Paper Review

SQUID: Deep Feature In-Painting for Unsupervised Anomaly Detection

YeongHyeon Park

Department of Electrical and Computer Engineering

SungKyunKwan University

JUNE 18-22, 2023
CVPR VANCOUVER, CANADA

THE UNIVERSITY OF SYDNEY
JOHNS HOPKINS UNIVERSITY

SQUID: Deep Feature In-Painting for Unsupervised Anomaly Detection

Tiange Xiang[1] Yixiao Zhang[2] Yongyi Lu[2] Alan L. Yuille[2]
Chaoyi Zhang[1] Weidong Cai[1] Zongwei Zhou[2,*]

[1]University of Sydney [2]Johns Hopkins University

GitHub: https://github.com/tiangexiang/SQUID

University of Sydney


Johns Hopkins University

About Author

Tiange Xiang

Other names ▸

FOLLOW

Stanford University
Verified email at stanford.edu - Homepage
computer vision deep learning medical imaging graphics

TITLE	CITED BY	YEAR
SQUID: Deep Feature In-Painting for Unsupervised Anomaly Detection T Xiang, Y Zhang, Y Lu, AL Yuille, C Zhang, W Cai, Z Zhou Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern …	6 *	2023
Seeing Beyond the Brain: Conditional Diffusion Model with Sparse Masked Modeling for Vision Decoding Z Chen, J Qing, T Xiang, WL Yue, JH Zhou CVPR 2023	9	2022
Towards bi-directional skip connections in encoder-decoder architectures and beyond T Xiang, C Zhang, X Wang, Y Song, D Liu, H Huang, W Cai Medical Image Analysis 78, 102420	2	2022
3d medical point transformer: Introducing convolution to attention networks for medical point cloud analysis J Yu, C Zhang, H Wang, D Zhang, Y Song, T Xiang, D Liu, W Cai arXiv preprint arXiv:2112.04863	21	2021
DSNet: A Dual-Stream Framework for Weakly-Supervised Gigapixel Pathology Image Analysis T Xiang, Y Song, C Zhang, D Liu, M Chen, F Zhang, H Huang, ... IEEE TMI 2022	8	2021
BiX-NAS: Searching Efficient Bi-directional Architecture for Medical Image Segmentation X Wang*, T Xiang*, C Zhang, Y Song, D Liu, H Huang, W Cai MICCAI 2021	20	2021
Partial graph reasoning for neural network regularization T Xiang, C Zhang, Y Song, S Liu, H Yuan, W Cai arXiv preprint arXiv:2106.01805	3	2021
Walk in the Cloud: Learning Curves for Point Clouds Shape Analysis T Xiang, C Zhang, Y Song, J Yu, W Cai ICCV 2021	154	2021
Two-Stage Monte Carlo Denoising with Adaptive Sampling and Kernel Pool T Xiang, H Yuan, H Huang, Y Shi arXiv preprint arXiv:2103.16115	1	2021
BiO-Net: Learning Recurrent Bi-directional Connections for Encoder-Decoder Architecture T Xiang, C Zhang, D Liu, Y Song, H Huang, W Cai MICCAI 2020	49	2020
DDM ^{2}: Self-Supervised Diffusion MRI Denoising with Generative Diffusion Models T Xiang, M Yurt, AB Syed, K Setsompop, A Chaudhari ICLR 2023	5 *	

About Author

Tiange Xiang 向天戈

I am a first-year Ph.D. student in Stanford Vision and Learning Lab (SVL) at the Stanford University. I'm currently rotating with Prof. Fei-Fei Li and Prof. Jiajun Wu.

I received my Bachelor's degree from The University of Sydney, where I was fortunate to work with Prof. Weidong Cai. I was awarded Honors Class I and **The University Medal**.

I have particular research interests on Machine Learning & Computer Vision.

Contact: {X @ Y}, where X=xtiange, Y=stanford.edu

Google Scholar / Github / Linkedin

Recent Works

SQUID: Deep Feature In-Painting for Unsupervised Anomaly Detection

Tiange Xiang[1] Yixiao Zhang[2] Yongyi Lu[2] Alan L. Yuille[2]
Chaoyi Zhang[1] Weidong Cai[1] Zongwei Zhou[2,*]

[1]University of Sydney [2]Johns Hopkins University

GitHub: https://github.com/tiangexiang/SQUID

Seeing Beyond the Brain: Conditional Diffusion Model with Sparse Masked Modeling for Vision Decoding

Zijiao Chen[1*] Jiaxin Qing[2*] Tiange Xiang[3] Wan Lin Yue[1] Juan Helen Zhou[1†]

[1]National University of Singapore, [2]The Chinese University of Hong Kong, [3]Stanford University

https://mind-vis.github.io

Recent Works

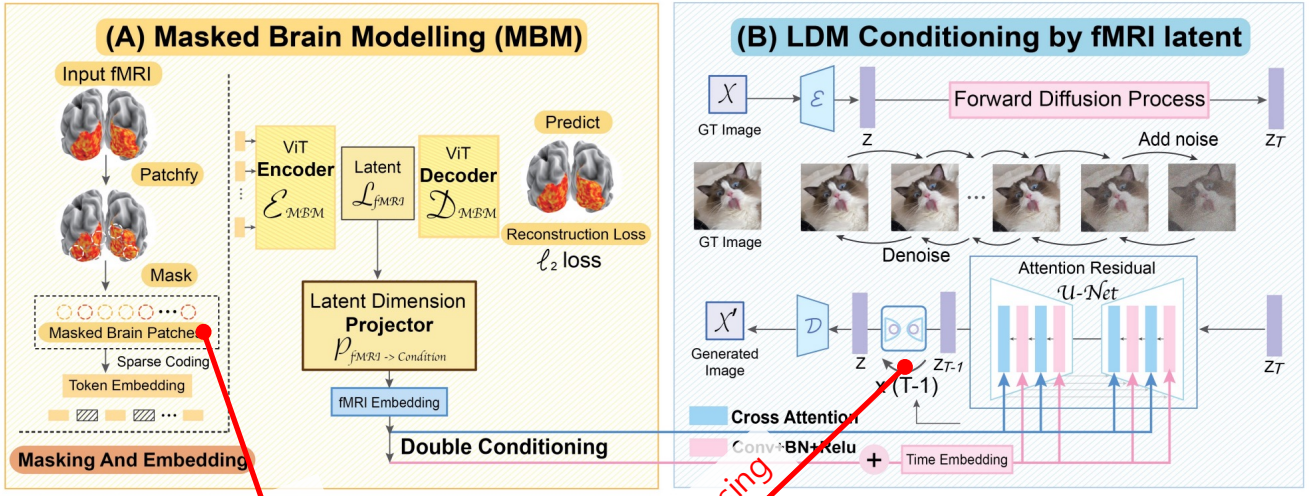


Figure 3. **MinD-Vis. Stage A**: Pre-train on fMRI with SC-MBM. We patchify, randomly mask the fMRI, and then tokenize them to large embeddings. We train an autoencoder ($\mathcal{E}_{\mathcal{MBM}}$ and $\mathcal{D}_{\mathcal{MBM}}$) to recover the masked patches. **Stage B (right):** Integration with the LDM through double conditioning. We project the fMRI latent ($\mathcal{L}_{fMRI}$) through two paths to the LDM conditioning space with a latent dimension projector ($\mathcal{P}_{fMRI \rightarrow Cond}$). One path connects directly to cross-attention heads in the LDM. The other path adds the fMRI latent to time embeddings. The LDM operates on a low-dimensional, compressed version of the original image (*i.e.* image latent), however, the original image is used in this figure for illustrations.

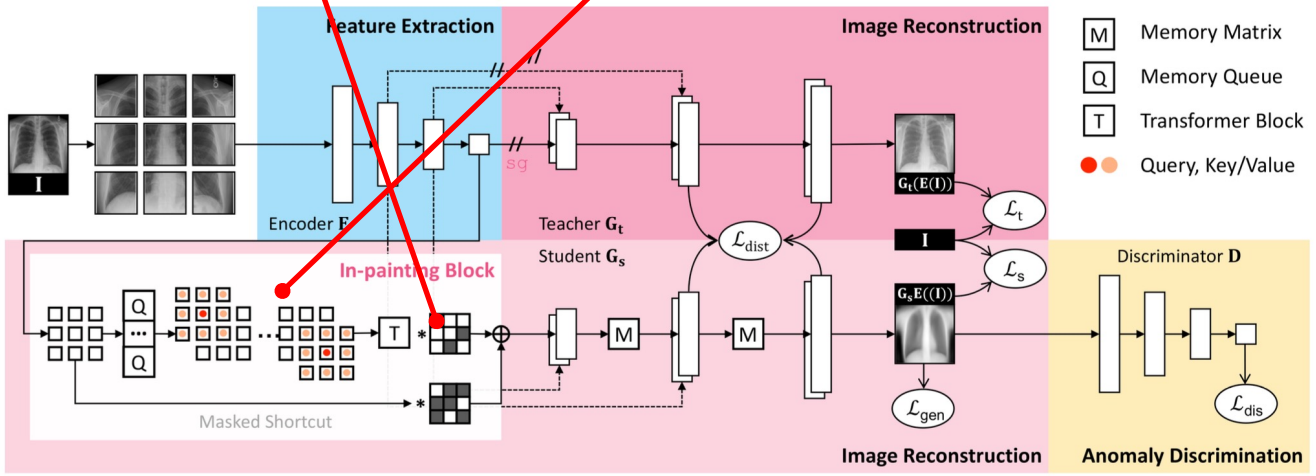


Figure 2. **SQUID.** We divide an input image into $N \times N$ non-overlapping patches and feed them into the encoder for feature extraction. Two generators will be trained to reconstruct the original image. Along with the reconstruction, a dictionary of anatomical patterns will be created and updated dynamically via a novel Memory Queue (§3.2); The teacher generator directly uses the features extracted by the encoder; the student generator uses the features augmented by our in-painting block (§3.3). The teacher and student generators are coupled through a knowledge distillation paradigm. We employ a discriminator to assess whether the image reconstructed by the student generator is real or fake. Once trained, it can also be used to detect anomalies in test images (§3.4).

Summary

* UAD: unsupervised anomaly detection

Summary

Proposal

- **Space-aware memory**: only access the information of patch itself
- **Memory Queue**: kind of dictionary for real normal patterns
- **Gumbel shrinkage**: trick for passing priority information while guaranteeing the gradient flow
- **Latent (masking) inpainting**: ease the issue of boundary artifacts from pixel-level inpainting

Contributions

- Shows patch features from memory matrix and memory queue
  - Memory queue can represent the pattern of training set well
- Provides comparative experiments with reference models
  - Ablation study with 8 modules
  - Results of hyperparameter tuning
- Enables true UAD
  - No need for filtered (labeled) normal samples for training

Limitations

- SQUID cannot localize anomalies at the pixel level
  - It measures anomaly score via trained discriminator
- Low scalability
  - Proposed method seems to be effective only on limited types of data
  - Consistent spatial relationships such as anatomical patterns
- Short explanation
  - Missing explanations can be interpreted via source code
  - Understanding of prior studies is necessary

Contribution (Authors claim) ?

- Best performing UAD method
- A synthetic dataset, digit anatomy
- Overcomes limitations in dominant UAD methods

Target Domain

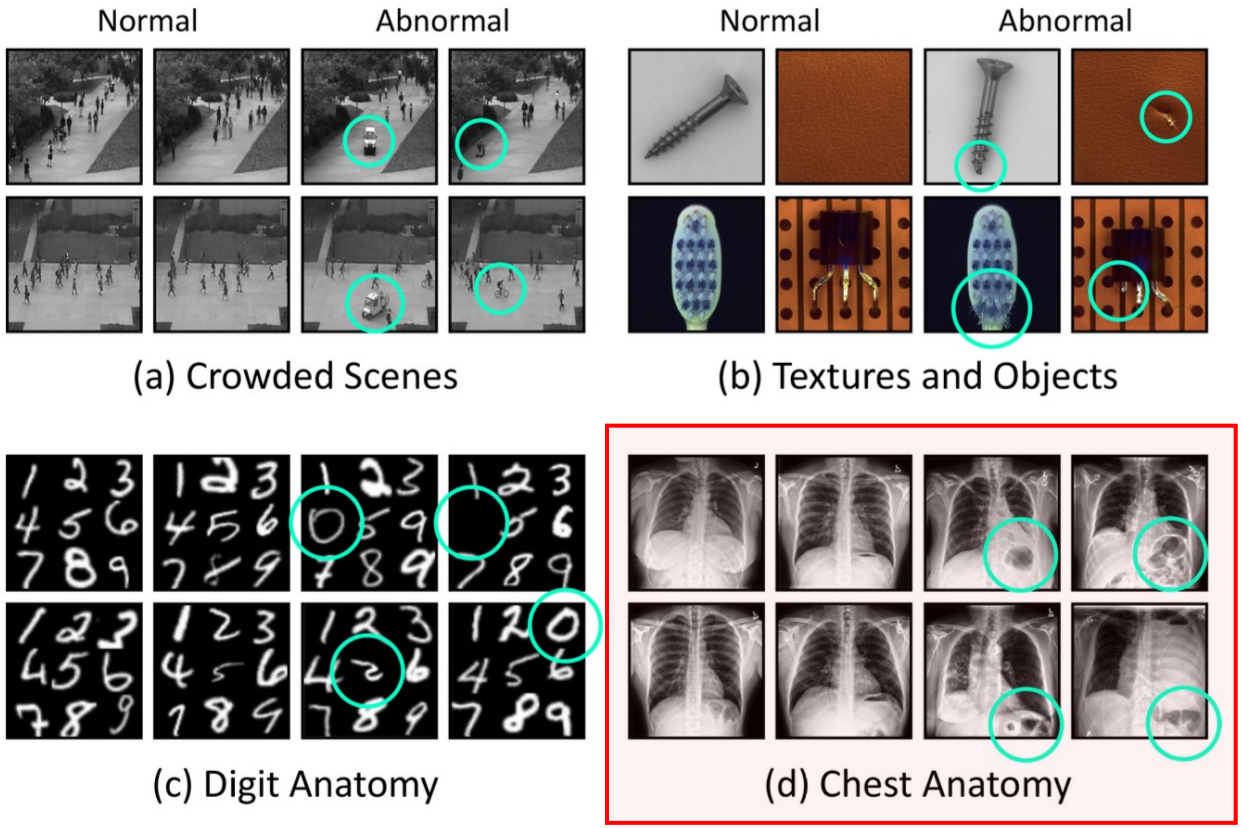


Figure 1. Anomaly detection in radiography images can be both easier and harder than in photographic images. It is easier because radiography images are spatially structured due to consistent imaging protocols. It is harder because anomalies in radiography images are subtle and require medical expertise to annotate.

