Progressive Neural Networks

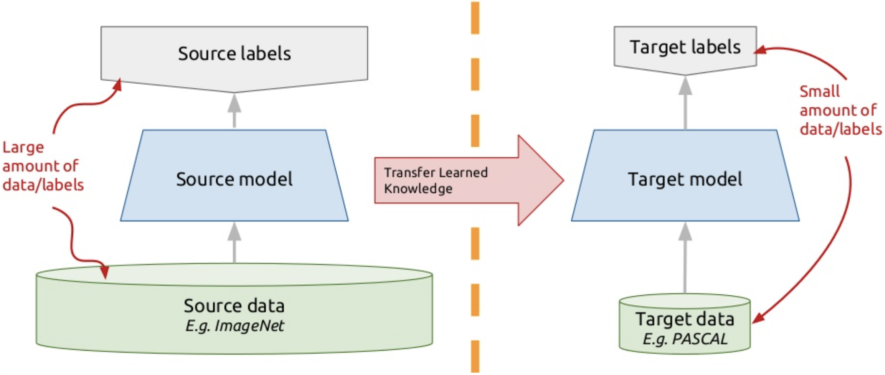
Trofimova Yuliya

Higher School of Economics

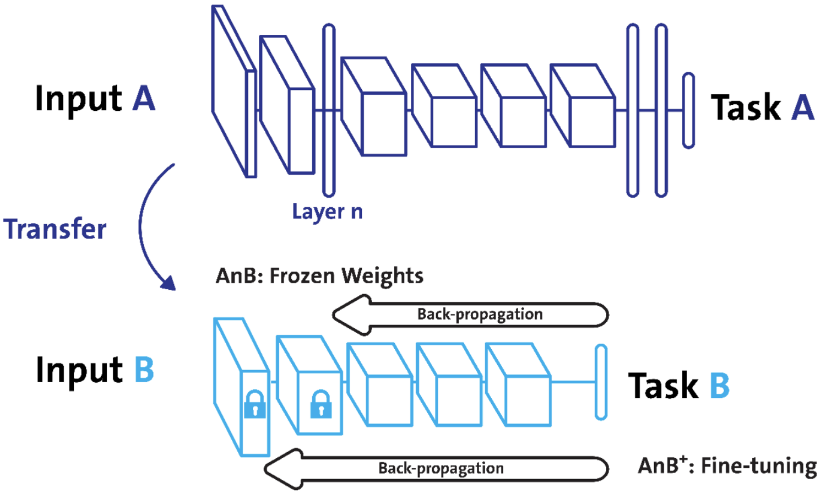
19 декабря 2019 г.

Overview

Transfer learning

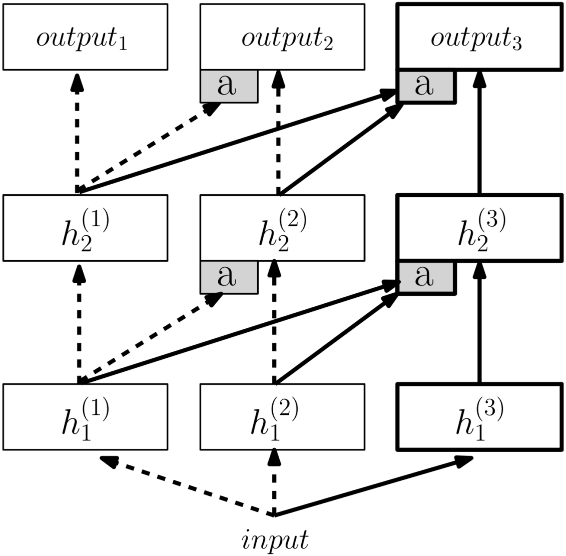


Finetuning



Input A
Task A
Layer n
Transfer
AnB: Frozen Weights
Back-propagation
Input B
Task B
Back-propagation
AnB+: Fine-tuning

Progressive Network



Progressive Network

$$h_i^{(k)} = f\left(W_i^{(k)} h_{i-1}^{(k)} + \sum_{j<k} U_i^{(k:j)} h_{i-1}^{(j)}\right)$$

- L layers
- $h_i^{(k)} \in \mathbb{R}^{n_i}$, with n_i - number of units at layer $i \leq L$
- $\theta^{(k-1)}$ are "frozen"
- $W_i^{(k)} \in \mathbb{R}^{n_i \times n_{i-1}}$ - weight matrix of layer i of column k
- $U_i^{k:j} \in \mathbb{R}^{n_i \times n_j}$ - lateral connections from layer $i-1$ of column j, to layer i of column k and h_0 is the network input
- $f(x) = max(0, x)$ - element-wise non-linearity

