Alleviating Catastrophic Forgetting with Modularity for Continuously Learning Linked Open Data

Lu Chen and Masayuki Murata

***Abstract*—Nowadays, Linked Open Data is spreading year by year, and its further utilization is expected. Because of the large size of data, Linked Open Data is attempted to learn by using neural networks. Since the data is still scaling in various region, unlike the existing neural network specialized to learn only one region, a neural network which can continuously learn wide region of knowledge is needed. However, neural network is known in its problem, catastrophic forgetting, which is to lose previously acquired skills when learning a new skill. Though existing researches said enhancing modularity can overcome this problem since it can reduce interference between tasks, those researches consider the number of learning tasks is given in advance, and it is not applicable for continuous learning. In this paper, we propose a design approach of neural network reducing modularity expecting that unspecialization can mitigate catastrophic forgetting for continuous learning. Our results show that, although, as we can expect, a neural network with high modularity can mitigate forgetting for tasks learned just before because of the low interference, a neural network with low modularity is better for the worst case when evaluating for all the tasks it learned in the past.**

***Index Terms*—Neural network topology design, catastrophic forgetting, linked open data, modularity.**

I. Introduction

Nowadays, Linked Open Data is spreading year by year, and its further utilization is expect. Because of the large size of data, Linked Open Data is attempted to learn by using neural networks[1]. Since the data is continuously scaling in various region, unlike the existing neural network specialized to learn one region, a neural network which can continuously learn wide region of knowledge is needed.

However, neural network is known in its problem, catastrophic forgetting, which is to lose previously acquired skills when learning a new skill. In [2], Ellefsen *et al.* said enhancing modularity can reduce interference between tasks by separating functionality into physically distinct modules, so that it can overcome catastrophic forgetting. However, since this consider the number of learning tasks is given in advance, it is not applicable for continuous learning. Therefore, a new approach can mitigate catastrophic forgetting for continuously learning wide region of knowledge is needed [3].

In studies of human brain, it is said, although functional brain network has high modularity in some high level, mod-ularity is low in a lower level [4]. Also, there is a research in complex network field says compare to a specialized system which is highly constrained, an unspecialized system has more potential to evolve in various ways [5]. Inspired from these studies, we propose a design approach of neural network reducing modularity expecting that unspecialization can mitigate catastrophic forgetting for continuous learning. From comparing with a neural network with high modularity which is said can have low interference between learning tasks, we show that although, as we can expect, a neural network with high modularity can mitigate forgetting for tasks learned just before, a neural network with low modularity is better for the worst case when evaluating for all the tasks it learned in the past.

Note that tasks we used are categorization. In order to apply to continuously learning wide region of knowledge, we change the learning method. In general, categorization is learned by assign output nodes of a neural network to categories. However, this restrict the number and kind of categories. Therefore, to learn categorization for continuously learning wide region of knowledge, we express category by tree structure, and learn it by using a neural network.

The rest of this paper is organized as follow. In Sec. II, we explain the designing approach for continuously learning wide region of knowledge. In Sec. III, we explain the category tree. In Sec. IV, we explain the evaluation results. Finally, we conclude our paper in Sec. V.

II. Design Approach

In this section, we explain the designing approach. To design a neural network for continuously learning, different from [2], we design a neural network with multiple learning subset without considering the number of tasks but considering modularity. In Sec. II-A, we explain the modularity of subsets, and in Sec. II-B, we explain the designing approach.

A. Modularity of Subsets

Calculation of modularity of subsets is explained in this section. We calculate modularity after reconstruct the neural network and partitioning the neural network based on the subsets.

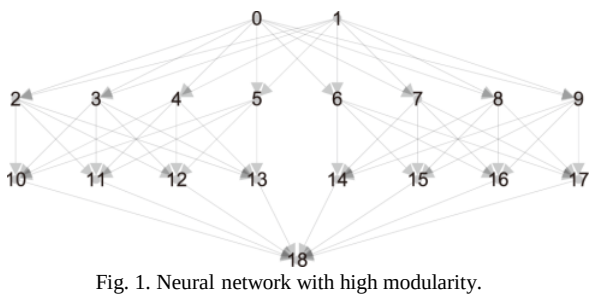

Fig. 1. Neural network with high modularity.

Manuscript received January 9, 2018; revised April 9, 2018.

Lu Chen and Masayuki Murata are with Osaka University, Suita, Osaka, Japan (e-mail: l-chen@ist.osaka-u.ac.jp, murata@ist.osaka-u.ac.jp).

DOI: 10.7763/IJCTE.2018.V10.1196

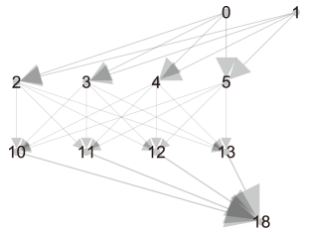

Fig. 2. Neural network with high modularity (subset 1).

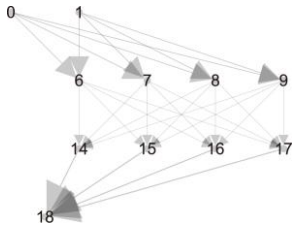

Fig. 3. Neural network with high modularity (subset 2).

