Universität Hamburg
Department Informatik
Knowledge Technology, WTM

Progressive Deep Neural Networks

Seminar Paper

Knowledge Processing with Neural Networks

Cindy Ching

Matr.Nr. 6831706

cindyching88@gmail.com

July 13, 2017

Abstract

Continual learning and transfer learning are important key factors in creating human-like artificial intelligence. Standard methods for transfer learning with neural networks lack the ability to learn tasks consecutively and preserve important information. Progressive neural networks are designed as novel approach to learn multiple tasks sequentially and by that to achieve continual learning. Compared to traditional transfer learning architectures with single column networks, progressive neural networks use multiple columns which are laterally interconnected. Transfer learning experiments including learning multiple Atari games in sequence demonstrated a better transfer learning performance of progressive neural networks in comparison with single-column networks and are therefore a promising stepping stone towards continual learning.

Contents

1 Introduction **2**

2 Traditional Transfer Learning **3**

3 Concept of Progressive Neural Networks **4**

4 Experiments on Progressive Neural Networks **6**

5 Results **8**

6 Discussion **10**

7 Conclusion **11**

8 References **12**

1 Introduction

Striving towards the vision of creating an artificial agent whose intelligence is at least on human level requires the machine to have the ability to learn and execute multiple complex tasks.
When humans learn new tasks they are capable of building on top of pre-existing experiences and by that, to accumulate a diverse skill set. The ability to exercise *transfer learning*, which is using already acquired knowledge and applying it to other tasks or situations in order to learn new things faster, is crucial for constructing human-like artificial intelligence. This makes transfer learning an important research subject in the field of machine learning.

The standard approach to transfer learning in neural networks is to train a neural network to master a specific source task and then retrain it towards a new target task. During this process its weights are adjusted to fit the new task. If the tasks are similar, the learned features from the first task may allow a faster learning of the new task [1, 2]. This is very useful especially for deep neural networks such as sophisticated image classifiers with complex and rich features hierarchies. For instance, training deep convolutional neural networks for object recognition in images requires large datasets and computational time. Instead of training such a network from scratch, layers containing features from already trained networks can be reapplied in the new network, where only the classification part is trained towards a different dataset [1, 2].

A network that reuses extracted features requires less time for learning a new domain and has therefore a higher learning efficiency. However, when a pretrained neural network learns a new target domain it may forget the task it originally specialized on and undergoes so called *catastrophic forgetting* [3, 4].
To learn consecutive tasks without catastrophic forgetting is a challenging topic tackled by Rusu et al. As a solution the research group developed Progressive

neural networks which are immune to catastrophic forgetting [3] and are therefore a significant contribution to the field of continual learning, the ability to learn new information while retaining the old information. Their work in this field is presented in this seminar paper.

2 Traditional Transfer Learning

Traditionally, a deep neural network is trained by feeding it with a large amount of data and given time to learn how to perform a specific task. As standard method *backpropagation* is used where a network is trained by gradually updating its weights to minimize the cost function and to achieve a desired output [5]. Such already trained networks are called pretrained networks and are used for transfer learning.

In order for a pretrained network to learn a new task, the first layers of the source network are copied to the target network, serving as its base (see Figure 1). The base can then either be *frozen* or *finetuned*. When layers are frozen, their weights remain constant during training. Otherwise their weights are allowed to adjust to the target domain [6]. Freezing is also known as the *feature extraction* approach, where already learned features are reused for classification and recognition tasks [7]. Finetuning the base network is called the *weight initialization* approach, since the starting values of the weights change eventually during training on the target domain.

Layers are frozen in case of dealing with a small target dataset with a large number of parameters since finetuning can result in overfitting. However, freezing the layers may lead to a performance loss when the frozen extracted features are too specific [6].

