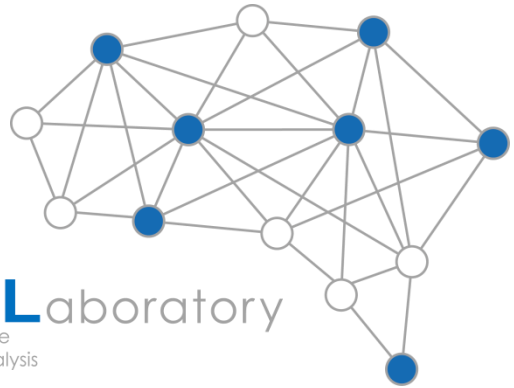
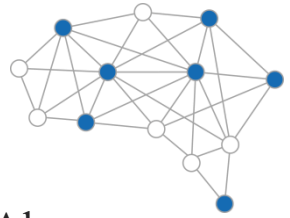
StackNet: Stacking feature maps for Continual learning

Jangho Kim*, Jeesoo Kim*, Nojun Kwak
(*Equal Contribution)



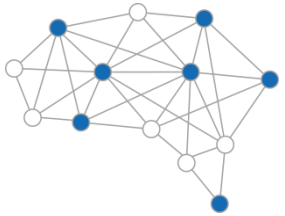
MIPAL aboratory
Machine Intelligence
& Pattern Analysis

Introduction



- Usually, artificial neural network (ANN) models consist of a finite number of filters. Also, parameters and operations in the ANN do not possess the ability corresponding to the memory system of human brains.
- This structural limit leads to the problem called catastrophic forgetting, i.e., the newly coming information diverts the model from previously leaned knowledge
- In this paper, we propose a network which efficiently uses the capacity of the network for continual learning without degrading the performance of previous tasks
- We have built our method using two complementary components which are **StackNet** and **Index module**

Related works

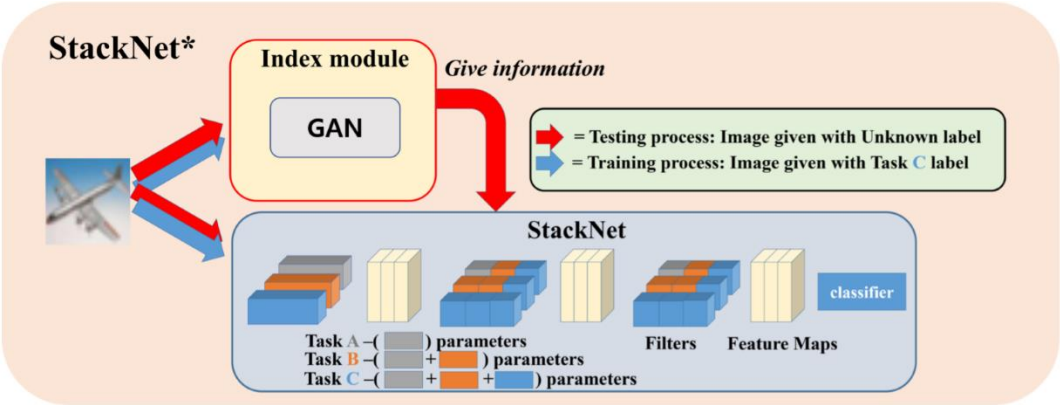


- Learning without forgetting (LwF) [1]
  - A method trying to retrain {the model by using the outcomes of the previously learned model as anchoring targets.
  - When training a new task, a considerably small learning rate is used to prevent the model from losing the learned knowledge.
- PackNet [2]
  - A method using the weight pruning method.
  - Leaving the remaining weights fixed after pruning, the model can solely focus on the task on interest.
  - Despite of its efficiency and performance, this method, referred as PackNet, requires the prior knowledge of a given input

[1] Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017): 2935-2947.
[2] Mallya, Arun, and Svetlana Lazebnik. "Packnet: Adding multiple tasks to a single network by iterative pruning." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

Method

- After training the **StackNet** from the previous task with a certain amount of parameters, additional parameters in the convolutional modules are stacked and trained with the next task.
- In order to determine which combination of filters to use, we adopted a **index module** which can distinguish the origin of a given input sample.
- The combined **index module** and **StackNet** can prevent the catastrophic forgetting using different parameters for different tasks under the constraint network capacity.



StackNet* = StackNet + Index module

Index module (GAN)

- we introduce the index module inspired by [3] using several different methods, which is able to estimate the task of the given input data and inform this to the StackNet to specify which filters to use.
- We train a task-specific generative model using a GAN to generate pseudo-samples of each task.
- A task-wise binary classifier is trained to classify whether the given input is from the task or not.
- Index module can figure out the task of input data with maximum probability across the set of classifiers.

(a) Generated samples with the generator of index module (b) Real samples

(a) Generated samples with the generator of index module (b) Real samples

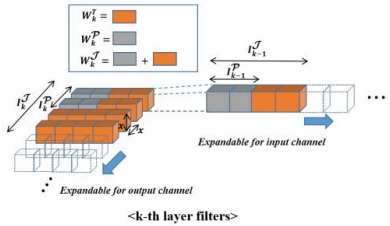
(a) Generated samples with the generator of index module (b) Real samples

MIPAL aboratory
Machine Intelligence & Pattern Analysis

[3] Li, Zhizhong, and Derek Hoiem. "Learning without forgetting." *IEEE transactions on pattern analysis and machine intelligence* 40.12 (2017): 2935-2947.