Adapters

Define vector of anterior features of dimensionality $n_{i-1}^{(<k)}$

$$h_{i-1}^{(<k)} = [h_{i-1}^{(1)} \cdots h_{i-1}^{(j)} \cdots h_{i-1}^{(k-1)}]$$

$$h_i^{(k)} = \sigma\left(W_i^{(k)} h_{i-1}^{(k)} + U_i^{(k:j)} \sigma(V_i^{(k:j)} \alpha_{i-1}^{(<k)} h_{i-1}^{(<k)})\right)$$

- $\alpha_{i-1}^{(<k)}$ - random small value
- $V_i^{(k:j)}$ - projection onto an n_i dimensional subspace

When k grows, number of parameters in lateral connections is in the same as $\Theta^{(1)}$

Transfer Analysis

Unlike finetuning, progressive nets do not destroy the features learned on prior tasks.

- **Average Perturbation Sensitivity**: inject Gaussian noise at isolated points in the architecture (e.g. a given layer of a single column)
- **Average Fisher Sensitivity**

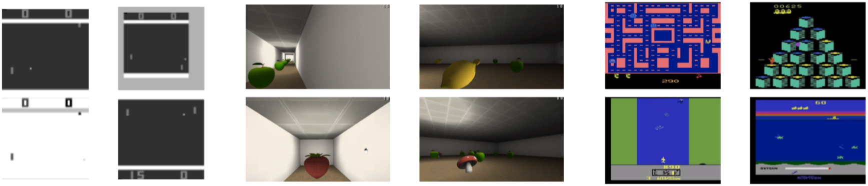
$$\hat{F}_i^{(k)} = \mathbb{E}_{\rho(s,a)} \left[\frac{\partial \log \pi}{\partial \hat{h}_i^{(k)}} \frac{\partial \log \pi}{\partial \hat{h}_i^{(k)}}^T \right] \qquad \mathrm{AFS}(i,k,m) = \frac{\hat{F}_i^{(k)}(m,m)}{\sum_k \hat{F}_i^{(k)}(m,m)}$$

- $\hat{F}$ - diagonal Fisher matrix of policy π with respect to the normalized activations at each layer $\hat{h}_i^{(k)}$
- $\rho(s,a)$ - state-action distribution from target task

Reinforcement learning

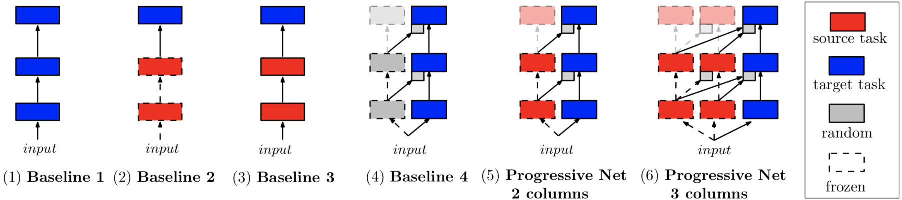
The k-th column defines a policy $\pi^{(k)}(a|s)$ taking as input a state s given by the environment, and generating probabilities over actions