In contrast, when the target dataset is large or the number of parameters are small, overfitting is unlikely to occur and finetuning is the more suitable option to be applied to the base network [6]. Regardless, which of the two methods is used, when trained towards the target domain the important weights for the

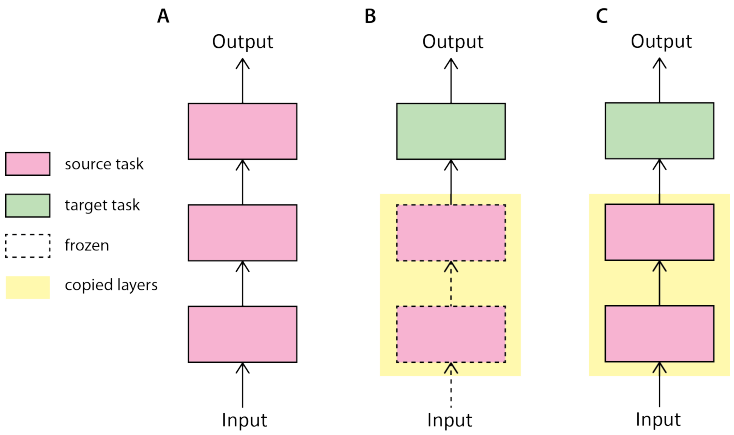


Figure 1: Traditional transfer learning. A) The source network. Its layers are trained towards a source task. B) Transfer learning network. Its first layers are copied from the source network and frozen when trained towards a target task. C) Transfer learning network. Its architecture is the same as network B, but all layers are allowed to learn.

source domain change. The network may be specialized on the target domain but its ability to carry out the source domain is lost [4]. This scenario of catastrophic forgetting impairs continual learning, which is the ability to successively learn tasks and to accumulate knowledge without memory loss.

3 Concept of Progressive Neural Networks

The traditional methods for transfer learning described in the previous section use single-column networks, whereas progressive neural networks consist of multiple columns that are laterally connected with each other. Transfer learning with progressive neural networks undergoes the following process: As a first step, a neural network with L number of layers is trained to perform an initial task. This neural network forms the initial *column* in the progressive neural network [3] (see Figure 2). In order to learn a second task, another column is added and its weights are initialized randomly. The outputs of layer l of the initial column serve as additional inputs to layer $l+1$ of the second column. As a measure against catastrophic forgetting, the weights of the first column are *frozen*, which means that their values remain constant during training [3](see Figure 3).

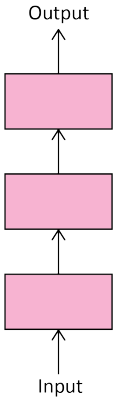


Figure 2: First column of a progressive neural network.

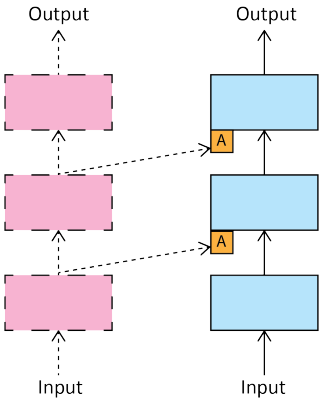


Figure 3: Pogressive neural network with two columns.

Following the same principle, to train the model to perform a third task, a third column with randomly initialized weights is added. The weights of the previous columns are frozen and the output layers l are connected to the neurons of the third column in layer $l + 1$ (see Figure 4).

With a growing number of columns the dimensionality increases. To counter this problem, the lateral connections between the columns are facilitated by *adapters*. The purpose of adapters is to assist in initial conditioning and to reduce dimensionality by replacing each linear lateral connection with a single hidden layer MLP. This ensures that the number of parameters coming from the lateral connections remains in the same order, as more tasks are added [3]. The idea behind this architecture, in which the network is trained towards consecutive tasks in a column-wise

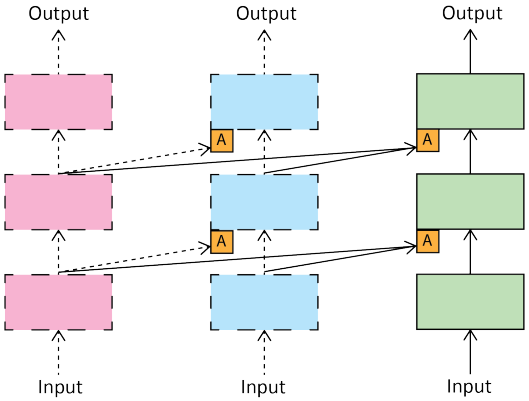


Figure 4: Progressive neural network with three columns.

