Loss of Plasticity in Deep Continual Learning

«Потеря пластичности в задачах непрерывного глубинного обучения»

Романова Марина, БПМИ212

Problems of continual learning

- Catastrophic forgetting – неспособность DL-моделей запомнить полезную информацию прошлые данные.
- Loss of plasticity – потеря DL-моделью способности обучаться на новых данных.

Пластичность — это способность модели продолжать обучение на новых данных.

Continual ImageNet

Net: 3 convolutional layers + 3 fully connected layers ->

Задача: 2000 последовательных датасетов для бинарной классификации. 600 тренировочных, 100 тестовх изображений.

Оптимизатор: SGD+momentum

Loss: cross-entropy loss

Momentum: 0.9.

Step-sizes: 0.01, 0.001, and 0.0001

Network Architecture for Continual ImageNet			
Layer 1: Convolutional + Max-Pooling			
Number of Filters	32	Activation	ReLU
Convolutional Filter Shape	(5,5)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(1,1)
Layer 2: Convolutional + Max-Pooling			
Number of Filters	64	Activation	ReLU
Convolutional Filter Shape	(3,3)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(1,1)
Layer 3: Convolutional + Max-Pooling			
Number of Filters	128	Activation	ReLU
Convolutional Filter Shape	(3,3)	Convolutional Filter Stride	(1,1)
Max-Pooling Filter Shape	(2,2)	Max-Pooling Filter Stride	(1,1)
Layer 4: Fully Connected			
Output Size	128	Activation	ReLU
Layer 5: Fully Connected			
Output Size	128	Activation	ReLU
Layer 6: Fully Connected			
Output Size	2	Activation	Linear

Continual ImageNet

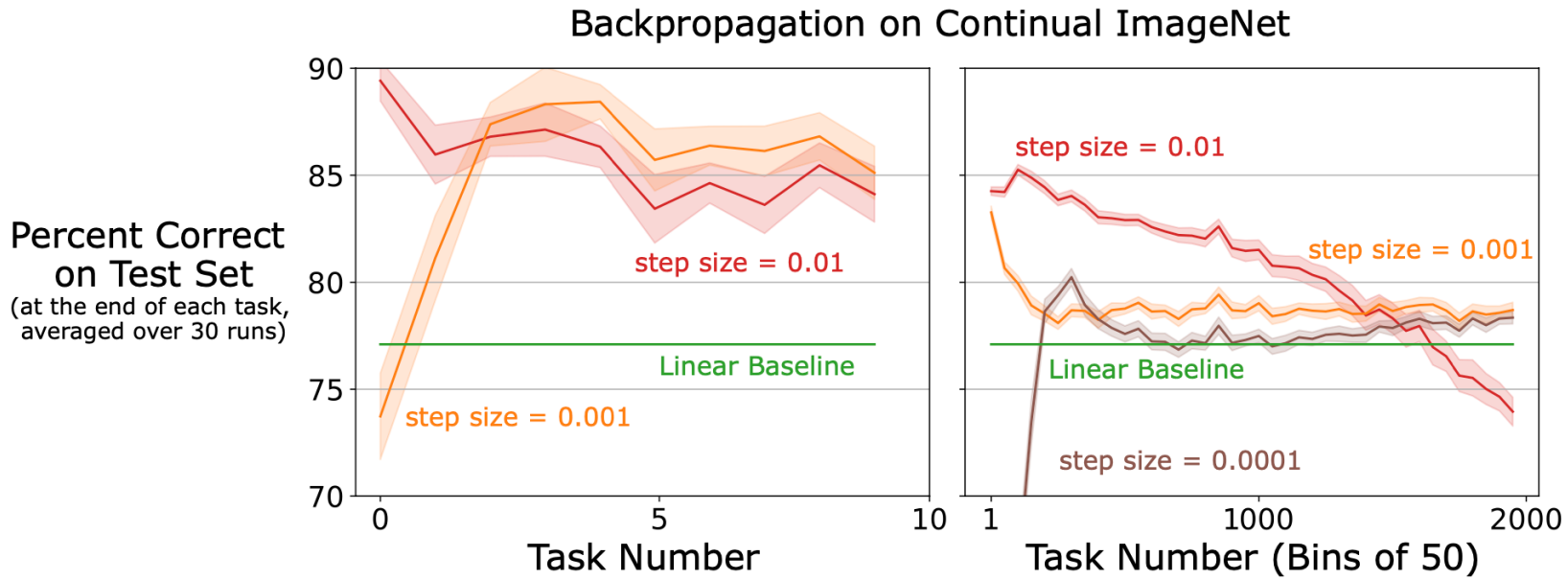
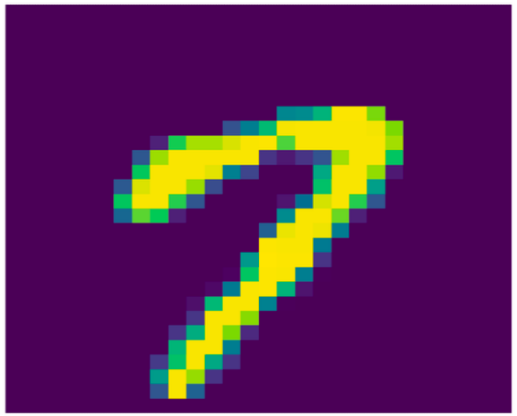


