A. Architectures of SQUID

Our SQUID consists of an encoder, a student (main) generator, a teacher generator, and a discriminator. All of the network architectures are built with plain convolution, batch normalization, and ReLU activation layers only. The architecture details of the encoder are shown in Table 3. For an input radiography image (sized of 128×128), we first divide it into 2×2 non-overlapping patches (sized of 64×64). The encoder then extracts the patch features.

As mentioned in §3.1, the student and teacher generators were constructed identically. The only difference is that additional Memory Matrices are placed in the student generator. The architecture details of the student generator are shown in Table 4. Skip connections from the encoder are only enabled at such levels that Memory Matrices are used. After the last Memory Matrix, the non-overlapping patches are put back as a whole for further reconstruction.

As shown in Table 5, the discriminator was constructed in a more lightweight style. Note that the images are discriminated at their full resolution (*i.e.* 128×128) rather than in patches.

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

B. Additional Results

B.1. Extensive Ablation Studies

In this section, we ablate three components in SQUID to fully validate their necessity and effectiveness.

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**

(1) Convolutional vs. Transformer Layers: In our proposed in-painting block, a transformer layer is used to aggregate the encoder extracted patch features, and the Memory Queue augmented "normal" features. However, one may wonder if a simple convolution layer can also suffice. We conducted experiments by replacing the transformer layer with a convolutional layer while preserving other structures.

(2) Soft vs. Hard Masked Shortcuts: In our proposed masked shortcut, skipped and in-painted features are aggregated using a binary gating mask. The intuitive question is whether such "hard" gating is necessary and a weighted "soft" addition can also achieve comparable results. To this end, instead of following Eq. 2, we conducted experiments by aggregating the patch features $\mathcal{F}$ through:

$$\mathcal{F}' = (1 - \rho) \cdot \mathcal{F} + \rho \cdot \texttt{inpaint}(\mathcal{F}), \tag{3}$$

where ρ was set to 95%, same as the best setting in SQUID.

(3) Pixel-level vs. Feature-level In-painting: As discussed in §3.3, raw images usually contain larger noise and artifacts than features, so we proposed to achieve the in-painting at the feature level rather than at the image level [41, 54, 87]. To validate our claim, we have conducted experiments on carrying out the in-painting at the pixel level. Instead of using a transformer layer to in-paint the extracted patch features, we randomly zeroed out parts of the input patches with 25% probability and let SQUID in-paint the distorted input images. All other settings and objective functions remain unchanged.

Summary: The results of the above three additional ablative experiments are presented in Table 6. Without using the transformer layer, masked shortcut, and feature-level in-painting as proposed, the AUC, Acc, and F1 scores decreased by at least 8%, 4%, and 3%, respectively, compared with the full SQUID setting.

B.2. Patch-MemAE

MemAE [17] with Memory Matrix is the primary baseline that we considered in this work. To further verify the effectiveness of our proposed space-aware setting, we trained additional MemAE models on patches segmented from different spatial location of input images. These multiple space-specific models were trained separately with

Table 7. We apply space-specific strategy to one of the strongest counterparts (MemAE [17]). In addition, the ensemble of spatial-aware models demands a *higher* degree of computational costs (4× more than ours), while our work proposed to encode this spatial information into the feature dictionary, ultimately requiring only one model—its efficiency is pronounced.

Method	AUC (%)	Acc (%)	F1 (%)
MemAE [17]	77.8±1.4	56.5±1.1	82.6±0.9
Patch-MemAE (Δ)	↑0.5	↑18.5	↓1.3
Full SQUID	**87.6±1.5**	**80.3±1.3**	**84.7±0.8**

their unique space-specific patches and were then evaluated through an ensemble style to compare with our SQUID. The results are reported in Table 7.

The results of the this experiment indicate that although improvements can be observed on AUC and Acc, such space-specific ensemble upgrade still performs inferior than SQUID. Moreover, we found such ensemble of models demands a much *higher* degree of computational costs (*4*× more than ours), while in our work, we proposed to encode this spatial information into the feature dictionary, ultimately requiring only one model. Both effectiveness and efficiency are pronounced.

C. Creating DigitAnatomy

The pseudocode of creating our new benchmark dataset (DigitAnatomy in §4.1) is provided in Algorithm 1. In practice, we have implemented the algorithm into an off-the-shelf data loader that can be amended to many other different datasets (*e.g.* SVHN, CIFAR, ImageNet).