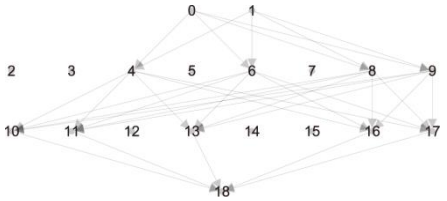

Fig. 4. Neural network with low modularity.

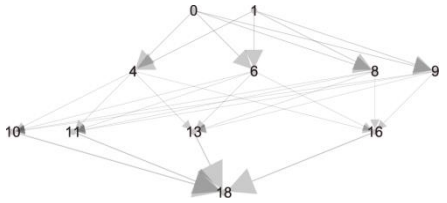

Fig. 5. Neural network with low modularity (subset 1).

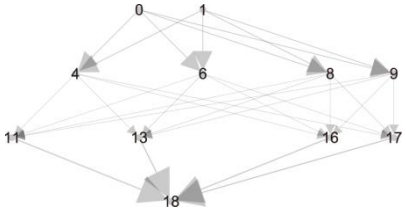

Fig. 6. Neural network with low modularity (subset 2).

We assume topology of a neural network and nodes and links of each learning subsets is given. Let the set of nodes in input layer be V_I, the set of nodes in output layer be V_O, the links in the neural network be E. Let the ath subset be S_a, and the set of nodes in S_a be V_a, and the set of links be E_a, and the total number of subsets be max_S. To calculate modularity, we reconstruct the topology of the neural network. Let the new topology be T_{new}. We redesign input nodes and connections from them, and then redesign output nodes and connections to them. Let the total number of nodes in input layer be the sum of input nodes in each subsets. We indicate the nodes in input layer in T_{new} as V_I^{new}. The number of nodes in V_I^{new} is

$$|V_I^{new}| = |V_I| * \max\_S \quad (1)$$

Let each node in V_I^{new} represent a input node in the subsets. Let a input node y_{in} represent a input node z_{in} in subset a connect to nodes in $W_I(z_{in}, a)$:

$$W_I(z_{in}, a) = \{w|(Z_{in}, w) \in E_a\} \quad (2)$$

We also do this for nodes in output layer. When indicate the nodes in output layer in T_{new} as V_O^{new}, the number of nodes in V_O^{new} is

$$|V_O^{new}| = |V_O| * \max\_S \quad (3)$$

Let each node in V_O^{new} represent a output node in the subset. Let an output node y_{out} represent an output node z_{out} in subset a connect from nodes in $W_O(z_{out}, a)$:

$$W_O(z_{out}, a) = \{w|(z_{out}, w) \in E_a\} \quad (4)$$

Nodes in hidden layer and links between nodes in hidden layer is not changed.

Next, we partition T_{new} in to max_S modules by the following algorithm. We describe the x th module as mod_x. Let a input node y_{in} in T_{new} represent a input node z_{in} in subset a belong to module mod_a, and let a output node y_{out} in T_{new} represent a output node z_{out} in subset a belong to module mod_a. Let the probability of a node in hidden layer belong to mod_a depends on the nodes it connected from. Let a node in hidden layer be y and the nodes it connected from be W_F. The probability of y belong to mod_x is, $p(y, mod_x)$:

$$p(y, mod_x) = \frac{|\{z|z \in W_F, z \text{ belongs to } mod_x\}|}{|W_F|} \quad (5)$$

The modularity is calculated using T_{new}. For calculating modularity, we used measurement of modularity for directed graph [6]:

$$modularity(A_{ij}) = \frac{1}{m}\sum_{ij}\left[A_{ij} - \frac{k_i^{in}k_j^{out}}{m}\right]\delta_{md(i),md(j),} \quad (6)$$

where A_{ij} is adjacency matrix of T_{new}, k^{in} is in degree of node k, k^{out} is out degree of node k, m is total number of links in T_{new}. $\delta_{md(i),md(j)}$ is 1 if node i and node j belong to the same module, otherwise it is 0. $md(x)$ describe the module which node x belong to.

B. Designing Approach using Modularity of Subsets

We explain the designing approach in this section. The basic idea is to design a neural network with multiple learning subset without considering the number of tasks but only considering modularity. To generate a neural network with low modularity, firstly, we generate a neural network with high modularity, than we generate that with low modularity by using simulated annealing.

We generate a neural network with high modularity by assigning nodes to subsets without duplication. We assume the size of a neural network, such as number of input nodes max_I, number of output nodes max_O, number of hid- den layer max_H, number of nodes in each hidden layer max $L(x)$, number of subsets max_S is given. Since we do not know the tasks to learn in advance, we use the same input nodes and output nodes for every task. Therefore, we do not

contain input nodes and output nodes to any subset. To maximize the learning speed for each subset, we let the number of nodes in each layer be the same. Then, we connect nodes in the same subset, and let the links belong to the subset.

