Efficient Deep Space Filling Curve

Wanli Chen, Xufeng Yao, Xinyun Zhang, Bei Yu

Chinese University of Hong Kong

ICCV23 PARIS

Space Filling Curve

Space filling curve is defined as a single curve that can fill the entire space without any intersection. They have many applications in computer vision, such as:

- Data linearization
- Feature compression
- Cross-modality feature transformation

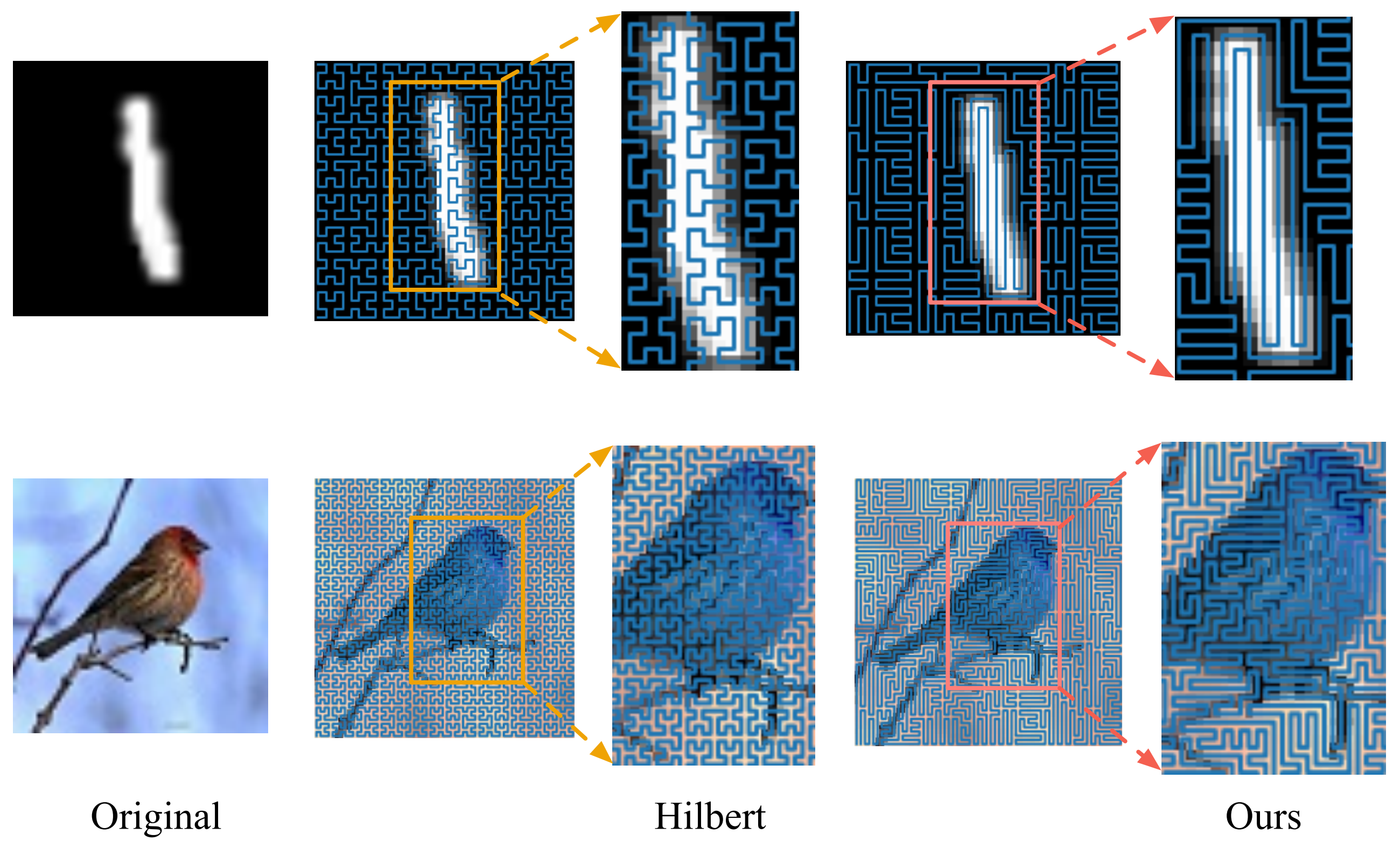


Figure 1. Comparison between our method and Hilbert curve. It can be found that the SFC generated by our model is determined by the image context while Hilbert curve has a fixed structure.