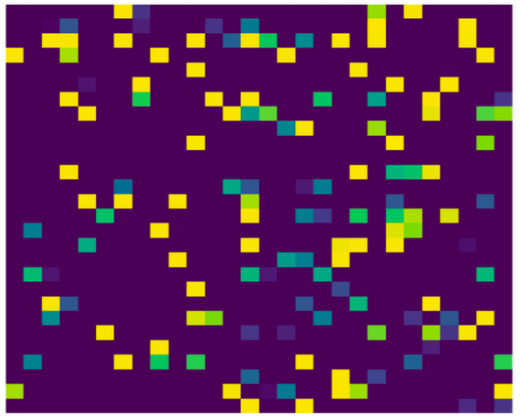
Figure 1: **Loss of plasticity on a sequence of ImageNet binary classification tasks**. The first plot shows performance over the first ten tasks, which sometimes improved initially before declining. The second plot shows performance over 2000 tasks, over which the loss of plasticity was extensive. The learning algorithm was backpropagation applied in the conventional deep-learning way.

Online Permuted MNIST

Net: feed-forward neural networks with three hidden layers

Layer size: 100, 1.000, 10.000





Задача: 800 последовательнх датасетов MNIST с одним алгоритмов перестановок пикселей.

Permutation rate: 10.000, 100.000, or 1.000.000 examples

Оптимизатор: SGD+momentum

Loss: cross-entropy loss

Online Permuted MNIST

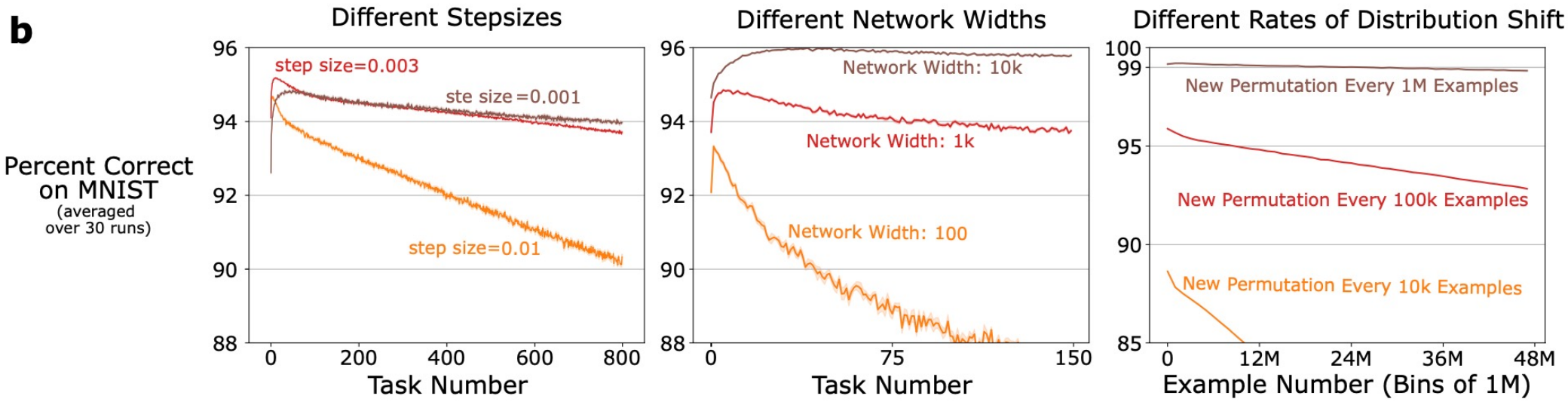
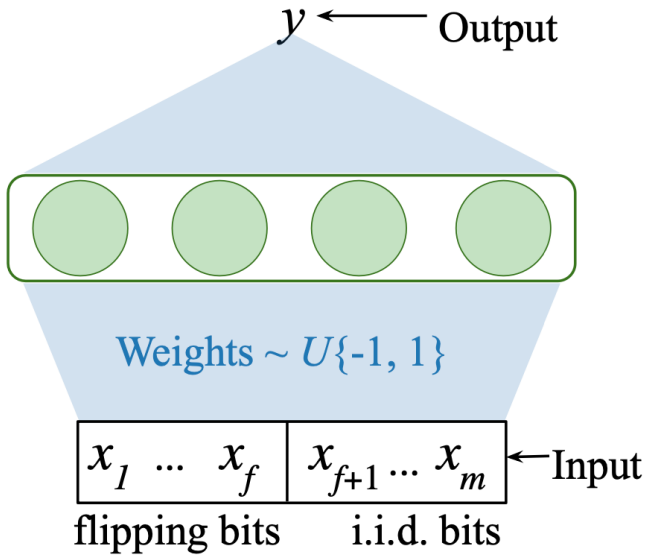