We generate a neural network with low modularity by using simulated annealing to minimize modularity of subset. Neighbor state is generated by exchanging nodes in the subsets and rewire the links. First, we randomly select a subset, then we randomly select a neuron in the subset. Let the selected neuron be n_{old}. Then we randomly select a neuron which is in the same layer with n_{old}, but is not included in the selected subset. Let the newly selected neuron be n_{new}. Then, we add n_{new} and links $W_N(n_{old})$ to the subset, where $W_N(n_{old})$ is

$$W_N(n_{old}) = \{w|(w, n_{old}) \in E\}. \quad (7)$$

Finally, we remove n_{old} and links connected to it from the subset. Since number of nodes and links used for learning one task affect the learning speed, we align them in every subset. The fitness function used in simulated annealing is modularity of subsets. Initial temperature is 100, and the cooling rate is 0.995. We confirmed that a neural network with small modularity is generated.

We show the example of generated subsets. For example, when set max_I to 2, max_O to 1, max_H to 2, max_L(1) to 8, max_L(2) to 8, and max_S to 2, Fig. 1 is the neural network with high modularity, and Fig. 2 and Fig. 3 are the subsets. A neural network with low modularity generated using this settings is Fig. 4. Fig. 5 and Fig. 6 are the subsets. This is obtained after adapting simulated annealing for 1000 steps. The modularity of the neural network with high modularity is 0.5, and that with low modularity is -0.05398. Note, these neural networks are used in the evaluation in Sec. IV.

III. Learning Category Tree by Neural Network

In neural networks designed for specialized usage, categories are often assign to output nodes. However, since the number and the kind of the categories are fixed, it is not suitable for wide region continuous learning, where the categories would expand. Therefore, instead of using the traditional method, we express the categories using tree structure, and learning the connections of the structure using a neural network. To create such neural network, we extend the work of Stanley *et al.* [7], which presented a neural network learning connections of a graph. We use the example data in Table I for explanation. It is to categorize animals by using its attributes.

The tree structure consists of three types of nodes and directed connections between them. One type of node is describing animals. Another type of node is describing attribute. In order to avoid confusion, we call the first type "animal node", the second type "attribute node". For convenience, we call these two types of nodes, "object node". The last type of node is describing categories. We call them "category node". Connections exist only between animal node and category node, and between category node and attribute node, and between category node themselves. Directions are only allowed from animal node to category node, and from category node to attribute node. Category nodes consist a single directed line, so that nodes close to root can describe higher/abstract level of category. For convenience, let the number of category nodes be max_C. We consider the category tree express "an animal" has "an attribute" if there is a path from the corresponding animal node to the corresponding attribute node. For example, for the data in Table I, Fig. 7 is the category tree express the data necessary and sufficient. From Fig. 7, we can say canary and eagle belongs to one category and shark belongs to another.

TABLE I: Example of Animals and Attributes

Animal	Attribute
Canary	Wings, Feathers, Fly, Skin, Eats, Breathes
Eagle	Wings, Feathers, Fly, Skin, Eats, Breathes
Shark	Skin, Eats, Breathes

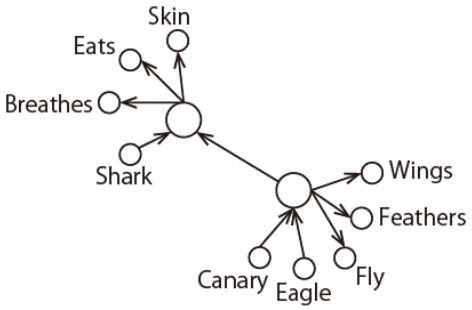

Fig. 7. Example of a category tree.

In the tree structure, we fix the category node, and using neural network to learn which category node the object node should connect to. In detail, our neural network has two input nodes and one output node. The input information describe an object node, and the output information describe which category node it connected to. For one input node, we input 1 if the object node describes animal, and 0 if the object node describes attribute. For the other input node, we input the node ID of the object node. The ID is given by a unique value. We consider, for example, when max_C is 2, then when the output is larger than 0, it express that the object node connect to the first category node, and otherwise it connect to the second category node. Therefore, a category tree can be obtained when finishing input all the object node describing animals and attribute in a task to a neural network.

To measure the correctness of the expression of category tree, we use F-measure. Though the above example can have the most successful categorization as Fig. 7, the dataset in a task cannot always have the perfectly correct answer. Therefore, here we use the F-measure to let the category tree express the correct animal-attribute pair necessary and sufficient. The F-measure is computed between all the animal-attribute pair can be obtained from category tree and the animal-attribute pair inputted. Instead of the normal F-measure, we use the weighted F-measure [8], since this can make more appropriate category tree. The weight we used is 0.01.

The generative algorithm we used is based on NEAT [7], [9] and HyperNEAT [7]. In the generative algorithm, an individual is a topology of a neural network. Mate is to generate offspring by letting the link weight and node

activation function of an offspring be that of what in either parent. Mutation is to change weight of selected link or change the activation function of a selected node. For the parameters in genetic algorithm, since the task is not fixed, we try to use random value as much as possible. However, in considering of the calculation time, *stagnation_age* and *min_elitism_size* is set to a random integer between 10 to 20. And following that, *young_age* and *old_age* is set to a random integer between 1 to 10 and 10 to 20. Since *target_species* and *population_size* affects the fitness improvement importantly, we set a random integer from 1 to 2 for *target species*, and we set 100 for *population_size*. Also, in considering of the calculation time, to reduce the search space of weights, we made an integer list in advance, and the link weight is selected from the list. The range of the integers in the list is between -1 multiply the maximum ID number and the maximum ID number. The size of the list is 10 since we find it can improve the fitness.