Adaptive Space Filling Curve

Aadaptive SFCs are more suitable for data linearization since they are generated according to the data distributionsThese data-adaptive SFCs are formed through the Cover-and-Merge algorithm. The generation of adalptive SFCs is shown as follow:

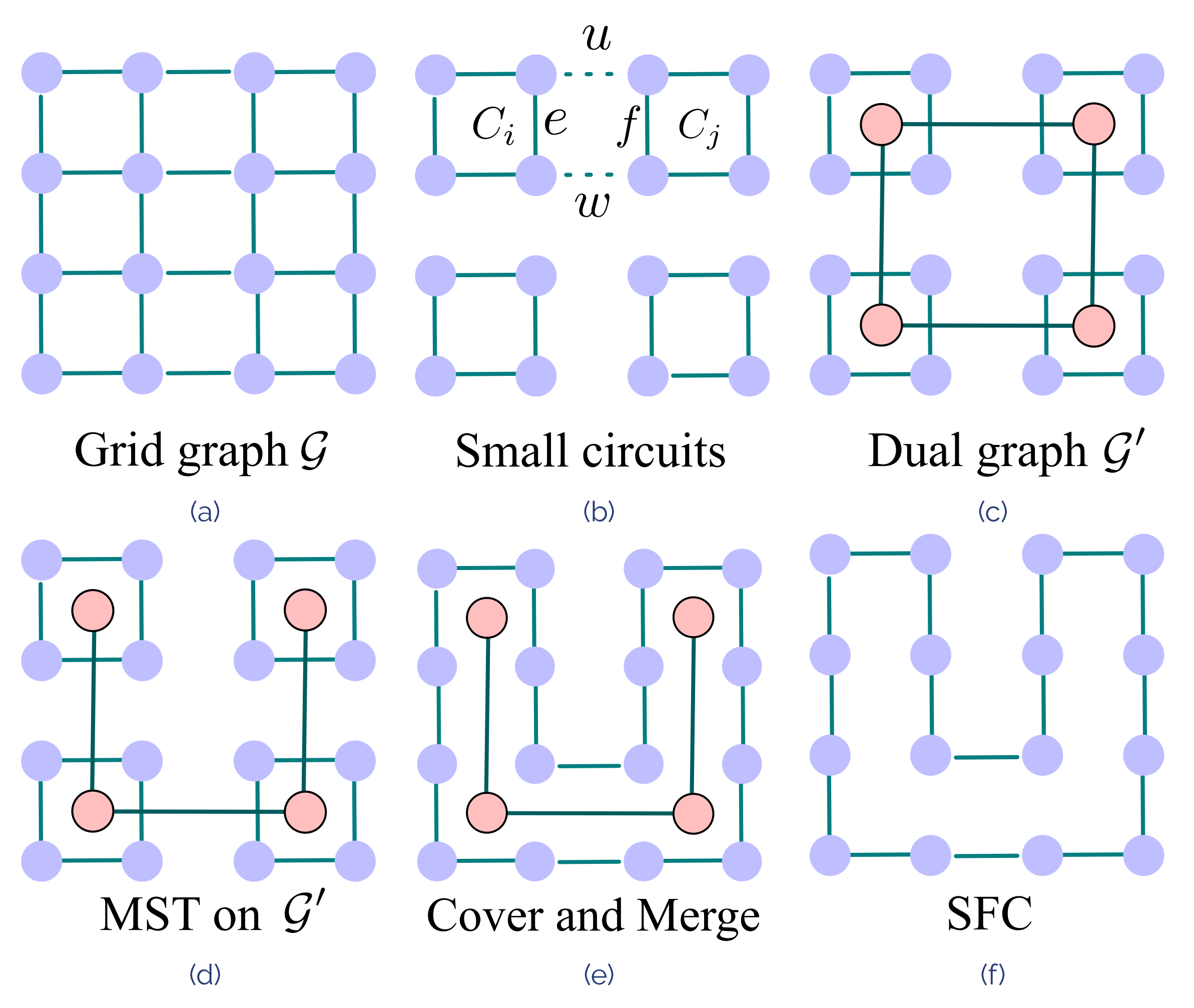
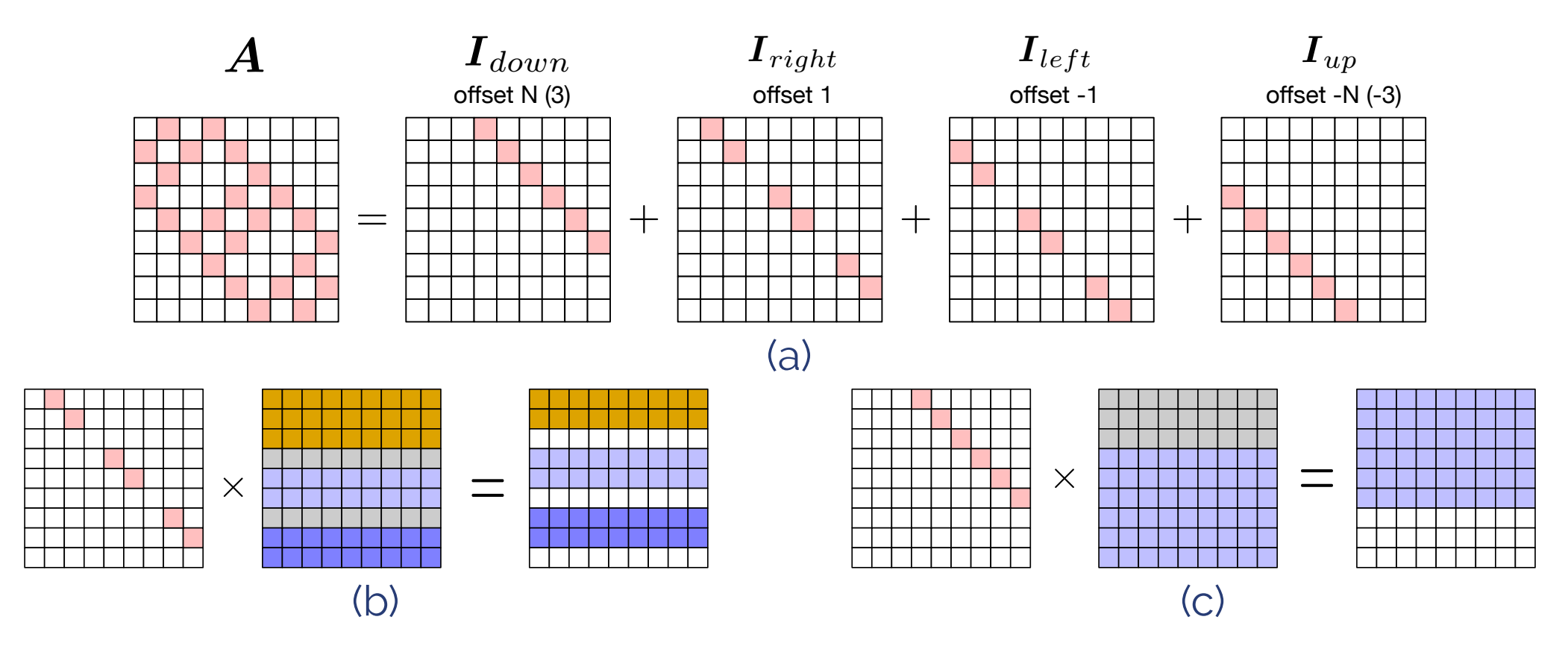


