Dataset Distillation и Generative Teaching Networks

Сергей Исаев, 182

В чём идея Dataset Distillation?

- Нам уже знакома дистилляция нейронных сетей. С помощью большой и умной нейронной сети мы обучаем маленькую, которая почти не проигрывает в качестве.
- Решим “ортогональную” задачу — по большой обучающей выборке (a.k.a. dataset) построим маленькую, которая сможет обучить нейросеть за один спуск.
- MNIST мы сможем сжать до 10 картинок (по одной на цифру).

Как мы обычно обучаем нейросети?

Обучающая выборка: $\mathbf{x} = \{x_i\}_{i=1}^{N}$

Хотим найти нейросеть, чтобы ошибка была минимальна:

$$\theta^* = \arg\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} \ell(x_i, \theta) \triangleq \arg\min_{\theta} \ell(\mathbf{x}, \theta)$$

Для этого делаем много градиентных спусков:

$\theta_{t+1} = \theta_t - \eta \nabla_{\theta_t} \ell(\mathbf{x}_t, \theta_t)$, где $\mathbf{x}_t$ — минибатч.

Dataset Distillation для фиксированной сети

Пусть у нас есть нейросеть θ_0, то есть нам известна её архитектура и начальные веса.

Хотим найти такой шаг $\tilde{\eta}$ и маленькую обучающую выборку $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$,

чтобы обучить θ_0 за один спуск: $\theta_1 = \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0)$

Для этого минимизируем ошибку сети θ_1 на исходной выборке:

$$\tilde{\mathbf{x}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0) = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \ell(\mathbf{x}, \theta_1) = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \ell(\mathbf{x}, \theta_0 - \tilde{\eta}\nabla_{\theta_0}\ell(\tilde{\mathbf{x}}, \theta_0))$$

Dataset Distillation для сети со случайной инициализацией

Теперь мы знаем только архитектуру сети, а про начальные веса знаем лишь то, что они берутся из распределения $p(\theta_0)$.

Нам необходимо решить эту задачу (всё ещё хотим обучить нейросеть за один градиентный спуск):

$$\tilde{\mathbf{x}}^*, \tilde{\eta}^* = \arg\min_{\tilde{\mathbf{x}}, \tilde{\eta}} \mathbb{E}_{\theta_0 \sim p(\theta_0)} \mathcal{L}(\tilde{\mathbf{x}}, \tilde{\eta}; \theta_0)$$

Для этого во время каждого шага минимизации $\mathcal{L}$ берём случайную нейросеть из $p(\theta_0)$.

Немножко псевдокода

Input: $p(\theta_0)$: distribution of initial weights; M: the number of distilled data
Input: α: step size; n: batch size; T: the number of optimization iterations; $\tilde{\eta}_0$: initial value for $\tilde{\eta}$

Initialize $\tilde{\mathbf{x}} = \{\tilde{x}_i\}_{i=1}^{M}$ randomly, $\tilde{\eta} \leftarrow \tilde{\eta}_0$
for each training step $t = 1$ to T **do**
    Get a minibatch of real training data $\mathbf{x}_t = \{x_{t,j}\}_{j=1}^{n}$
    Sample a batch of initial weights $\theta_0^{(j)} \sim p(\theta_0)$
    for each sampled $\theta_0^{(j)}$ **do**
        Compute updated parameter with GD: $\theta_1^{(j)} = \theta_0^{(j)} - \tilde{\eta}\nabla_{\theta_0^{(j)}}\ell(\tilde{\mathbf{x}}, \theta_0^{(j)})$
        Evaluate the objective function on real training data: $\mathcal{L}^{(j)} = \ell(\mathbf{x}_t, \theta_1^{(j)})$
    end for
    Update $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \alpha\nabla_{\tilde{\mathbf{x}}}\sum_j \mathcal{L}^{(j)}$, and $\tilde{\eta} \leftarrow \tilde{\eta} - \alpha\nabla_{\tilde{\eta}}\sum_j \mathcal{L}^{(j)}$
end for
Output: distilled data $\tilde{\mathbf{x}}$ and optimized learning rate $\tilde{\eta}$