IV. Evaluation

We use real data for evaluation. We evaluated not only for tasks learned just before, but also for tasks learned a while before. In Sec. IV-A, data used for evaluation is explained. Evaluation measurement is explained in Sec. IV-B. The results of comparing a neural network with high modularity and one with low modularity is shown in Sec. IV-C. We show some examples of obtained category trees in Sec. IV-D. The validity of parameters used in simulation is discussed in Sec. IV-E.

A. Data for Evaluation

As sample data for evaluation, we chose the real data obtained from BBC Nature [10]. The dataset consists with animals and their attributes. Since a wide region of animals are obtainable, it is suitable for our aim, which is to design a neural network for wide region continuous learning. In detail, we obtained dataset by starting from "Amphibians", which is relatively in the middle level, and then, we trace the abstract level or subdivision level of that species recursively. The length of the obtained data is more than 50, and we show a part of it in Table II. As we mentioned before, since it is unable to know where to punctuate the data as a single task, we punctuate the data in a certain length. Here, we punctuate every 3 data for a single task. Though it is true that the number is quite small and should be improved, since the focus in this paper is modularity, we do not take it as a problem here.

TABLE II: A Part of Obtained LOD

Animal	Attribute
Texas blind salamander	Adapted to swimming, Chemical communication, Metamorphosis, Neoteny, Egg layer, Tactile sense, Carnivorous, Cave dweller
Lamp shells	Intertidal zone, Sea bed, Estuaries, Shallow seas
Bony fish	Shallow seas, Tactile sense, Lakes and ponds, Deep ocean, Brackish water, Open ocean, Estuaries, Adapted to swimming, Wetlands
Cartilaginous fish	Tactile sense, Sea bed, Carnivorous, Shallow seas, Open ocean, Reefs, Adapted to swimming
Pangolins	Adapted to climbing, Tropical dry forest, Chemical communication, Rainforest, Tropical grassland, Tropical coniferous forest
Leptictidium	Adapted to running, Adapted to jumping, Viviparous
Armadillos	Burrower, Adapted to running, Viviparous
Shrews	Moles and relatives
Elephant shrews	Adapted to running, Active at birth, Territorial, Viviparous
Malayan colugo	Adapted to gliding, Maternal care, Herbivorous, Chemical communication, Nocturnal, Rainforest, Tree dweller, Territorial, Mangroves, Viviparous, Acoustic communication

B. Evaluation Measurement

We evaluated not only for tasks learned just before, but also for tasks learned a while before. In detail, we evaluate not only the task learned just before, but also the tasks learned from 5 before to just before, and all the tasks learned before. We indicate each task set as PT1, PT5, PTALL. Let the i th learned task be T_i, Let the last generation of neural network generated for T_i be NN^i. Then, let the F-measure of the category tree when input T_i to neural network NN^i be F- measure (T_i, NN^i). When assume current learning task is c, result for PT1 is

$$\text{F-measure}(T_{c-1}, NN^c) \tag{8}$$

result for PT5 is

$$\min\{\text{F-measure}(T_{c-1}, NN^c) | i = 1,2,3,4,5\} \tag{9}$$

result for PTALL is

$$\min\{\text{F-measure}(T_{c-1}, NN^c) | i = 1, \dots, \max_i\} \tag{8}$$

C. Evaluation Results

The topology we used for the evaluation is obtained by setting max_I to 2, max_O to 1, max_H to 2, max_L(1) to 8, max_L(2) to 8, max_S to 2. For a neural network with high modularity, we only used one kind of topology. For a neural network with low modularity, we generated 20 kinds of

topology by using different random seeds. The topology of the neural networks are shown in Sec. II-B. The modularity of the neural network with high modularity is 0.5, and that with low modularity are around 0. We indicate the neural network with high modularity as H^{high}, and the neural networks with low modularity as H^{low}. In the evaluation below, the results are all done in 20 trial. For the neural network with high modularity, we do the simulation by different 20 random seeds. And, for the neural networks with low modularity, we do the simulation for each topology by a random seed.

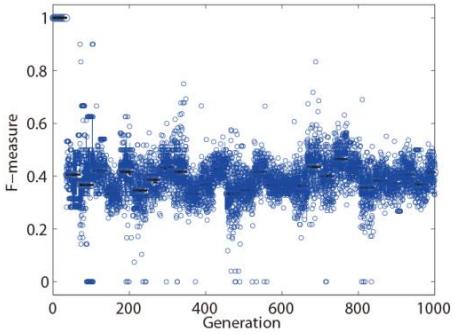

Fig. 8. Fitness of H^{low} for PT1.

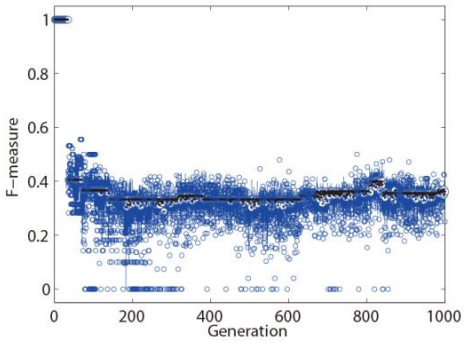

Fig. 9. Fitness of H^{low} for PT5.

