Supplementary Material

Network architecture In the tables below, we show the detailed architecture of network modules of HA-GAN, including G^A, G^L, G^H, D^L, D^H, E^H and E^G.

TABLE I. Architecture of the G^A Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	1 × 1024
Dense	-	512×4×4×4
Conv3D	3×3×3, 1	512×4×4×4
GroupNorm+ReLU	-	512×4×4×4
Interpolation	-	512×8×8×8
Conv3D	3×3×3, 1	512×8×8×8
GroupNorm+ReLU	-	512×8×8×8
Interpolation	-	512×16×16×16
Conv3D	3×3×3, 1	256×16×16×16
GroupNorm+ReLU	-	256×16×16×16
Interpolation	-	256×32×32×32
Conv3D	3×3×3, 1	128×32×32×32
GroupNorm+ReLU	-	128×32×32×32
Interpolation	-	128×64×64×64
Conv3D	3×3×3, 1	64×64×64×64
GroupNorm+ReLU	-	64×64×64×64

TABLE II. Architecture of the G^L Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	64×64×64×64
Conv3D	3×3×3, 1	32×64×64×64
GroupNorm+ReLU	-	32×64×64×64
Conv3D	3×3×3, 1	16×64×64×64
GroupNorm+ReLU	-	16×64×64×64
Conv3D	3×3×3, 1	1×64×64×64
Tanh	-	1×64×64×64

TABLE III. Architecture of the G^H Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	64×64×64×64
Interpolation	-	64×128×128×128
Conv3D	3×3×3, 1	32×128×128×128
GroupNorm+ReLU	-	32×128×128×128
Interpolation	-	32×256×256×256
Conv3D	3×3×3, 1	1×256×256×256
Tanh	-	1×256×256×256

TABLE IV. Architecture of the E^H Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	1×32×256×256
Conv3D	4×4×4, 2	32×16×128×128
GroupNorm+ReLU	-	32×16×128×128
Conv3D	3×3×3, 1	32×16×128×128
GroupNorm+ReLU	-	32×16×128×128
Conv3D	4×4×4, 2	64×8×64×64
GroupNorm+ReLU	-	64×8×64×64

TABLE V. Architecture of the E^G Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	64×64×64×64
Conv3D	4×4×4, 2	32×32×32×32
GroupNorm+ReLU	-	32×32×32×32
Conv3D	4×4×4, 2	64×16×16×16
GroupNorm+ReLU	-	64×16×16×16
Conv3D	4×4×4, 2	128×8×8×8
GroupNorm+ReLU	-	128×8×8×8
Conv3D	4×4×4, 2	256×4×4×4
GroupNorm+ReLU	-	256×4×4×4
Conv3D	4×4×4, 1	1024×1×1×1

TABLE VI. Architecture of the D^L Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	1×64×64×64
Conv3D	4×4×4, 2	32×32×32×32
SpectralNorm+LeakyReLU	-	32×32×32×32
Conv3D	4×4×4, 2	64×16×16×16
SpectralNorm+LeakyReLU	-	64×16×16×16
Conv3D	4×4×4, 2	128×8×8×8
SpectralNorm+LeakyReLU	-	128×8×8×8
Conv3D	4×4×4, 2	256×4×4×4
SpectralNorm+LeakyReLU	-	256×4×4×4
Conv3D	4×4×4, 1	1×1×1×1
Reshape	-	1

TABLE VII. Architecture of the D^H Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	1×32×256×256
Conv3D	4×4×4, 2	16×16×128×128
SpectralNorm+LeakyReLU	-	16×16×128×128
Conv3D	4×4×4, 2	32×8×64×64
SpectralNorm+LeakyReLU	-	32×8×64×64
Conv3D	4×4×4, 2	64×4×32×32
SpectralNorm+LeakyReLU	-	64×4×32×32
Conv3D	2×4×4, 2	128×2×16×16
SpectralNorm+LeakyReLU	-	128×2×16×16
Conv3D	2×4×4, 1	256×1×8×8
SpectralNorm+LeakyReLU	-	256×1×8×8
Conv3D	1×4×4, 1	512×1×4×4
SpectralNorm+LeakyReLU	-	512×1×4×4
Conv3D	1×4×4, 1	128×1×1×1
SpectralNorm+LeakyReLU	-	128×1×1×1
Dense	-	64
SpectralNorm+LeakyReLU	-	64
Dense	-	32
SpectralNorm+LeakyReLU	-	32
Dense	-	1