SQUID

Space-aware Memory **Qu**eues for **I**n-painting and **D**etecting anomalies from radiography images

Overall

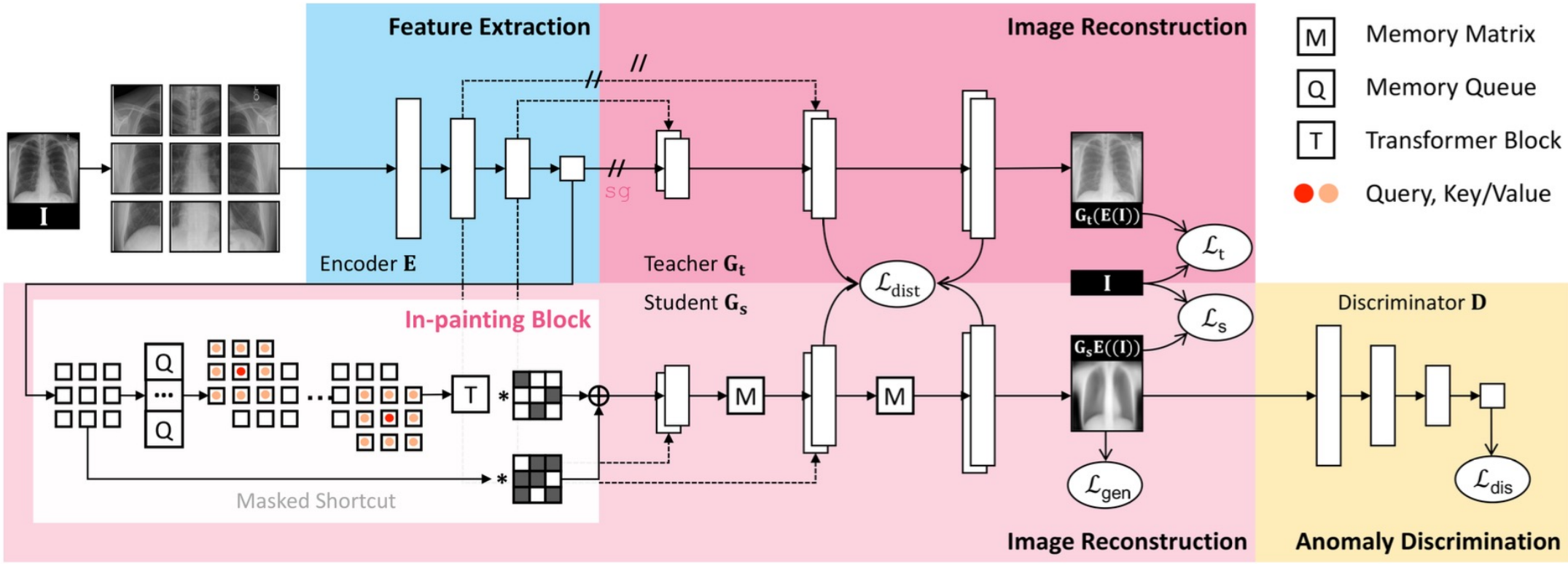


Figure 2. **SQUID.** We divide an input image into $N \times N$ non-overlapping patches and feed them into the encoder for feature extraction. Two generators will be trained to reconstruct the original image. Along with the reconstruction, a dictionary of anatomical patterns will be created and updated dynamically via a novel Memory Queue (§3.2); The teacher generator directly uses the features extracted by the encoder; the student generator uses the features augmented by our in-painting block (§3.3). The teacher and student generators are coupled through a knowledge distillation paradigm. We employ a discriminator to assess whether the image reconstructed by the student generator is real or fake. Once trained, it can also be used to detect anomalies in test images (§3.4).

https://github.com/tiangexiang/SQUID/blob/b5cacbb32c99d9a782e484a2731e4580a5b807de/models/inpaint.py

Detail of SQUID

Table 3. Encoder structure in SQUID.

Level	#Channels	Resolution
Input	1	$(2 \times 2) \times (64 \times 64)$
1	32	$(2 \times 2) \times (32 \times 32)$
2	64	$(2 \times 2) \times (16 \times 16)$
3	128	$(2 \times 2) \times (8 \times 8)$
4	256	$(2 \times 2) \times (4 \times 4)$

Table 4. Student and teacher generator structures in SQUID. S&M denotes the usage of skip connections and Memory Matrix. Note that there is no Memory Matrix placed in the teacher generator.

Level	#Channels	*w/* S&M	Resolution
4	256	✓	$(2 \times 2) \times (4 \times 4)$
3	128	✓	$(2 \times 2) \times (8 \times 8)$
2	64		32×32
1	32		64×64
Output	1		128×128

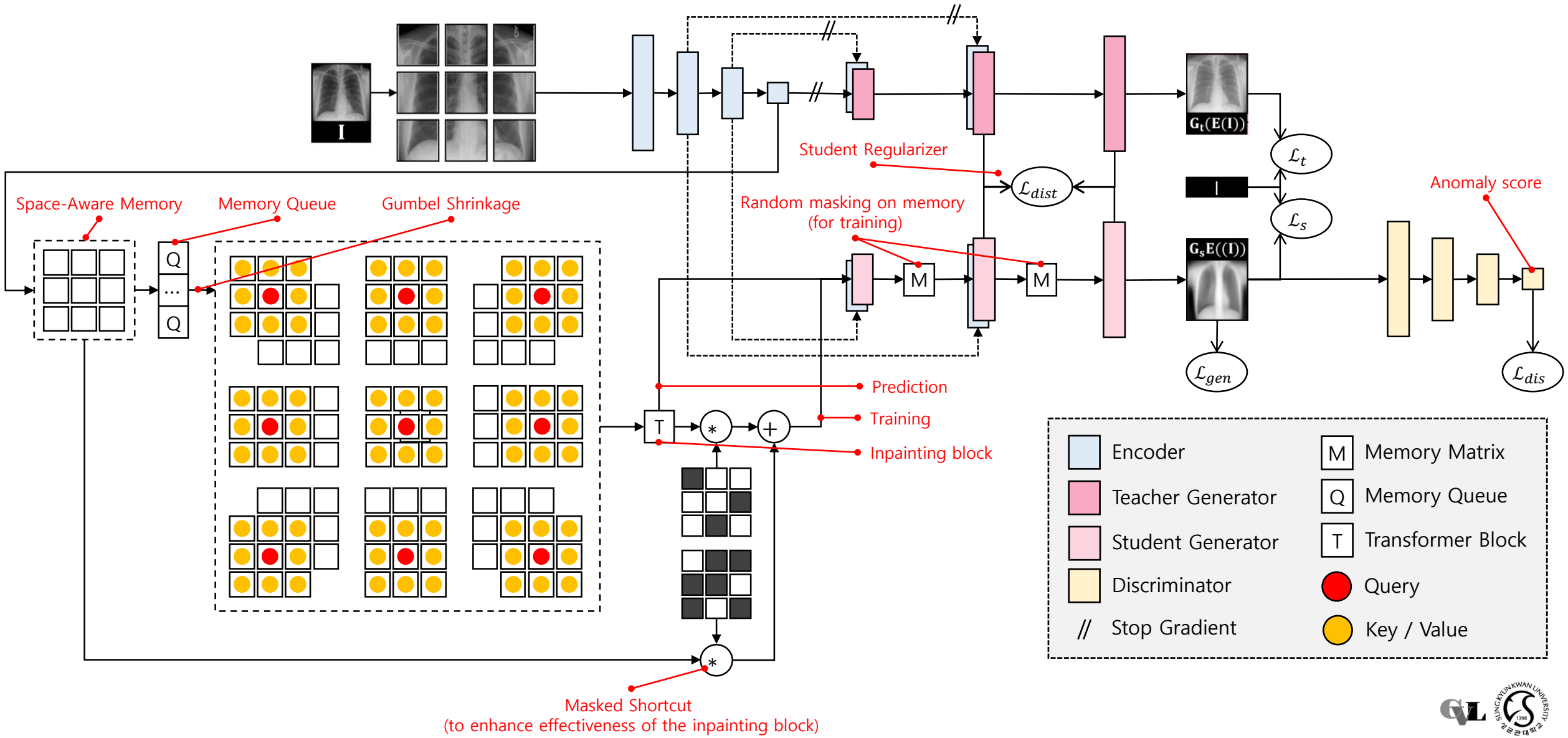
Table 5. Discriminator structure in SQUID.

Level	#Channels	Resolution
Input	1	128×128
1	16	64×64
2	32	32×32
3	64	16×16
4	128	8×8
5	128	4×4
Output	1	1×1

No information for Transformer
(Inpainting Block)
But it is implemented as following sequence.

- 1 x 1 convolution
- Multi-head attention (single layer)
- MLP (two layers)
- 1 x 1 convolution

Overall (Full Structure)



