Reproducibility Challenge: Incremental Learning through Deep Adaption

Zichao Yan[1], Jianing Sun[2]

[1]Department of Computer Science
[2]Department of Electrical Computer Engineering

McGill University

Incremental Learning

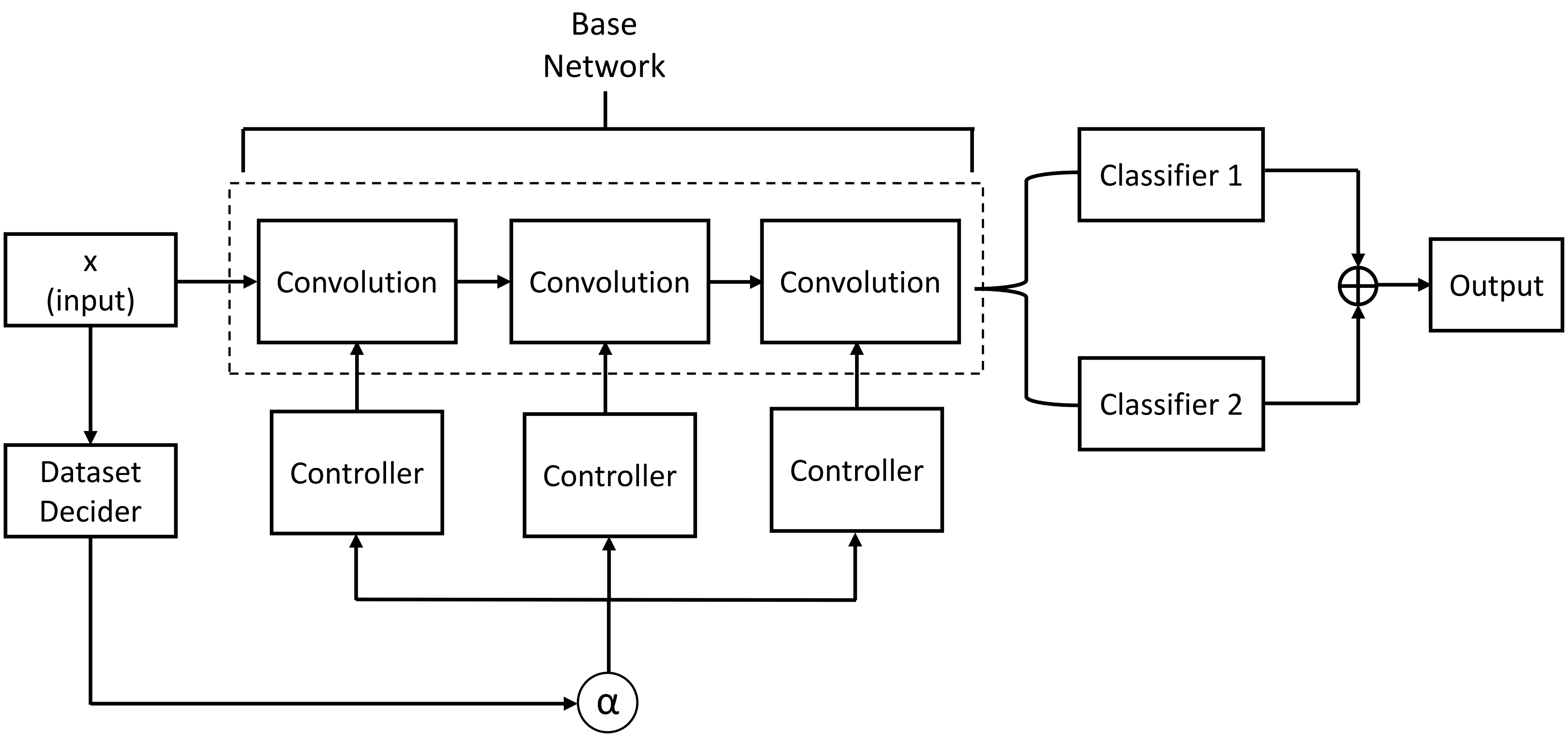
Let’s start from the problem of Deep Neural Network

- Typically a separate model needs to be trained for each new task
- Given two tasks of a totally different modality or nature, each would require a different architecture or computations