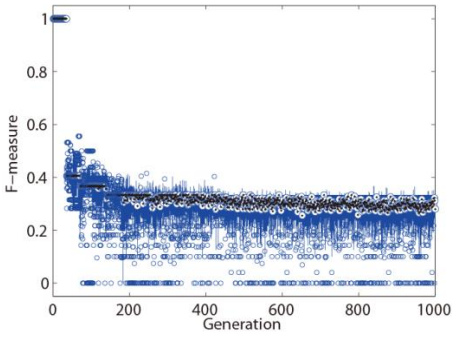

Fig. 10. Fitness of H^{low} for PTALL.

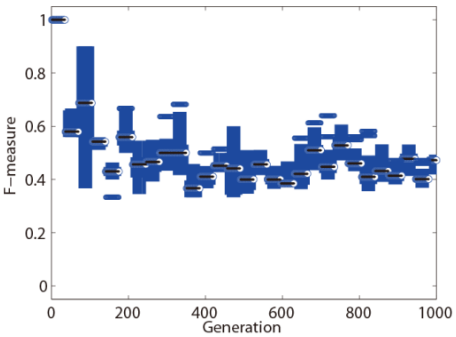

Fig. 11. Fitness of H^{high} for PT1.

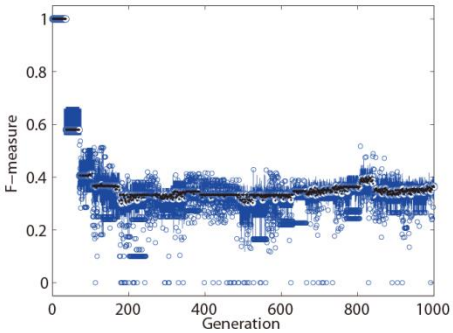

Fig. 12. Fitness of H^{high} for PT5.

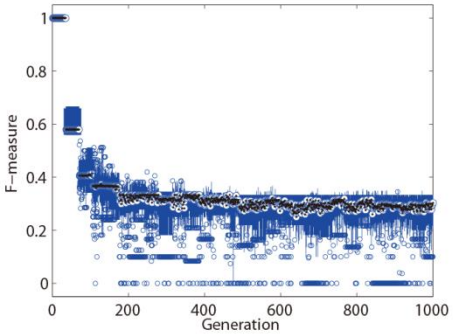

Fig. 13. Fitness of H^{high} for PTALL.

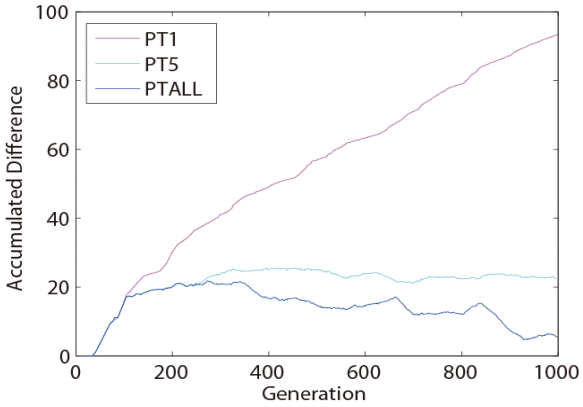

Fig. 14. Accumulated difference of value of H^{high} minus value of H^{low}.

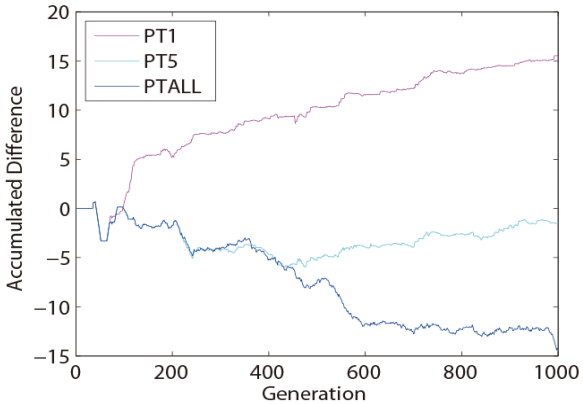

Fig. 15. Accumulated difference of value of H^{high} minus value of H^{low} (with topology 2/8/8/8/1).

Results are from Fig. 8 to Fig. 13. Fig. 8 and Fig. 11 are results evaluated by PT1. Fig. 9 and Fig. 12 are results evaluated by PT5. Fig. 10 and Fig. 13 are results evaluated by PTALL. All of the results are boxplot of the 20 trials. The dot in the middle is median. X-axis indicates generation, and y-axis indicates F-measure. For the first task, since there is

no previous task, we set 1 for all the results. From the results, we can see, for PT1, the median and the minimum of F-measure of H^{high} is higher than that of H^{low} in most of the generations. However, in larger generations for PT5 and PTALL, the minimum of F-measure becomes more frequently in H^{high} than in H^{low}. To show more clearly, we show the accumulated difference in Fig. 14. Y-axis is accumulation of difference obtained by minus the minimum value of H^{low} from that of H^{high}. This means, when the gradient is plus, the minimum value of H^{high} is better than that of H^{low}, and when the gradient is minus, vice versa. From Fig. 14, we can see that, as we can expect, for evaluation by PT1, the minimum value of H^{high} is better than that of H^{low} in all the generations. Then, for PTALL, though the minimum value of H^{high} is better at first, from around the 400th generation, the minimum value of H^{low} become better. And, for PT5, after the 100th generation, the result is almost the same. This might could be explain by the specification of neural networks. Since a neural network with high modularity is specified to the newly learned two tasks, it performs significantly bad for tasks learned a while before. On the other hand, a neural network with low modularity perform bad to the task just learned before, it performs comparatively better for tasks learned a while before.