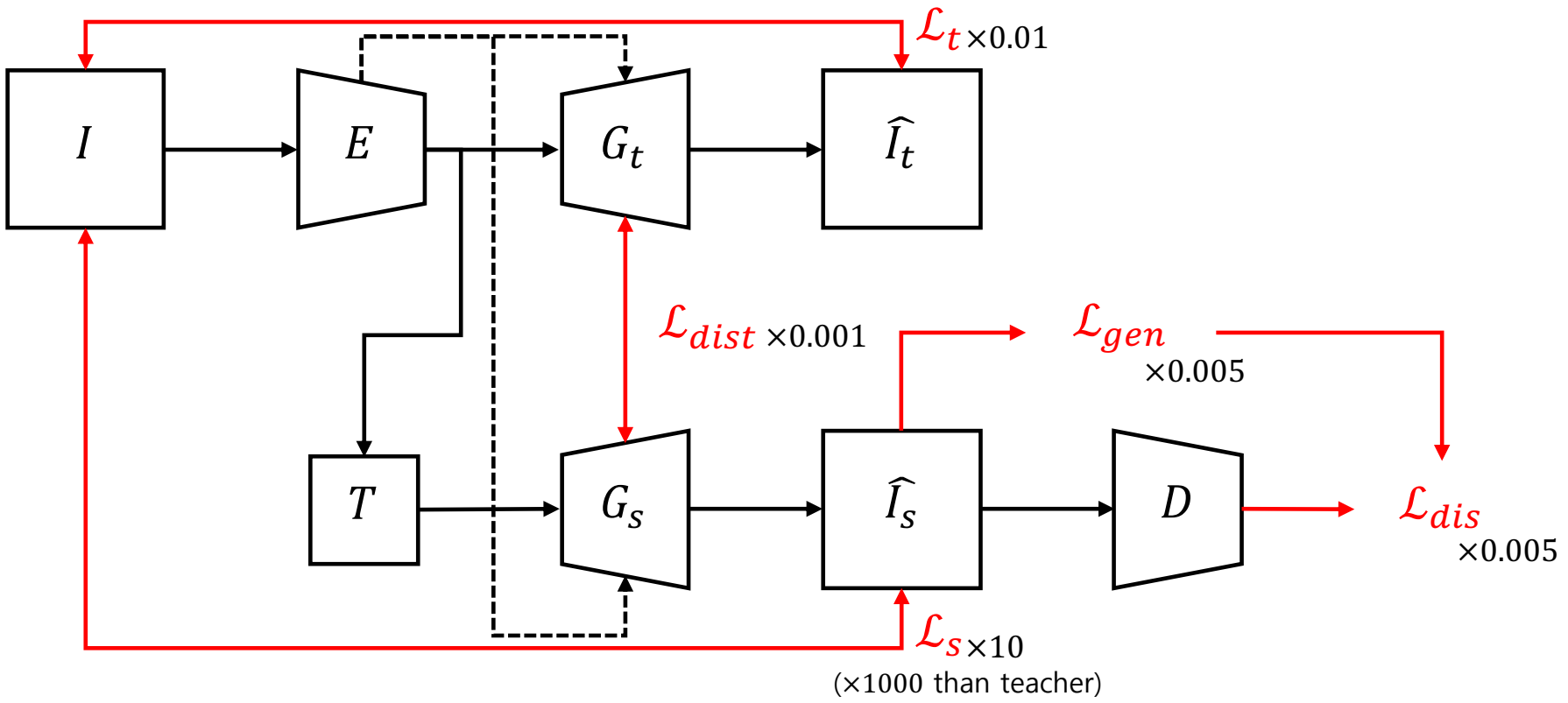
Training

3.5. Loss Function

SQUID is optimized by five loss functions. The mean square error (MSE) between input and reconstructed images is used for both teacher and student generators. Concretely, $\mathcal{L}_{\text{t}} = (\mathbf{I} - \mathbf{G}_{\text{t}}(\mathbf{E}(\mathbf{I})))^2$ and $\mathcal{L}_{\text{s}} = (\mathbf{I} - \mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I})))^2$ for the teacher and student generators, respectively, where $\mathbf{I}$ denotes the input image. Following the knowledge distillation paradigm, we apply a distance constraint between the teacher and student generators to all levels of features: $\mathcal{L}_{\text{dist}} = \sum_{i=1}^{l}(\mathcal{F}_{\text{t}}^i - \mathcal{F}_{\text{s}}^i)^2$, where l is the level of features used for knowledge distillation, $\mathcal{F}_{\text{t}}$ and $\mathcal{F}_{\text{s}}$ are the intermediate features in the teacher and student generators, respectively. In addition, we employ an adversarial loss (similar to DCGAN [55]) to improve the quality of the image generated by the student generator. Specifically, the following equation is minimized: $\mathcal{L}_{\text{gen}} = \log(1 - \mathbf{D}(\mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I}))))$. The discriminator seeks to maximize the average of the probability for real images and the inverted probability for fake images: $\mathcal{L}_{\text{dis}} = \log(\mathbf{D}(\mathbf{I})) + \log(1 - \mathbf{D}(\mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I}))))$. In summary, SQUID is trained to *minimize* the generative loss terms $(\lambda_{\text{t}}\mathcal{L}_{\text{t}} + \lambda_{\text{s}}\mathcal{L}_{\text{s}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{gen}}\mathcal{L}_{\text{gen}})$ and to *maximize* the discriminative loss term $(\lambda_{\text{dis}}\mathcal{L}_{\text{dis}})$.

4.4. Implementation Details

We utilized common data augmentation strategies such as random translation within the $[-0.05, +0.05]$ range and a random scaling of $[0.95, 1.05]$. The Adam [34] optimizer was used with a batch size of 16 and a weight decay of $1e$-5. The learning rate was initially set to $1e$-4 for both the generator and the discriminator and then decayed to $2e$-5 in 1000 epochs following the cosine annealing scheduler. The discriminator is trained at every iteration, while the generator is trained every two iterations. We set loss weights as $\lambda_{\text{t}} = 0.01$, $\lambda_{\text{s}} = 10$, $\lambda_{\text{dist}} = 0.001$, $\lambda_{\text{gen}} = 0.005$, and $\lambda_{\text{dis}} = 0.005$. We divide the input images in 2×2 non-overlapping patches, fix the shortcut mask probability at $\rho = 95\%$, and activate only the top 5 similar patterns in the Gumbel Shrinkage. The impact of these hyper-parameters is studied in §5.3. The architectures of our generators and discriminator are detailed in Appendix A.



MemAE

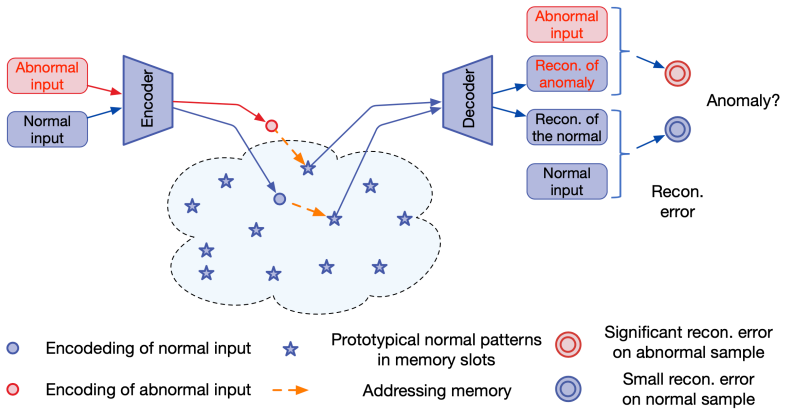


Figure 1. Anomaly detection via the proposed MemAE. After training on the dataset only with normal samples, the memory in MemAE records the prototypical normal patterns. Given an *abnormal* input, MemAE retrieves the most relevant *normal* patterns in memory for reconstruction, resulting in an output significantly different to the abnormal input. To simplify the visualization, we assume only one memory item is addressed here.

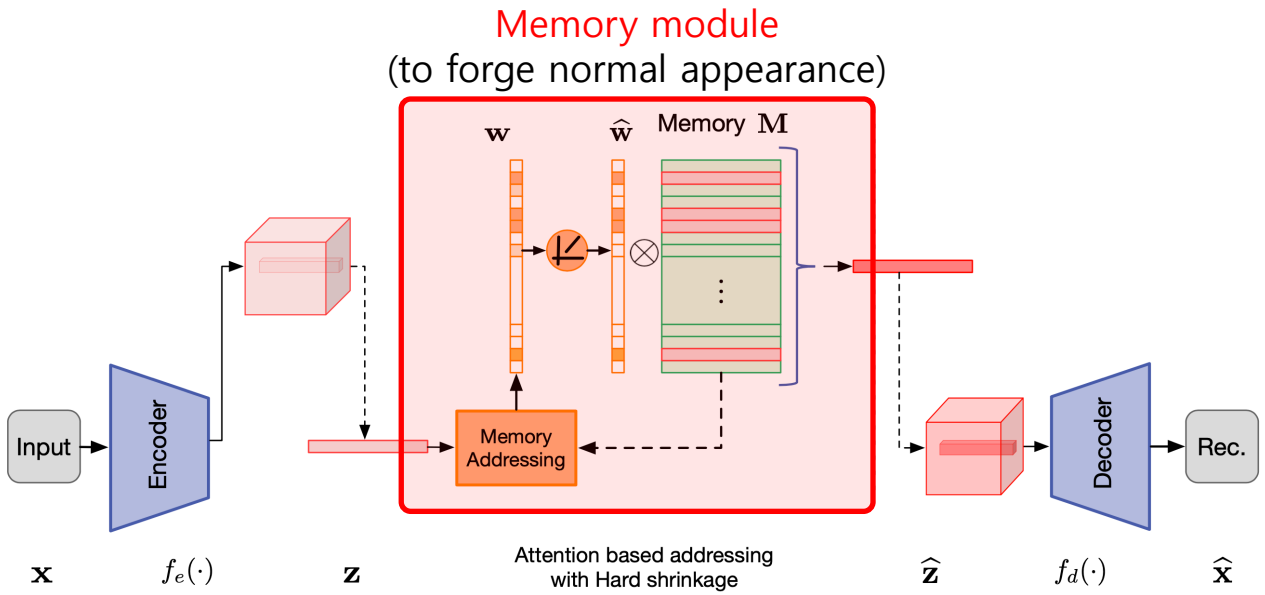


Figure 2. Diagram of the proposed MemAE. The memory addressing unit takes the encoding $\mathbf{z}$ as query to obtain the soft addressing weights. The memory slots can be used to model the whole encoding or the features on one pixel of the encoding (as shown in the figure). Note that $\hat{\mathbf{w}}$ is normalized after the hard shrinkage operation.

Space-Aware Memory [1/2]

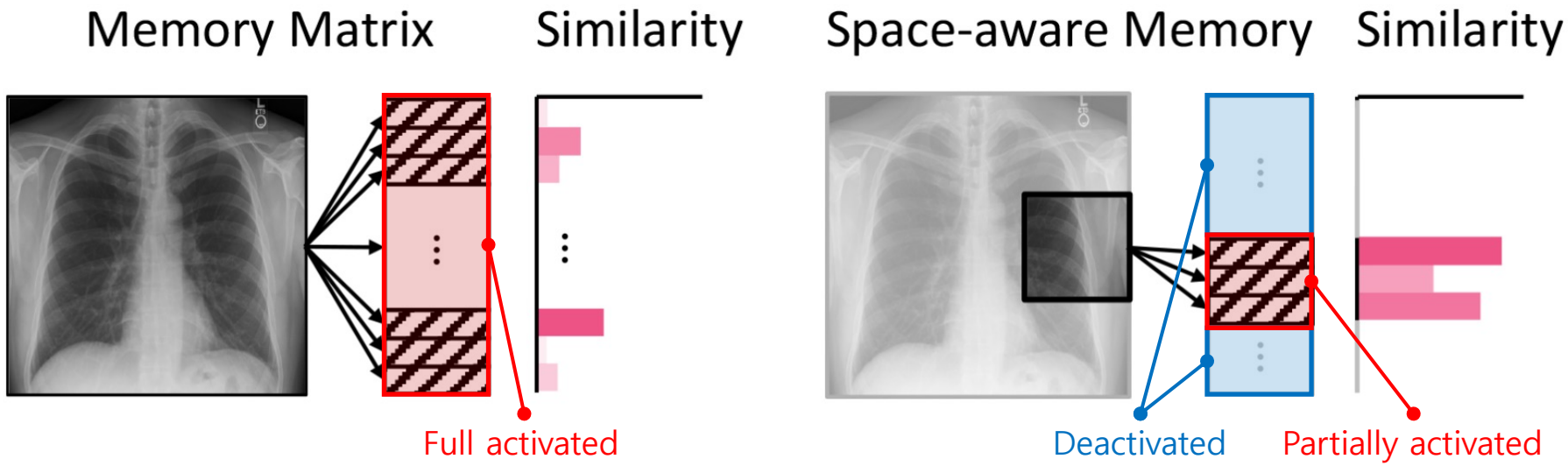
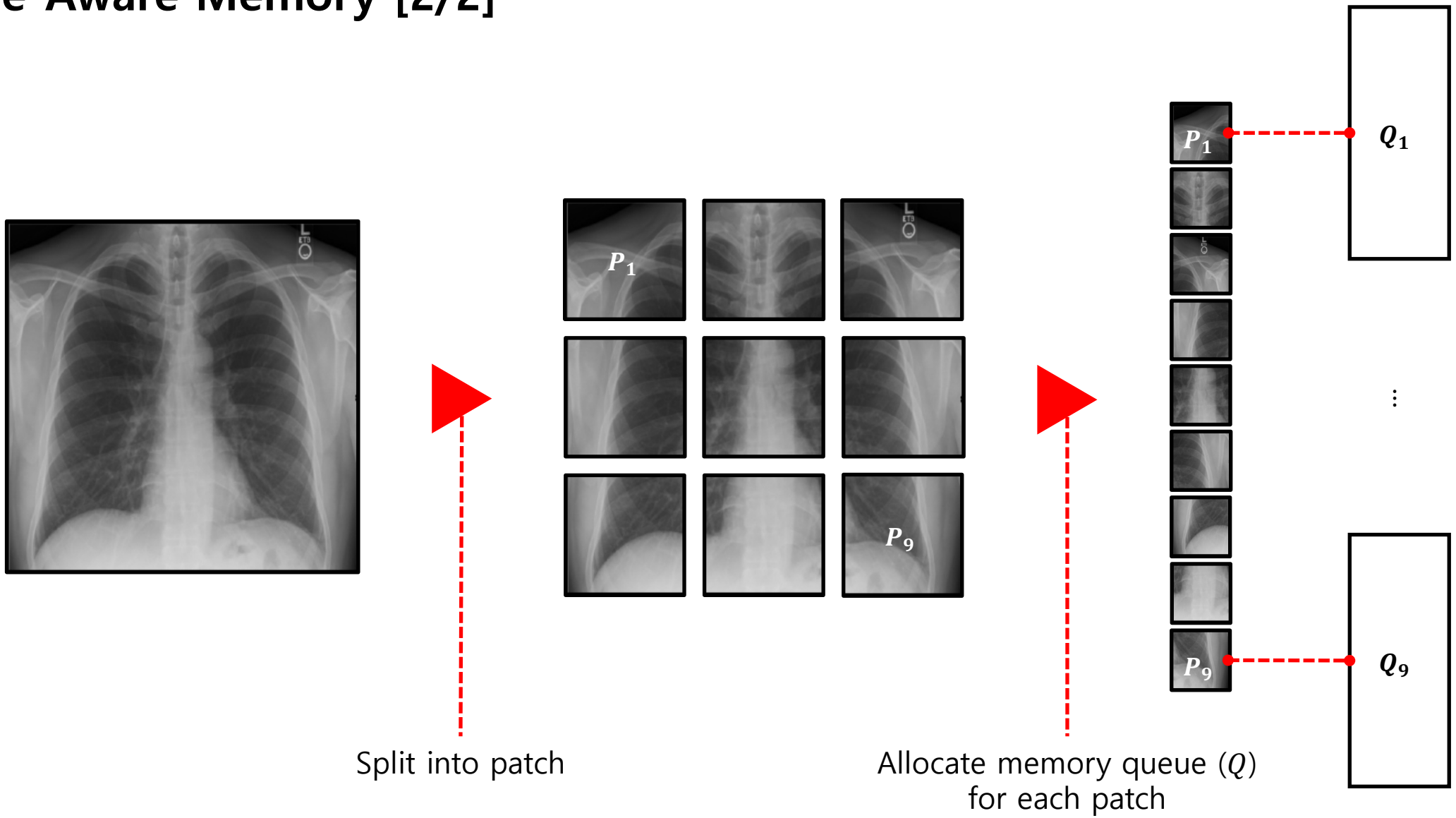


Figure 3. **Space-aware memory.** For unique encoding of location information, we restrict each patch to be only accessible by a non-overlapping region in the memory.

