# Machine Teaching

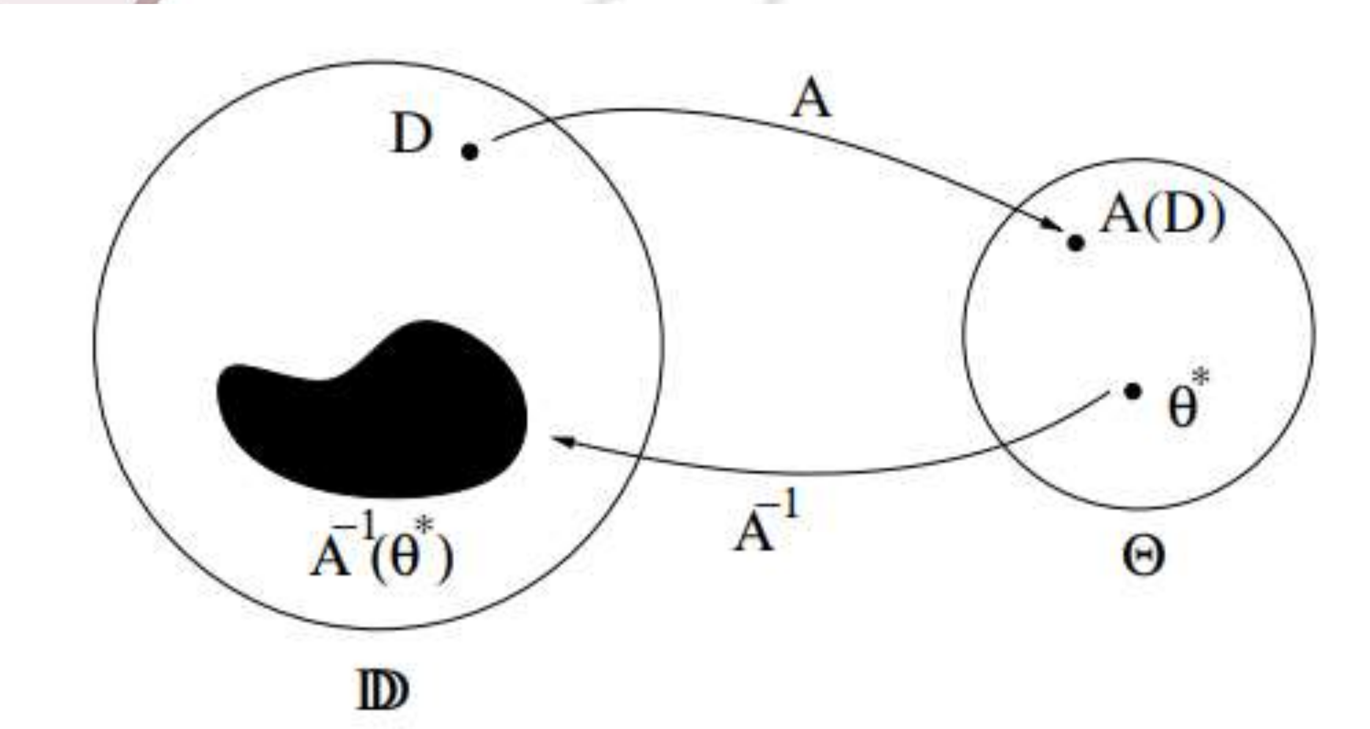
The inverse problem of machine learning

Yi-Fan Liou

# Distillation

- Model distillation
  - Classical method
  - Annoying teacher
- Dataset distillation
  - CVPR2018
    - FAIR
  - ICLR2019
    - …..a sad story……

# Machine Teaching




“Machine Teaching: An Inverse Problem to Machine Learning and an Approach Toward Optimal Education”

D 屬於 𝔻，同時可以透過A投射到Θ。

我們已經知道在Θ有一組參數$\theta *$是可以明確找出在𝔻當中特定的dataset，我們能不能找出這個$A^{-1}$

目標:
如果可以找出一組特定資料對應到最佳模型，我們就能拿這些資料來訓練出另外一台機器。

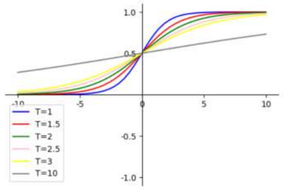
# Distilling knowledge in a neural network

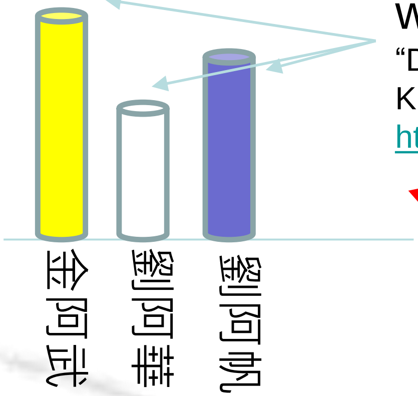
Teacher net

student net

Results can use:

1. Softmax with temperature
2. Logits (better)




Perturbation the logits would be better.