StackNet

- We propose an efficient way to allocate the capacity of a network according to the given tasks.
- While training the StackNet with several tasks, each task uses a different part of the StackNet.
- we utilize a network divided into several parts and refer it as StackNet where each part takes charge of a particular task.
- we introduce a filter index I^J (J task). The filter index I^J defines the range of filters to use for the specific task J in every layers.
- To avoid Forgetting, when we train the new weight for new task, we freeze the previous weights and only update the new weights for new task.



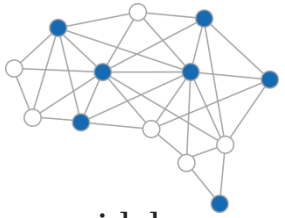
Algorithm 1 StackNet Training

Input: $W^{\mathcal{J}}, I^{\mathcal{P}}, I^{\mathcal{J}}$

Output: $W^{\mathcal{J}*}$

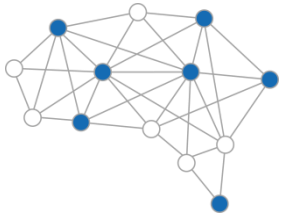
1: Freeze the $W^{\mathcal{P}} \subset W^{\mathcal{J}}$ using the information $I^{\mathcal{P}}$
2: Initialize the $W^{\mathcal{T}}$ with $W^{\mathcal{P}}$ and a random noise
3: $W^{\mathcal{J}*} \leftarrow \arg\min(\mathcal{L}_c)$
{Update W^T using backpropagation}

Results



- we verify the effectiveness of our method with MNIST, SVHN and CIFAR-10 which are widely used to evaluate image classification performance. Then, we evaluate our method on two subsets of ImageNet which is a real world image dataset
- We compare our method to various methods such as Learning without forgetting (LwF) and PackNet as well as networks trained for a single target task which shows the performance without conducting continual learning.
- To save the training time, two subsets of the ImageNet dataset each having 50 randomly chosen classes have been used for evaluation. They are referred to as ImageNet-A and ImageNet-B respectively in this paper.

Results



Datasets	Methods	Old(%)	New(%)	Avg. (%)
MNIST ↓ SVHN	single network (MNIST)	99.49	–	–
	single network (SVHN)	–	92.82	–
	LwF ($T = 1$)	98.27	86.40	92.34
	LwF ($T = 2$)	97.33	86.97	92.15
	PackNet	99.45	91.49	95.47
	StackNet*	99.43	92.20	95.82
	StackNet	99.43	92.37	95.90
SVHN ↓ CIFAR-10	single network (SVHN)	92.94	–	–
	single network (CIFAR)	–	79.69	–
	LwF ($T = 1$)	91.76	70.57	81.17
	LwF ($T = 2$)	90.19	71.94	81.07
	PackNet	92.84	76.78	84.81
	StackNet*	90.05	76.62	83.34
	StackNet	92.21	76.70	84.46

Datasets	Methods	MNIST(%)	SVHN(%)	CIFAR(%)	Avg(%)
MNIST ↓ SVHN ↓ CIFAR-10	single network (MNIST)	99.44	–	–	–
	single network (SVHN)	–	92.94	–	–
	single network (CIFAR)	–	–	79.69	–
	LwF ($T = 1$)	95.06	86.80	68.98	83.61
	LwF ($T = 2$)	86.09	85.07	69.63	80.26
	PackNet	99.38	91.93	66.34	85.88
	StackNet*	99.41	89.36	74.93	87.90
	StackNet	99.41	91.84	75.02	88.76

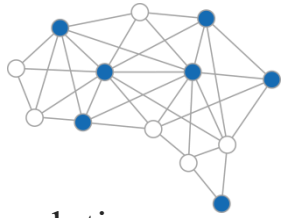
< Mean classification results on the basic datasets with LeNet like networks>

StackNet* = StackNet + Index module

datasets	Methods	Old(%)	New(%)	Avg. (%)
ImageNet-A ↓ ImageNet-B	single network (ImageNet-A)	83.28	–	–
	single network (ImageNet-B)	–	85.28	–
	LwF ($T = 1$)	82.2	86.72	84.46
	LwF ($T = 2$)	80.92	86.96	83.94
	PackNet	82.16	88.72	85.44
	StackNet	83.30	88.66	85.98

<ImageNet-A and ImageNet-B with ResNet-50>

StackNet – Only "Index module" Results

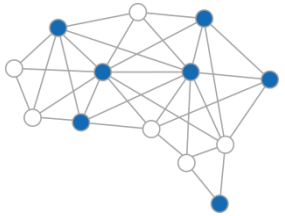


- Generative models usually have difficulties in generating realistic images with high resolution and there are few researches experimented on ImageNet data

Datasets	Methods	Old(%)	New(%)	Avg. (%)
MNIST→ SVHN	GAN	100	99.94	99.97
	Baseline	61.65	97.17	79.41
SVHN→ CIFAR-10	GAN	97.00	99.90	98.45
	Baseline	89.39	85.86	87.63

Datasets	Methods	MNIST(%)	SVHN(%)	CIFAR(%)	Avg(%)
MNIST→ SVHN CIFAR-10	GAN	100	96.65	99.83	98.83
	Baseline	52.88	90.89	75.7	73.16

<Experimental results of the index module on the basic datasets>



Thank you