Space-Aware Memory [2/2]



Memory Queue [1/3]

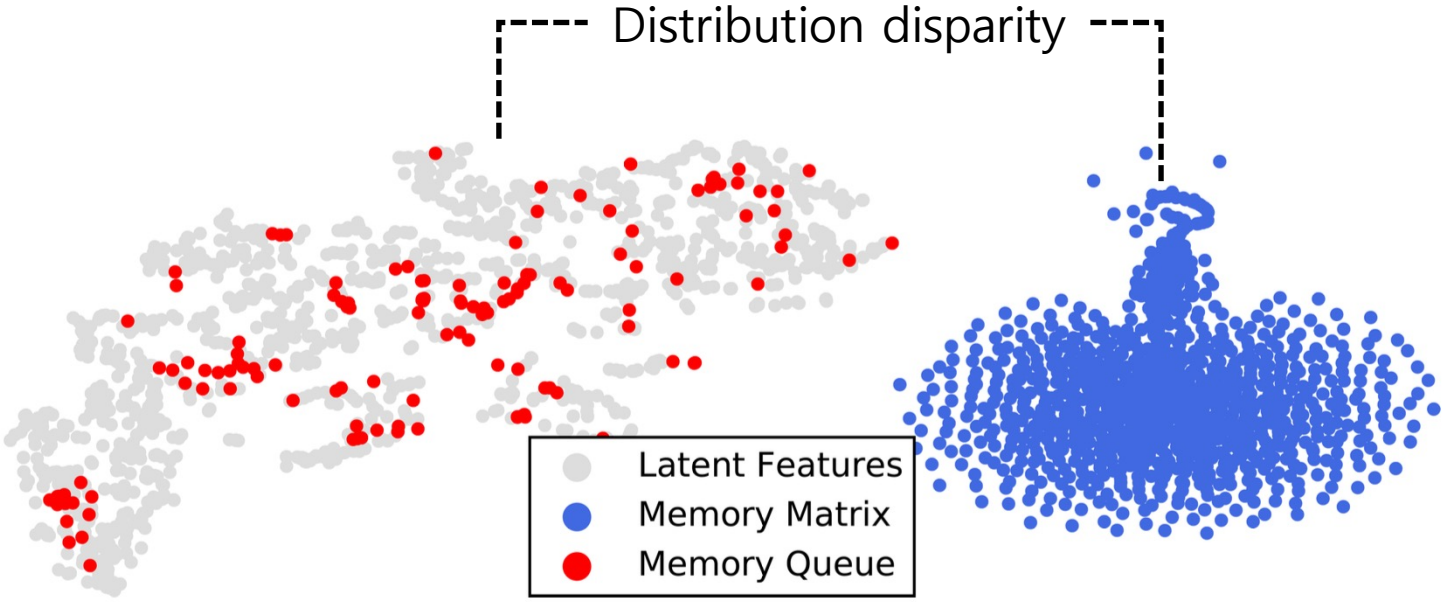


Figure 4. t-SNE visualizations of patterns in Memory Matrix, Memory Queue, and patch features of the training samples [73]. Patterns in the Memory Matrix are far away from the distribution of patch features, while patterns in the Memory Queue (as copies of previously seen features) share a similar distribution.

Memory Queue [2/3]

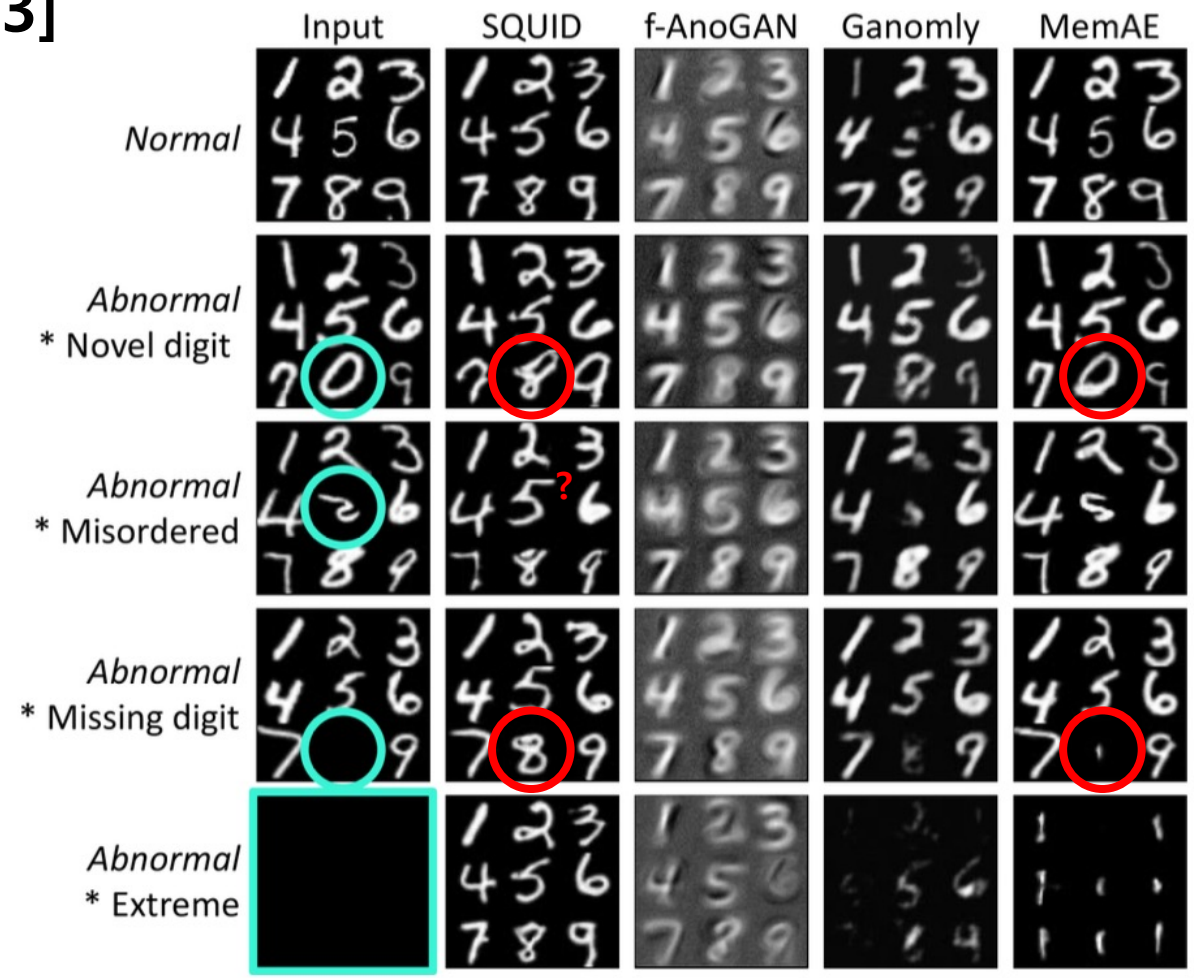
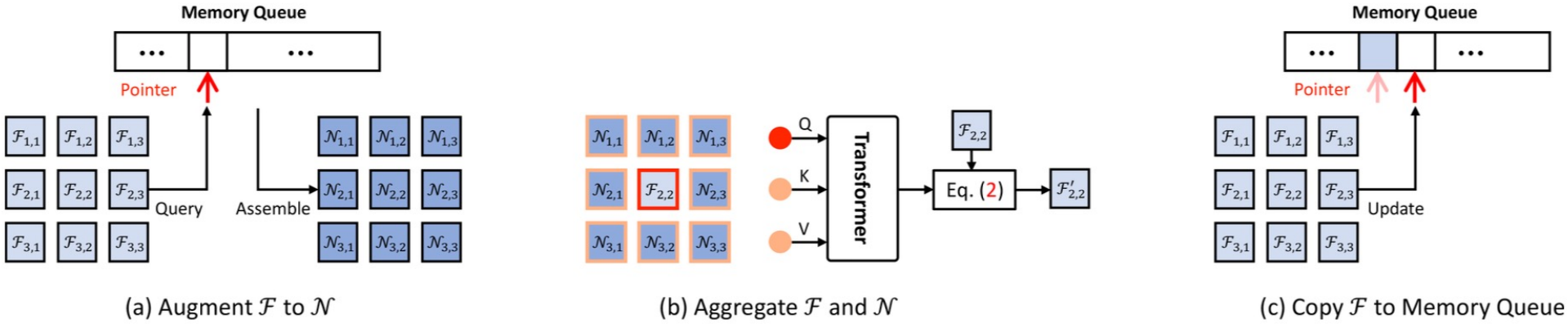


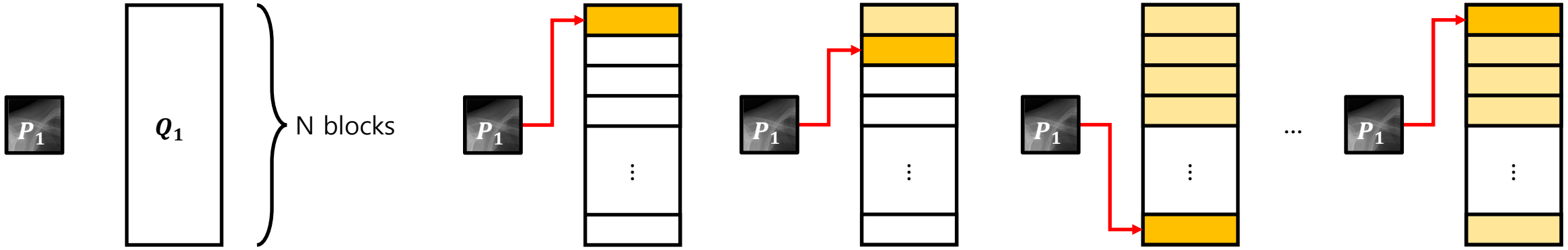
Figure 6. **Reconstruction results on DigitAnatomy.** Our feature-level in-painting is more robust to amplified noise and pixel variance than the existing pixel-level in-painting methods. More visualization can be found in Appendix D.

Memory Queue [3/3]



(a) Augment $\mathcal{F}$ to $\mathcal{N}$ (b) Aggregate $\mathcal{F}$ and $\mathcal{N}$ (c) Copy $\mathcal{F}$ to Memory Queue

Figure 5. **Three-step workflow of our in-painting block.** (a) Each non-overlapping patch feature $\mathcal{F}$ is queried to a unique region in Memory Queue, and the most similar items are assembled to $\mathcal{N}$. (b) Each center patch feature $\mathcal{F}$ and its eight neighbors $\mathcal{N}$ are used as query and key/value, respectively, to a Transformer layer for in-painting. (c) Each Memory Queue region copies its corresponding patch features $\mathcal{F}$ into the memory by maintaining a pointer. Note that this step is only performed during training.