Incremental Learning aims at enhancement of knowledge

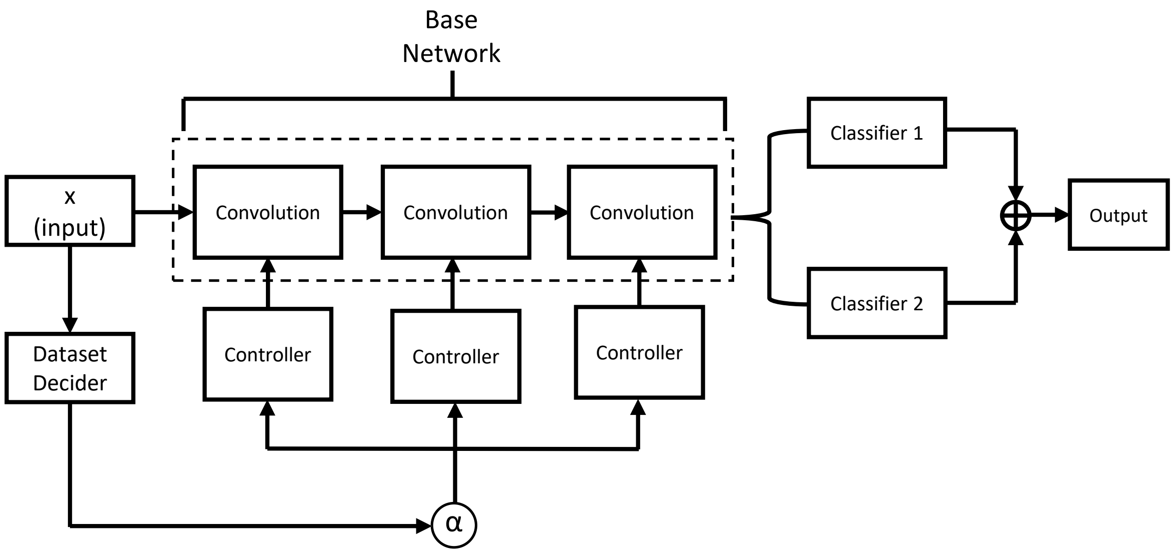
- Our goal is to enable a network to learn a set of related tasks one by one (be learned incrementally).
- Our method could closely approach the average performance of full-transfer learning though requiring a fraction of the parameters (22%)
- Full-transfer learning: double the parameters (100%)
  easy for catastrophic forgetting

Approach



- Each convolutional layer of a base network is modified by re-combining its weights through a controller module
- A binary switching vector α controls the output of the network

Adaption Representations



- Given two tasks, T1 and T2, we then learn a base network N to solve T1
- We augment N so that it will be able to solve T2 as well

For each convolutional layer $\emptyset_l$ in N, let $F_l \in R^{C_0 \times C_i \times k \times k}$, where C_0 is the number of output features, C_l is the number of inputs, $k \times k$ is the kernel size

Denote by $\widetilde{F}_l \in R^{C_0 \times D}$ the matrix whose rows are the flattened versions of the filters of $\widetilde{F}_l$, where $D = C_i \cdot k \cdot k$. Let $f \in R^{C_i \times k \times k}$ be a filter from F_l whose values are

$$f^1 = \begin{pmatrix} f^1_{11} & \cdots & f^1_{1k} \\ & \ddots & \\ & & f^1_{kk} \end{pmatrix}, \cdots, f^i = \begin{pmatrix} f^i_{11} & \cdots & f^i_{1k} \\ & \ddots & \\ & & f^i_{kk} \end{pmatrix}$$

The flattened version of f is a row vector $\tilde{f} = (f^1_{11}, \dots, f^1_{kk}, \dots, \dots f^i_{11}, \dots, f^i_{kk}) \in R^D$

Adaption Representations

Hence, $\widetilde{F_l^a} = W_l \cdot \widetilde{F_l}$

where $W_l \in R^{C_0 \times C_0}$ is a weight matrix defining linear combinations of the flattened filters of F_l, resulting in C_0 new filters

Unflattening $\widetilde{F_l^a}$ to its original shape results in $F_l^a \in R^{C_0 \times C_i \times k \times k}$, which we call the adapted filters of layer $\emptyset_l$.

Using the symbol $a \otimes b$ as shorthand for *flatten b -> matrix multiply by a -> unflatten*, then we can write