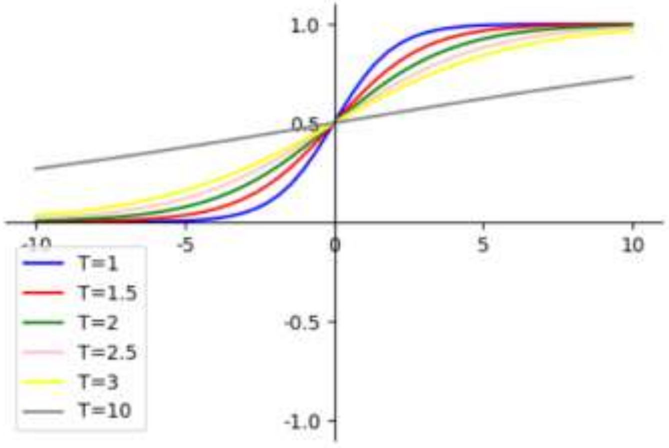
“Deep Model Compression: Distilling Knowledge from Noisy Teachers”
https://arxiv.org/pdf/1610.09650.pdf

loss

results

Ground truth

loss




Teacher nets are thought to lean much “empirical” information which would contain in the categories not activated.

Distilling the Knowledge in a Neural Network


Geoffrey Hinton[*†] Google Inc. Mountain View geoffhinton@google.com

Oriol Vinyals[†] Google Inc. Mountain View vinyals@google.com

Jeff Dean Google Inc. Mountain View jeff@google.com

# Teaching assistant

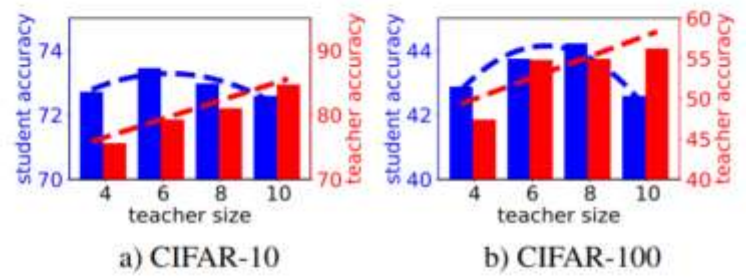



*Figure 2.* Distillation performance with increasing teacher size. The number of convectional layers in student is 2.

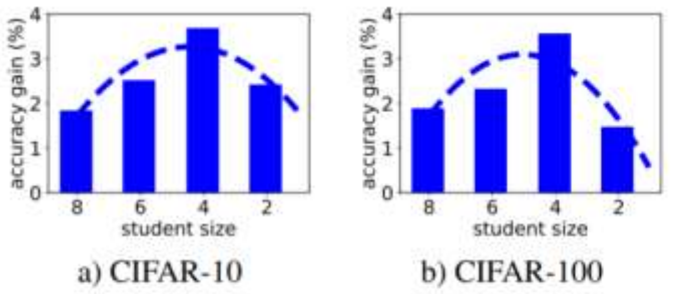
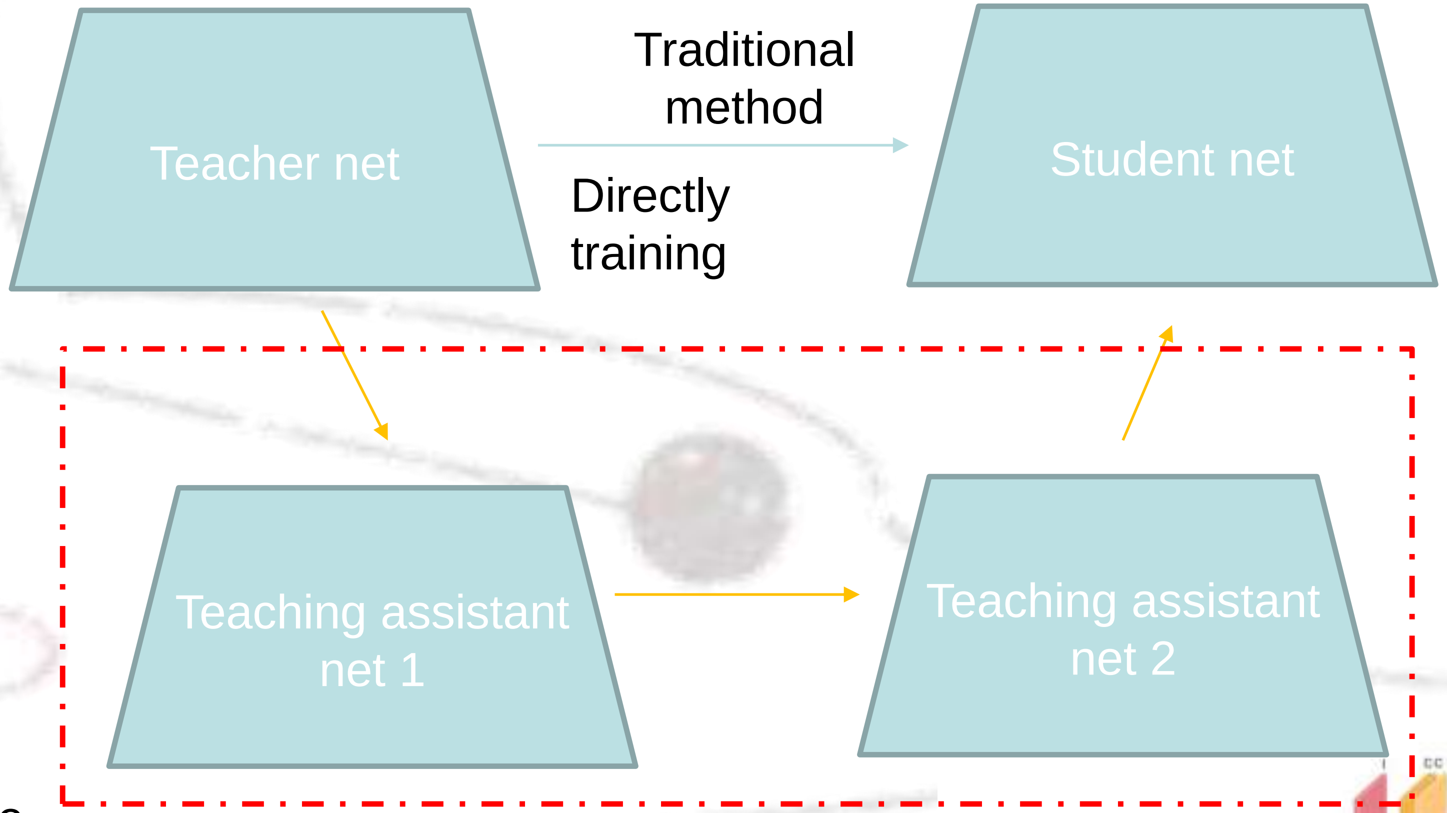