Dataset Distillation для нескольких шагов и эпох

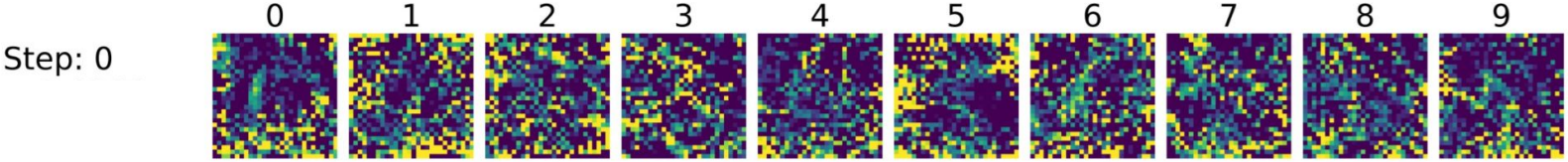
До сих пор мы учили датасет, который обучает нейросеть только за один шаг.

Но мы можем обучить несколько тренировочных выборок и шагов для каждого шага:

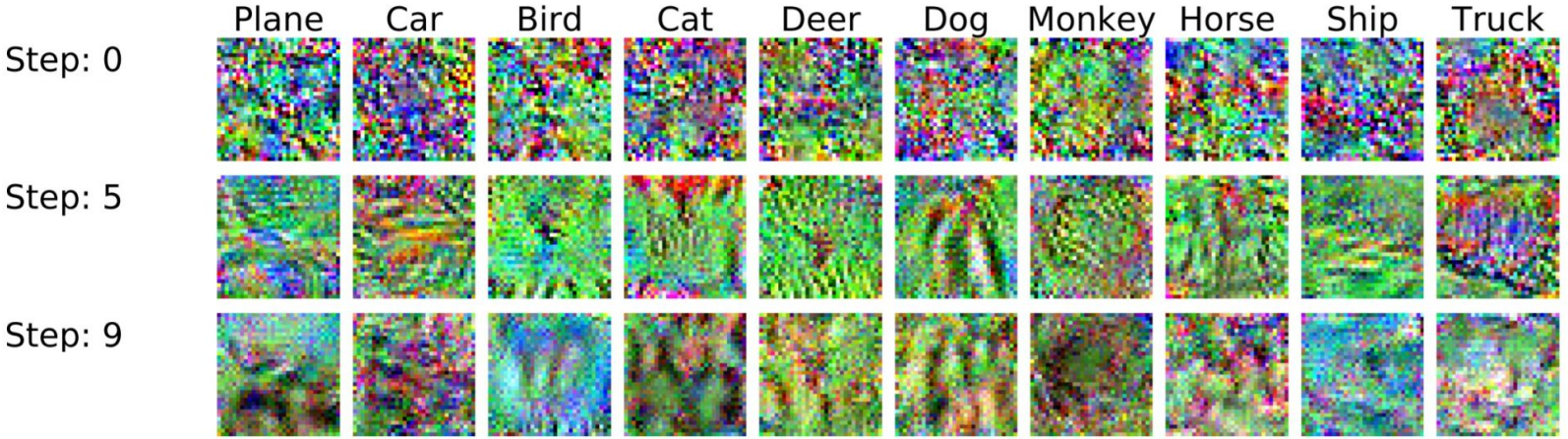
$$\theta_{i+1} = \theta_i - \tilde{\eta}_i \nabla_{\theta_i} \ell(\tilde{\mathbf{x}}_i, \theta_i)$$

Это довольно дорогая задача, ведь градиент придётся пропускать через всю цепочку.