To confirm the tendency of the obtained results do not relay on the topology of neural networks, we do the same evaluation for neural networks with different topology. The neural network we used is generated by setting max_H to 3 and max_L(3) to 8. Other settings are as same as the one before, which means the neural network is 2/8/8/8/1. We evaluate this neural network by using the same parameters. The accumulated difference is shown in Fig. 15. From the result, we can see the tendency is as same as before, which is, for PT1, the neural network with high modularity is better than that with low modularity, and for PTALL, the neural network with low modularity become better.

D. Obtained Category Tree

We show the examples of category trees we obtained during the simulation. Fig. 16 to Fig. 19 are examples of category trees obtained in the simulation. Fig. 16 is obtained in 139th generation. Fig. 17 is obtained in 102th generation. Fig.18 is obtained in 160th generation. Fig. 19 is obtained in 224th generation.

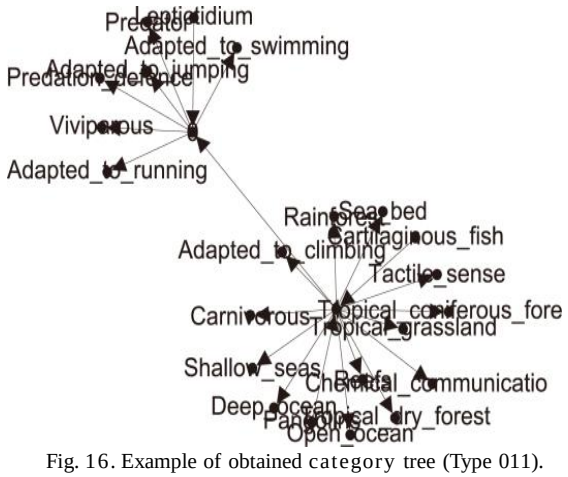

Fig. 16. Example of obtained category tree (Type 011).

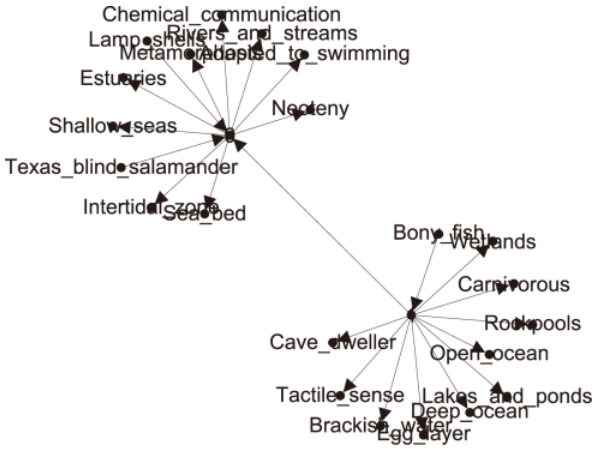

Fig. 17. Example of obtained category tree (Type 001).

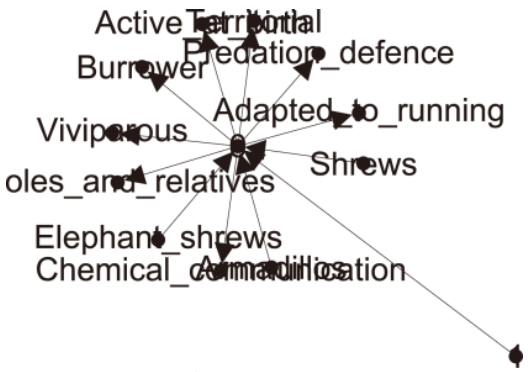

Fig. 18. Example of obtained category tree (Type 000).

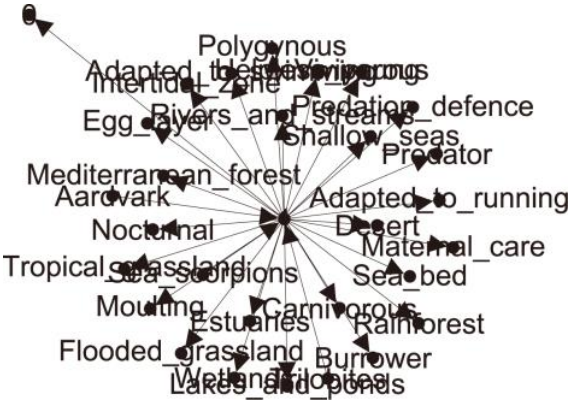

Fig. 19. Example of obtained category tree (Type 111).