*Figure 3.* Percentage of distilled student performance increase over the performance when it learns from scratch with varying student size. The teacher consists of 10 convolutional layers.



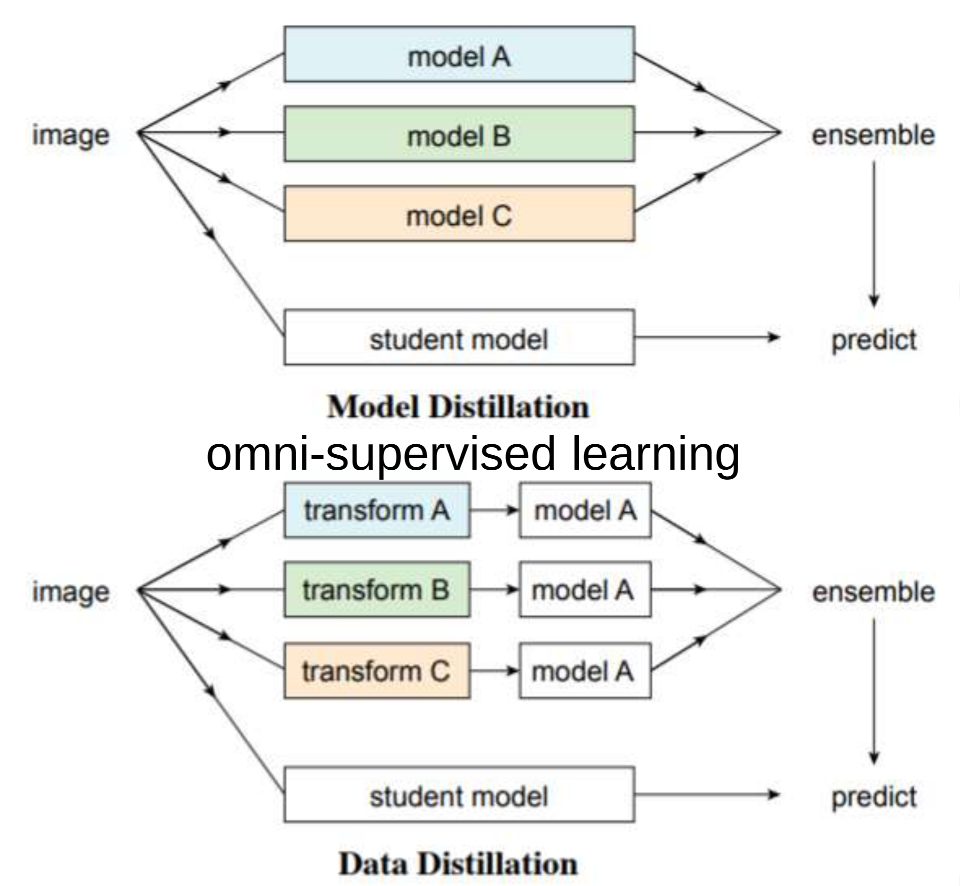

Using several intermediated size neural networks (TA nets)

1. When the power of teachers increase, the student did not have more power.
2. Enlarging the size of student, the power is still not increased.


*Figure 1.* Teacher assistant fills the gap between student & teacher

"Improved Knowledge Distillation via Teacher Assistant: Bridging the Gap Between Student and Teacher", https://arxiv.org/pdf/1902.03393.pdf

# Dataset Distillation – CVPR2018




omni-supervised learning

The comparism of model distillation and dataset distillation.