Результат на модели с фиксированными весами

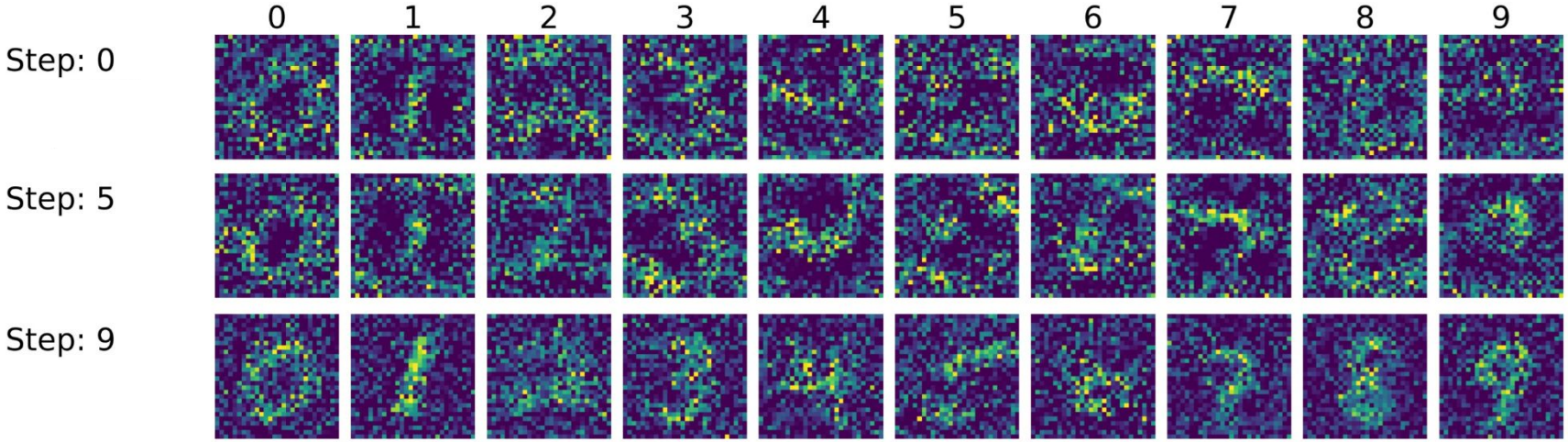


(a) MNIST. Theses distilled images train a fixed initializations from 12.90% test accuracy to 93.76%.



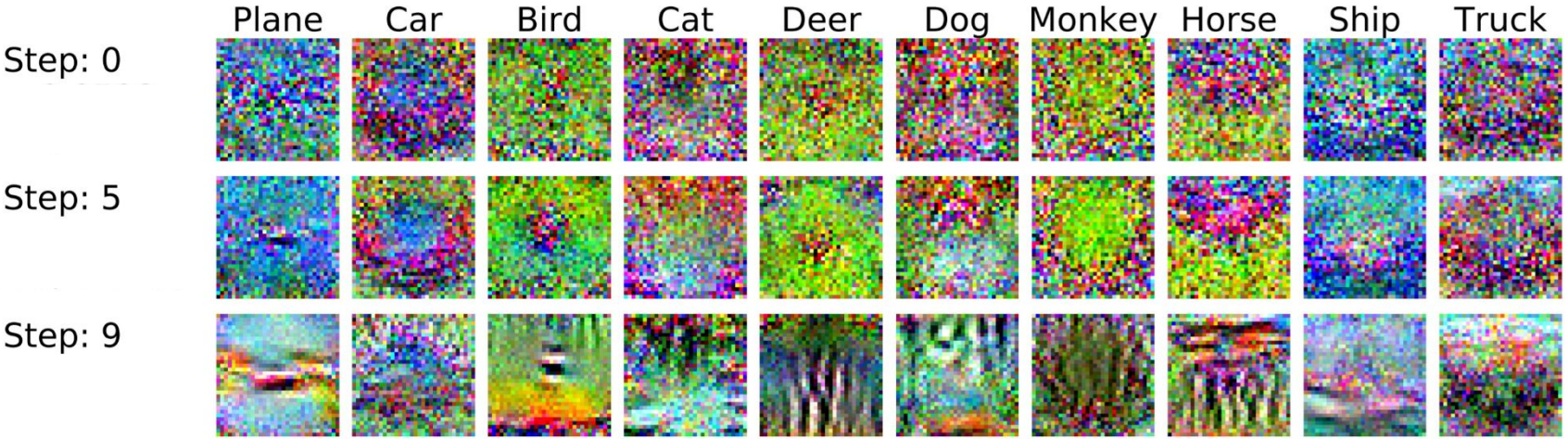
(b) CIFAR10. These distilled images train a fixed initialization from 8.82% test accuracy to 54.03%.

