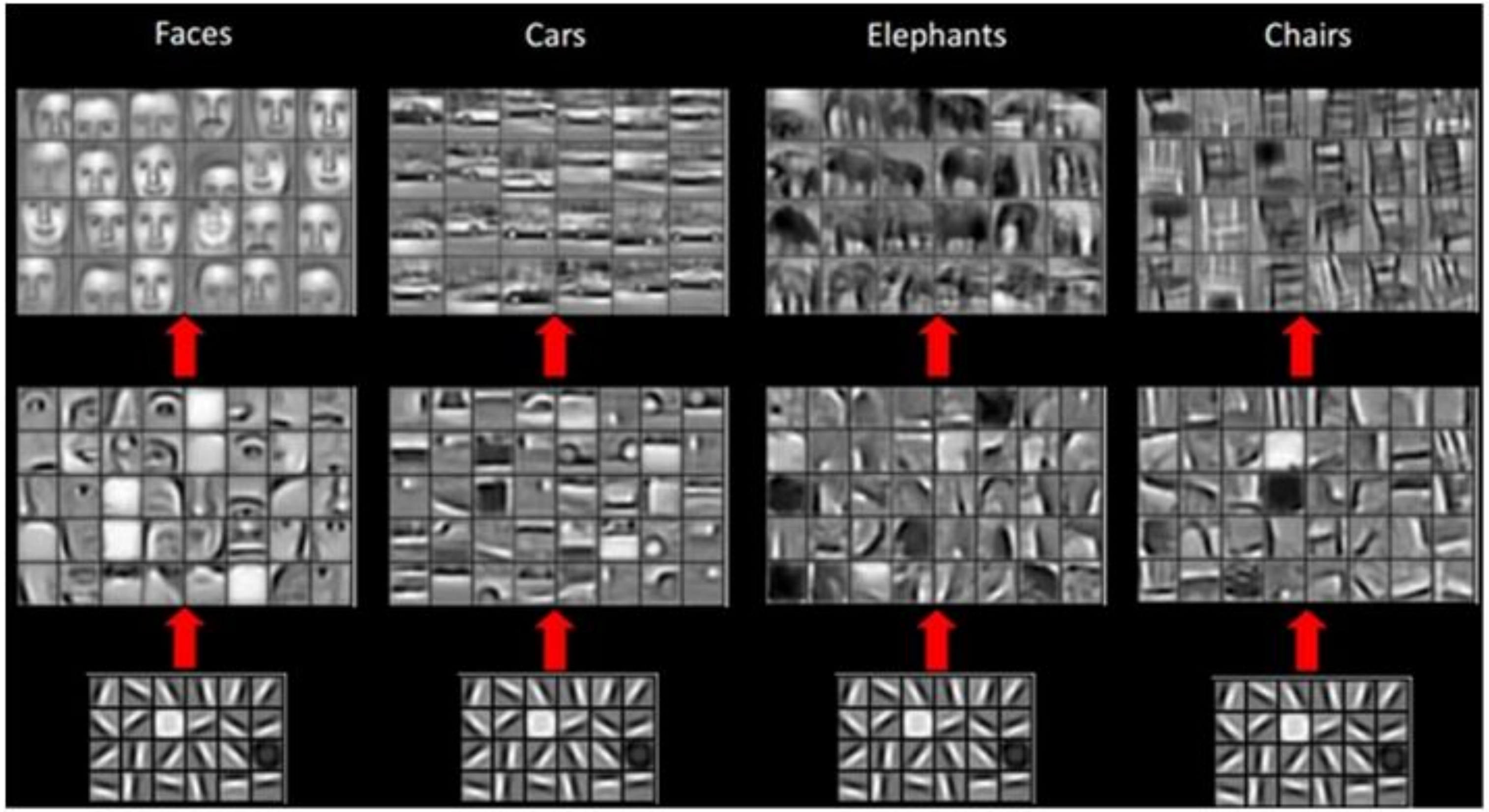
Tensor Decomposition via Core Tensor Networks