Figure 2. The process of generating adaptive SFCs.

In the cover step, we create a grid graph $\mathcal{G}$, generate small circuits, and construct a dual graph $\mathcal{G}'$. After setting $\mathcal{G}'$'s edge weights, we proceed to the merge step. We find the MST τ in $\mathcal{G}'$, link the circuits in $\mathcal{G}$ according to τ, and obtain a new SFC. The Cover-and-Merge algorithm turns SFC generation into finding a Hamiltonian path in $\mathcal{G}$.

EGCN: Fast Curve Generation

Now we look at the adjacency matrix of grid graph. Obviously, the computation of GCN in grid graph can be decomposed into the summation of 3 diagonal matrix multiplication.



We use the three diagonals instead of the entire adjacency matrix for GCN computation. Our EGCN shows good efficiency compared with other GCN methods.

Multi-Stage MST: Robust Curve Generation

We generate different Multi-Stage Minimum Spanning Trees (MSTs) on the different output stage of ResNet backbone.

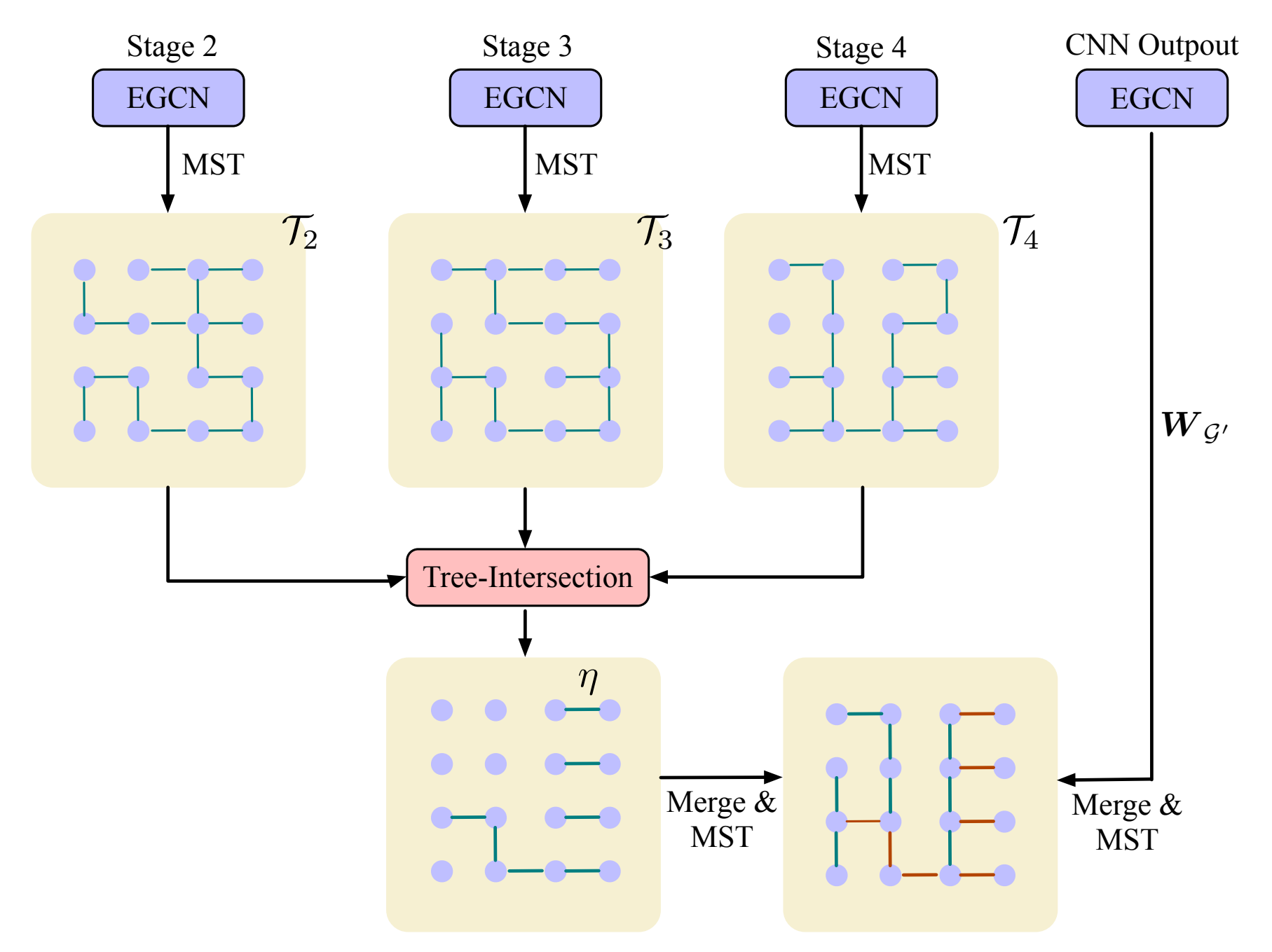


Figure 3. Multi-stage MST.

This design makes the final output more stable.

Siamese Network Training: Faster Convergence

We design a siamese network training process. Specifically, we train two neural networks together. The upper branch uses the original input while the lower branch uses the input with Gaussian noise.