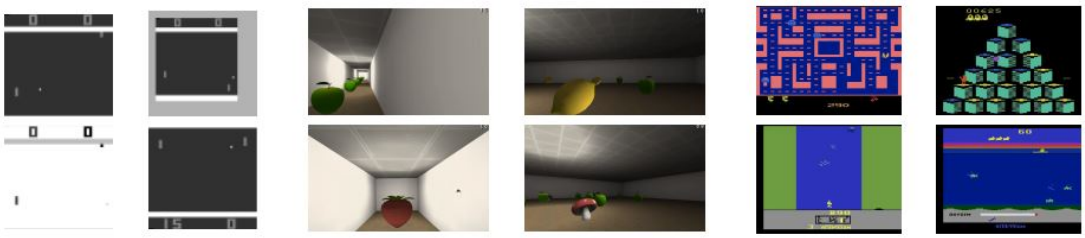
fashion, is to preserve knowledge and promote continual learning. In the following sections, progressive neural networks are put into practice, to evaluate their performance regarding transfer and continual learning.

4 Experiments on Progressive Neural Networks

The performance of progressive neural networks was investigated in the following three different experimental set-ups:

Pong soup: In this experiment, the models were initially trained on the Atari Pong game before learning another variation of the game. Different synthetically created variations included alterations such as adding noise to the input (*Noisy*), altering the background to white or black (*White*, *Black*), scaling the input by 75 % (*Zoom*), and feeding a vertically and/or horizontally flipped input (*V-flip, H-flip* and *VH-Flip*) (see Figure 5a). For two-column progressive networks, the first column was trained on *Pong, Noisy* or *flip* and the following column on each of the other variants. As a unit to measure the performance *transfer scores* were computed [3].

Atari games: Here, feature transfer was investigated using three randomly selected Atari games [8]. When compared to the synthetic variants in the Pong soup



(a) Pong variants (b) Labyrinth games (c) Atari games

Figure 5: Selection of different task domains used in the experiments [3].

experiment, the games in this experiment are not as similar. The games differ far more from each other in terms of visuals, controls and strategy and demand a higher transfer performance than in Pong soup. The baselines as well as multi-column progressive networks were trained on these source games and their transfer performance on the target game was examined. Progressive networks with more than one source column were trained successively. For example, in a three-column progressive network, the initial column was trained on the first game. The following column on the second game, and the last column on the third game [3] (see Figure 5c).

Labyrinth: In this setting, the agent learns to move through a 3D maze outputting a set of actions which can either be looking up, down, left or right and moving forward, backwards, left or right. Further the agent earns positive or negative scores by eating "good" or "bad" items while manoeuvring through maps of different levels [3]. Each column is trained on a level map in the game, allowing to observe to what extent previously learned mazes influences the network on learning new mazes [3] (see Figure 5b).

The experiments were carried out on two- and three-column progressive networks and four baselines comparing the respective results with each other(see Figure 6). These baselines show different network architechture and learning approaches. In the network of *baseline 1*, all layers of the single column are trained towards a target task representing training from a blank slate. In *baseline 2* the single column was

first trained on a source task and is then trained on a target task. Here, only the outer layer of network is able to adjust to the new task, whereas the other layers remain frozen. *Baseline 3* is the same as *baseline 2* but without frozen layers, allowing all layers to be adjusted to the target task. This network architecture is constructed for finetuning. *Baseline 4* has progressive network architecture and consists of two columns. The first column is frozen and set up randomly. The second column is trained on a target task.

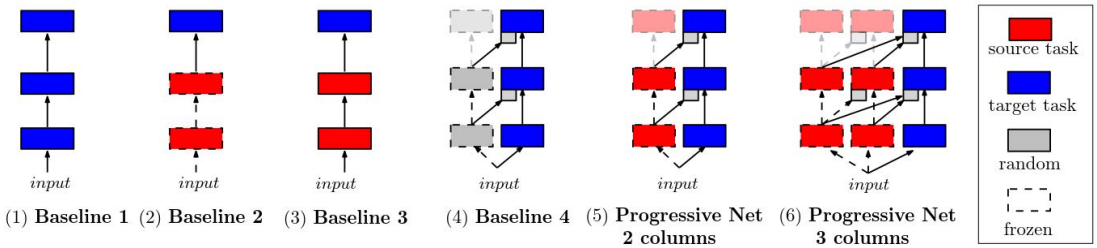


Figure 6: Schematic illustration of the different baselines as well as the two- and three-column progressive networks [3]. (1) Baseline 1: The single-column network represents training from a blank slate. (2) Baseline 2: A single column that was pretrained on a source task and finetuned on a new target task. Here, only the output layer is trained and the other layers are frozen.(3) Baseline 3: Similar to baseline 2. However, all layers can be finetuned. (4) Baseline 4: A two-column progressive network, where the first column is initialized randomly and frozen.