Jianfu Zhang, Zerui Tao, Qibin Zhao, and Liqing Zhang
RIKEN AIP, Shanghai Jiao Tong University, Lanzhou University

Assumptions

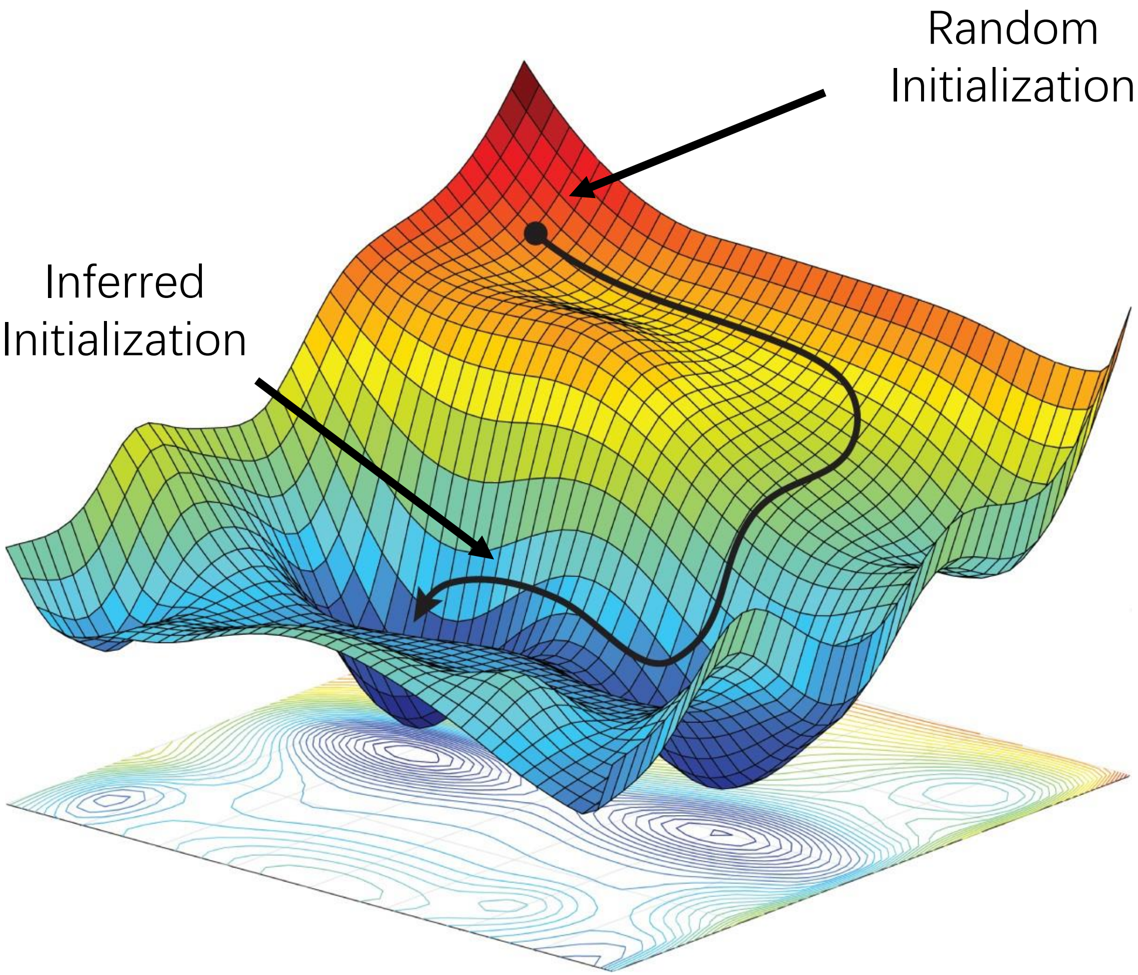
- The mappings of a bunch of tensors might be shared or highly correlated.
- Leverage the correlated information might be helpful for convergence