Figure 2: **a:** Left: An MNIST image with the label '7'; Right: A corresponding permuted image. **b:** Loss of plasticity in Online Permuted MNIST is robust over step sizes, network sizes, and rates of change.

Slowly-Changing Regression



Задача: 3М последовательных векторов с лейблами, сгенерированными target net.

Инициализация весами Кайминга

U(-b, b) $b = gain * \sqrt{\frac{3}{num\_inputs}}$

gain(tanh) = 5/3 gain(ELU)= $\sqrt{2}$

gain(sigmoid) =1 gain(Swish)=$\sqrt{2}$

gain(Leaky-ReLU)= $\sqrt{\frac{2}{(1+\alpha\text{^}2)}}$

gain(ReLU)= $\sqrt{2}$

slowly-changing regression Problem Parameters		
Parameter Name	Description	Value
m	Number of input bits	21
f	Number of flipping bits	15
n	Number of hidden units	100
T	Duration between bit flips	10,000 time steps
Bias	Include bias term in input and output layers	True
θ_i	LTU Threshold	$(m+1)\cdot\beta - S_i$
β	Proportion used in LTU Threshold	0.7
Learning Network Parameters		
Parameter Name		Value
Number of hidden layers		1
Number of units in each hidden layer		5

Slowly-Changing Regression

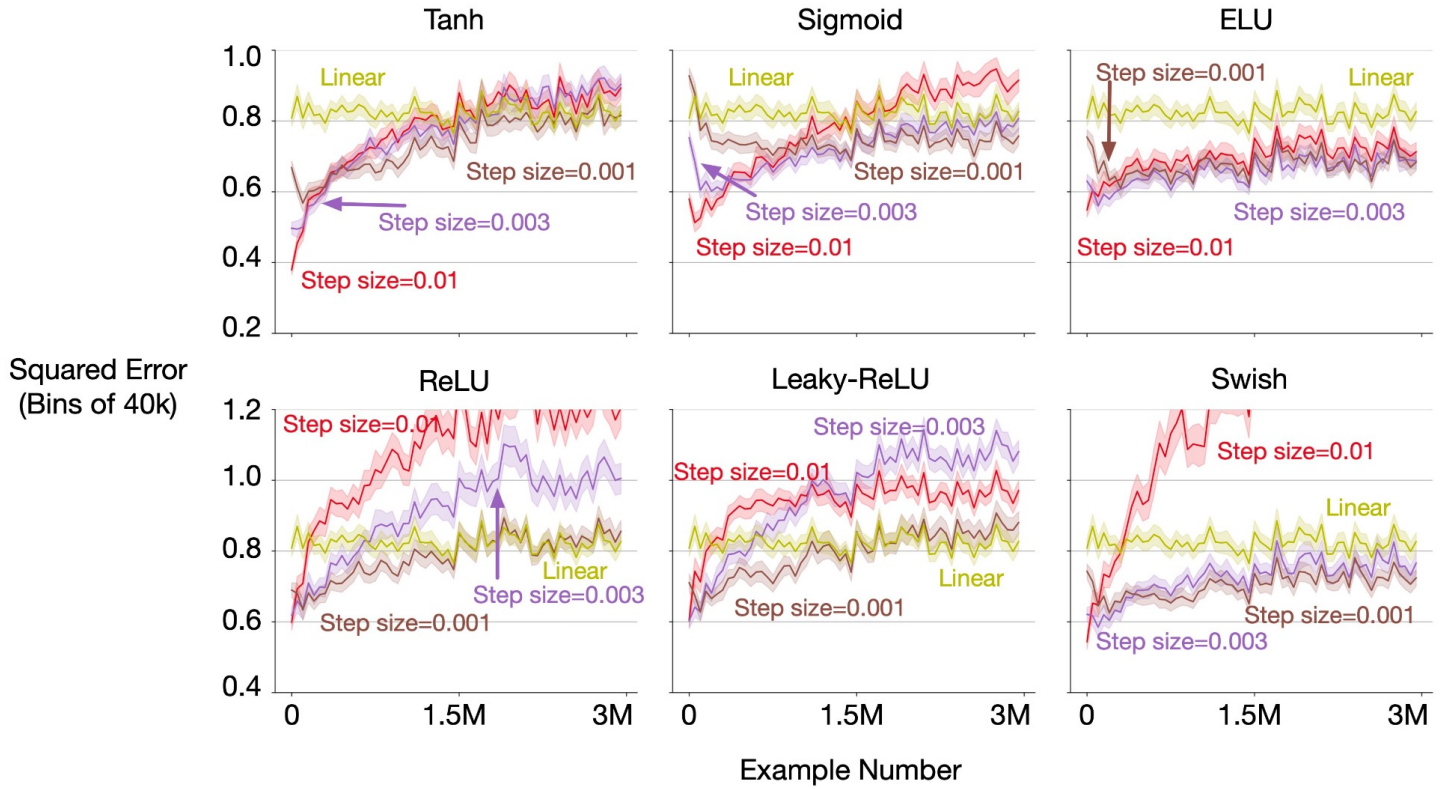
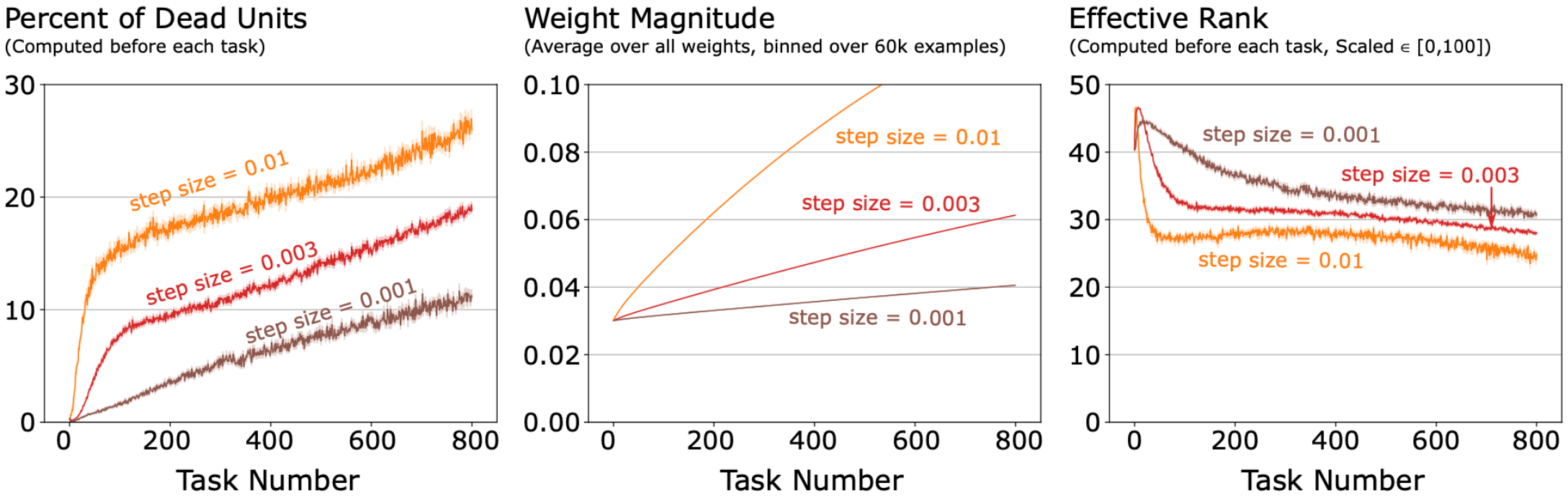


Figure B.10: Loss of plasticity is robust across different activations.