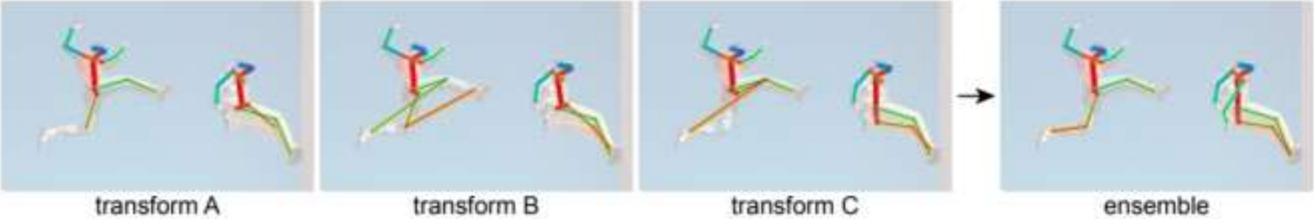
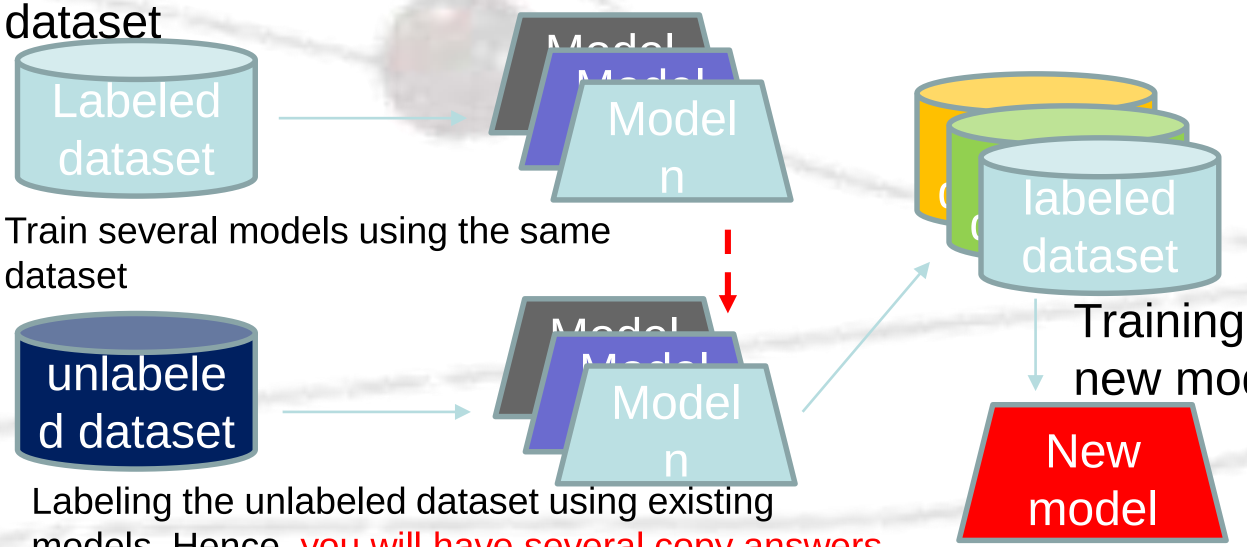



Figure 2. **Ensembling keypoint predictions from multiple data transformations can yield a single superior (automatic) annotation.** For visualization purposes all images and keypoint predictions are transformed back to their original coordinate frame.

Status:

If you have only limited labeled dataset, and much unlabeled dataset




Train several models using the same dataset

Labeling the unlabeled dataset using existing models. Hence, you will have several copy answers with the same dataset.