Assumptions

- Train DNN models with auxiliary samples

- With pre-trained models, infer the initialization to fast and accurate tensor decomposition



Gradient Descent

- Tensor completion:
- Tensor denoising:
- Tensor decomposition:
- Objective function:
- Gradient descent:

  $\mathcal{G}_i^{(k)} \leftarrow \mathcal{G}_i^{(k)} - \lambda \nabla_{\mathcal{G}_i^{(k)}} L$
- Convergence:

  $|L_t - L_{t-1}| < thd$

$$t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} \times w_{i_1,\ldots,i_N}$$

$$t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} + e_{i_1,\ldots,i_N}$$

$$\mathcal{T} \approx \mathcal{X} = \ll \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)} \gg$$

$$L = \frac{1}{M} \sum_{i=1}^{M} \|\mathcal{W}_i * (\mathcal{T}_i - \mathcal{X}_i)\|_F^2$$

Algorithm 1 Gradient Descent for Tensor Decomposition.

Require: Input data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.

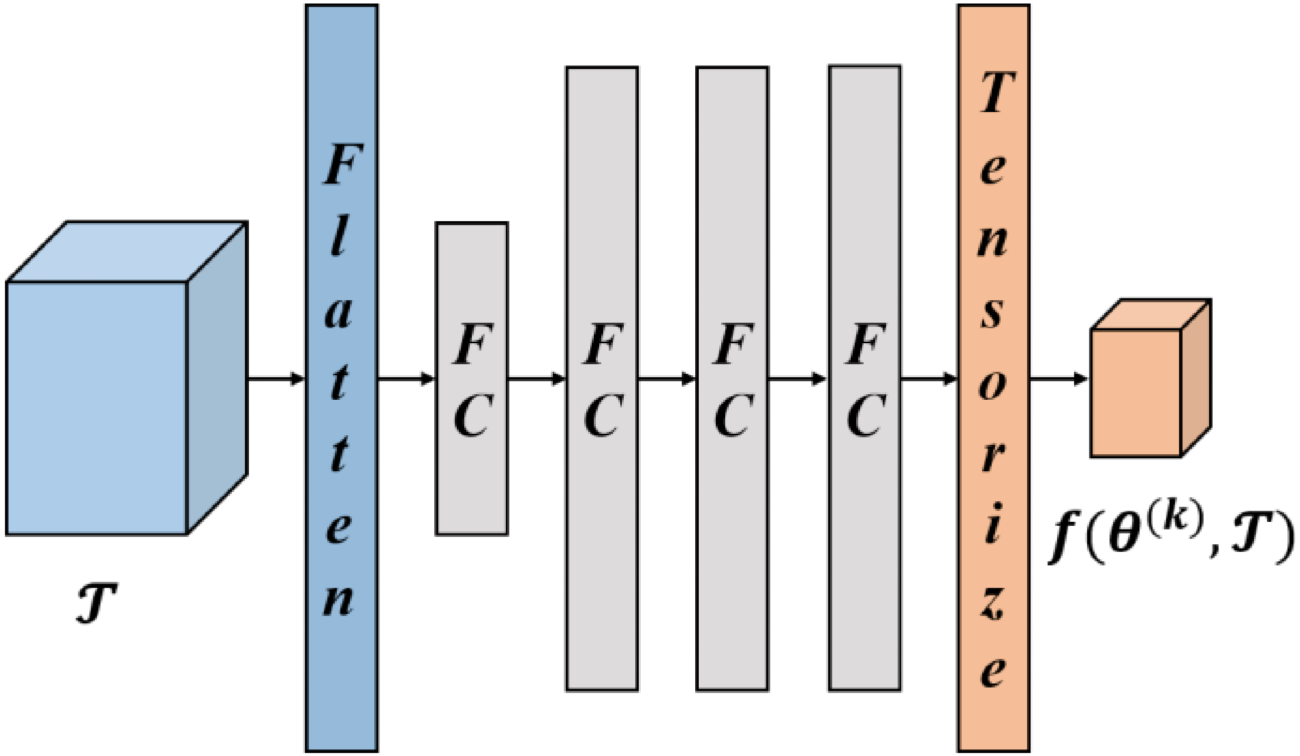
Ensure: Model parameters $\{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M$.

```
Randomly initialize {G_i^(1), ..., G_i^(N)}_{i=1}^M.
while Not converged do
    Calculate loss function L based on Eq. 5.
    Update {G_i^(1), ..., G_i^(N)}_{i=1}^M based on Eq. 6.
end while
```

