Exploring Continual Learning Using Incremental Architecture Search

Shenyang Huang[1], Vincent François-Lavet[1], Guillaume Rabusseau[2], Joelle Pineau[1]
shenyang.huang@mail.mcgill.ca, vincent.francois-lavet@mcgill.ca, guillaume.rabusseau@umontreal.ca, joelle.pineau@mcgill.ca
RL lab, McGill University, Montreal, Canada[1]
Université de Montreal, Montreal, Canada[2]

Abstract

The emerging paradigm of continual learning requires a machine learning model to acquire new information continuously while retaining existing knowledge. In this work, we focus on class-incremental learning where new classes of the same dataset arrive sequentially. We combine network transformation techniques with automatic neural architecture design approach to develop a new method for class-incremental learning: incremental architecture search (IAS). When a new image class arrives, IAS selects competitive architectures based on the best model from the previous step thus avoiding the computational cost to sample many potential architectures from scratch. IAS shows promising results in image classification tasks on the MNIST and Fashion-MNIST datasets when compared with directly training on all available classes from a random initialization. Our experiments also show that the changes in learning task caused by the arrival of a new class have different effects on individual classes.

1 Introduction

As an important step towards artificial intelligence, a machine learning model must be able to accommodate new information while retaining previous knowledge: this ability is referred to as continual lifelong learning [1]. In this work, we focus on class-incremental learning where new classes are introduced sequentially [3]. To adapt neural networks to this setting, we present a dynamic architecture approach: Incremental Architecture Search (IAS). IAS accommodates new classes into the model and increases the model capacity as new classes arrive. However, catastrophic forgetting or catastrophic interference remains a major challenge for any model aiming to tackle the continual learning problem [2]. Catastrophic forgetting refers to the phenomenon where learning new information interferes with existing knowledge. To minimize this effect, we continually train our model with past data when new information is introduced.

The source code for IAS is available at `https://github.com/shenyangHuang/IAS`.

1.1 Class-incremental learning

In class-incremental learning, the difficulty of the learning task increases sequentially as each new class arrives. In addition, factors such as the arrival order of the classes, the initial classes known as base knowledge and the number of new classes introduced at each step all change the learning task. In this work, we provide all training data from one new class to the model at each step and the arrival order is determined by the default dataset classification labels such as the incremental order from digit 0 to 9 in MNIST handwritten digit dataset [8]. In Section 3, we observe that the change in learning task caused by the arrival of a new class has different effects on individual classes.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

1.2 Net2Net and automatic architecture design

In 2015, Chen et al. [4] proposed Net2Net techniques to enable a rapid transfer of information from one neural network to another. More specifically, Net2WiderNet allows one to widen layers of the network (by expanding existing hidden layers with additional neurons) and Net2DeeperNet allows one to deepen a network (by adding new identity layers). Both transformations preserve the function computed by the neural network. Cai et al. [5] proposed a new framework for efficient architecture search by exploring the architecture space based on the current network and reusing its weights. Similarly, we combine Net2WiderNet and Net2DeeperNet to sample new neural architectures without having to retrain the network from scratch using a random initialization. In Section 2, we discuss how automatic neural architecture design is leveraged in IAS. In Appendix A, we also explain how IAS naturally tackles the capacity saturation problem in continual learning.

Mendoza et al. [6] discussed the concept of AutoML which aims to provide efficient and off-the-shelf learning systems that avoid the tedious tasks of manually selecting the right algorithm, hyperparameters and neural architecture. In IAS, only the hyperparameters and architecture for the base knowledge need to be manually selected (such as the initial architecture and learning rate). These choices only concern a single and shallow model in contrast to a naive approach of picking a new architecture for each incremental step (and at later steps, the architecture search space would become increasingly large). Therefore, IAS can be seen as a case of AutoML for class-incremental learning because the number of hyperparameters and architectures that need to be manually selected are greatly reduced.

2 Methodology

Past approaches that utilized neural networks for class-incremental learning prevented catastrophic forgetting by withholding or limiting the modification of network weights on previous tasks. One example is reinforced continual learning (RCL) proposed by Xu et al. [7], a novel approach that achieves image classifications by freezing network weights for previous tasks and only training newly added filters. In contrast to RCL, training procedure in IAS adjusts all weights of the network and employs Net2Net techniques to enlarge the current architecture.