Gumbel Shrinkage [1/3]

The Gumbel-Max trick (Gumbel, 1954; Maddison et al., 2014) provides a simple and efficient way to draw samples z from a categorical distribution with class probabilities π:

Non-differentiable

$$z = \texttt{one\_hot}\left(\arg\max_i \left[g_i + \log \pi_i\right]\right) \tag{1}$$

where $g_1 \ldots g_k$ are i.i.d samples drawn from Gumbel$(0, 1)$[1]. We use the softmax function as a continuous, differentiable approximation to arg max, and generate k-dimensional sample vectors $y \in \Delta^{k-1}$ where

Differentiable

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)} \qquad \text{for } i = 1, \ldots, k. \tag{2}$$

Gumbel Shrinkage [2/3]

MemAE, Hard Shrinkage

$$\widehat{w}_i = h(w_i; \lambda) = \begin{cases} w_i, & \text{if } w_i > \lambda, \\ 0, & \text{otherwise}, \end{cases} \quad (6)$$

SQUID, Gumbel Shrinkage

$$\mathbf{w}' = \overbrace{\mathtt{sg}(\mathtt{hs}(\mathbf{w}, \mathtt{topk}(\mathbf{w})) - \phi(\mathbf{w}))}^{\text{Forward only}} + \overbrace{\phi(\mathbf{w})}^{\text{Forward \& Backward}}, \quad (1)$$

- $\boldsymbol{w}$: similarity between image feature and memory entries
- $\boldsymbol{sg}$: stop gradient
- $\boldsymbol{hs}$: hard shrinkage $hs(\cdot, t)$ with threshold t (same as λ)
- $\boldsymbol{topk}$: top-k most similar entries (set to 5)
- $\boldsymbol{\phi}$: softmax

Gumbel Shrinkage [3/3]

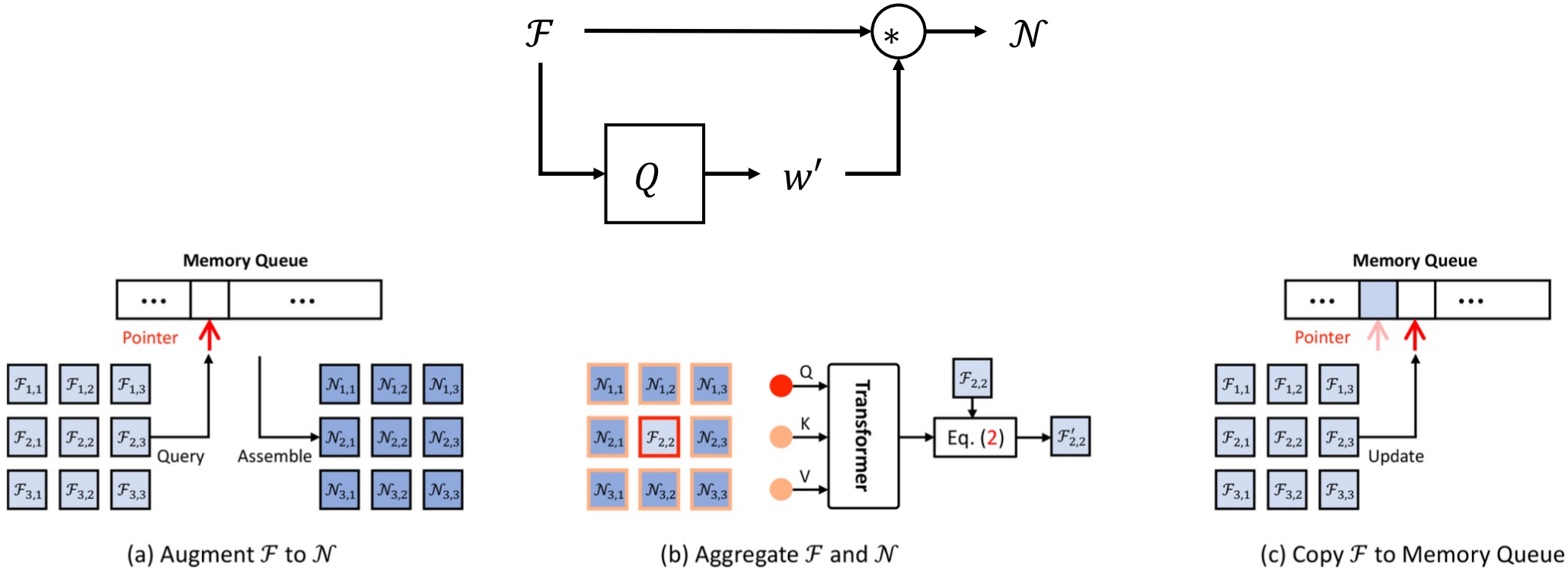


Figure 5. **Three-step workflow of our in-painting block.** (a) Each non-overlapping patch feature $\mathcal{F}$ is queried to a unique region in Memory Queue, and the most similar items are assembled to $\mathcal{N}$. (b) Each center patch feature $\mathcal{F}$ and its eight neighbors $\mathcal{N}$ are used as query and key/value, respectively, to a Transformer layer for in-painting. (c) Each Memory Queue region copies its corresponding patch features $\mathcal{F}$ into the memory by maintaining a pointer. Note that this step is only performed during training.

- Ashish Vaswani, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).
  - torch.nn.MultiheadAttention. https://pytorch.org/docs/stable/generated/torch.nn.MultiheadAttention.html

Latent Inpainting

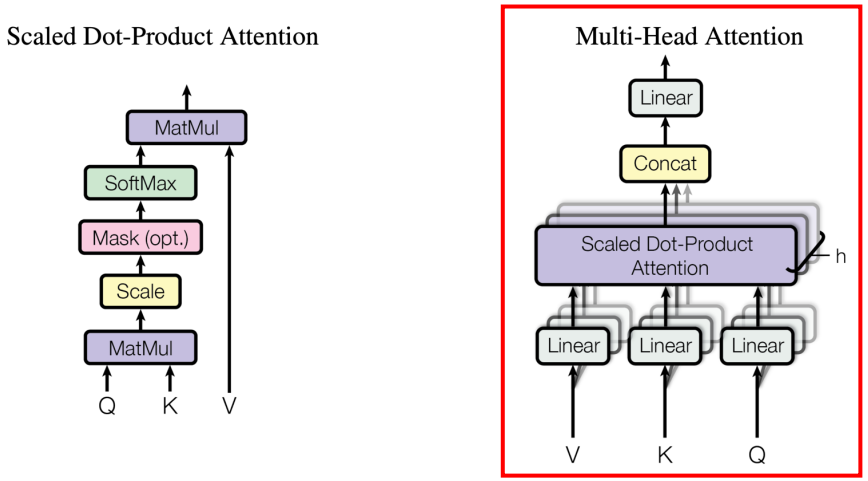
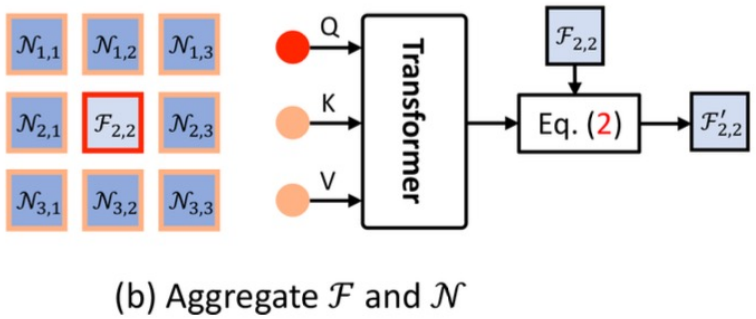


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

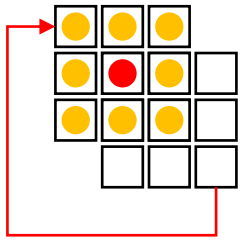
MULTIHEADATTENTION

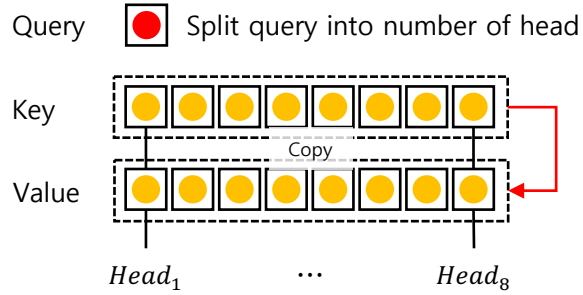
CLASS torch.nn.MultiheadAttention(*embed_dim*, *num_heads*, *dropout=0.0*, *bias=True*, *add_bias_kv=False*, *add_zero_attn=False*, *kdim=None*, *vdim=None*, *batch_first=False*, *device=None*, *dtype=None*) [SOURCE]

- **query** (*Tensor*) – Query embeddings of shape (L, E_q) for unbatched input, (L, N, E_q) when `batch_first=False` or (N, L, E_q) when `batch_first=True`, where L is the target sequence length, N is the batch size, and E_q is the query embedding dimension `embed_dim`. Queries are compared against key-value pairs to produce the output. See "Attention Is All You Need" for more details.
- **key** (*Tensor*) – Key embeddings of shape (S, E_k) for unbatched input, (S, N, E_k) when `batch_first=False` or (N, S, E_k) when `batch_first=True`, where S is the source sequence length, N is the batch size, and E_k is the key embedding dimension `kdim`. See "Attention Is All You Need" for more details.
- **value** (*Tensor*) – Value embeddings of shape (S, E_v) for unbatched input, (S, N, E_v) when `batch_first=False` or (N, S, E_v) when `batch_first=True`, where S is the source sequence length, N is the batch size, and E_v is the value embedding dimension `vdim`. See "Attention Is All You Need" for more details.

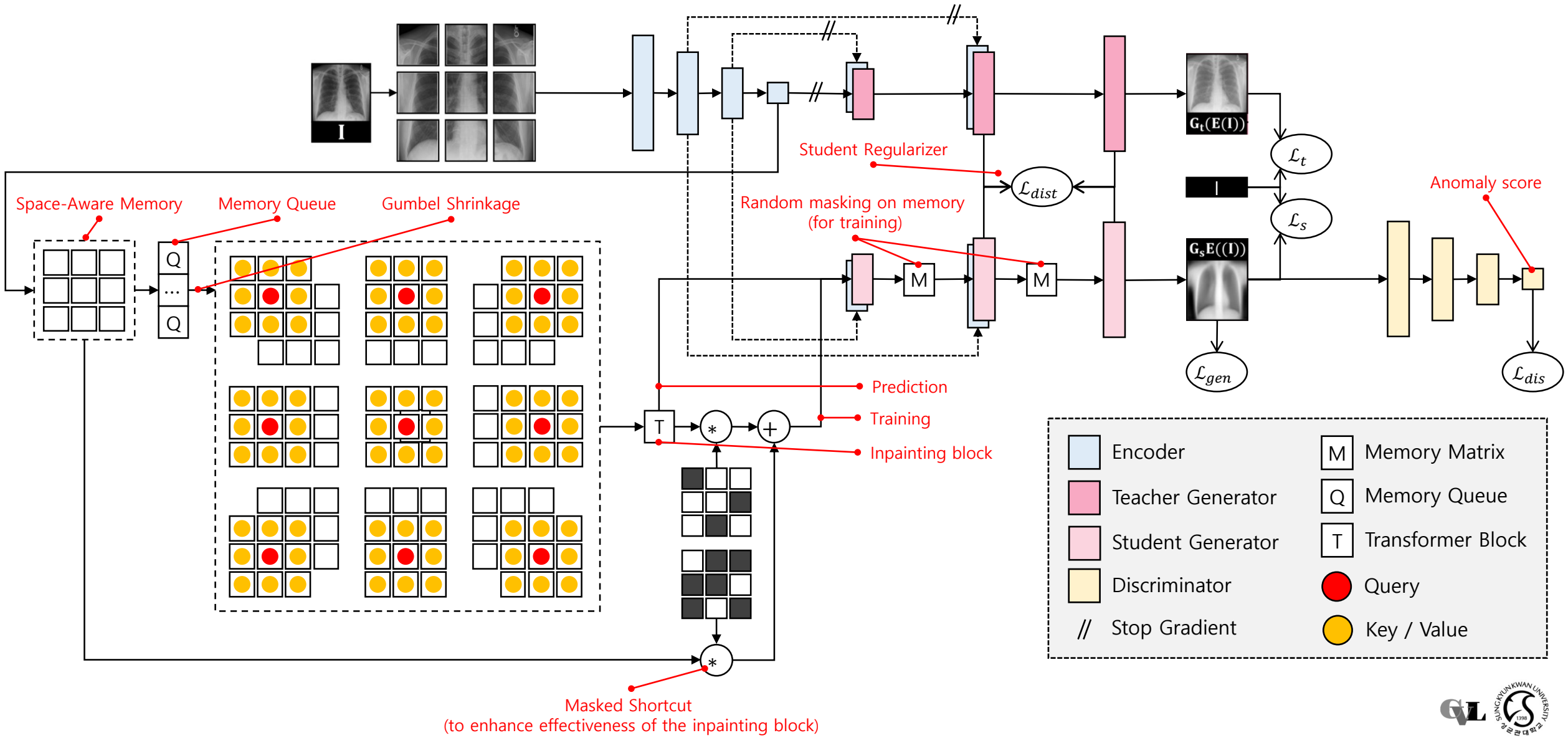


(b) Aggregate $\mathcal{F}$ and $\mathcal{N}$





Overall (recap)