Core Tensor Network

- $f(\theta^{(k)}, \mathcal{T})$: (main core tensor) The function representation for the network to learn the k-th core tensor with network parameter θ
- Learn the function with multi-layer perceptron
- Combine bias core tensor:

$$\mathcal{G}_i^{(k)} = f(\theta^{(k)}, \mathcal{T}_i) + \mathcal{B}_i^{(k)}$$



Gradient Descent for Core Tensor Networks

- Initialize the core tensors with random projections of the input tensors

- Main core tensors share the same model parameter for all the input tensors

$$\theta^{(k)} \leftarrow \theta^{(k)} - \lambda \nabla_{\theta^{(k)}} L, \mathcal{B}_i^{(k)} \leftarrow \mathcal{B}_i^{(k)} - \lambda \nabla_{\mathcal{B}_i^{(k)}} L.$$

Algorithm 2 Gradient Descent for Core Tensor Networks.

Require: Input data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.

Ensure: Model parameters $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.

Ensure: Model parameters $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$.

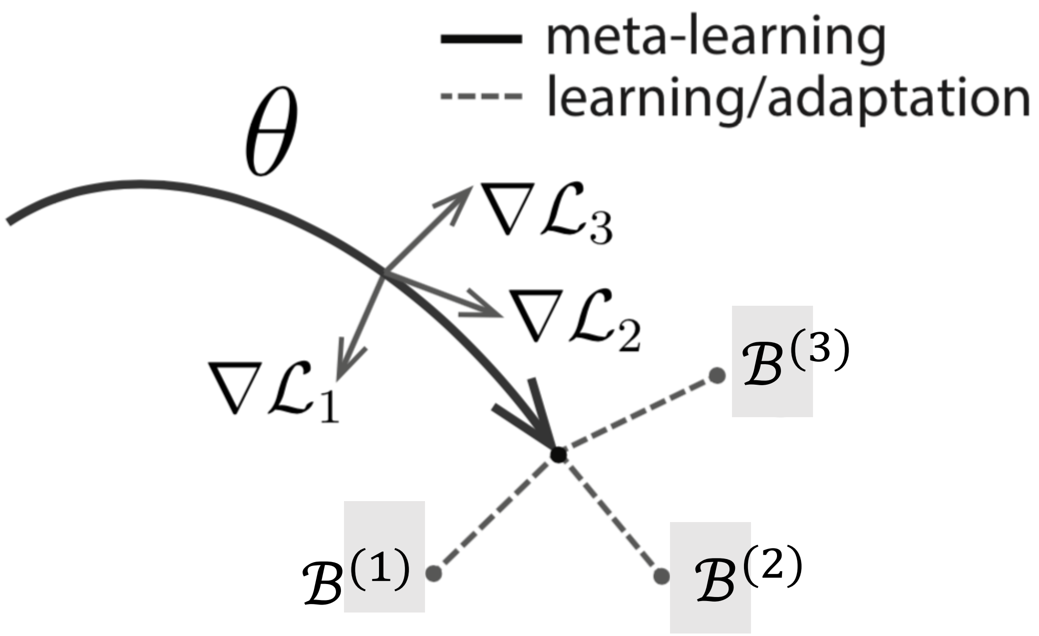
Randomly initialize $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$ with zeros.
while Not converged **do**
    Calculate loss function L based on Eq. 5.
    Update $\theta^{(k)}$, $\{\mathcal{B}_i^{(k)}\}_{i=1}^{M}$ for all k based on Eq. 6.
end while

Core Tensor Network with Meta-Learning

Algorithm 3 Transfer Learning for Core Tensor Networks.