2.1 Incremental architecture search

Incremental architecture search (IAS) describes a general algorithm that expands the current model when a new class arrives. A flow chart of IAS can be seen in Figure 1.

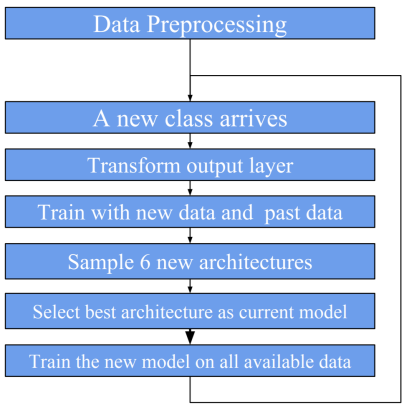

Figure 1: Flow chart of incremental architecture search

Once data for a new class arrives, one neuron is added to the output layer. In IAS, we use softmax activation funcion for the output layer while all previous layers use ReLU activation function. The training performance thus instantly decreases due to random predictions for samples from this new class. Before automatic architecture search, the current model is first trained with all available data (including the newly arrived class) using early stopping (training is terminated only when the validation loss stops improving for 20 epochs). This ensures that any increase in the performance of a sampled architecture is only caused by the improvement in neural architecture (and not by simply learning to discriminate the new classes). To limit the cost of automatic architecture search, IAS samples 6 new architectures based on selected guidelines that consist of several Net2Net transformations that are likely to improve the performance (and these guidelines are listed in Appendix C). The weights in each sample architecture is transferred using Net2Net techniques from the current model and then trained for 20 epochs on all available data. We use the average validation accuracy across all classes to select the best architecture. After the search, the newly selected model is trained with all available data using early stopping (with the same stopping criteria as before). The above steps are repeated each time a new class arrives.

2.2 Data Preprocessing

If the entire dataset is given at once, it is possible to utilize data normalization methods that involve statistical information of the dataset such as mean μ or standard deviation σ^2. However, to avoid any bias towards the distribution of unseen images, we simply normalize each pixel value x into $(x/128 - 1)$ thus effectively rescaling them into the range $[-1, 1]$. When the data for a new class arrives, it is split into 10% validation set and 90% training set.

2.3 Output layer transformation

The softmax function captures a categorical distribution and is often used as the output layer activation function for classification tasks. Each class has an associated probability that is represented by a specific output neuron. In a class-incremental learning setting, the model has no knowledge on the number of unseen categories. Therefore, assuming there are ℓ number of seen classes, the output layer contains exactly ℓ neurons. When a new class is introduced in the dataset, we need to add one additional neuron in the output layer to represent the $(\ell + 1)$-th class. This process is illustrated in Figure 2. We initialize the new weights with a normal distribution of mean $\mu = 0$ and standard deviation $\sigma^2 = 0.35$ while the new bias term is set to 0. Further discussion can be seen in Appendix B.

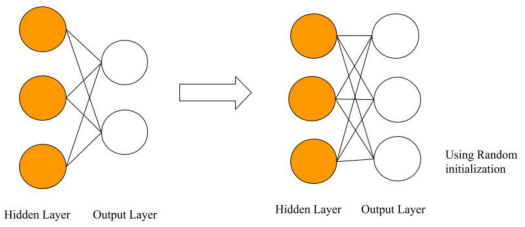

Figure 2: Add one output neuron when a new class arrives.

3 Experiments

3.1 Baseline

We consider two baselines. First, we use a fixed MLP architecture with 2 hidden layers of 256 neurons each and it is referred as fixed architecture in Figure 3 and 4 (this architecture achieves competitive test accuracies for a multilayer perceptron learner on both MNIST and Fashion-MNIST dataset if trained with all 10 classes). Second, at each step, we also train a neural network from scratch with a random initialization using the neural architecture selected by IAS and we call it random initialization. All models are trained using the RMSProp backpropagation algorithm [10] with a learning rate of 0.0001.

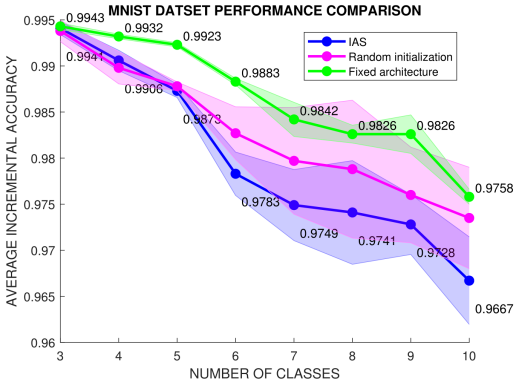

Figure 3: Performance comparison between IAS, fixed architecture and random initialization when learning MNIST dataset incrementally.