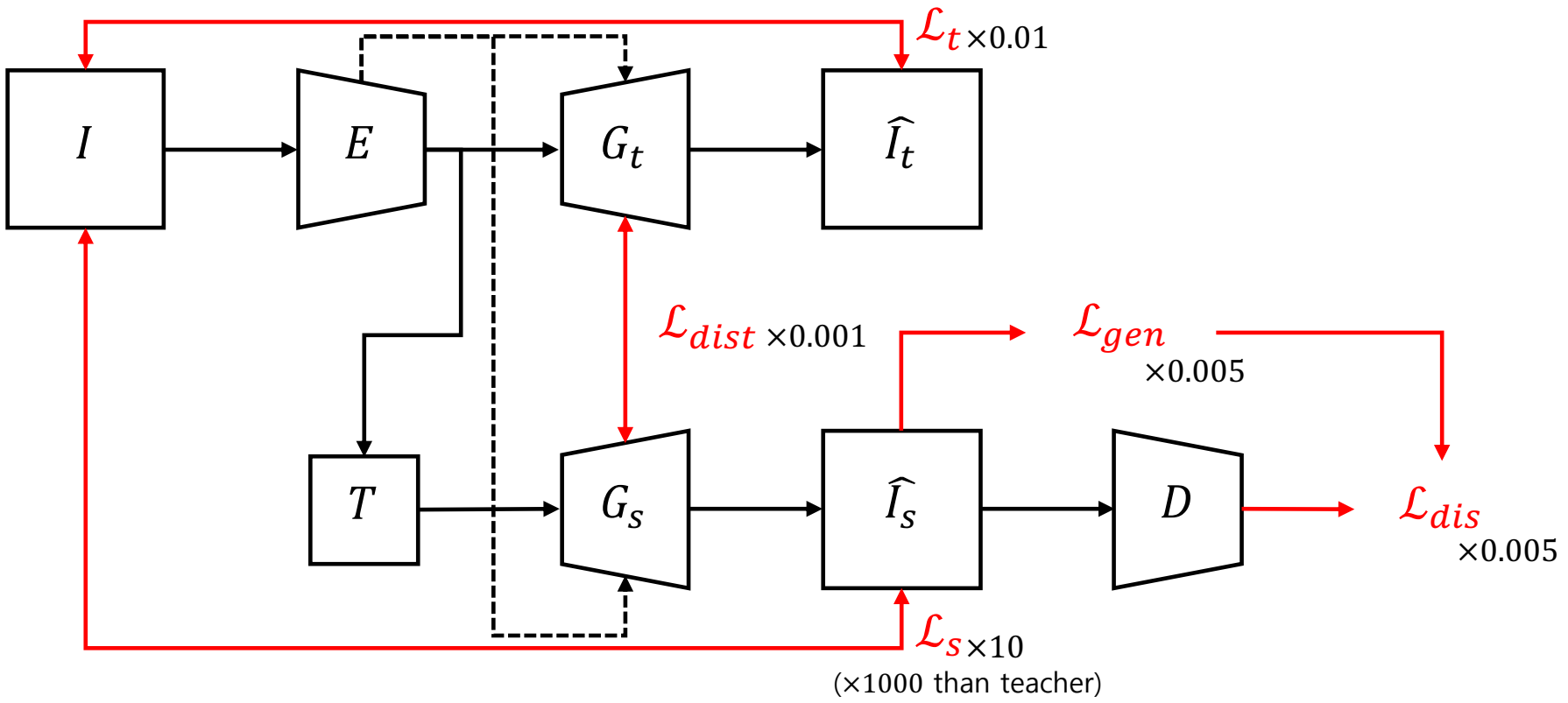
Training (recap)

3.5. Loss Function

SQUID is optimized by five loss functions. The mean square error (MSE) between input and reconstructed images is used for both teacher and student generators. Concretely, $\mathcal{L}_{\text{t}} = (\mathbf{I} - \mathbf{G}_{\text{t}}(\mathbf{E}(\mathbf{I})))^2$ and $\mathcal{L}_{\text{s}} = (\mathbf{I} - \mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I})))^2$ for the teacher and student generators, respectively, where $\mathbf{I}$ denotes the input image. Following the knowledge distillation paradigm, we apply a distance constraint between the teacher and student generators to all levels of features: $\mathcal{L}_{\text{dist}} = \sum_{i=1}^{l}(\mathcal{F}_{\text{t}}^i - \mathcal{F}_{\text{s}}^i)^2$, where l is the level of features used for knowledge distillation, $\mathcal{F}_{\text{t}}$ and $\mathcal{F}_{\text{s}}$ are the intermediate features in the teacher and student generators, respectively. In addition, we employ an adversarial loss (similar to DCGAN [55]) to improve the quality of the image generated by the student generator. Specifically, the following equation is minimized: $\mathcal{L}_{\text{gen}} = \log(1 - \mathbf{D}(\mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I}))))$. The discriminator seeks to maximize the average of the probability for real images and the inverted probability for fake images: $\mathcal{L}_{\text{dis}} = \log(\mathbf{D}(\mathbf{I})) + \log(1 - \mathbf{D}(\mathbf{G}_{\text{s}}(\mathbf{E}(\mathbf{I}))))$. In summary, SQUID is trained to *minimize* the generative loss terms $(\lambda_{\text{t}}\mathcal{L}_{\text{t}} + \lambda_{\text{s}}\mathcal{L}_{\text{s}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}} + \lambda_{\text{gen}}\mathcal{L}_{\text{gen}})$ and to *maximize* the discriminative loss term $(\lambda_{\text{dis}}\mathcal{L}_{\text{dis}})$.

4.4. Implementation Details

We utilized common data augmentation strategies such as random translation within the $[-0.05, +0.05]$ range and a random scaling of $[0.95, 1.05]$. The Adam [34] optimizer was used with a batch size of 16 and a weight decay of $1e$-5. The learning rate was initially set to $1e$-4 for both the generator and the discriminator and then decayed to $2e$-5 in 1000 epochs following the cosine annealing scheduler. The discriminator is trained at every iteration, while the generator is trained every two iterations. We set loss weights as $\lambda_{\text{t}} = 0.01$, $\lambda_{\text{s}} = 10$, $\lambda_{\text{dist}} = 0.001$, $\lambda_{\text{gen}} = 0.005$, and $\lambda_{\text{dis}} = 0.005$. We divide the input images in 2×2 non-overlapping patches, fix the shortcut mask probability at $\rho = 95\%$, and activate only the top 5 similar patterns in the Gumbel Shrinkage. The impact of these hyper-parameters is studied in §5.3. The architectures of our generators and discriminator are detailed in Appendix A.



Experiments

Reconstruction Quality

DigitAnatomy (Synthetic Data)

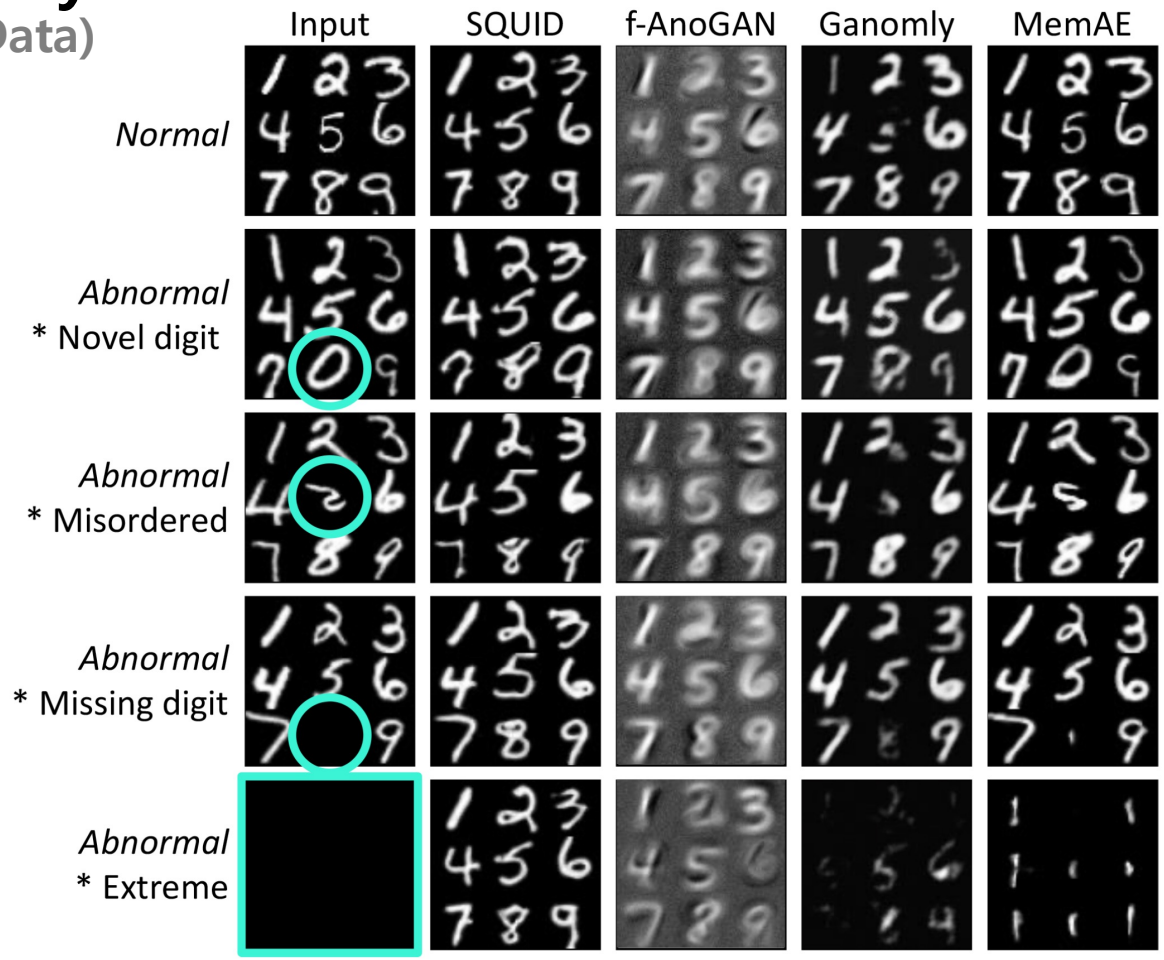


Figure 6. **Reconstruction results on DigitAnatomy.** Our feature-level in-painting is more robust to amplified noise and pixel variance than the existing pixel-level in-painting methods. More visualization can be found in Appendix D.

[1] Daniel S. Kermany, et al. "Identifying medical diagnoses and treatable diseases by image-based deep learning." cell 172.5 (2018): 1122-1131.
[2] Jeremy Irvin, et al. "Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison." Proceedings of the AAAI conference on artificial intelligence. Vol. 33. No. 01. 2019.

Anomaly Detection

Chest X-ray (Real Data)

	Training		Test	
	Normal	**Abnormal**	**Normal**	**Abnormal**
ZhangLab Chest X-ray [1]	1,349	3,883	234	390
Stanford CheXpert [2]	5,249	23,671	250	250

Table 1. Benchmark results on the test sets of the two datasets.

ZhangLab	Ref & Year	AUC (%)	Acc (%)	F1 (%)
Auto-Encoder	-	59.9	63.4	77.2
VAE [35]	Arxiv'13	61.8	64.0	77.4
Ganomaly [1]	ACCV'18	78.0	70.0	79.0
f-AnoGAN [61]	MIA'19	75.5	74.0	81.0
MemAE [17]	ICCV'19	77.8±1.4	56.5±1.1	82.6±0.9
MNAD [53]	CVPR'20	77.3±0.9	73.6±0.7	79.3±1.1
SALAD [82]	TMI'21	82.7±0.8	75.9±0.9	82.1±0.3
CutPaste [40]	CVPR'21	73.6±3.9	64.0±6.5	72.3±8.9
PANDA [56]	CVPR'21	65.7±1.3	65.4±1.9	66.3±1.2
M-KD [59]	CVPR'21	74.1±2.6	69.1±0.2	62.3±8.4
IF 2D [50]	MICCAI'21	81.0±2.8	76.4±0.2	82.2±2.7
PaDiM [12]	ICPR'21	71.4±3.4	72.9±2.4	80.7±1.2
IGD [10]	AAAI'22	73.4±1.9	74.0±2.2	80.9±1.3
SQUID	-	**87.6±1.5**	**80.3±1.3**	**84.7±0.8**

CheXpert	Ref & Year	AUC (%)	Acc (%)	F1 (%)
Ganomaly [1]	ACCV'18	68.9±1.4	65.7±0.2	65.1±1.9
f-AnoGAN [61]	MIA'19	65.8±3.3	63.7±1.8	59.4±3.8
MemAE [17]	ICCV'19	54.3±4.0	55.6±1.4	53.3±7.0
CutPaste [40]	CVPR'21	65.5±2.2	62.7±2.0	60.3±4.6
PANDA [56]	CVPR'21	68.6±0.9	66.4±2.8	65.3±1.5
M-KD [59]	CVPR'21	69.8±1.6	66.0±2.5	63.6±5.7
SQUID	-	**78.1±5.1**	**71.9±3.8**	**75.9±5.7**

?