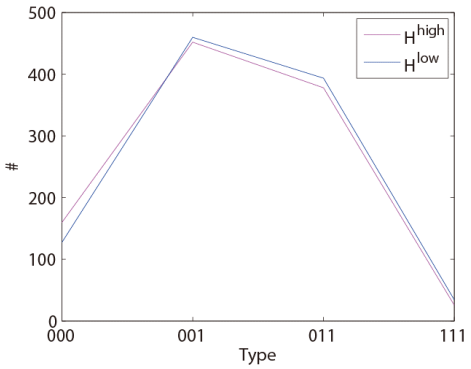

Fig. 20. Types of obtained category tree.

From analyzing the category trees, we find that some of them is not successfully categorized. For example, Fig. 18 and Fig. 19 are not successfully categorized since all the

object nodes are connected to one category node, and no object node is connected to the other category node. On the other hand, Fig. 16 and Fig. 17 are successfully categorized since two object node is connected to one category node, and the other object node is connected to the other category node. We show the number of each type we obtained in during simulation in Fig. 20. X-axis is the type of obtained category tree. Type 011 means that an object node connect to a category node with ID 0 and two object nodes connect to a category node with ID 1 such as Fig. 16. Type 001 means that two object nodes connect to a category node with ID 0 and one object node connect to a category node with ID 1 such as Fig. 17. Type 000 means that all of the three object nodes connect to a category node with ID 0 such as Fig. 18. Type 111 means that all of the three object nodes connect to a category node with ID 1 such as Fig. 19. And, Type 011 and Type 001 are successfully categorized, while Type 000 and Type 111 are not. The average number of all the trials of each type we obtained is showed in Fig. 20. From Fig. 20, we can see that the number of Type 000 and Type 111 is much lower than the number of Type 011 and Type 001. Also, we can see that this tendency do not depend on the modularity of the neural network.

E. Validity of Parameters

In this section, we demonstrate the parameters we used for generative algorithm as explained in Sec. III are reasonable. *Population_size* is the size of a population, and *target_species* is the number of individuals in one species. The algorithm cannot find better individual when *population_size* or *target species* is too small, while computational time increase as they become larger. Therefore, from comparing with results obtained with smaller *population_size* and larger *target species*, we show that the values we used in simulations are enough to find better individuals during the learning process.

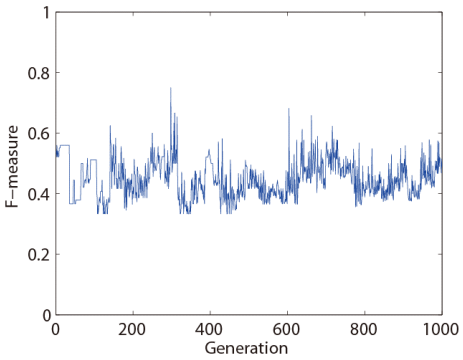

Fig. 21. Fitness for the neural network with low modularity.

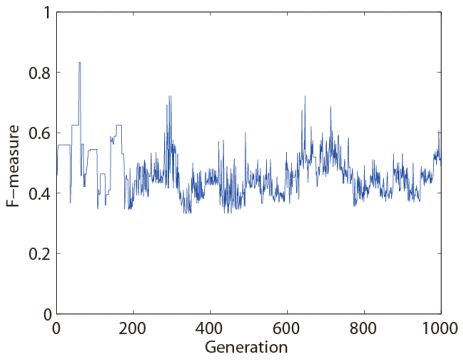

Fig. 22. Fitness for the neural network with high modularity.

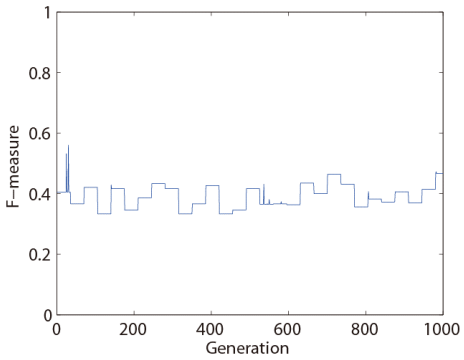

Fig. 23. Fitness when set *population_size* to 10 for the neural network with low modularity.

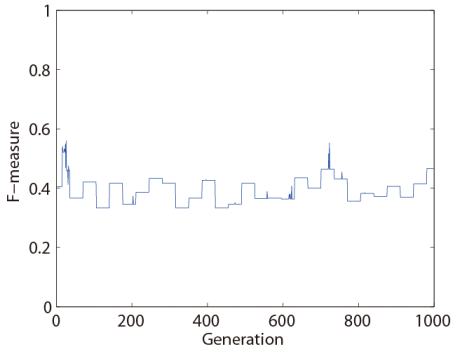

Fig. 24. Fitness when set *population_size* to 10 for the neural network with high modularity.

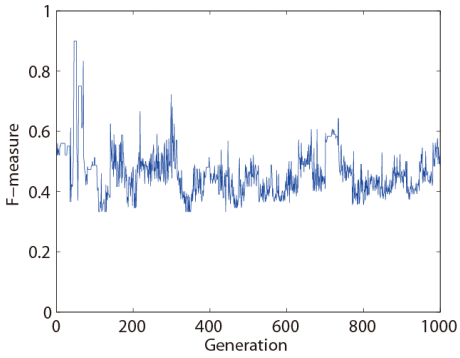

Fig. 25. Fitness when set *target_species* to a random value from 5 to 6 for the neural network with low modularity.