5 Results

The previously described experiments generated various results presented in this section and demonstrate the performance of progressive networks in terms of transfer learning in comparison with other baselines.

The results of the Pong soup experiment (see Figure 7) show that baseline 2, where only the output layer is finetuned, has the worst performance. The dominating blue colour in the transfer matrix (see Figure 7a) indicates that it mostly failed to learn new Pong variants. Only the transfer from Pong to Black or White and from Noisy to Black or White was successful. In contrast, baseline 3 was able to learn all games via finetuning exhibiting high positive transfer scores. Progressive networks

show even higher performance, as their median and mean transfer percentages exceed those of baseline 3. In comparison, baseline 4 has lower transfer percentages. It has the same network architecture as a two-column progressive network but with a randomly initialised first column instead of being trained on a source game. This lack of information seems to lead to a lower transfer performance as compared to baseline 3 and to progressive networks. Moreover, Figure 7b show that the two-column progressive neural network requires less steps and has steeper learning curves than baseline 1 and 3 when transferring from Pong to White or Pong to Flip.

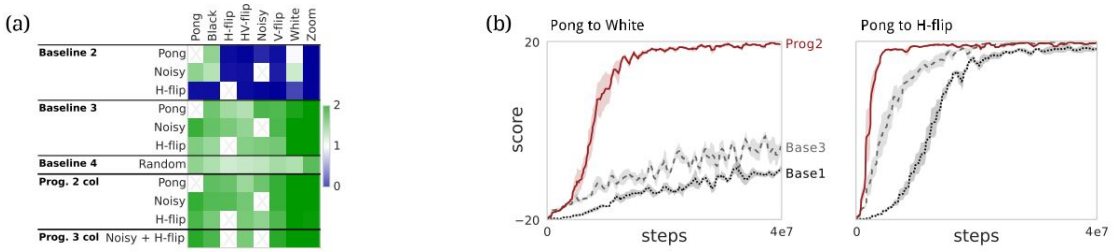


Figure 7: Results of the Pong soup experiment [3]. (a) Shows the transfer matrix. Transfer scores are colour indicated and range from 0 (dark blue) to 2 (dark green). (b) Learning curves for Pong soup.

In the Atari games, progressive neural networks achieved positive transfer scores in 8 out of 12 cases and only failed to transfer twice, whereas the finetuning baseline 3 only transferred positively in 5 of 12 games (see Figure 8b). Again, the learning curves of the progressive networks are steeper those of baseline 1 to 3 with the exception of the two-columm progressive network (see Figure 8b) when transferring from the source games to *Boxing*. However, progressive neural networks could not outperform the other baselines when transferring to the game *Gopher*, as demonstrated by the overlapping learning curves.

Similarly, in the Labyrinth games progressive neural networks resulted in better overall transfer performance (see Figure 9), which can be derived from the higher transfer percentages (see Table 1) as well.

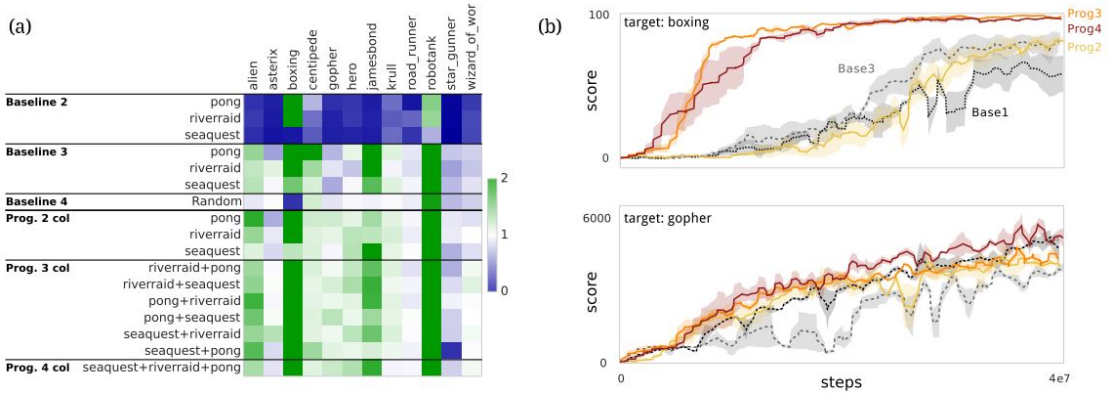


Figure 8: Results of the Atari games experiment [3]. (a) Shows the transfer matrix. Transfer scores are colour indicated and range from 0 (dark blue) to 2 (dark green). (b) Learning curves for Atari target games.