Результат на модели со случайными весами (MNIST)



(a) MNIST. These distilled images unknown random initializations to $79.50\% \pm 8.08\%$ test accuracy.

Результат на модели со случайными весами (CIFAR10)

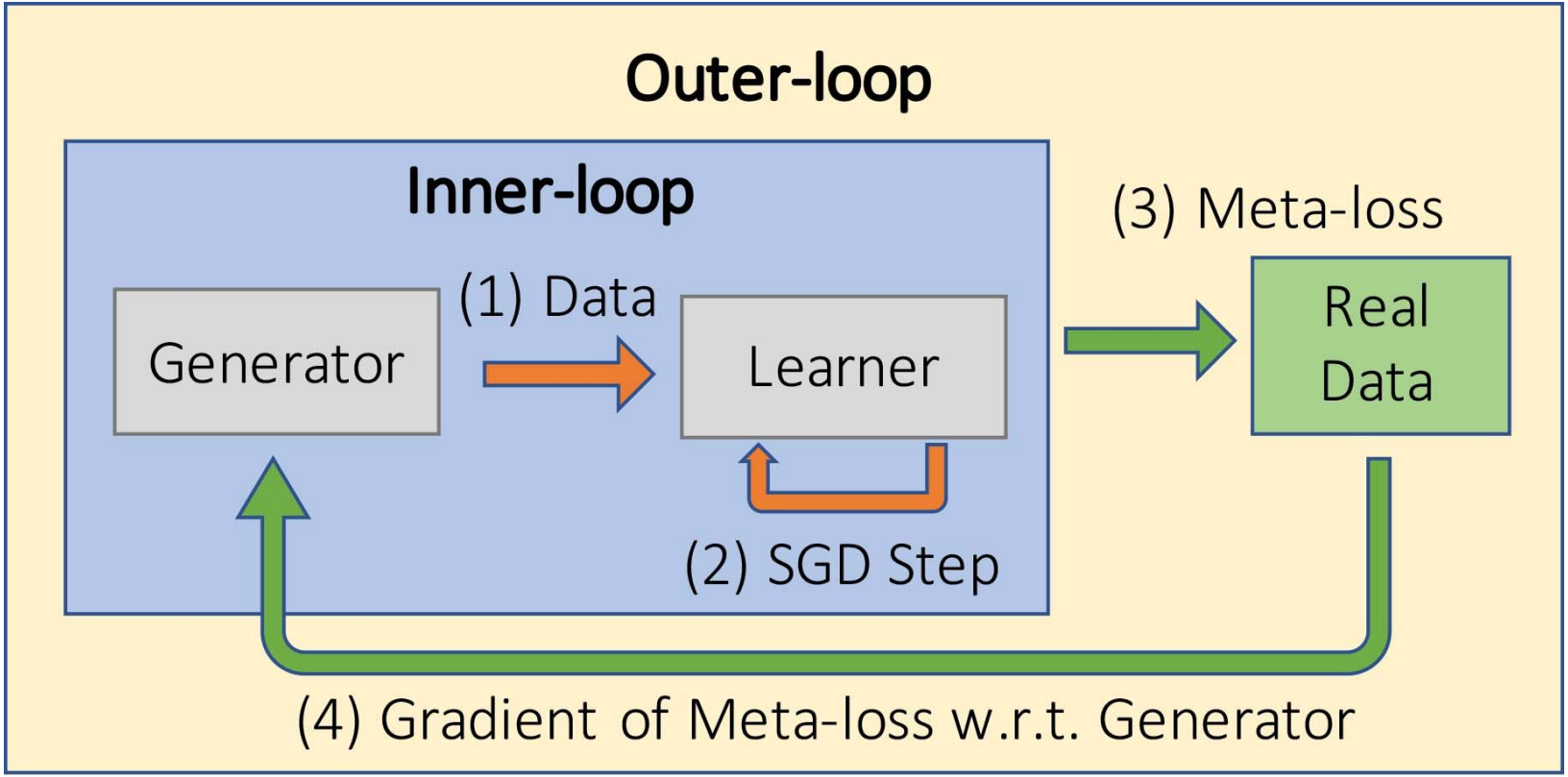


(b) CIFAR10. These distilled images unknown random initializations to $36.79\% \pm 1.18\%$ test accuracy.