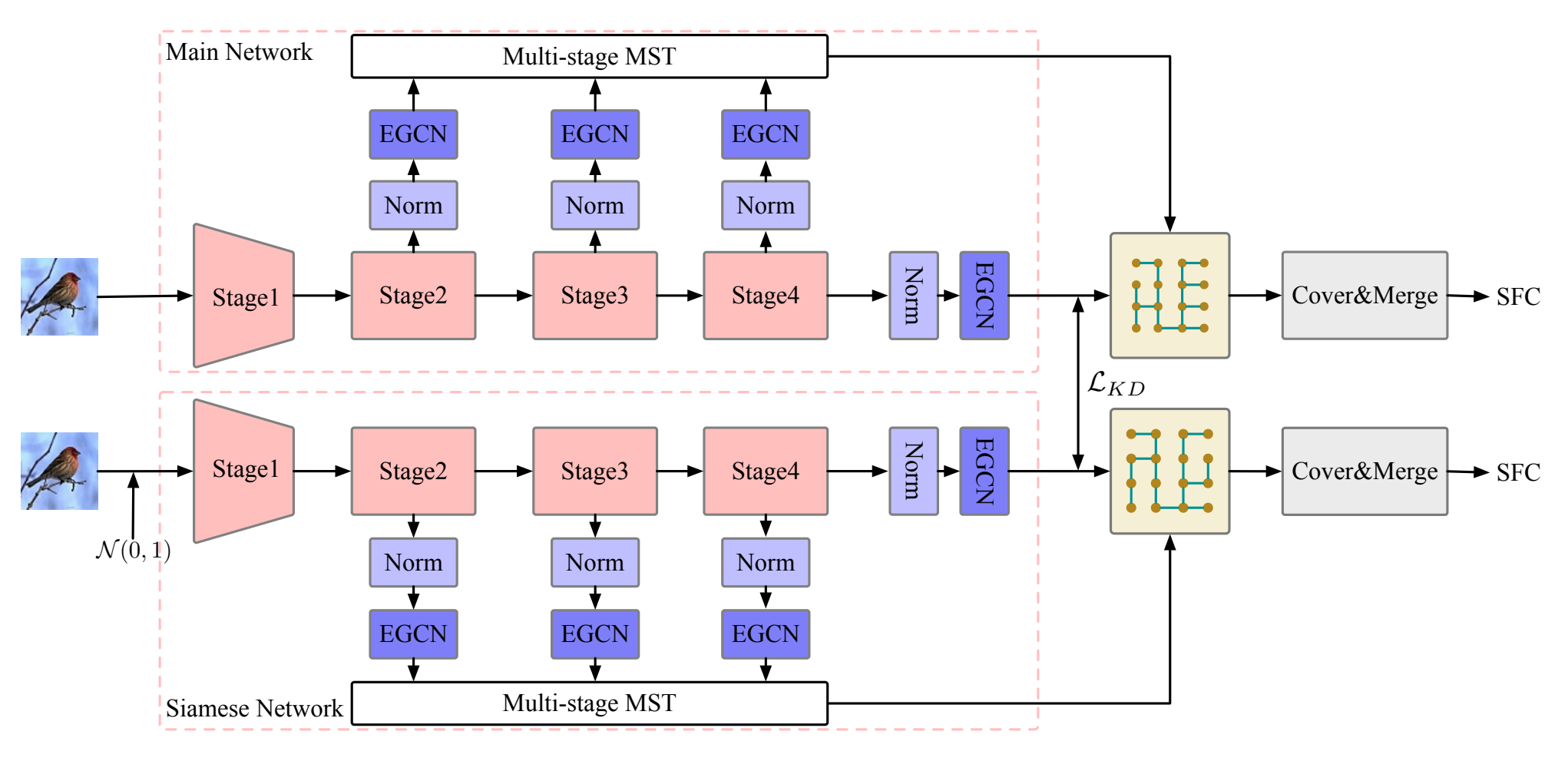


Figure 4. The main framework of our model.

We minimize the KL-divergence between two networks.