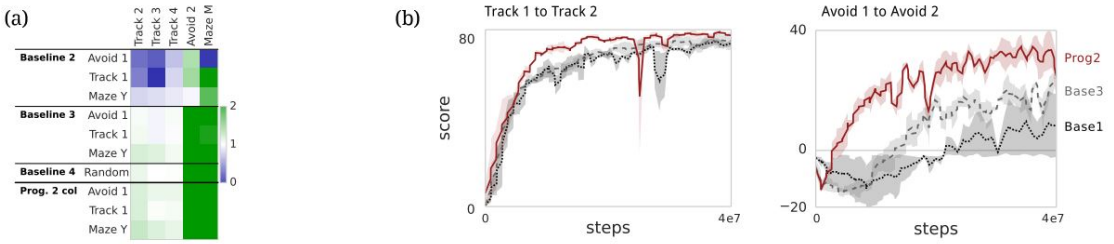


Figure 9: Results of the Labyrinth experiment [3]. (a) Shows the transfer matrix. Transfer scores are colour indicated and range from 0 (dark blue) to 2 (dark green). (b) Learning curves for Labyrinth.

	Pong Soup		**Atari**		**Labyrinth**	
	Mean (%)	Median (%)	Mean (%)	Median (%)	Mean (%)	Median (%)
Baseline 1	100	100	100	100	100	100
Baseline 2	35	7	41	21	88	85
Baseline 3	181	160	133	110	235	112
Baseline 4	134	131	96	95	185	108
Progressive 2 col	209	169	132	112	**491**	**115**
Progressive 3 col	**222**	**183**	140	111	—	—
Progressive 4 col	—	—	**141**	**116**	—	—

Table 1: Transfer percentages of the three experimental set-ups [3]. In all three experiments transfer percentages of progressive neural networks are higher than the baselines, demonstrating a better overall transfer learning performance.

6 Discussion

The results of the three experiments described in the previous section demonstrate that progressive neural networks yield higher transfer scores and steeper learning

curves than networks on which the finetuning method was applied. In most transfer tasks, progressive neural networks outperformed the other baselines. In the Atari games, they received highest transfer scores when features of the first columns were enhanced by features of the new column. In contrast, transfer scores were the lowest when no learning of new visual features occurred. At first glance, this result appears to be counter-intuitive, since presumably, the transfer score should be high when features from target domains were already learned in previous source domains. An explanation offered by the authors, is a bias induced by the learned source task leading to a fast convergence to a local minimum which is a common problem in transfer learning [3].

7 Conclusion

In the past, commonly used methods for transfer learning were feature extraction and weight initialization on single-column networks. Both approaches have proven that learning new tasks with pre-trained networks can be more efficient than training a network from scratch [7, 6, 2, 1].Especially for deep complex networks, such as image object classifiers built on Convolutional Neural Network architecture these approaches are very useful since they can reduce the necessary time and data volume required for training. Nevertheless, the single-column networks are prone to catastrophic forgetting, which is harmful for the process of continual learning. Progressive neural networks however are unaffected by this event and are able to learn multiple sequential tasks, as demonstrated by Rusu et al. In addition, their overall transfer learning performance was better than in the other methods. Still it should be mentioned that the performance might be better, but with each added task, the number of parameters increases quadratically. Another limitation is that the network cannot automatically identify the task label to find the most suitable column for carrying out a task [3]. Instead, this information needs to be fed externally. All in all, progressive neural networks have proven to be a promising concept for transfer and continual learning.

8 References

[1] L. Hertel, E. Barth, T. Kaster, and T. Martinetz, "Deep convolutional neural networks as generic feature extractors," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2015 September, 2015.

[2] Y. Zhan, Y. Chen, Q. Zhang, and X. Kang, "Image forensics based on transfer learning and convolutional neural network," in *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '17, (New York, NY, USA), pp. 165–170, ACM, 2017.

[3] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *CoRR*, vol. abs/1606.04671, 2016.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks.," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 114, no. 13, pp. 3521–3526, 2017.

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

[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.

[7] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1717–1724, 2014.

[8] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *CoRR*, vol. abs/1207.4708, 2012.