"Data Distillation: Towards Omni-Supervised Learning", CVPR2018

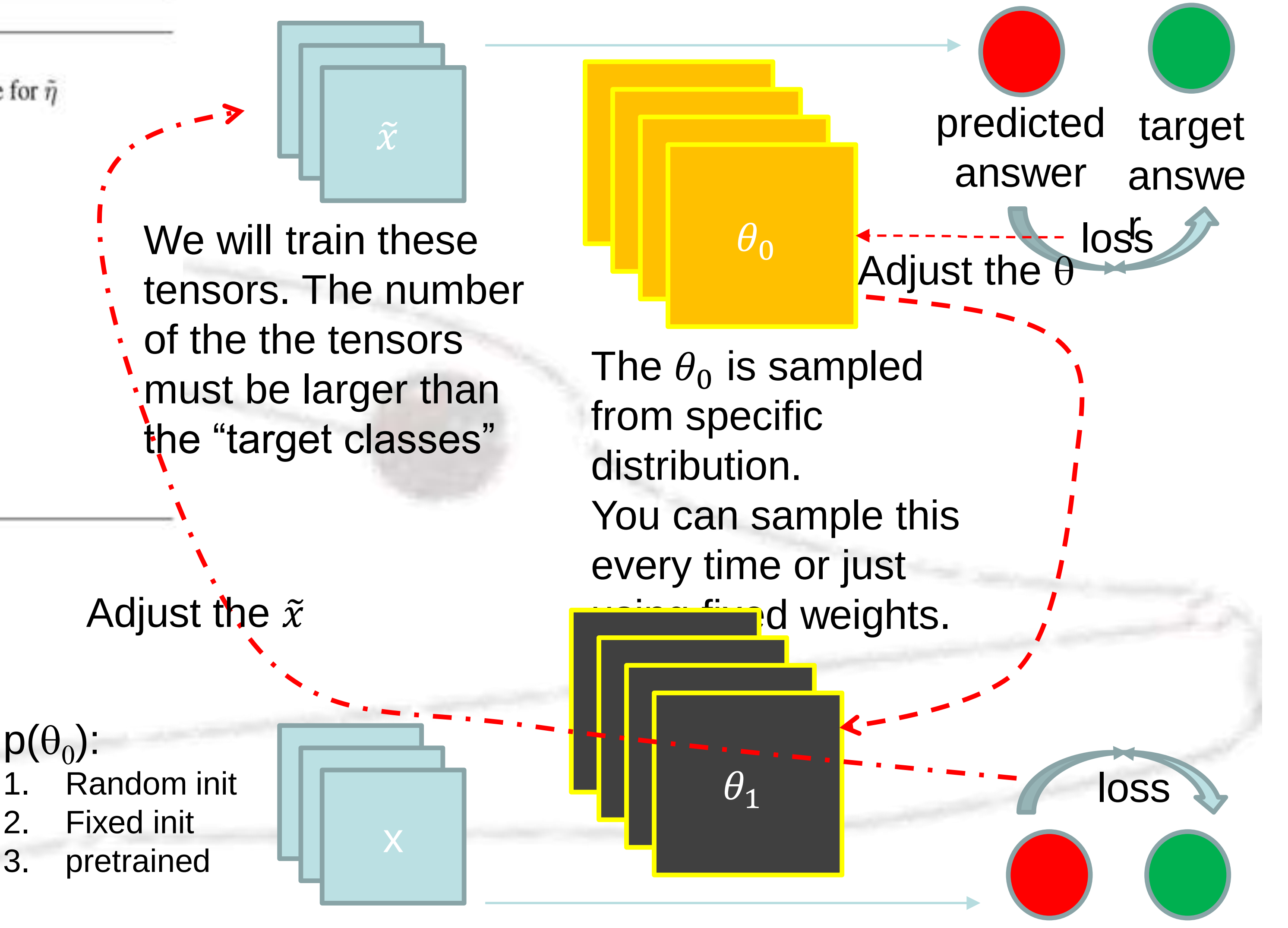
# Data Distillation – MIT & FAIR & UC Berkley

**Algorithm 1** Dataset Distillation

**Input:** $p(\theta_0)$: distribution of initial weights; $M$: the number of distilled data
**Input:** $\alpha$: 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 $\tilde{\eta}$

