Reimplementation of Progressive Neural Networks

Hengda Shi, Gaohong, Liu, Jintao Jiang

Department of Computer Science, UCLA

Catastrophic Forgetting

Fine-tuning [1] is a classic transfer learning example that has the problem of catastrophic forgetting.

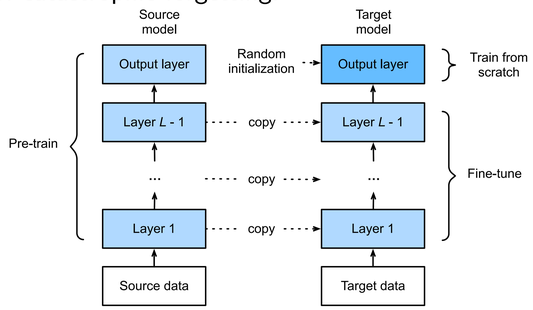

Figure 1: Fine Tuning

The neural network would be trained on the source domain with massive data, and the last layer would be replaced by a new fully connected layer on the target domain.

The problem is that layers would be unfrozen to further fine tuning the neural network. During the retraining on the target domain, information stored in the previous weights would be forgotten.

Progressive Neural Networks [3]

Starts with a single column: a DNN having L layers with hidden activation $h_i^{(1)}$, n_i the number of units at layer i, and parameters $\Theta^{(1)}$.

This generalizes to K tasks as follows:

$$h_i^{(k)} = f(\mathbf{W}_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} \mathbf{U}_i^{(k:j)} h_{i-1}^{(j)})$$

$\mathbf{W}_i^{(k)} \in \mathbb{R}^{n_i \times n_{i-1}}$: the weight matrix of layer i of column k

$\mathbf{U}_i^{(k:j)} \in \mathbb{R}^{n_i \times n_{i-1}}$: lateral connections from layer $i-1$ of column j to layer i of column k

h_0: network input

f: element-wise activation function

When switching to a second task, the parameters $\Theta^{(1)}$ are "frozen" and a new column with parameters $\Theta^{(2)}$ is instantiated (with random initialization), where layer $h_i^{(2)}$ receives input from both $h_{i-1}^{(2)}$ and $h_{i-1}^{(1)}$ via lateral connections.

Progressive Neural Networks[3] (cont'd)

Advantage of Progressive Neural Networks

- Catastrophic forgetting is prevented by instantiating a new neural network (a *column*) for each task being solved.
- Learning can be accelerated by utilizing learned feature from previous columns via lateral connections.
- K independent tasks can be solved at the end of the training.

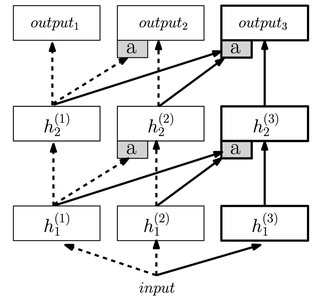

Figure 2: An Example of K=3 Progressive Neural Network

Asynchronous Advantage Actor-Critic Method (A3C) [2]

Utilizing the Progressive Neural Networks as a function approximator, this paper uses A3C method to update agent policy. Instead of having only one agent interacting with the environment, A3C method emphasizes on creating multiple local agents interacting with different scenarios in the environment, and update the global model asynchronously.

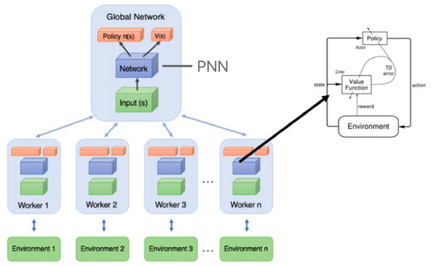

Figure 3: A3C

Result

Experiment on Atari game of Pong. The results of training two columns on the Pong variants including:

- *Noisy* (frozen Gaussian noise is added to the inputs)
- *Black* (black background)
- *White* (white background)
- *Zoom* (input is scaled by 75and translated)
- *V – flip*, *H – flip*, *VH – flip* (input is horizontally and/or vertically flipped)

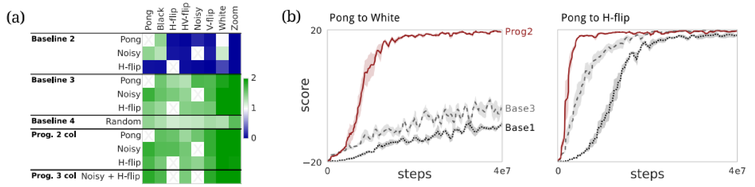

Figure 4: (a) Transfer matrix. Colours indicate transfer scores (clipped at 2). For progressive nets, the first column is trained on Pong, Noisy, or H-flip (table rows); the second column is trained on each of the other pong variants (table columns). (b) Example learning curves.

References

- Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., et al.
  Unsupervised and transfer learning challenge: a deep learning approach.
  In *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning workshop-Volume 27* (2011), pp. 97–111.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K.
  Asynchronous methods for deep reinforcement learning.
  CoRR abs/1602.01783 (2016).
- Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R.
  Progressive neural networks, 2016.