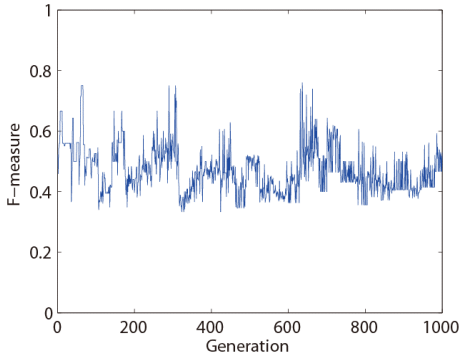

Fig. 26. Fitness when set *target_species* to a random value from 5 to 6 for the neural network with high modularity.

Fig. 21 and Fig. 22 are the results of one trial obtained from the simulations above, which means those are results when *target_species* is set to a random integer from 1 to 2, and *population_size* is set to 100. Fig. 21 is the result of a neural network with low modularity, and Fig. 22 is the result of a neural network with high modularity. Fig. 23 and Fig. 24 are the results of the same trial when change *population_size* to

10. We can see from comparing Fig. 21 and Fig. 22 with Fig. 23 and Fig. 24 that fitness improves much more obviously in the former. This means the value should be set to 100. Fig. 25 and Fig. 26 are the results of the same trial when change *target_species* to a random value from 5 to 6. We can see from comparing Fig. 21 and Fig. 22 with Fig. 25 and Fig. 26 that, although the *target_species* is larger, the improvement of the growth of fitness during the learning process is unnoteworthy. This means the value we set for the simulations above is large enough.

V. Conclusion and Future Work

In this paper, we propose a design approach of neural network reducing modularity in order to mitigate catastrophic forgetting for continuously learning wide region of knowledge of Linked Open Data. Our results show that, although, as we can expect, a neural network with high modularity can mitigate forgetting for tasks learned just before because of the low interference, a neural network with low modularity is better for the worst case when evaluating for all the tasks it learned in the past.

For future works, the evaluation should be tested for more wide range of data. The learning accuracy should be improved. Generalization to tasks other than Linked Open Data is needed. Also, theoretical analysis is needed.

References

[1] I. Tiddi, M. d'Aquin, and E. Motta, "Using neural networks to aggregate linked data rules," in *Proc. International Conference on Knowledge Engineering and Knowledge Management*, Nov. 2014, vol. 8876, pp. 547–562.

[2] K. O. Ellefsen, J. B. Mouret, and J. Clune, "Neural modularity helps organisms evolve to learn new skills without forgetting old skills," *PLoS Computational Biology*, vol. 11, p. e1004128, Apr. 2015.

[3] L. Chen and M. Murata, "Mitigate catastrophic forgetting for continuously learning linked open data using modularity," in *Proc. International Conference on Innovation in Artificial Intelligence*, Mar. 2018.

[4] D. Meunier, R. Lambiotte, A. Fornito, K. Ersche, and E. T. Bullmore, "Hierarchical modularity in human brain functional networks," *Frontiers in Neuroinformatics*, vol. 3, p. 37, 2009.

[5] M. Prokopenko, F. Boschetti, and A. Ryan, "An information-theoretic primer on complexity, self-organization, and emergence," *Complexity*, vol. 15, pp. 11–28, Sept. 2009.

[6] E. A. Leicht and M. E. Newman, "Community structure in directed networks," *Physical Review Letters*, vol. 100, p. 118703, Mar. 2008.

[7] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial Life*, vol. 15, pp. 185–212, Apr. 2009.

[8] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Applied Soft Computing*, vol. 6, pp. 119–138, Jan. 2006.

[9] J. Huizinga, J. B. Mouret, and J. Clune, "Does aligning phenotypic and genotypic modularity improve the evolution of neural networks?" in *Proc. the Genetic and Evolutionary Computation Conference*, pp. 125–132, July 2016.

[10] BBC. (2017). NATURE WILDLIFE. [Online]. Available: http://www.bbc.co.uk/nature/wildlife

Lu Chen received her M.E. and D.E. in information science and technology from Osaka University in 2013 and 2016, respectively. In October 2016, she joined Central Research Laboratories NEC as a researcher, and become a visiting researcher in Osaka University.

Masayuki Murata received his M.E. and D.E. in information science and technology from Osaka University in 1984 and 1988, respectively. In April 1984, he joined the Tokyo Research Laboratory IBM Japan as a researcher. From September 1987 to January 1989, he was an assistant professor with the Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. From 1992 to 1999, he was an associate professor with the Graduate School of Engineering Science, Osaka University, and since April 1999, he has been a professor. He moved to the Graduate School of Information Science and Technology, Osaka University, in April 2004. He has published more than 300 papers in international and domestic journals and conferences. His research interests include computer communication networks and performance modeling and evaluation. He is a fellow of the IEICE and a member of the IEEE, the Association for Computing Machinery (ACM), the Internet Society, and the Information Processing Society of Japan.