x = {d, t}
d => features
T => label

Learning rate is learnable

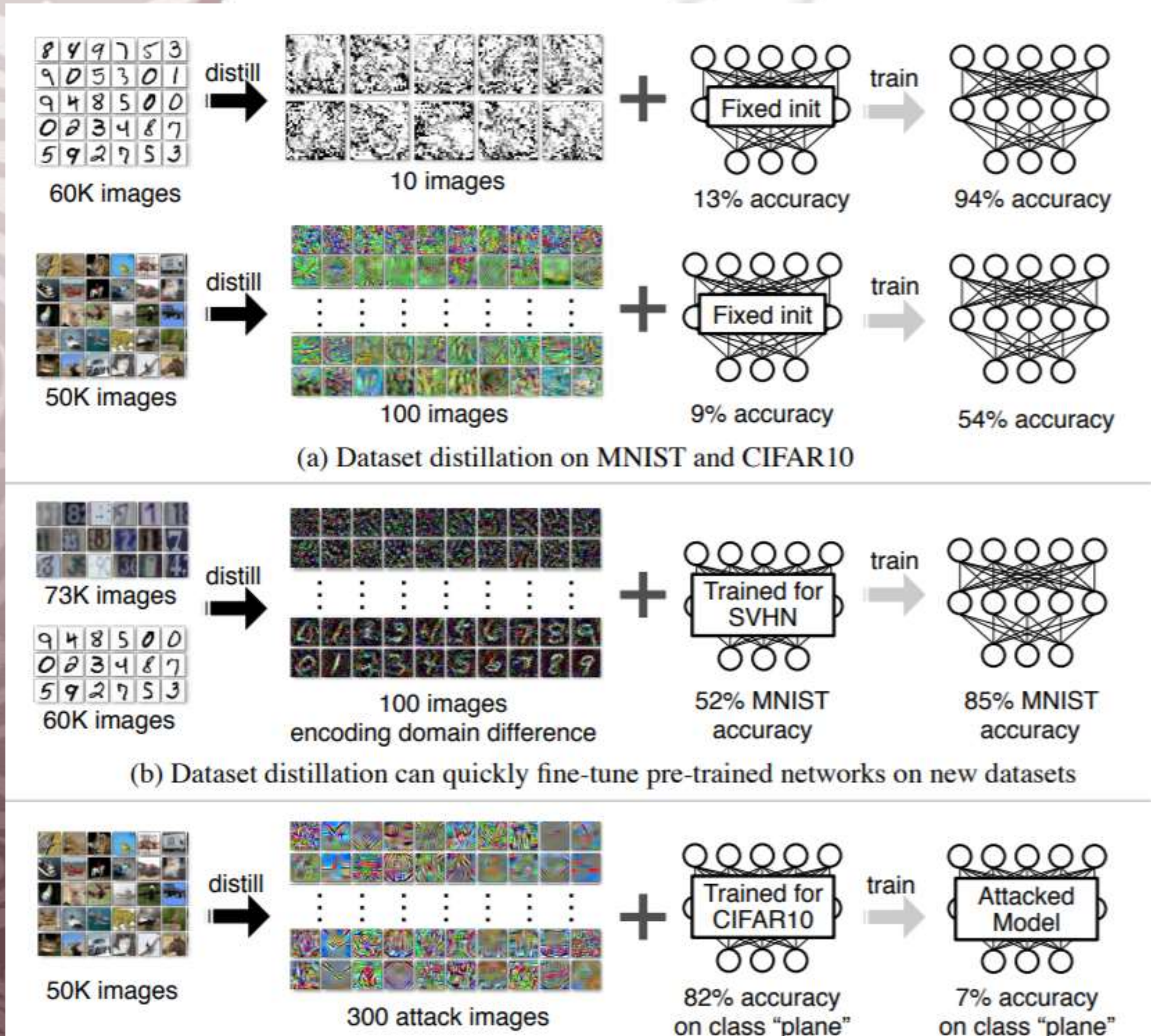



OpenReview.net Search ICLR 2019 Conference

**Original idea, but lacks a practical use case.**
*ICLR 2019 Conference Paper593 Area Chair1*
14 Dec 2018 (modified: 21 Dec 2018) ICLR 2019 Conference Paper593 Meta Review Readers: Everyone
**Metareview:** The reviewers agree that the idea for dataset distillation is novel, however it is unclear how practica
significantly improved through the addition of new baselines, however ultimately the performance is not quite go
advocate strongly on its behalf. Perhaps the paper would be better motivated by finding a realistic scenario in wh
to use this approach over reasonable alternatives.
**Confidence:** 3: The area chair is somewhat confident
**Recommendation:** Reject

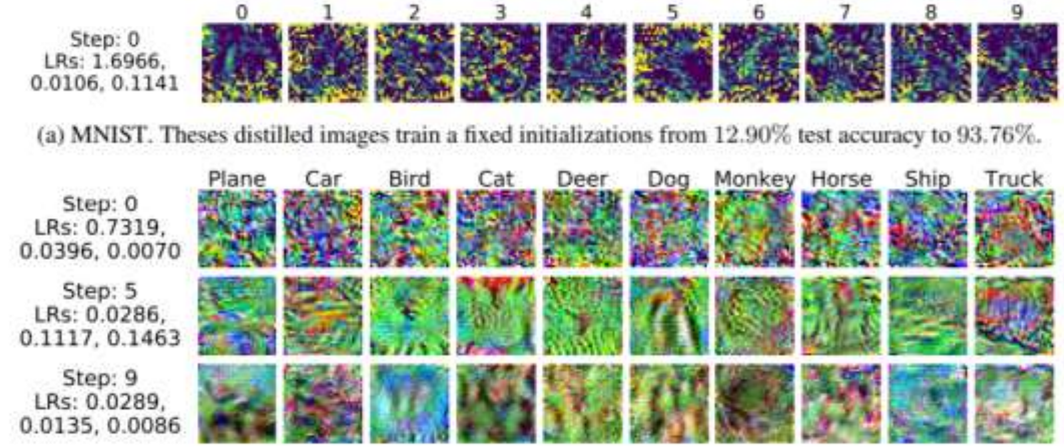
https://openreview.net/forum?id=Sy4lojC9tm