Visualization

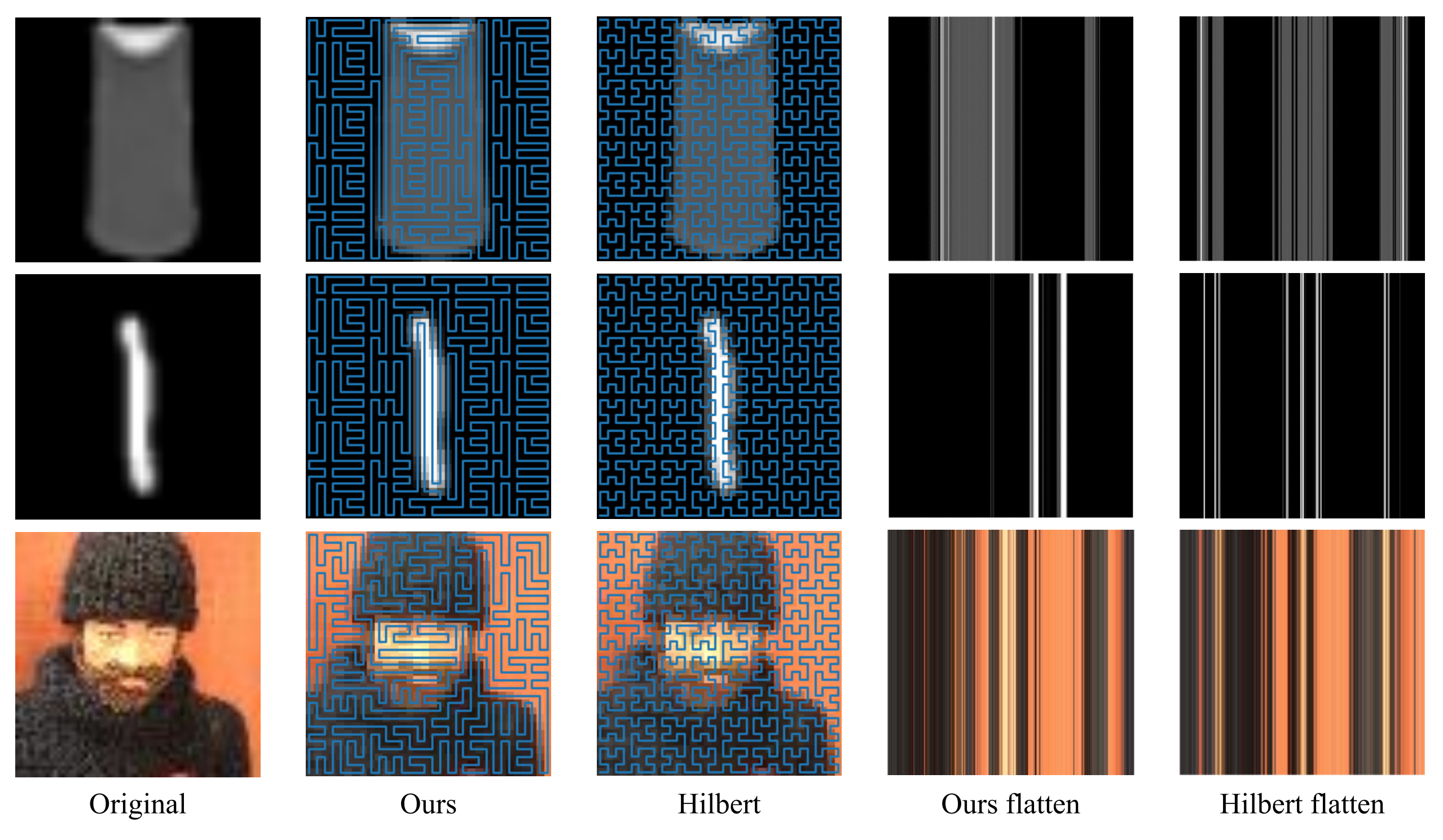


Figure 5. Visualized results of our proposed methods.

Experimental Results

Dataset	Method	Autocorrelation	LZW Code Length (bytes)
MNIST	Zigzag	0.207	175.4
	Hilbert	0.475	182.7 (+7.3)
	Dafner	0.401	-
	NSFC	0.558	171.1 (-4.3)
	Ours	**0.625**	**158.3 (-17.1)**
Fashion-MNIST	Zigzag	0.552	425.8
	Hilbert	0.723	427.3(+1.5)
	Dafner	0.704	-
	NSFC	0.786	412.4 (-13.4)
	Ours	**0.834**	**400.7 (-25.1)**
Tiny-imagenet (32×32)	Zigzag	0.811	925.1
	Hilbert	0.874	927.6 (+2.5)
	Dafner	0.896	909.0 (-16.1)
	NSFC	0.913	904.9 (-20.2)
	Ours	**0.936**	**888.7 (-36.4)**
Tiny-imagenet (64×64)	Zigzag	0.719	-
	Hilbert	0.773	-
	Dafner	0.779	-
	NSFC	-	-
	Ours	**0.826**	-

Table 1. Comparison between different methods.

Performance of EGCN

Method	GPU	Params	Inference time
GCN	104M	1.6M	11ms
GAT	120M	1.8M	12ms
SGC	88M	0.8M	9ms
FastGCN	96M	0.5M	8ms
EGCN	**32M**	**0.3M**	**4ms**

Table 2. Comparison between different GCN methods. All methods are tested by using grid graphs with size 64×64 and batch size 512.

Training Time Comparison

Learning Scheme	AC	Training Time/epoch	Params
NSFC	0.593	1867s	22.9M
Ours	**0.625**	**72s**	22.3M

Table 3. Comparison between our scheme and NSFC.