Understanding Loss of Plasticity



Formally, consider a matrix $\Phi \in \mathbb{R}^{n \times m}$ with singular values σ_k for $k = 1, 2, ..., q$, and $q = \max(n, m)$. Let $p_k = \sigma_k / \|\boldsymbol{\sigma}\|_1$, where $\boldsymbol{\sigma}$ is the vector containing all the singular values, and $\|\cdot\|_1$ is the ℓ^1-norm. The effective rank of matrix Φ, or erank(Φ), is defined as

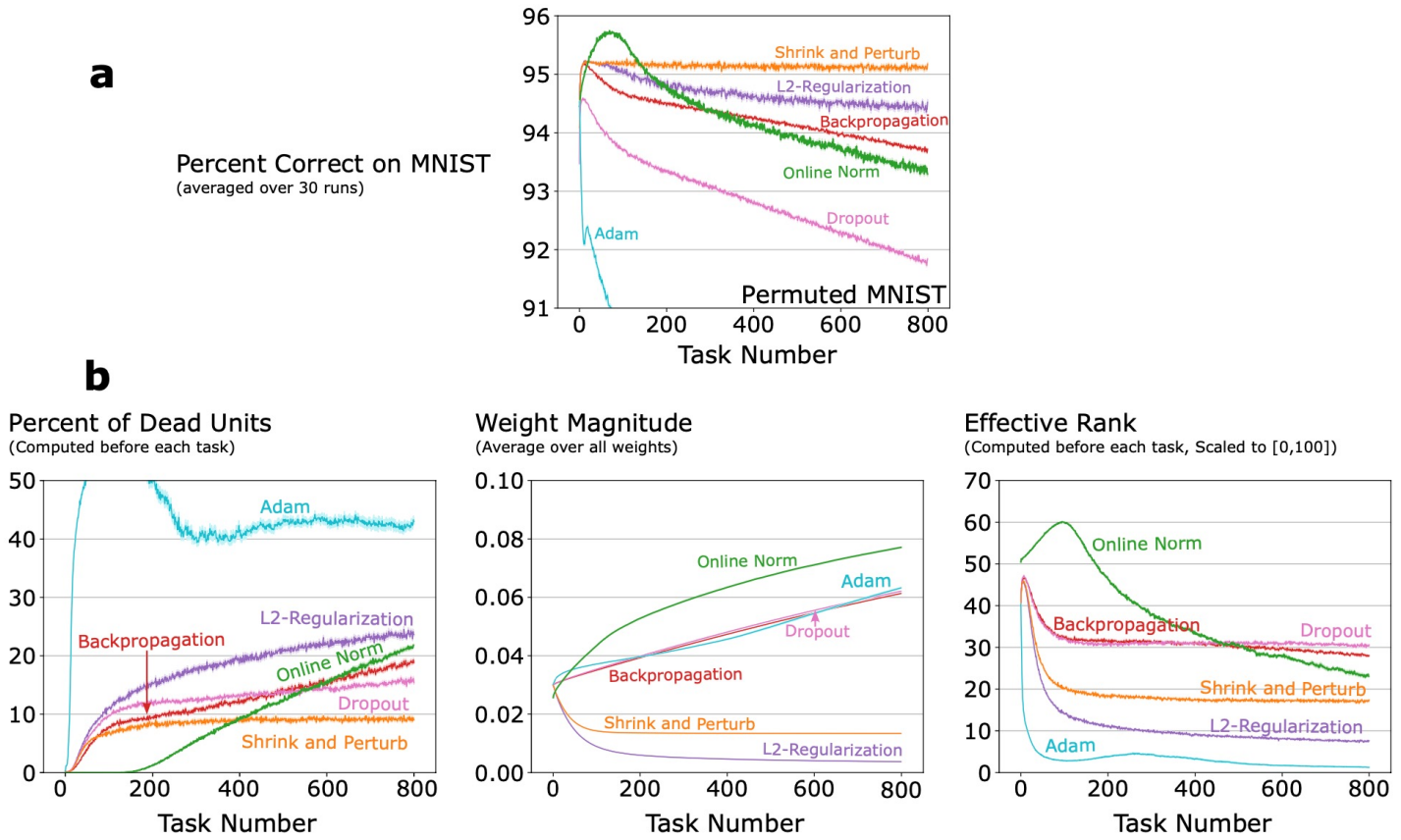
$$\text{erank}(\Phi) \doteq \exp\{H(p_1, p_2, ..., p_q)\}, \text{where } H(p_1, p_2, ..., p_q) = -\sum_{k=1}^{q} p_k \log(p_k). \quad (1)$$

Note that the effective rank is a continuous measure that ranges between one and the rank of matrix Φ.