# Data distillation - applications

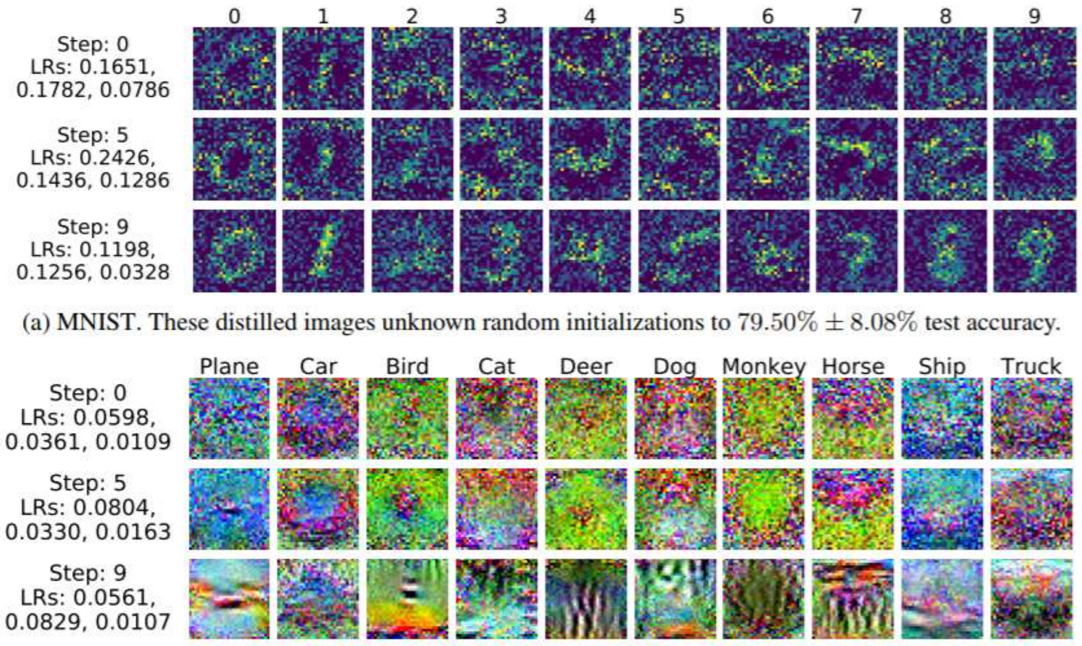



Poisoning images

Fixed initialization




Random initialization



# The Dataset module of Tensorflow

# The environment setting

- Tensorflow
  - Version : 1.13.0
  - CUDA 10
  - Python 3.6

- Why Dataset module
  - The dataset manipulating often cost 80% of the source code.
  - Some skills is regular, such as the “training-validation-test” datasets creating, dataset mapping, etc.
  - These routing operations can be merge into some module

# The basic concept of the Dataset module

- Class
  - Dataset
    - The basic container for storage data for further usage
  - Iterator
    - Access the data
    - Functions
      - make_make_one_shot_iterator: the elements will be used only once; needn't initialization.
      - make_initializable_iterator : the dataset can be reused by setting new parameters
  - Options
    - Providing the information of tf.data.Dataset
  - FixedLengthRecordDataset
    - Mainly designing for binary filesTFRecordDataset
    - Handling the data with TFRecorder
  - TextLineDataset
    - Handling the text data

# Dataset architecture

dataset

Element 1
Element 2
Element 3
……..
Element n

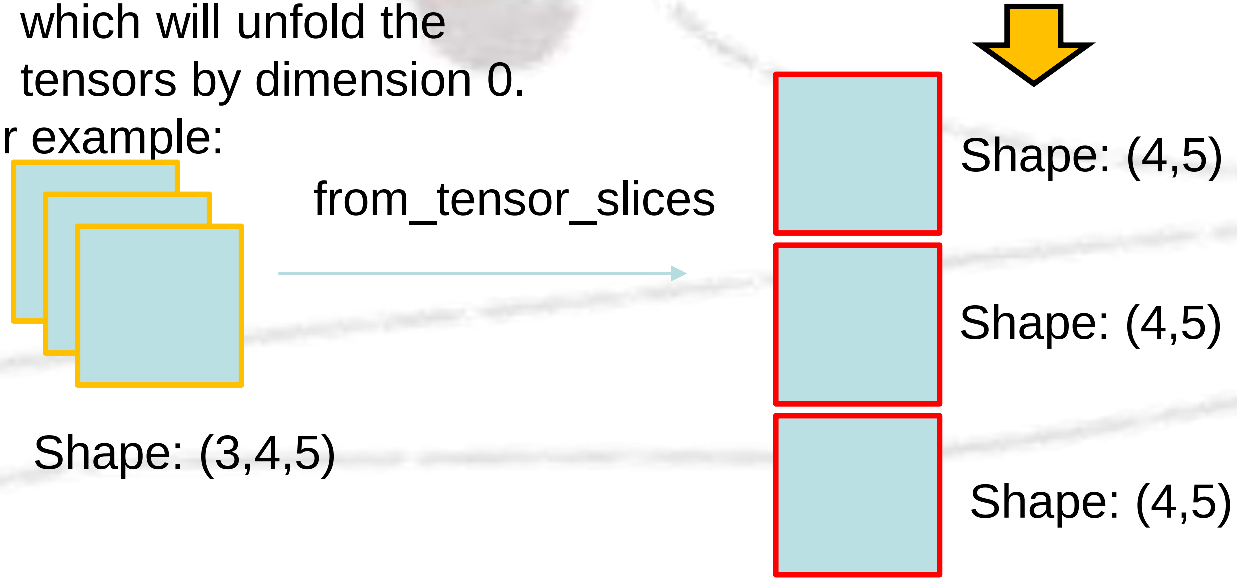
Each element has the same structure, like:
(Img 1, label 1)
(img 2, label 2)
……
(img n, label n)

The Dataset module use pieces of whole dataset

1. We need to cut the whole data into small pieces.
2. tf.data.Dataset.from_tensor_slices help use to complete this mission which will unfold the tensors by dimension 0.

You can do anything, like creating data batch or mapping the piecies into some functions. Because this is the smallest unit.

for example:




https://www.tensorflow.org/guide/datasets
https://www.tensorflow.org/api_docs/python/tf/data/Dataset

# Dataset 幾個基本操作

```python
import numpy as np
import tensorflow as tf

data_s = np.reshape([i for i in range(3 * 4 * 5)], [3,4,5])
# print(data_s)

dataset = tf.data.Dataset.from_tensor_slices(data_s)
# print(dataset)

dataset_iter = dataset.make_initializable_iterator()
dataset_go = dataset_iter.get_next()

with tf.Session() as sess:
    # they need to be initialized
    sess.run(dataset_iter.initializer)
    print(sess.run(dataset_go))
    print(sess.run(dataset_go))

    # they can be initialized again. All stuff will restart again
    sess.run(dataset_iter.initializer)
    print(sess.run(dataset_go))
    print(sess.run(dataset_go))