Algorithm 1 Creating DigitAnatomy

```
# a function to pick random digit instances
def pick_random(class, single_digits):
    # random pick an image with size: [28, 28]
    pick_digit = random.choice(single_digits[class])
    return pick_digit


# load MNIST digits with shape: [10, 1000, 28, 28]
single_digits = load_MNIST()

# all possible conditions
conditions = ['normal', 'missing',\
              'misorder', 'flipped', 'novel']

output = torch.zeros(3, 28, 3, 28)

# loop over digit 1-9 in order
for idx in range(1,10):

  # randomly pick a condition
  condition = random.choice(conditions)

  if condition == 'normal':
    digit = pick_random(idx, single_digits)
  # anatomy of missing digit
  elif condition == 'missing':
    digit = torch.zeros(28,28)
  # anatomy of disorder digit
  elif condition == 'misorder':
    ridx = randrom.randint(1,10)
    digit = pick_random(ridx, single_digits)
  # anatomy of flipped digit
  elif condition == 'flipped':
    digit = pick_random(idx, single_digits)
    digit = digit[::-1,::-1]
  # anatomy of novel digit
  elif condition == 'novel':
    digit = pick_random(0, single_digits)

  output[idx // 3, :, idx % 3, :] = digit

# combine all patches together
output = output.view(28 * 3, 28 * 3)
```

D. Visualization Results

D.1. Visualizations on DigitAnatomy

More reconstruction results of SQUID and the compared methods [1, 17, 61] are shown in Figure 11. Our observations from these additional results are aligned with the ones discussed in §5.1. SQUID can capture *every* appearing anomaly (highlighted in light blue) in the images and augment them back to the normal closest forms. On the contrary, although MemAE restores the normal digits the best, it is limited in detecting a few anomaly types (*e.g.* misordered and missing digits). Ganomaly is not able to perfectly recover the normal digits and also cannot generate meaningful reconstructions on the abnormal ones. f-AnoGAN, on the other hand, memorizes and generates an exemplary normal pattern that fails to respond to different inputs.

D.2. Visualizations on Chest Radiography

Figure 12 and Figure 13 show more reconstruction results of our SQUID on the ZhangLab Chest X-ray and Stanford CheXpert datasets. We observed that our method is capable of translating the input image to its "normal" counterpart and assigning larger anomaly scores to abnormal cases.

When inputting normal images, SQUID will try to reconstruct the inputs as well as possible. Due to the usage of memory modules, our framework could hardly degenerate to function as an identity mapping from inputs to outputs. Therefore, the reconstruction of normal inputs cannot perfectly recover every single detail.

When inputting abnormal images, SQUID will make larger impacts by combining previously seen normal features together into such abnormal ones. Since the generator is not trained on such hybrid features, the reconstruction results could demonstrate more obvious artifacts and blurs.

After our framework converges, the optimized discriminator can perceptually capture such inconsistencies between reconstructed normal and abnormal images and achieve anomaly detection.