Качество выходит довольно низкое. Нужно как-то улучшить алгоритм генерации тренировочной выборки.

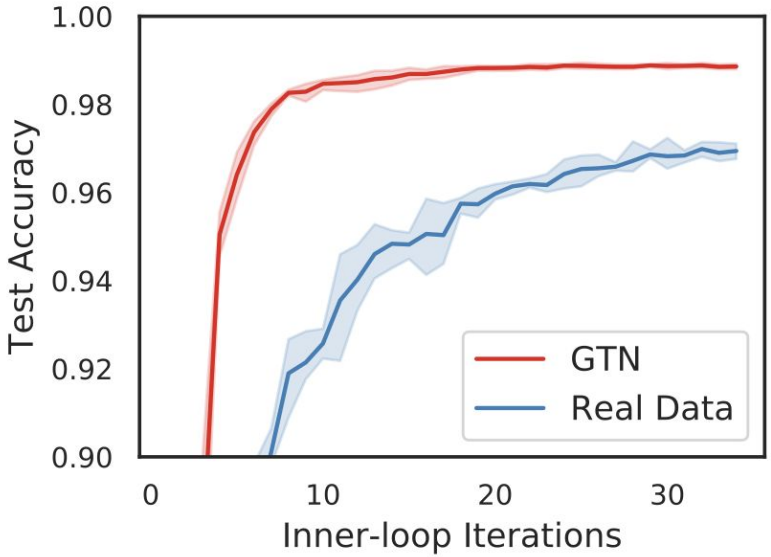
Generative Teaching Networks

- Вместо тренировочной выборки будем обучать нейросеть-генератор, которая по шуму будет строить примеры для нейросети-студента.
- Такой подход гораздо мощнее, поэтому в процессе обучения нейросети-генератора будем подкидывать нейросеть не только со случайными начальными весами но и со случайной архитектурой.

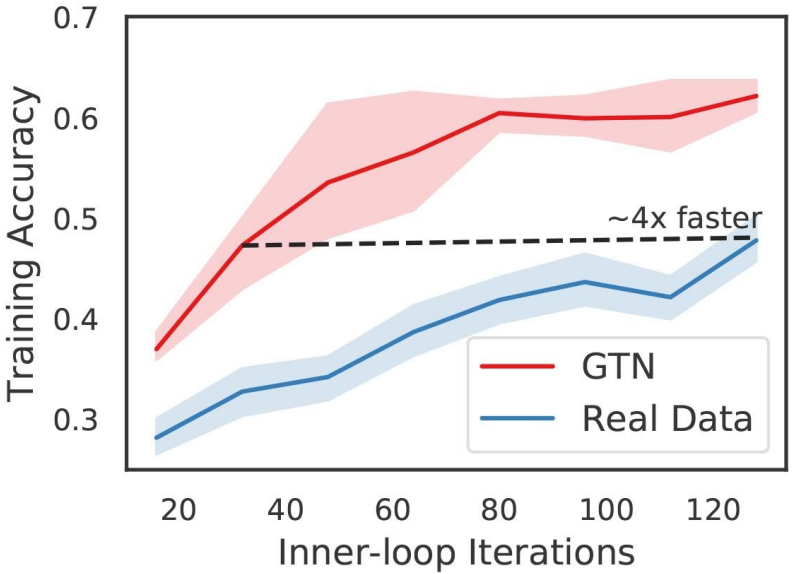


The generator (a deep neural network) generates synthetic data that a newly created learner neural network trains on. After training on GTN-produced data, the learner is able to perform well on the target task despite never having seen real data.

GTN: Результаты



MNIST



CIFAR10

Качество обучения на таких синтетических примерах значительно выше.
Также мы избавились от зависимости данных от архитектуры сети.