Existing Deep-Learning Methods for Mitigating Loss of Plasticity

- **L2-регуляризация.** Штрафы за большие веса модели. Борется с повышением средних весов нейронов
- **Shrink-and-perturb.** Уменьшает веса + шум. Борется с всем: повышением среднего веса, мертвыми нейронами и падением эффективного ранга слоев.
- **Dropout.** Обнуляет веса небольшой части нероном. Борется с падением эффективного ранга слоев.
- **Batch normalization.** Нормализует веса слоев. Борется с мертвыми нейронами и падением эффективного ранга слоев.
- **Adam.** Вариация SGD, хорош с нестационарными функциями потерь. Борется с потерей пластичности.

Existing Deep-Learning Methods for Mitigating Loss of Plasticity



Continual Backpropagation

Stochastic Gradient Descent with Selective Reinitialization

На каждом временном шаге проводит:

- Реинициализация весов нейронов неэффективных участков модели небольшими значениями.
- Градиентный спуск

На кажом шаге реинициализируются 0 веса ρ самых бесполезных нейронов. Они получают защиту от реинициализации на m [maturity threshold] шагов.

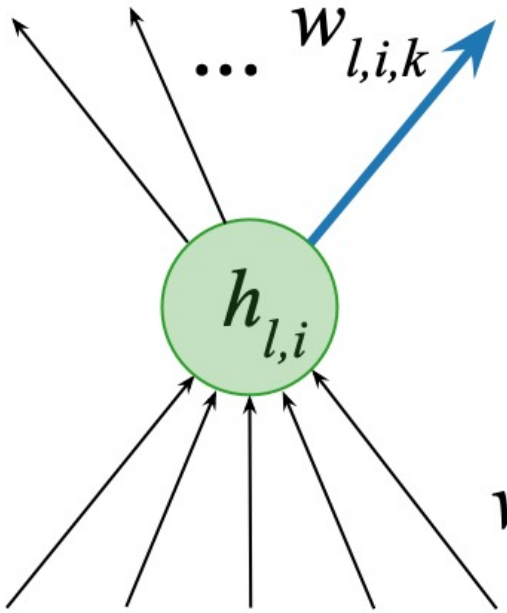
Показатель полезности

- Полезность вклада / contribution utility

Первая часть измеряет вклад нейрона в работу его потребителей. Потребителем является любой нейрон, которая использует продукцию данного нейрона. Потребителем могут быть другие скрытые нейроны или выходные нейноры сети.

- Способность к адаптации / adaptation-utility

Вторая часть полезности измеряет способность нейронов к адаптации, интуитивно пытается определить, насколько быстро скрытый нейрон измерения может изменить функцию, которую он представляет.



$$y_{l,i,t} = \frac{|h_{l,i,t} - \hat{f}_{l,i,t}| * \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|}{\sum_{j=1}^{n_{l-1}} |w_{l-1,j,i,t}|}$$

Contribution utility

In a feed-forward neural network, the contribution-utility, $c_{l,i,t}$, of the ith hidden unit in layer l at time t is updated as

$$c_{l,i,t} = \eta * c_{l,i,t-1} + (1-\eta) * |h_{l,i,t}| * \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|, \tag{2}$$

where $h_{l,i,t}$ is the output of the i^{th} hidden unit in layer l at time t, $w_{l,i,k,t}$ is the weight connecting the i^{th} unit in layer l to the k^{th} unit in layer $l+1$ at time t, n_{l+1} is the number of units is layer $l+1$.

$$f_{l,i,t} = \eta * f_{l,i,t-1} + (1-\eta) * h_{l,i,t}, \tag{3}$$

$$\hat{f}_{l,i,t} = \frac{f_{l,i,t-1}}{1-\eta^{a_{l,i,t}}}, \tag{4}$$

$$z_{l,i,t} = \eta * z_{l,i,t-1} + (1-\eta) * |h_{l,i,t} - \hat{f}_{l,i,t}| * \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|, \tag{5}$$

Adaptation utility + Final utility

$$y_{l,i,t} = \frac{|h_{l,i,t} - \hat{f}_{l,i,t}| * \sum_{k=1}^{n_{l+1}} |w_{l,i,k,t}|}{\sum_{j=1}^{n_{l-1}} |w_{l-1,j,i,t}|}$$

$$u_{l,i,t} = \eta * u_{l,i,t-1} + (1 - \eta) * y_{l,i,t},$$

$$\hat{u}_{l,i,t} = \frac{u_{l,i,t-1}}{1 - \eta^{a_{l,i,t}}}.$$

Continual Backpropagation

Algorithm 1: Continual backpropagation (CBP) for a feed-forward network with L hidden layers