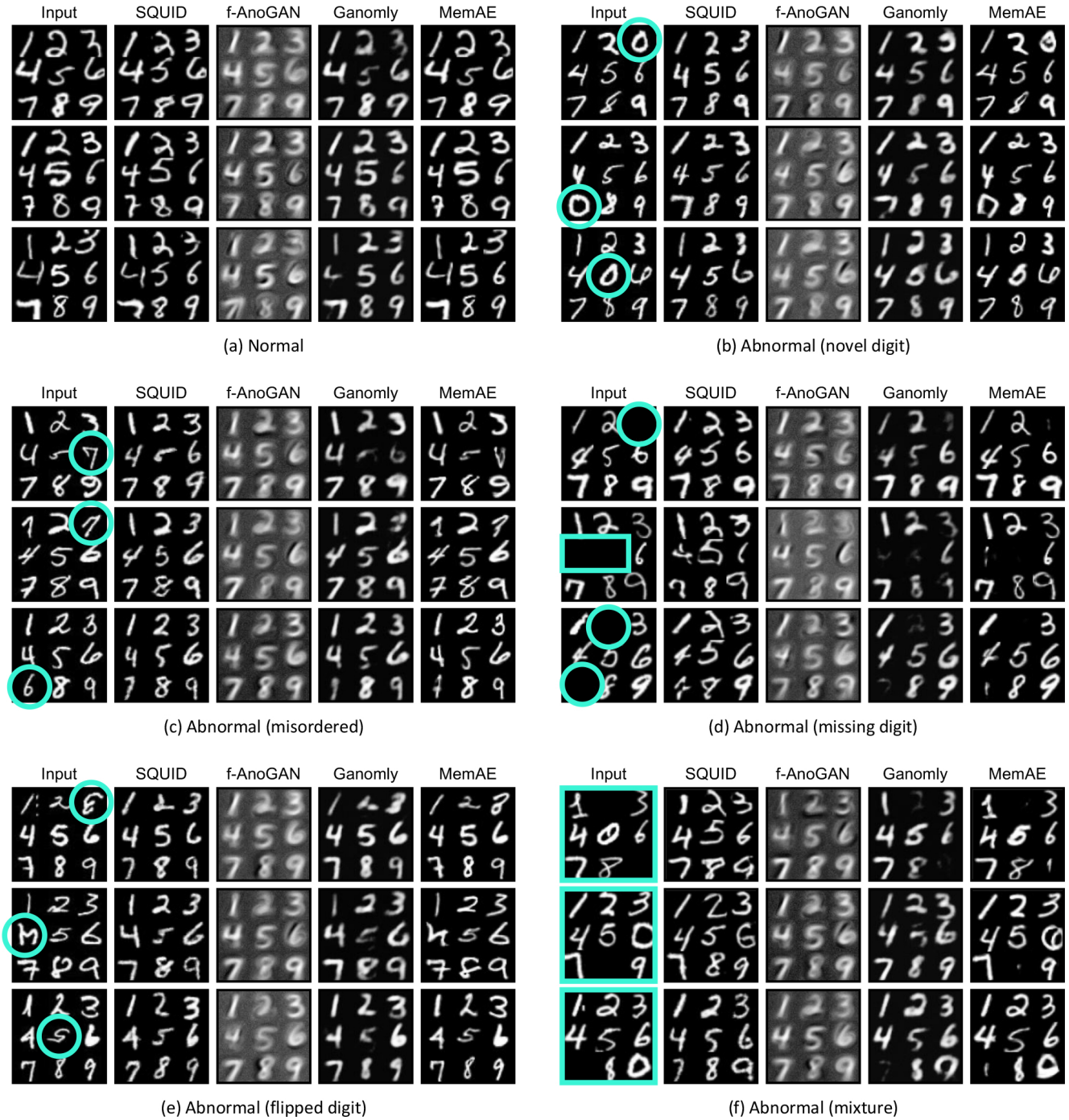

Figure 11. Comparisons of reconstruction results on DigitAnatomy of our SQUID, f-AnoGAN [61], Ganomaly [1], and MemAE [17]. Anomalies are highlighted in light blue.

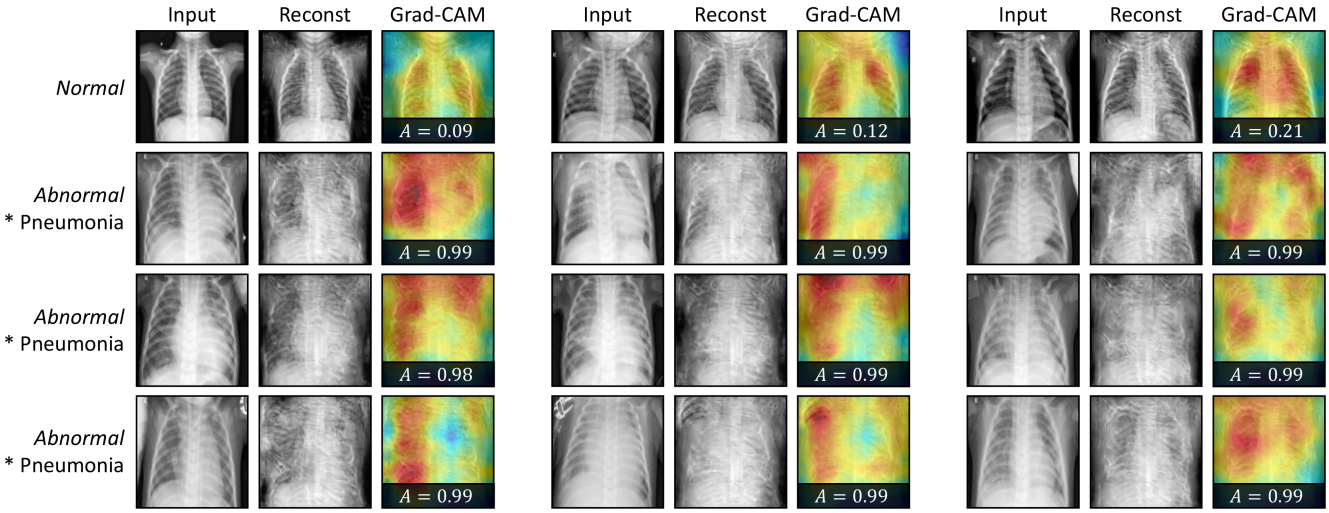

Figure 12. Reconstruction results of SQUID on the ZhangLab Chest X-ray dataset. The corresponding Grad-CAM heatmaps along with anomaly scores are shown as well.