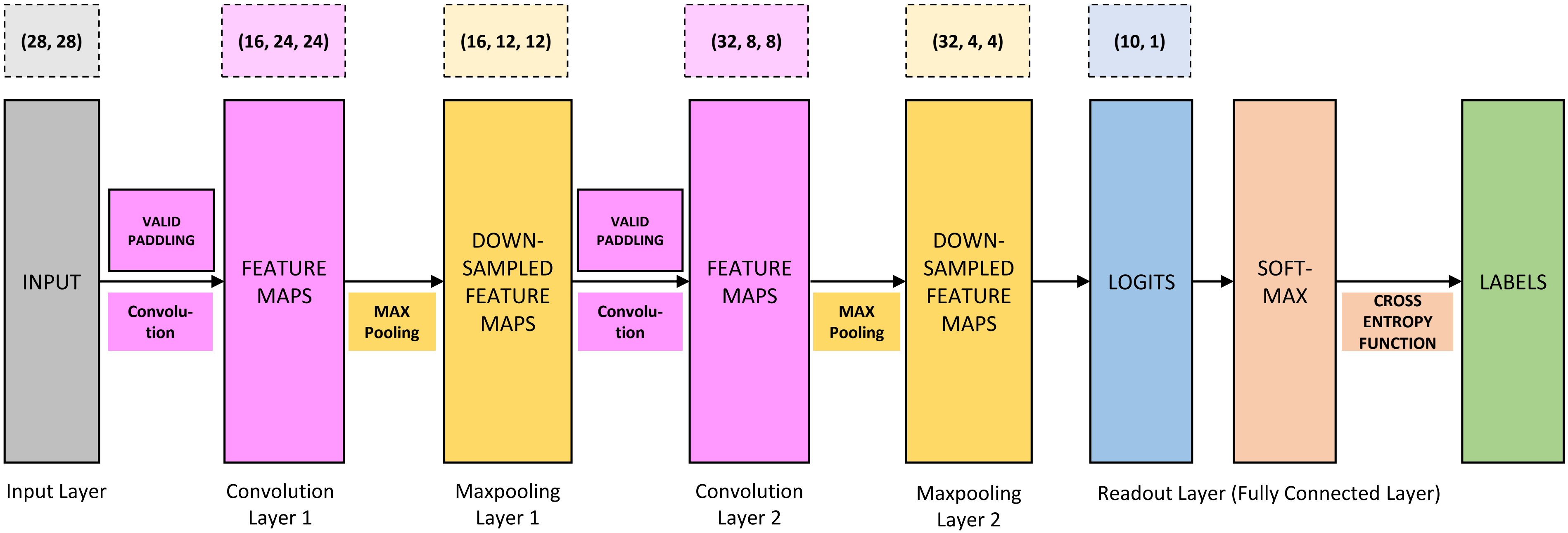
$$F_l^a = W_l \otimes F_l$$

Let x_l be the input of $\emptyset_l$ in the adapted network. For a given switching parameter $\alpha \in \{0,1\}$, we can get the output of the modified layer as follows:

$$\boldsymbol{x_{l+1} = [\alpha(W_l \otimes F_l) + (1-\alpha)F_l] * x_l + \alpha b_l^{\alpha} + (1-\alpha)b_l}$$

Experiments – 2 layer CNN

- Standard 2-layer CNN model:



- New model with controller module:

```
CNNModel (
 (cnn1): controlledConv2 (
  (conv): Conv2d(3, 16, kernel_size=(5, 5), stride=(1, 1))
 )
 (relu1): ReLU ()
 (maxpool1): MaxPool2d (size=(2, 2), stride=(2, 2), dilation=(1, 1))
 (cnn2): controlledConv2 (
  (conv): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
 )
 (relu2): ReLU ()
 (maxpool2): MaxPool2d (size=(2, 2), stride=(2, 2), dilation=(1, 1))
 (fc1): Linear (800 -> 10)
)
```

Experiments – VGG

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Results

Dataset

- CIFAR10

2-layer CNN	Iteration: 500. Loss: 1.9349169731140137. Accuracy: 31.22 Iteration: 1000. Loss: 1.9069454669952393. Accuracy: 35.47 Iteration: 1500. Loss: 1.4878935813903809. Accuracy: 41.57 Iteration: 2000. Loss: 1.4290344715118408. Accuracy: 46.17 Iteration: 2500. Loss: 1.5525197982788086. Accuracy: 47.4 Iteration: 3000. Loss: 1.2857474088668823. Accuracy: 49.0 Iteration: 3500. Loss: 1.4767930507659912. Accuracy: 49.43 Iteration: 4000. Loss: 1.3399474620819092. Accuracy: 49.73
VGG-13	Iteration: 500. Loss: 1.4679466485977173. Accuracy: 53.1 Iteration: 1000. Loss: 1.0690292119979858. Accuracy: 63.31 Iteration: 1500. Loss: 0.9382166266441345. Accuracy: 67.93 Iteration: 2000. Loss: 0.8654510378837585. Accuracy: 70.92 Iteration: 2500. Loss: 0.7722991704940796. Accuracy: 72.63 Iteration: 3000. Loss: 0.5586891770362854. Accuracy: 73.8 Iteration: 3500. Loss: 0.7460127472877502. Accuracy: 74.71 Iteration: 4000. Loss: 0.3032959997653961. Accuracy: 75.82

- We have presented a method to adapt an existing network to new tasks while fully preserving the existing representation, and intuitively VGG should work better than a 2-layer CNN

Future work