Reconstruction Quality

Chest X-ray (Real Data)

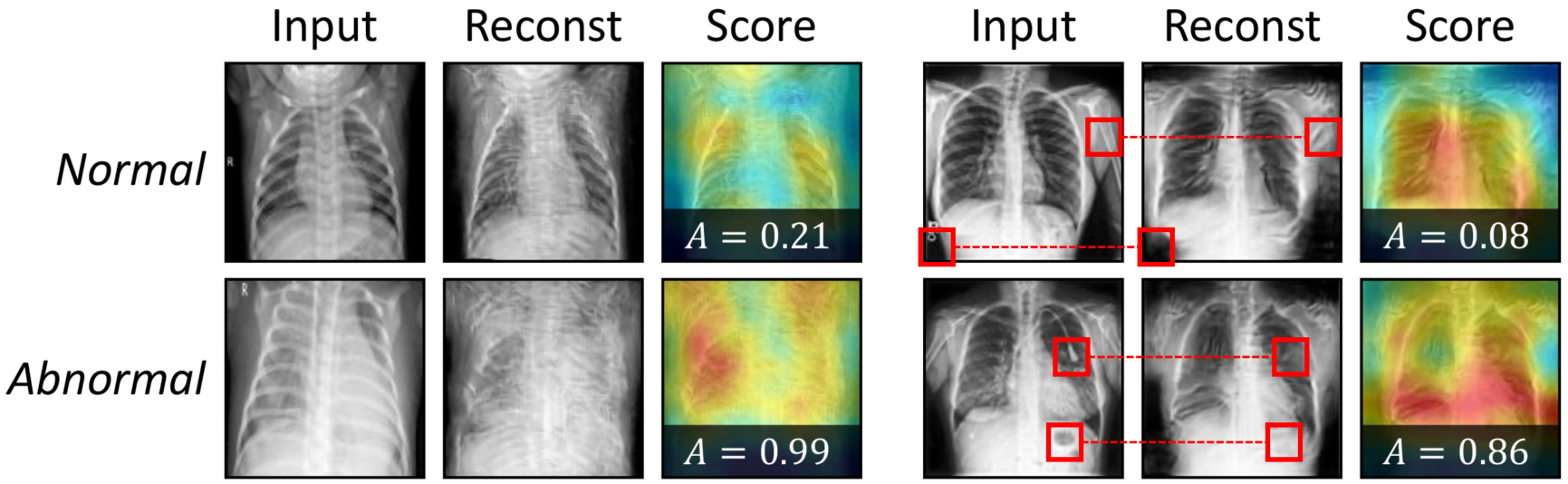


Figure 8. Reconstruction results of SQUID on the two datasets, associated with the corresponding anomaly scores (defined in §3.4). A larger score indicates a higher probability of being abnormal. More visualization can be found in Appendix D.

Ablation Study / Hyperparameter Tuning

Table 2. Performance benefits from all the components in SQUID.

Method	AUC (%)	Acc (%)	F1 (%)
w/o Space-aware Memory	77.6±0.5	75.5±0.5	82.5±0.6
w/o In-painting Block	80.9±2.1	75.8±1.5	81.6±1.3
w/o Gumbel Shrinkage	81.1±0.9	77.6±0.9	81.3±0.8
w/o Knowledge Distillation	81.2±0.8	75.2±0.7	81.3±0.8
w/o Stop Gradient	81.7±4.3	76.7±2.8	82.5±1.6
w/o Memory Queue	82.5±1.1	78.6±0.9	81.7±1.1
w/o Masked Shortcuts	82.5±1.3	76.4±0.8	82.3±1.1
w/o Decoder Memory	82.9±1.2	77.4±1.1	81.2±0.5
Full SQUID	**87.6±1.5**	**80.3±1.3**	**84.7±0.8**

Table 6. The extensive results indicate that all proposed techniques in SQUID are essential for a high overall performance.

Method	AUC (%)	Acc (%)	F1 (%)
Convolution Layers	76.9±3.3	74.2±3.3	80.7±2.7
Transformer Layers (Δ)	↑10.7	↑6.1	↑4.0
Soft Masked Shortcut	79.7±3.4	76.1±2.7	80.7±2.3
Hard Masked Shortcut (Δ)	↑7.9	↑4.2	↑4.0
Pixel-level In-painting	79.1±0.4	74.4±1.6	81.3±0.9
Feature-level In-painting (Δ)	↑8.5	↑5.9	↑3.4
Full SQUID	**87.6±1.5**	**80.3±1.3**	**84.7±0.8**

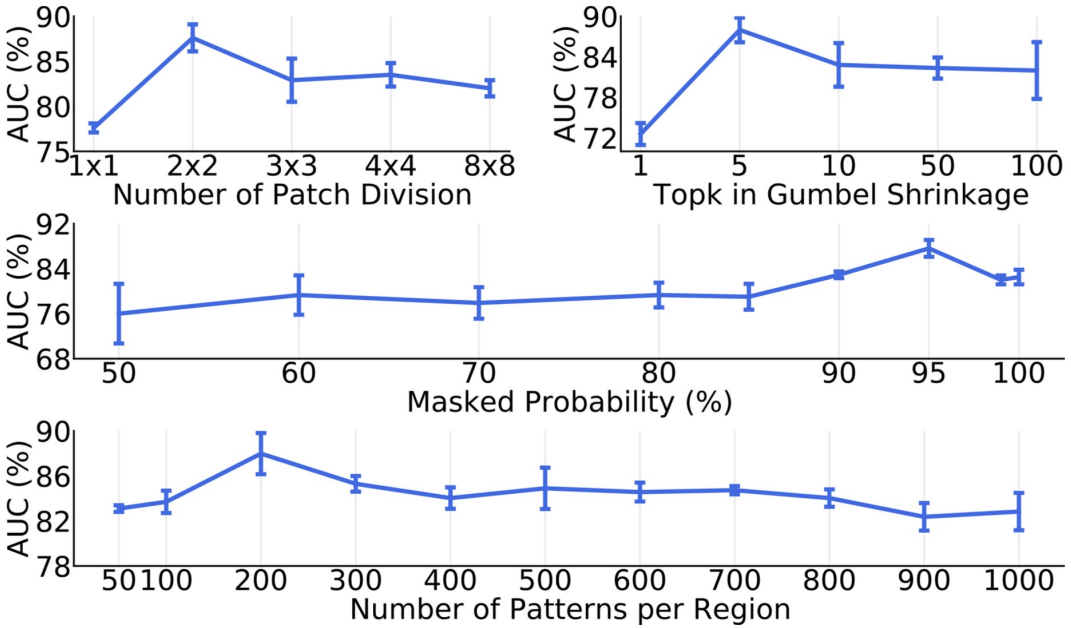


Figure 9. Hyper-parameter ablations.

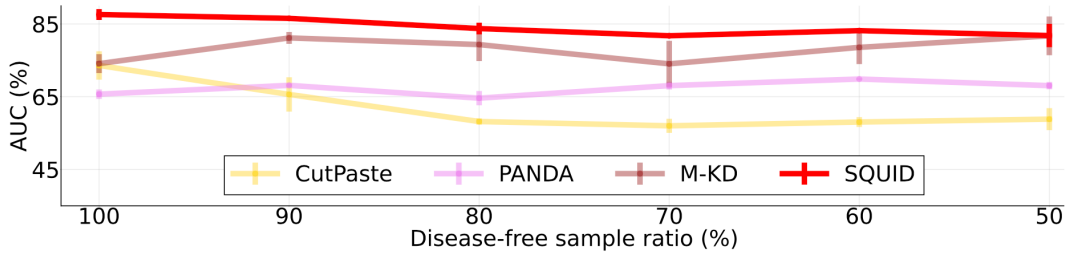


Figure 10. Results of mixing normal/abnormal training samples.

Conclusion

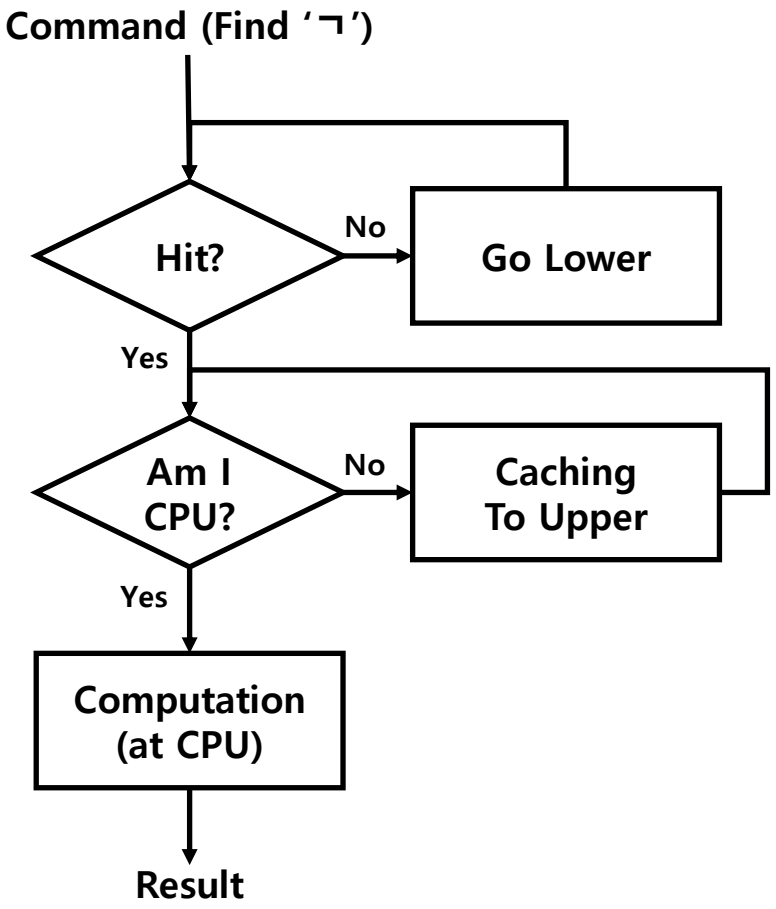
Acceptable Approaches

- **Space-aware memory**: only access the information of patch itself
- **Memory Queue**: kind of dictionary for real normal patterns
- **Gumbel shrinkage**: trick for passing priority information while guaranteeing the gradient flow