Set: step size α, replacement rate ρ, decay rate η, and maturity threshold m (e.g. 10^{-4}, 10^{-4}, 0.99, and 100)

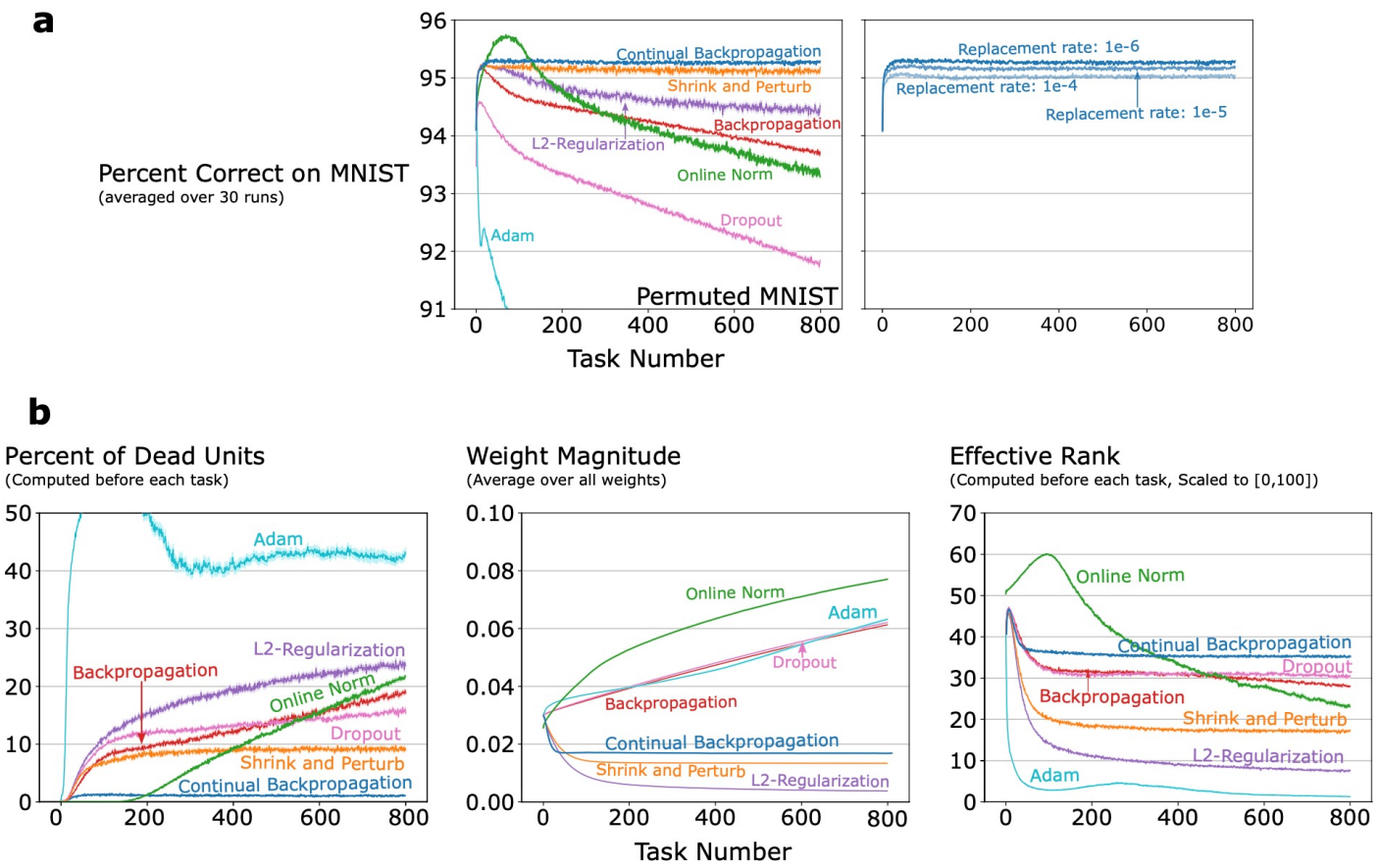
Initialize: Initialize the weights $\mathbf{w}_0, ..., \mathbf{w}_L$. Let, $\mathbf{w}_l$ be sampled from a distribution d_l

Initialize: Utilities $\mathbf{u}_1, ..., \mathbf{u}_L$, average activation $\mathbf{f}_1, ..., \mathbf{f}_l$, and ages $\mathbf{a}_1, ..., \mathbf{a}_L$ to 0