Require: Training data $\{\mathcal{T}'_1, \ldots, \mathcal{T}'_{M'}\}$.
Require: Test data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.
Ensure: Model parameters $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
Ensure: Model parameters $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$.
Randomly initialize $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
for $iter$ in $1, \ldots, iter_{max}$ **do**
  Sample a batch $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ from training set.
  Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$ with zeros.
  for p in $1, \ldots, \gamma$ **do**
    Calculate L for $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ based on Eq. 5.
    Update $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$ based on Eq. 6.
  end for
  Calculate L for $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ based on Eq. 5.
  Update $\theta^{(k)}$ for all $k \in [N]$ based on Eq. 6.
end for
Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^{M}$ with zeros.
while Not converged **do**
  Calculate L for $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ based on Eq. 5.
  Update $\theta^{(k)}$, $\{\mathcal{B}_i^{(k)}\}_{i=1}^{M}$ for all k based on Eq. 6.
end while

- Pre-train the main core tensors and finetune the model to the test set



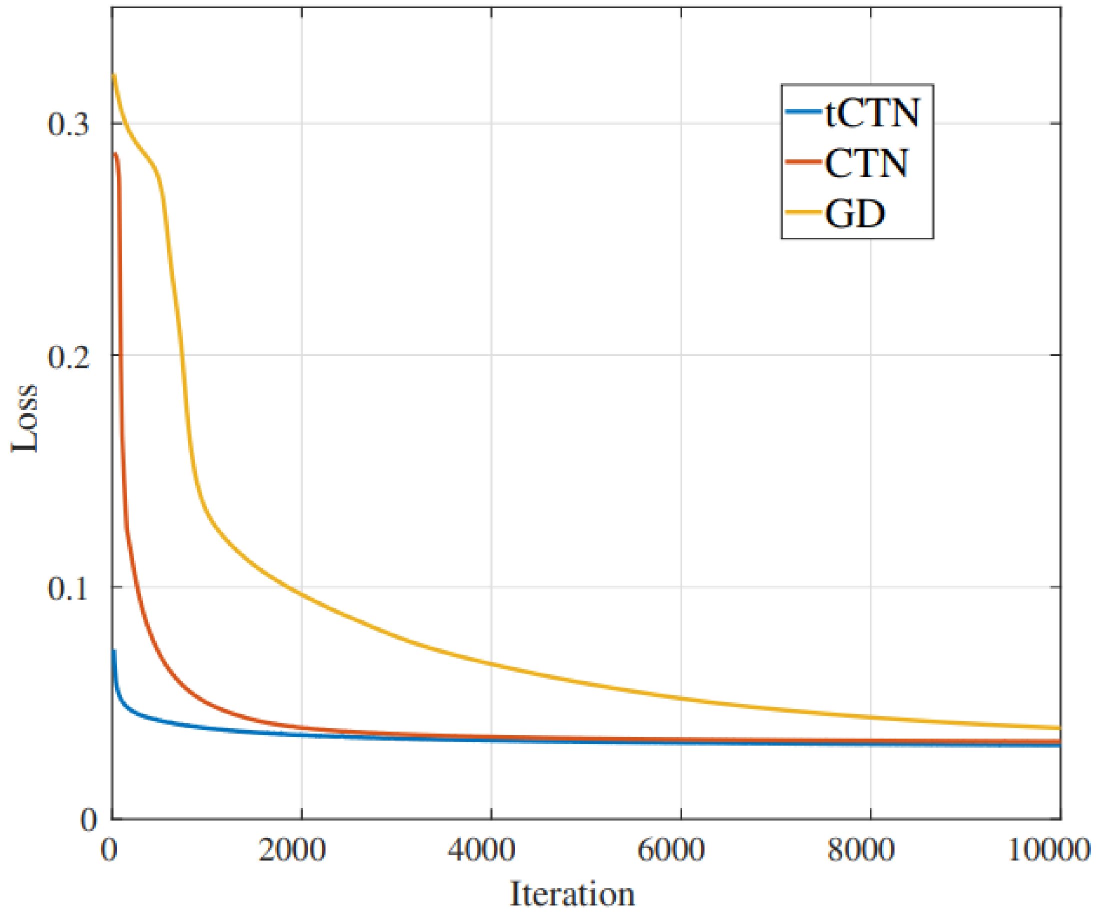
Experiments

- CTN converges much faster than GD

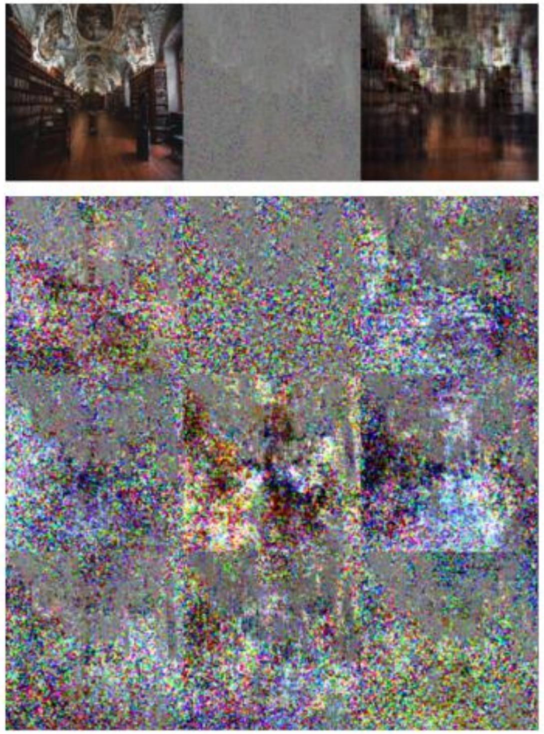
Metrics/Algorithms	GD	ALS	CTN	tCTN
RSE↓	0.1243	0.1229	0.1205	0.1201
Second per Image↓	22.2	109.3	6.55	0.57

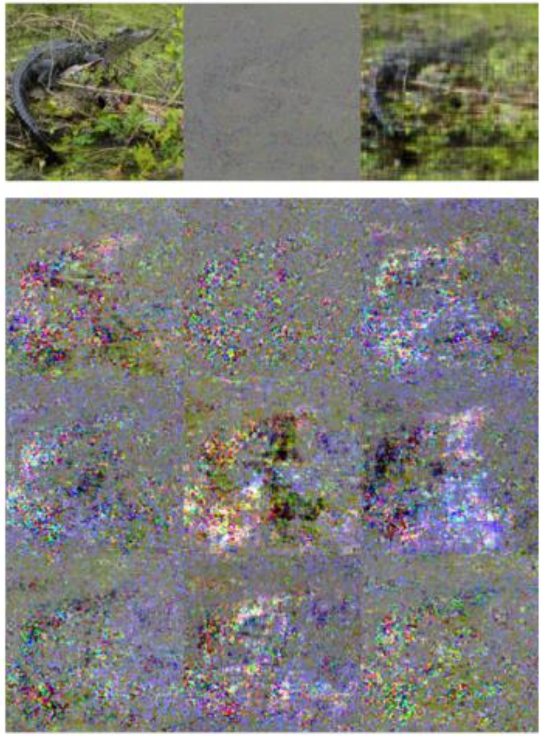
- Higher γ costs more time to train, $\gamma = 20$ performs best