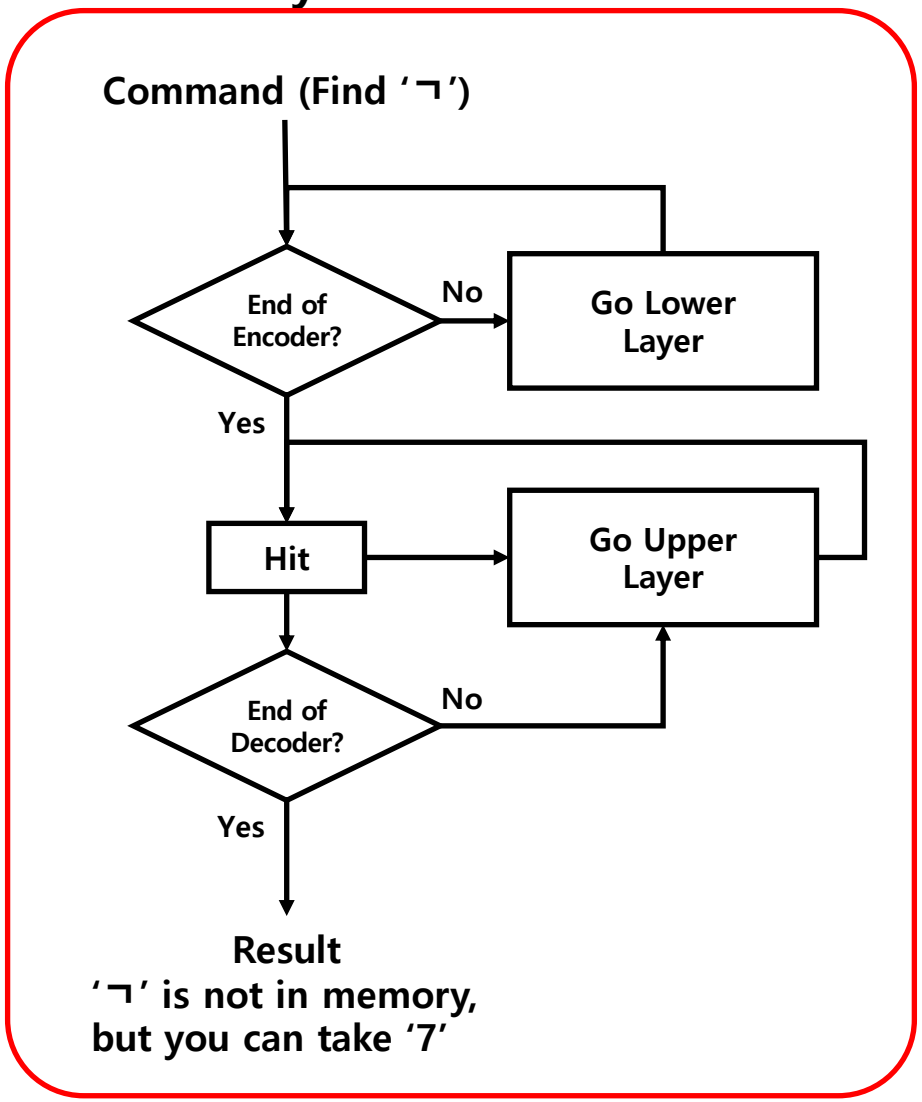
Appendix A

About MemAE

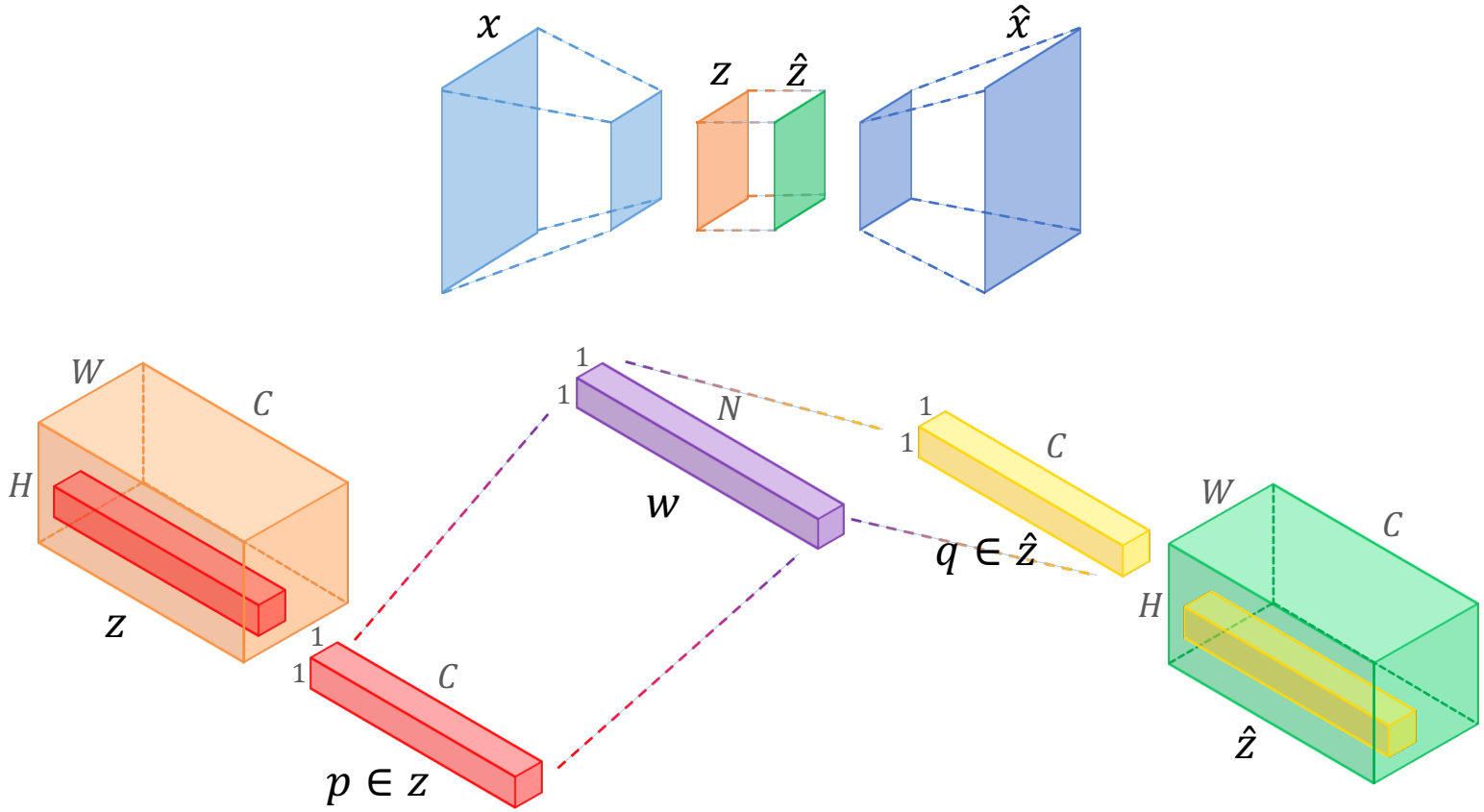
Concept of MemAE



Memory module of MemAE



MemAE [1/3]



MemAE [2/3]

Description

$$\hat{\mathbf{z}} = \mathbf{w}\mathbf{M} = \sum_{i=1}^{N} w_i \mathbf{m}_i, \tag{3}$$

$$w_i = \frac{\exp(d(\mathbf{z}, \mathbf{m}_i))}{\sum_{j=1}^{N} \exp(d(\mathbf{z}, \mathbf{m}_j))}, \tag{4}$$

$$d(\mathbf{z}, \mathbf{m}_i) = \frac{\mathbf{z}\mathbf{m}_i^{\mathsf{T}}}{\|\mathbf{z}\|\|\mathbf{m}_i\|}. \tag{5}$$

$$\hat{w}_i = h(w_i; \lambda) = \begin{cases} w_i, & \text{if } w_i > \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

$$\hat{w}_i = \frac{\max(w_i - \lambda, 0) \cdot w_i}{|w_i - \lambda| + \epsilon}, \tag{7}$$

Calculation

$$d(\mathbf{z}, \mathbf{m}_i) = \frac{\mathbf{z}\mathbf{m}_i^{\mathsf{T}}}{\|\mathbf{z}\|\|\mathbf{m}_i\|}. \tag{5}$$

$$w_i = \frac{\exp(d(\mathbf{z}, \mathbf{m}_i))}{\sum_{j=1}^{N} \exp(d(\mathbf{z}, \mathbf{m}_j))}, \tag{4}$$

$$\hat{w}_i = h(w_i; \lambda) = \begin{cases} w_i, & \text{if } w_i > \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

Hard Shrinkage

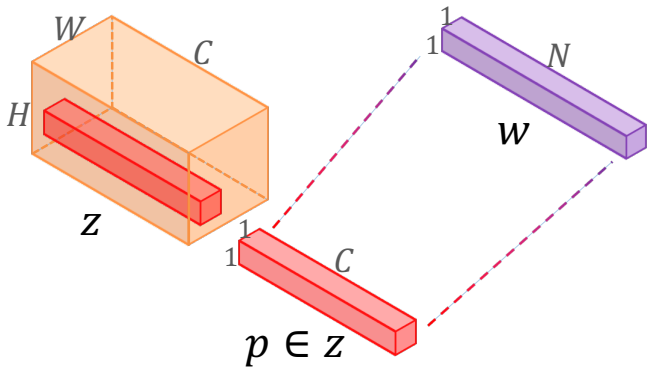
$$\hat{w}_i = \frac{\max(w_i - \lambda, 0) \cdot w_i}{|w_i - \lambda| + \epsilon}, \tag{7}$$

Hard Shrinkage
(trick for backprop.)

$$\hat{\mathbf{z}} = \mathbf{w}\mathbf{M} = \sum_{i=1}^{N} w_i \mathbf{m}_i, \tag{3}$$

Forged normal feature

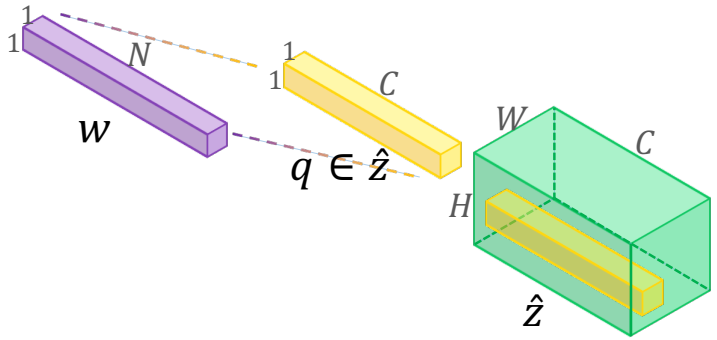
MemAE [3/3]



Dimension Expansion

$$d(\mathbf{z}, \mathbf{m}_i) = \frac{\mathbf{z}\mathbf{m}_i^{\mathsf{T}}}{\|\mathbf{z}\|\|\mathbf{m}_i\|}. \tag{5}$$

$$w_i = \frac{\exp(d(\mathbf{z}, \mathbf{m}_i))}{\sum_{j=1}^{N} \exp(d(\mathbf{z}, \mathbf{m}_j))}, \tag{4}$$



$$\hat{w}_i = h(w_i; \lambda) = \begin{cases} w_i, & \text{if } w_i > \lambda, \\ 0, & \text{otherwise}, \end{cases} \tag{6}$$

$$\hat{w}_i = \frac{\max(w_i - \lambda, 0) \cdot w_i}{|w_i - \lambda| + \epsilon}, \tag{7}$$

$$\hat{\mathbf{z}} = \mathbf{w}\mathbf{M} = \sum_{i=1}^{N} w_i \mathbf{m}_i, \tag{3}$$

Dimension Reduction

Appendix B

Source code for AD

Ganomaly

GANomaly-TF Public

Pin | Unwatch 3 | Fork 1 | Starred 10

master | 1 branch | 0 tags | Go to file | Add file | Code

YeongHyeon Update README.md fe3dd49 on Dec 2, 2020 34 commits

figures	readme	3 years ago
source	tf.v1	3 years ago
LICENSE	license	4 years ago
README.md	Update README.md	3 years ago
run.py	source refactoring	3 years ago

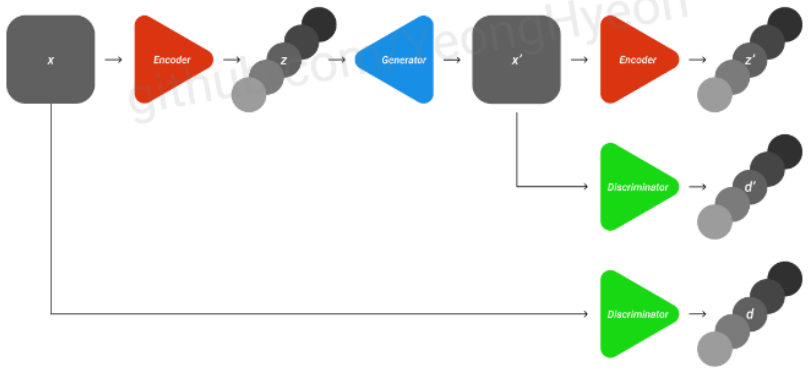
README.md

[TensorFlow] GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training

TensorFlow implementation of GANomaly with MNIST dataset.
PyTorch Version is also implemented.

Summary

GANomaly architecture



Simplified GANomaly architecture.

About

TensorFlow implementation of GANomaly (with MNIST dataset)

generative-adversarial-network
mnist-dataset
anomaly-detection
generative-neural-network

Readme
MIT license
Activity
10 stars
3 watching
1 fork

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Suggested Workflows

Based on your tech stack

Actions Importer — Set up
Automatically convert CI/CD files to YAML for GitHub Actions.

Django — Configure

f-AnoGAN

f-AnoGAN-TF Public

Pin | Unwatch 2 | Fork 1 | Starred 8

master | 1 branch | 0 tags | Go to file | Add file | Code

YeongHyeon readme 594542f on Nov 18, 2020 22 commits

figures	readme	3 years ago
source	source	3 years ago
LICENSE	license	3 years ago
README.md	readme	3 years ago
run.py	source	3 years ago

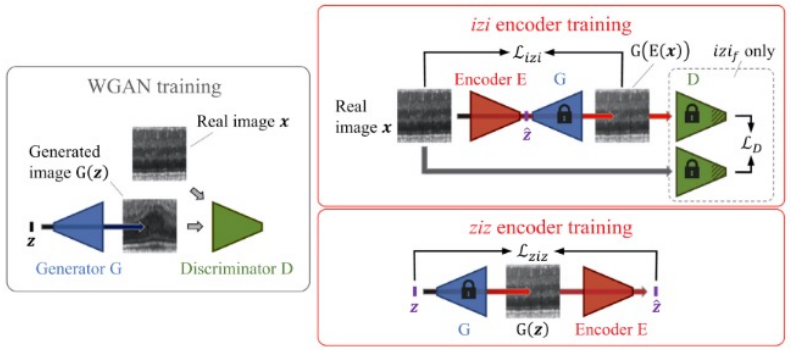
README.md

f-AnoGAN: Fast unsupervised anomaly detection with generative adversarial networks

TensorFlow implementation of f-AnoGAN with MNIST dataset [1].
The base model WGAN is also implemented with TensorFlow.

Summary

f-AnoGAN architecture



The architecture of f-AnoGAN [1].

About

TensorFlow implementation of f-AnoGAN (with MNIST dataset)

generative-adversarial-network
mnist-dataset
anomaly-detection
generative-neural-network

Readme
MIT license
Activity
8 stars
2 watching
1 fork

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

Python 100.0%

Suggested Workflows

Based on your tech stack

Actions Importer — Set up
Automatically convert CI/CD files to YAML for GitHub Actions.

SLSA Generic generator — Configure

MemAE

MemAE-TF2 Public

Unpin | Unwatch 3 | Fork 5 | Starred 21

master | 1 branch | 0 tags | Go to file | Add file | Code

YeongHyeon Merge pull request #1 from zeexan-dev/patch-2 ... 530a3c7 last month 6 commits

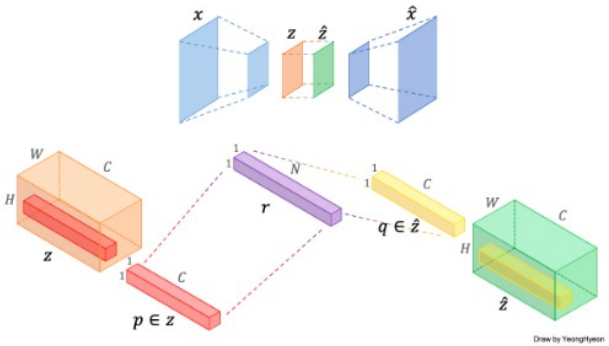
figures	figure	3 years ago
source	Update tf_process.py	last month
LICENSE	license	3 years ago
README.md	readme	3 years ago
run.py	source	3 years ago

README.md

[TensorFlow 2] Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection

TensorFlow implementation of Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection. [PyTorch Version] [TensorFlow 1 Version]

Architecture



Architecture of MemAE.

About

TensorFlow implementation of "Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection"

deep-learning tensorflow mnist-dataset convolutional-neural-networks convolutional-neural-network augmentation anomaly-detection memorizing memory-augmentation tensorflow2

Readme
MIT license
Activity
21 stars
3 watching
5 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package