3.2 MNIST performance

Our MNIST experiment starts with 2 initial classes: digits 0 and 1, the remaining 8 classes are provided one at a time and can be seen in Appendix E. Figure 3 shows the average incremental accuracy [3] at each class-incremental learning step (average test accuracy across all seen classes), averaged over three trials of the experiment. As the figure suggests, IAS performs comparatively to both random initialization and fixed architecture. It is worth noting that IAS starts with a small initial architecture of 1 hidden layer with 16 neurons. As more classes are introduced, the architecture evolution of IAS leads to a competitive architecture with 2 hidden layers. The detailed architecture evolution can be found in Appendix D. The shaded bands in Figure 3, 4 and 5 are standard deviations measured from three distinct trials.

3.3 MNIST and Fashion-MNIST

To investigate the scenario where two different datasets are learned incrementally, we design an experiment where all 10 MNIST classes are learned as base knowledge and one new Fashion-MNIST

class is introduced at each step of the class-incremental learning setting. The arrival order of classes can be seen in Appendix E. The Fashion-MNIST dataset is designed to be a drop-in replacement for the MNIST dataset thus no image resizing is necessary [9].

Figure 4 shows the average incremental accuracy over three trials. Both IAS and fixed architecture outperformed the random initialization baseline model which uses the same architecture as IAS. Notice that IAS and fixed architecture transfer weights from previous steps thus one possible reason for this observation is the positive forward transfer effect described by Lopez-Paz et al. [11]. We also observe that the performance of IAS are consistent with fixed architecture in earlier steps while in later steps, fixed architecture outperforms IAS. This could be attributed to the fact that Net2Net transformations only enlarge existing models thus IAS might have sampled unnecessarily large models in later steps.

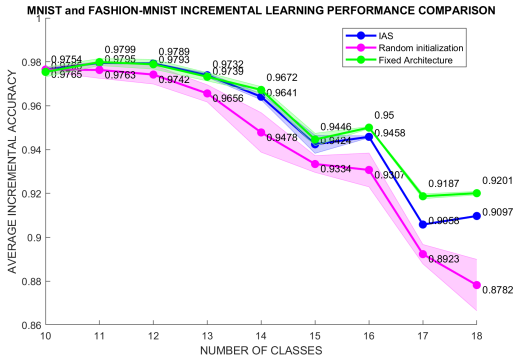

Figure 4: Performance comparison between IAS, fixed architecture and random initialization baseline during incremental learning experiment of MNIST and Fashion-MNIST classes.

Figure 5 shows the average test accuracies of MNIST digit 0,1,2 and Fashion-MNIST class 0,1,2 from 3 independent IAS trials. When more Fashion-MNIST classes are introduced, the test accuracies on MNIST classes are only affected to a limited extent (a more detailed figure can be seen in Appendix F). However, the decrease in average incremental accuracy seen in Figure 4 is mostly caused by the decrease in performance of Fashion-MNIST classes. For example, in Figure 5, classes such as Fashion-MNIST class 0 and 2 have a more severe drop in test acccuracy compared to the MNIST digit 0,1,2. Thus, the change in learning task induced by the introduction of a new class can have very different effects on the performance of individual classes.

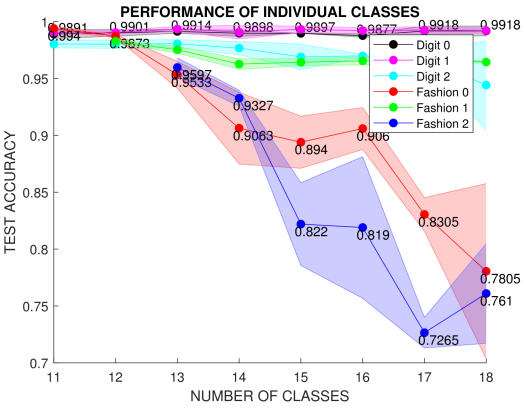

Figure 5: The performance of MNIST digit 0,1,2 and Fashion-MNIST class 0,1,2 during incremental learning experiment of MNIST and Fashion-MNIST dataset

4 Discussion

From the incremental learning experiment involving both MNIST and Fashion-MNIST dataset, we showed that performance on individual classes react differently to a change in learning task. Future research can further investigate this effect and develop more dynamic methods to account for any change in learning task. Next, IAS can be adapted for convolutional neural networks (CNN) as Net2Net can be easily adapted to CNNs. This will enable IAS to take on more challenging image classification tasks such as CIFAR-10 [12] and ImageNet [13]. In addition, novel approaches in architecture design such as the inclusion of a reinforcement learning meta controller [7] can improve and replace the current heuristics approach.