Применение синтетических данных

Рассмотрим две области применения синтетических данных:

- Synthetic Data for Malicious Data Poisoning
- применение синтетических данных для поиска оптимальной архитектуры нейросети (Neural Architecture Search)

Synthetic Data for Malicious Data Poisoning

- Пусть кто-то собирает обучающую выборку для нейросети, и у нас есть возможность закинуть в неё несколько своих картинок (можно даже одну).
- Наша цель сделать так, чтобы нейросеть направлено ошибалась на каком-то классе (например, красный свет светофора считала зелёным).
- С помощью любой из рассмотренных техник создаём синтетический пример, который обучает нейросеть выдавать неправильный класс.

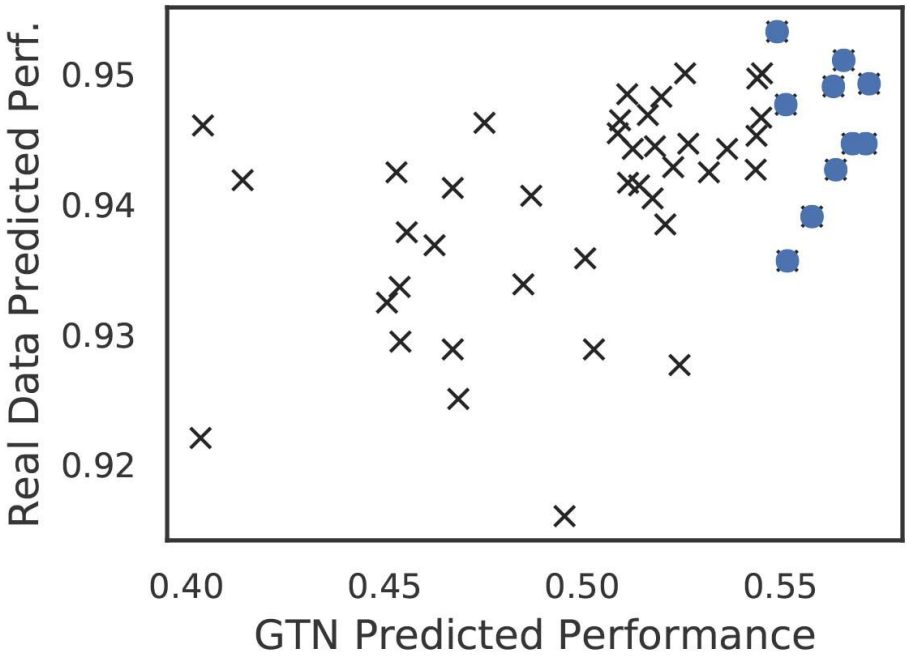
NAS (Neural Architecture Search)

Хотим подобрать архитектуру, которая лучше всего решает нашу задачу.

Полностью обучать каждую архитектуру очень долго. Поэтому будем быстро обучать архитектуру на синтетических данных и предположим, что раз архитектура хорошо сработала для синтетических данных, то и на реальных примерах она поведёт себя хорошо.

Получаем десятикратное ускорение перебора архитектур!

GTN-NAS: Результаты



Correlation plot between final performance after training for 30 seconds with GTN synthetic data compared to. four hours with real data for the top 50 percent of architectures according to the GTN estimate. The correlation is high enough (0.5582 Spearman rank-correlation) that selecting the top architectures according to the GTN estimate will also select architectures that are truly high-performing. Blue squares represent the top 10 percent of architectures according to the GTN estimate.

Итог

Мы разобрали:

- Dataset Distillation
- GTN (Generative Teaching Networks)
- Synthetic Data for Malicious Data Poisoning
- GTN for Neural Architecture Search

Источники

- https://arxiv.org/pdf/1811.10959.pdf (Dataset Distillation + Synthetic Data for Malicious Data Poisoning)
- https://arxiv.org/pdf/1912.07768.pdf (GTN + GTN-NAS)
- https://eng.uber.com/generative-teaching-networks/ (более читабельная выжимка предыдущей статьи)