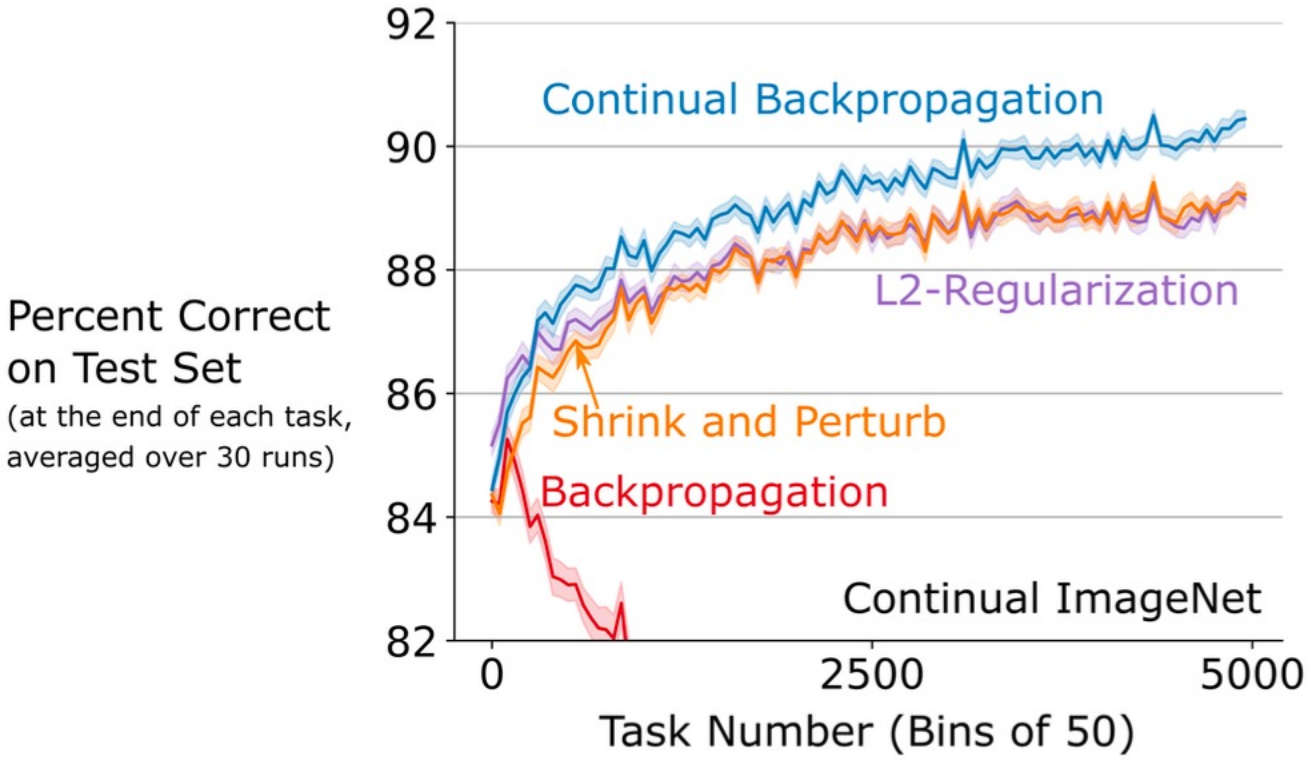
for *each input* x_t **do**

- **Forward pass:** pass input through the network, get the prediction, $\hat{y}_t$
- **Evaluate:** Receive loss $l(x_t, \hat{y}_t)$
- **Backward pass:** update the weights using stochastic gradient descent
- **for** *layer* l *in* $1 : L$ **do**
  - **Update age:** $\mathbf{a}_l \mathrel{+}= 1$
  - **Update unit utility:** Using Equations 4, 5, and 6
  - **Find eligible units:** Units with age more than m
  - **Units to reinitialize:** $n_l * \rho$ of eligible units with the smallest utility, let their indices be $\mathbf{r}$
  - **Initialize input weights:** Reset the input weights $\mathbf{w}_{l-1}[\mathbf{r}]$ using samples from d_l
  - **Initialize output weights:** Set $\mathbf{w}_l[\mathbf{r}]$ to zero
  - **Initialize utility, unit activation, and age:** Set $\mathbf{u}_{l,\mathbf{r},t}$, $\mathbf{f}_{l,\mathbf{r},t}$, and $\mathbf{a}_{l,\mathbf{r},t}$ to 0

Continual Backpropagation



Continual Backpropagation



Спасибо за внимание

Время для вопросов