```

定義data source。先製作一個shape為[3,4,5]的tensor。

切成slice放到dataset物件中。

初始化Dataset的iterator，並定義取用element的操作子。

使用以前記得初始化Dataset一下。

# 實用的進階操作- feedable dataset

```python
import numpy as np
import tensorflow as tf

data_s = tf.placeholder(tf.float32, [None,4,5])
dataset = tf.data.Dataset.from_tensor_slices(data_s)
dataset = dataset.batch(2) # This can set the batch size. "Ba
print(dataset)

dataset_iter = dataset.make_initializable_iterator()
dataset_go = dataset_iter.get_next()
print(dataset_go)


s = np.reshape([i for i in range(10 * 4 * 5)], [10,4,5])
with tf.Session() as sess:
    sess.run(dataset_iter.initializer, feed_dict={data_s:s})
    print(sess.run(dataset_go))
```

可以一次就把所有資料餵進去，再用dataset來切資料

使用方法都相同，也可以直接設定batch size。這樣可以一次fetch不只一個elements。

initial的時候就要提供餵了甚麼資料進去，這個operator才能啟動。

# The other practical operators

- map()
  - Transforming the input tensors to other tensor by specific function (usually use lambda simply)
- repeat()
  - Since the iterator will stop at the end, if we want to train for many epochs:

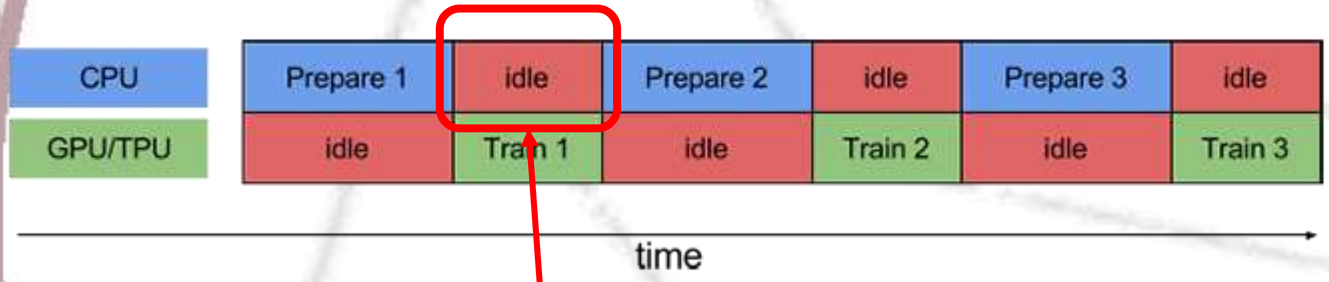
    dataset = dataset.repeat(10)  #repeat dataset 10 times, you can train this for 10 epochs
    dataset = dataset.repeat() #repeat infinity times. this would save the work of re-initialize dataset.
- shuffle()
  - randomly shuffle dataset would be needed for each epoch, so

    dataset = dataset.shuffle(buffer_size=100) # large buffering size makes more random
- tf.contrib.data.shuffle_and_repeat
  - repeat() will give infinity accessing right. But shuffle_and_repeat() can give the shuffle function before the next repeating.

    dataset = dataset.apply(tf.contrib.data.shuffle_and_repeat(buffer_size=100))
- batch()
  - setting the element fetch numbers
    - dataset = dataset.batch(5, True) # fetch 5 elements per time. abandon the last batch
      - 因為最後一個Batch常常都是未滿batch size的數量，例如上面的例子就是有可能會少於5個。如果不捨棄就用False(default)

# Dataset Prefetch

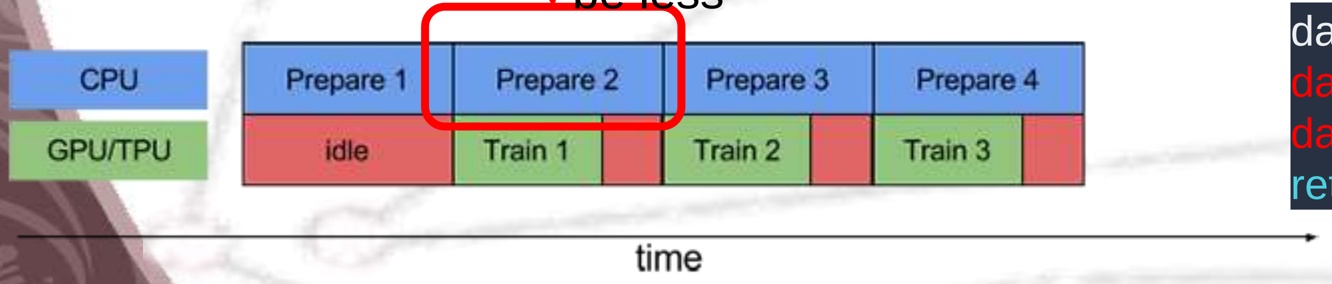
- The issue to the original dataset module is the computing resources wasting




How to

```
dataset = dataset.batch(batch_size=FLAGS.batch_size)
return dataset
```

if we asynchronized the thread, the idle time of CPU/GPU would be less




```
dataset = dataset.batch(batch_size=FLAGS.batch_size)
dataset = dataset.prefetch(buffer_size=FLAGS.prefetch_buffer_size)
return dataset
```

https://www.tensorflow.org/guide/performance/datasets