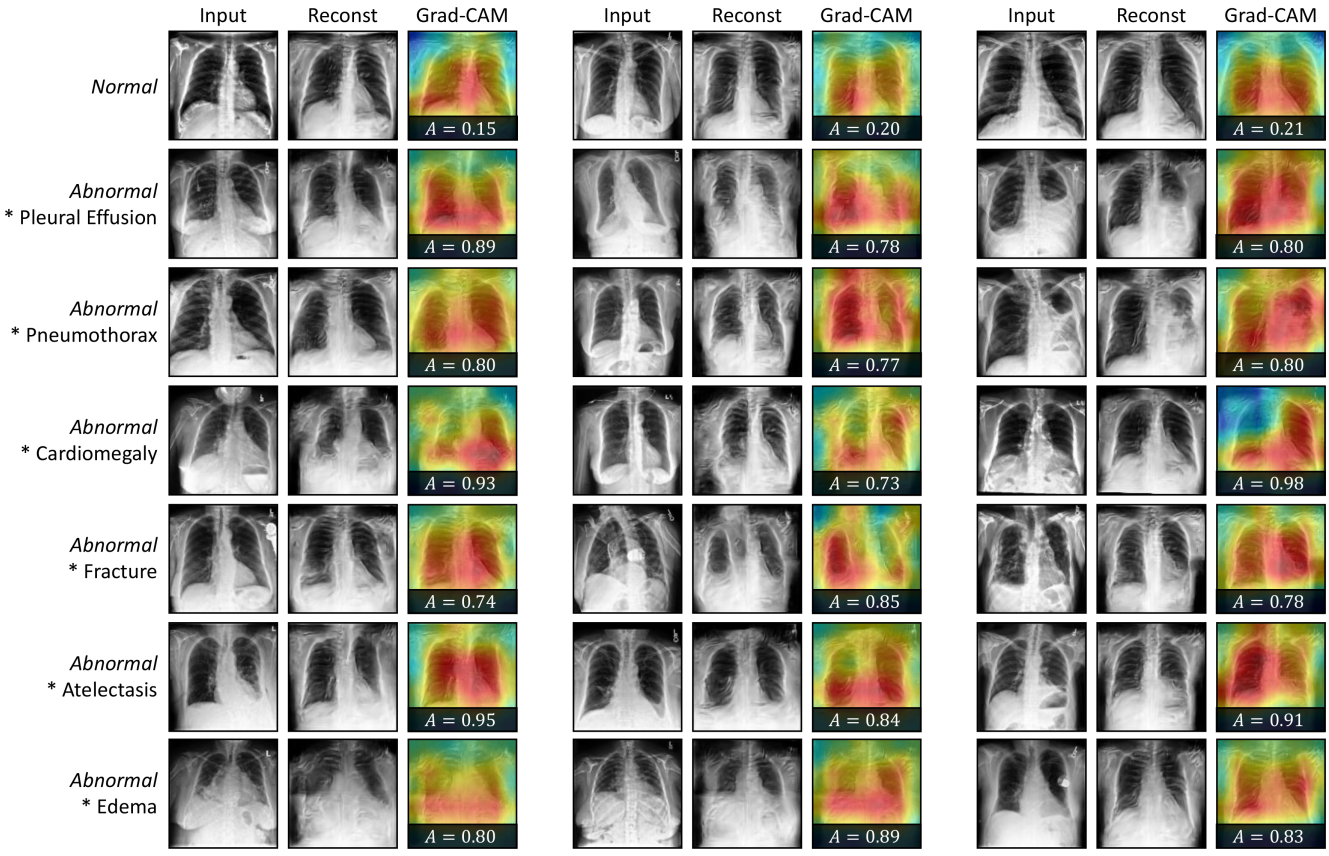

Figure 13. Reconstruction results of SQUID on the Stanford CheXpert dataset. Different disease types are separated into different rows. The corresponding Grad-CAM heatmaps along with anomaly scores are shown as well.