5 Conclusion

In this paper, we presented IAS, a new framework to explore continual learning. With the use of Net2Net techniques, it is possible for a trained neural network to transfer knowledge to another network. In this way, a neural network can continue to expand and adapt to newer tasks. IAS efficiently samples new architectures at each step by starting from the best architecture from the previous step. Experiments with MNIST and Fashion-MNIST dataset have shown that learning incrementally can be as competitive as retraining a network from scratch with a random initialization. We also observed that the change in learning task during class-incremental learning has different effects on individual classes and we provided some insights related to this phenomenon.

References

[1] Parisi, G.I. & Kemker, R. & Part, J.L. & Kanan, Christopher & Wermter, S. (2018) Continual Lifelong Learning with Neural Networks: A Review. *arXiv:1802.07569v2 [cs.LG]*

[2] McCloskey, M. & Cohen, N.J. (1989) Catastrophic interference in connectionist networks: The sequential learning problem. *The Psychology of Learning and Motivation, 24:104-169.*

[3] Rebuffi, S. & Kolesnikov, A. & Sperl, G. & Lampert, C.H. (2016) iCaRL: Incremental Classifier and Representation Learning. *arXiv:1611.07725v2 [cs.CV]*

[4] Chen, T. & Goodfellow, I. & Shlens, J. (2016) Net2Net: Accelerating Learning Via Knowledge Transfer. *arXiv:1511.05641v4 [cs.LG]*

[5] Cai, H. & Chen, T. & Zhang, W. & Yu, Y. & Wang, J. (2017) Neural Architecture Search with Reinforcement Learning. *arXiv:1707.04873v2 [cs.LG]*

[6] Mendoza, H. & Klein, A. & Feurer, M. & Springenberg, J.T. & Hutter, F. (2016) Towards Automatically-Tuned Neural Networks. *Proceedings of the Workshop on Automatic Machine Learning, PMLR 64:58-65, 2016.*

[7] Xu, J. & Zhu, Z. (2018) Reinforced Continual Learning. *arXiv:1805.12369 [cs.LG]*

[8] LeCun, Y. & Bottou, L. & Bengio, Y. & Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11):2278-2324, November 1998*

[9] Xiao, H. & Rasul, K. & Vollgraf, R. (2017) Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747v2 [cs.LG]*

[10] Hinton, G. & Srivastava, N. & Swersky, K. (2014) Overview of mini-batch gradient descent. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`

[11] Lopez-Paz, D. & Ranzato, M. (2017) Gradient Episodic Memory for Continual Learning. *arXiv:1706.08840v5 [cs.LG]*

[12] Krizhevsky, A. (2009) Learning Multiple Layers of Features from Tiny Images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`

[13] Deng, J. & Dong, W. & Socher, R. & Li, L.J. & Li, K. & Li, F.F. (2009) ImageNet: A Large-Scale Hierarchical Image Database. *In CVPR*

[14] Sodhani, S. & Chandar, S. & Bengio, Y. (2018) On Training Recurrent Neural Networks for Lifelong Learning. *arXiv:1811.07017*

Appendix A Capacity Saturation

Class-incremental learning also has the issue of capacity saturation where after a certain number of classes are introduced, a static deep learning architecture will reach its maximum learning capacity. Recent work by Sodhani et al. [14] pointed out that a continual learning agent needs to have expansion property so that the capacity can be increased on the fly. Realizing that both Net2WiderNet and Net2DeeperNet result in a larger network than before, IAS dynamically samples enlarged architectures at each step to ensure that capacity saturation is never reached during class-incremental learning. In Appendix D, we can see that in earlier steps, IAS maintains the initial architecture. After more than 5 classes are introduced, architectures with more capacity are chosen because they are able to tackle the increasing difficulty of the classification task.