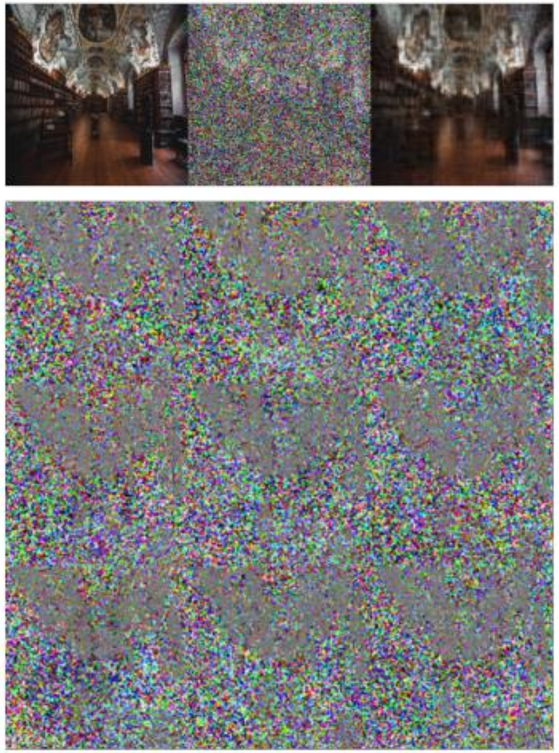
Metrics/Settings	0	5	10	20	50
RSE↓	0.1284	0.1246	0.1212	0.1201	0.1211
Batch per Second↑	26.53	19.53	13.44	9.93	4.68

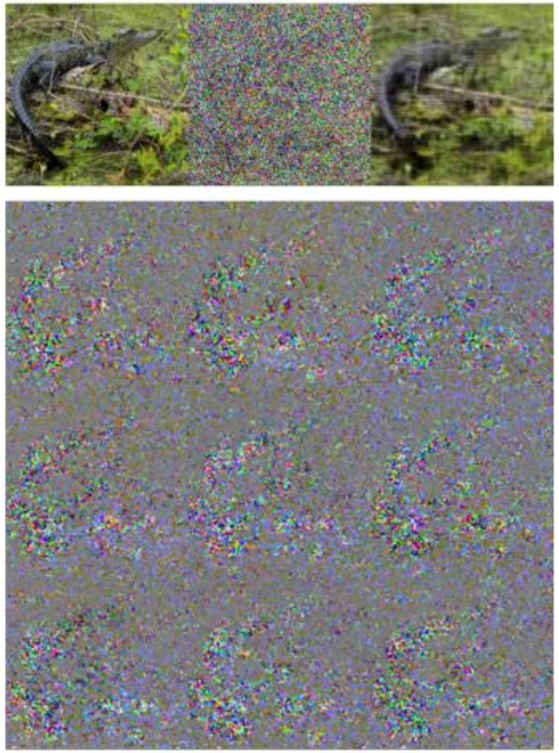


Experiments









Rate	Metric	BCPF	TT-WOPT	TR-ALS	TRLRF	CTN	tCTN
0.9	PSNR↑	19.69	23.21	22.22	22.27	23.50	23.53
	RSE↓	0.1868	0.1262	0.1396	0.1388	0.1205	0.1201
0.7	PSNR↑	25.18	25.36	24.51	26.82	27.78	27.79
	RSE↓	0.0993	0.0972	0.1072	0.0822	0.0736	0.0735

Level	Metrics	TT-WOPT	TR-ALS	CTN	tCTN
10dB	PSNR↑	19.24	19.69	20.17	20.33
	RSE↓	0.0849	0.0774	0.0686	0.0682
20dB	PSNR↑	19.48	20.03	20.23	20.34
	RSE↓	0.0804	0.0723	0.0682	0.0675

Future Works

- Replace MLP with CNN, which can go deeper and preserve local structures of the input tensors.
- Analyze the patterns of the core tensors.
- Theoretical analyses of the core tensor network.
- **Thanks for listening! E-mail: jianfu.zhang@riken.jp**