 44.29 | 8.57 | 12.14 | 45.24 | 99.52 | 100.00 |
| | (std) | 10.10 | 9.82 | 12.61 | 6.23 | 10.50 | 11.97 | 10.49 | 1.78 | 0.00 |
| | Test | 50.00 | 50.00 | 58.33 | 81.67 | 50.00 | 50.00 | 62.78 | 100.00 | 91.11 |
| | (std) | 0.00 | 0.00 | 14.75 | 21.24 | 0.00 | 0.00 | 15.33 | 0.00 | 14.74 |
| Tic-Tac | Train | 33.05 | 42.18 | 61.81 | 71.14 | 36.07 | 40.06 | 62.10 | 79.65 | 97.40 |
| | (std) | 9.04 | 6.68 | 3.83 | 2.85 | 9.33 | 7.18 | 4.52 | 3.54 | 0.24 |
| | Test | 0.00 | 8.29 | 12.08 | 24.97 | 0.00 | 8.39 | 13.22 | 38.96 | 61.24 |
| | (std) | 0.00 | 3.40 | 2.33 | 6.65 | 0.00 | 3.39 | 3.38 | 12.18 | 6.90 |
| Ionosph. | Train | 85.03 | 91.07 | 94.39 | 94.80 | 80.98 | 88.98 | 93.58 | 95.97 | 94.88 |
| | (std) | 1.92 | 1.54 | 1.55 | 2.58 | 2.32 | 2.01 | 1.53 | 0.86 | 1.32 |
| | Test | 89.54 | 92.48 | 94.19 | 93.94 | 90.00 | 92.26 | 94.40 | 94.74 | 84.62 |
| | (std) | 4.42 | 2.12 | 1.43 | 2.07 | 4.70 | 1.47 | 2.34 | 1.72 | 1.99 |
| Zoo | Train | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 99.81 |
| | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.60 |
| | Test | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 98.44 |
| | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 3.93 |
| Lenses | Train | 83.12 | 75.42 | 83.96 | 61.67 | 79.17 | 77.92 | 94.17 | 95.21 | 99.81 |
| | (std) | 15.15 | 17.60 | 21.51 | 24.72 | 14.99 | 15.46 | 9.12 | 5.97 | 0.60 |
| | Test | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 98.44 |
| | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 3.93 |
| Balance | Train | 94.09 | 95.62 | 95.98 | 93.35 | 93.94 | 95.36 | 96.11 | 94.65 | 92.29 |
| | (std) | 0.00 | 0.72 | 0.63 | 1.05 | 0.57 | 0.97 | 0.53 | 1.60 | 7.85 |
| | Test | 100.00 | 100.00 | 99.98 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 88.33 |
| | (std) | 0.00 | 0.00 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 8.50 |
| Robot | Train | 99.95 | 100.00 | 100.00 | 100.00 | 99.93 | 100.00 | 100.00 | 100.00 | 95.95 |
| (Four) | (std) | 0.15 | 0.00 | 0.00 | 0.00 | 0.17 | 0.00 | 0.00 | 0.00 | 1.69 |
| | Test | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 98.81 |
| | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 2.39 |
| Robot | Train | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 100.00 | 98.69 |
| (Twenty | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.75 |
| Four) | Test | 100.00 | 100.00 | 100.00 | 100.00 | 1.17 | 100.00 | 100.00 | 100.00 | 94.35 |
| | (std) | 0.00 | 0.00 | 0.00 | 0.00 | 2.17 | 0.00 | 0.00 | 0.00 | 6.21 |

provide modularity in knowledge representation and decision making so that it can be applied to problems where real-time implementations may have cases where full feature space or all the features are available. The results shows that the proposed algorithm can deliver similar or better performance to that of canonical (non-modular) backpropagation network. Hence, the proposed algorithm can train neural networks that do not lose performance although their

knowledge representation featured modularity. Moreover, although unexpected, the algorithm outperformed canonical backpropagation in several cases.

In future work, the method can be adapted for other types of problems that include image classification. Moreover, the modular method needs to be adapted further so that the trained network is operational even when the foundation or base subtask is unavailable during test phase. Uncertainty quantification through Bayesian inference methods for modular networks can also be explored. Furthermore, the method can be extended to the application of deep learning .

## References


1. Asuncion, A., Newman, D.: UCI machine learning repository. `http://archive.ics.uci.edu/ml/datasets.html` (2007)
2. Auda, G., Kamel, M.: Modular neural networks: a survey. International Journal of Neural Systems 9(02), 129–151 (1999)
3. Boutilier, C., Dearden, R., Goldszmidt, M.: Stochastic dynamic programming with factored representations. Artificial intelligence 121(1), 49–107 (2000)
4. Caruana, R.: Multitask learning. In: Learning to learn, pp. 95–133. Springer (1998)
5. Chandra, R., Ong, Y.S., Goh, C.K.: Co-evolutionary multi-task learning for dynamic time series prediction. CoRR abs/1703.01887 (2017), `http://arxiv.org/abs/1703.01887`
6. Chandra, R., Ong, Y.S., Goh, C.K.: Co-evolutionary multi-task learning with predictive recurrence for multi-step chaotic time series prediction. Neurocomputing 243, 21–34 (2017)
7. Geschwind, N., Behan, P.: Left-handedness: Association with immune disease, migraine, and developmental learning disorder. Proceedings of the National Academy of Sciences 79(16), 5097–5100 (1982)
8. Hunt, K.J., Sbarbaro, D., Żbikowski, R., Gawthrop, P.J.: Neural networks for control systemsa survey. Automatica 28(6), 1083–1112 (1992)
9. Johnson, M.K.: A multiple-entry, modular memory system. Psychology of Learning and Motivation, vol. 17, pp. 81 – 123. Academic Press (1983)
10. Lee, M.H., Meng, Q., Chao, F.: Developmental learning for autonomous robots. Robotics and Autonomous Systems 55(9), 750–759 (2007)
11. Misra, J., Saha, I.: Artificial neural networks in hardware: A survey of two decades of progress. Neurocomputing 74(13), 239 – 255 (2010), artificial Brains
12. Moon, S.W., Kong, S.G.: Block-based neural networks. IEEE Transactions on Neural Networks 12(2), 307–317 (Mar 2001)
13. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. 22(10), 1345–1359 (Oct 2010)
14. Sun, S.: A survey of multi-view machine learning. Neural Computing and Applications 23(7-8), 2031–2038 (2013)
15. Xu, Z., Sun, S.: Multi-source transfer learning with multi-view adaboost. In: International Conference on Neural Information Processing. pp. 332–339. Springer (2012)
16. Zeng, T., Ji, S.: Deep convolutional neural networks for multi-instance multi-task learning. In: Data Mining (ICDM), 2015 IEEE International Conference on. pp. 579–588 (Nov 2015)
17. Zheng, H., Geng, X., Tao, D., Jin, Z.: A multi-task model for simultaneous face identification and facial expression recognition. Neurocomputing 171, 515 – 523 (2016)