Appendix B Learning Complacency

The initialization method described in Section 2.3 ensures that the predicted conditional probabilities for the new class is sufficiently far away from 0 before training. If such probability is close to 0, it might prevent the gradients to be backpropagated properly through the new weights thus hindering the model's ability to learn the new class. As a result, the model would be unable to learn a new class while only maintaining knowledge of existing classes; we refer to this phenomenon as *learning complacency*. While the initialization scheme mentioned previously was sufficient to circumvent this issue in our experiments, a future improvement would be to initialize the new weights relatively to existing weights in the output layer.

Appendix C Simple architecture search

All changes are made to the last hidden layer H_n. Assume the number of neurons in last hidden layer is k

1. Transform H_n into having $2k$ neurons.
2. Transform H_n into having $4k$ neurons.
3. Add new layer H_{n+1} after H_n and H_{n+1} has k neurons.
4. Add new layers H_{n+1} and H_{n+2} after H_n and they both have k neurons.
5. Transform H_n into having $2k$ neurons and add new layer H_{n+1} after H_n having $2k$ neurons.
6. Transform H_n into having $4k$ neurons and add new layer H_{n+1} after H_n having $4k$ neurons.

Appendix D Architecture evolution of IAS

The following table shows the evolution of the architectures selected by IAS. Note that only structure of the hidden layers are shown with each element in the array indicating the number of neurons in that layer.

Table 1: Evolution of neural architecture selected by IAS

Number of classes	IAS trial 1	IAS trial 2	IAS trial 3
2	[16]	[16]	[16]
3	[16]	[16]	[16]
4	[16]	[16]	[16]
5	[16]	[16]	[16]
6	[32]	[16]	[16]
7	[64]	[16]	[32, 32]
8	[128]	[32]	[32, 32]
9	[256,256]	[64, 64]	[64, 64]
10	[256,256]	[128, 128]	[64, 128]

Appendix E Arrival order of classes

The follow tables shows the arrival order of various classes for MNIST and Fashion-MNIST experiments discussed in 3. MNIST digit 0 is shortened as D_0 and Fashion-MNIST class 0 is shortened as F_0. The classes inside the brackets indicates the initial classes. The corresponding class descriptions for each Fashion-MNIST label can be seen at `https://github.com/zalandoresearch/fashion-mnist`.

Table 2: class arrival order for each experiment

Experiment	Arrival order
MNIST	$(D_0, D_1) \leftarrow D_2 \leftarrow D_3 \leftarrow D_4 \leftarrow D_5 \leftarrow D_6 \leftarrow D_7 \leftarrow D_8 \leftarrow D_9$
Fashion-MNIST and MNIST	$(D_0, ..., D_9) \leftarrow F_0 \leftarrow F_1 \leftarrow F_2 \leftarrow F_3 \leftarrow F_4 \leftarrow F_5 \leftarrow F_6 \leftarrow F_7$

Appendix F Performance of individual MNIST classes

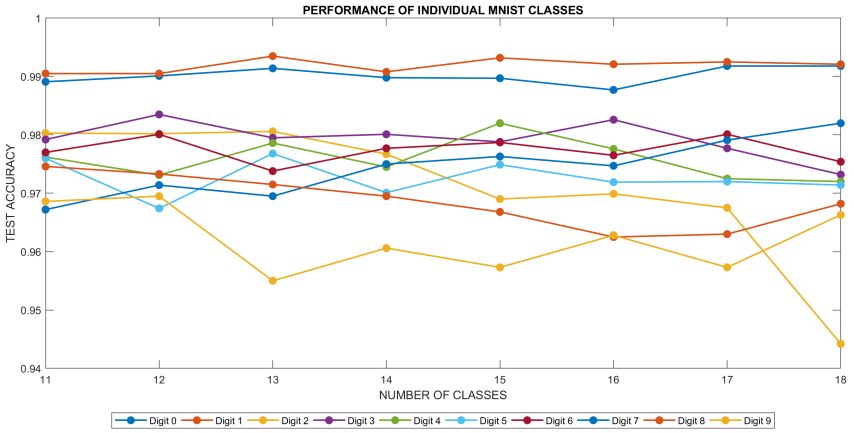

Figure 6: The performance of all 10 MNIST classes during incremental learning experiment with MNIST and Fashion-MNIST dataset. The results shown are average across three different trials.