$$\pi^{(k)}(a|s) = h_L^{(k)}(s)$$



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

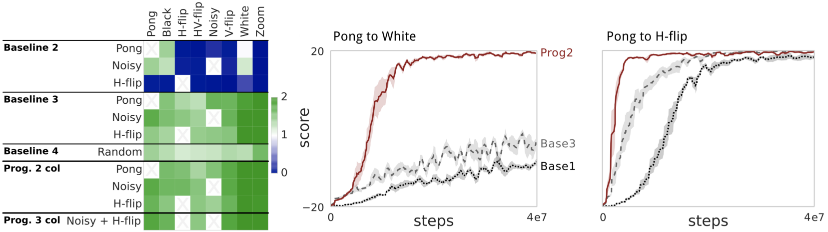
Experiments



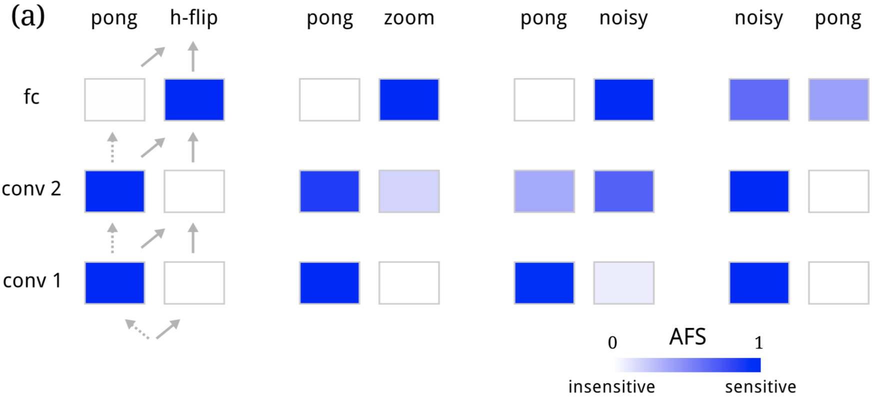
(1) **Baseline 1** (2) **Baseline 2** (3) **Baseline 3** (4) **Baseline 4** (5) **Progressive Net 2 columns** (6) **Progressive Net 3 columns**

Pong Soup

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



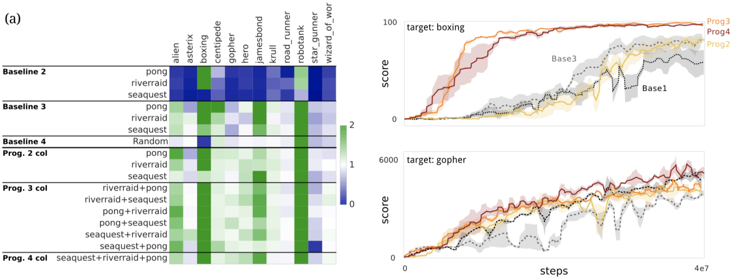
Pong Soup AFS



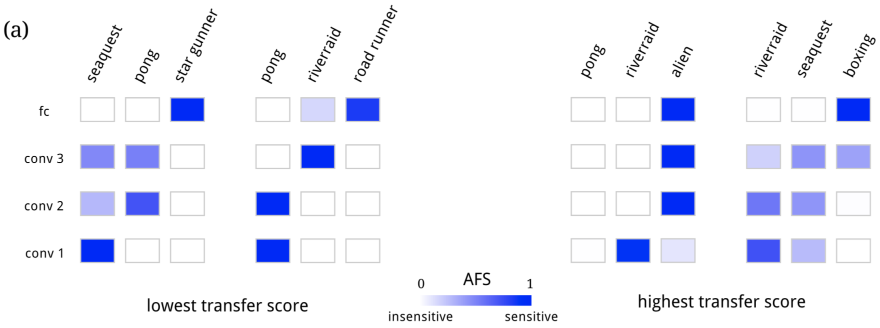
Conv1 - mid-vision, Conv2 - low-vision, fc - policy

Atari

Progressive nets result in positive transfer in 8 out of 12 target tasks and negative transfer is 2. **Baseline 3** in positive transfer with only 5 of 12.

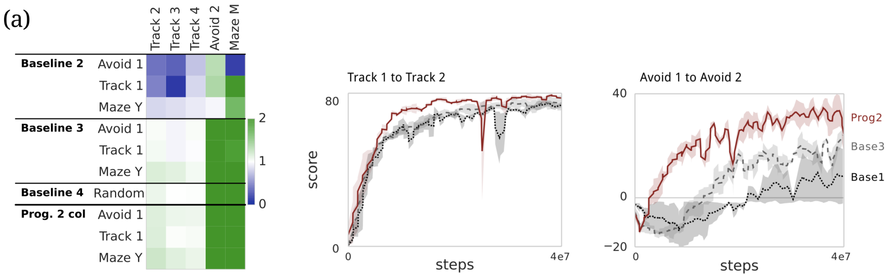


Atari AFS



- The most negative transfer coincides with complete dependence on the convolutional layers of the previous columns, and no learning of new visual features in the new column.
- The most positive transfer occurs when the features of the first two columns are augmented by new features

Labyrinth



- Seek Track 1: simple corridor with many apples
- Seek Track 2: U-shaped corridor with many strawberries
- Seek Track 3: Ω-shaped, with 90 turns, with few apples
- Seek Track 4: Ω-shaped, with 45 turns, with few apples
- Seek Avoid 1: large square room with apples and lemons
- Seek Avoid 2: large square room with apples and mushrooms
- Seek Maze M: M-shaped maze,with apples at dead-ends
- Seek Maze Y: Y-shaped maze,with apples at dead-ends

Conclusions

- Progressive neural networks are a stepping stone towards continual learning
- Progressive approach is able to effectively exploit transfer for compatible source and task domains

Вопросы

- Напишите формулу скрытого слоя для progressive network.
- Почему Progressive network лучше, чем finetuning всей модели, справляются с трансфером из ортогональной задачи?
- Какие зависимости наблюдаются между source и target task в случаях наибольшего негативного и позитивного переноса?