Datasets and Training Details For the COPDGene dataset, CT scans are acquired using multi-detector CT scanners (at least 16 detector channels). Volumetric CT acquisitions are obtained on full inspiration (200mAs). The initial voxel size is typically $0.68mm \times 0.68mm \times 0.54mm$. The dataset is available at `https://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/study.cgi?study_id=phs000179.v6.p2` For the GSP dataset, all imaging data were collected on matched 3T Tim Trio scanners (Siemens Healthcare, Erlangen, Germany) at Harvard University and Massachusetts General Hospital using the vendor-supplied 12-channel phased-array head coil. Structural data included a high-resolution (1.2mm isotropic) multi-echo T1-weighted magnetization-prepared gradient-echo image. The dataset is available at `https://dataverse.harvard.edu/dataverse/GSP` For fair comparison between different baseline models, we use the official implementation whenever it is available. We also provide training and validation curves in Fig. 1 and Fig. 2.

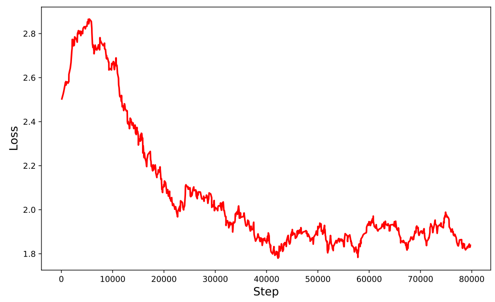

Fig. 1. Training loss curve of the generator for HA-GAN.

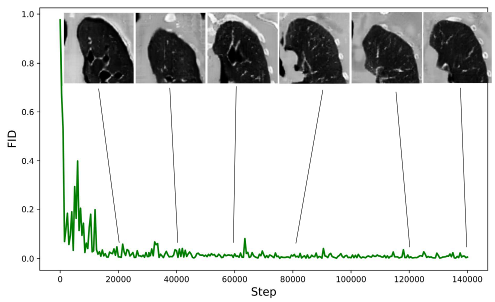

Fig. 2. Validation curve during training for HA-GAN. FID score is used as the validation metric. The images cropped on left upper lobe shown above demonstrate the quality of random image synthesis during training.

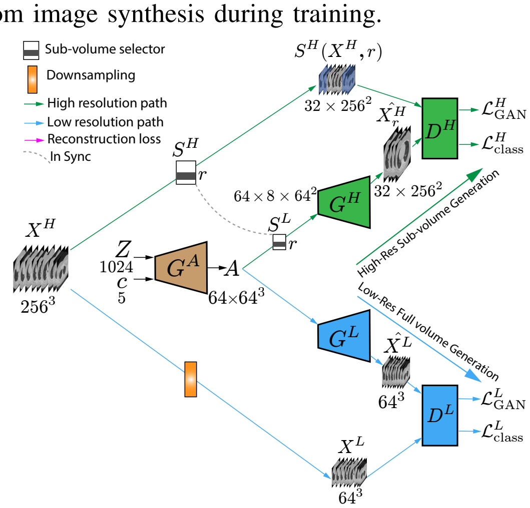

Fig. 3. Architecture of conditional HA-GAN for data augmentation (encoder is hidden here to improve clarity).

Data Augmentation for Supervised Learning In this section, we provide more details about our experiment of data augmentation for supervised learning. In Fig. 3, we show the architecture of conditional HA-GAN for data augmentation. Compared with original HA-GAN, two modifications are made to the original HA-GAN architecture: 1) Besides the latent variable Z sampled from Gaussian distribution, the generator module $G^A(Z; c)$ also takes a 5-class one-hot code variable $c \sim p_c$ as input, which indicates which class the generated images should be. 2) The discriminator gives both a probability distribution over real/fake classification (same as original HA-GAN) and a probability distribution over the class labels $P(C|X)$. In this way, the discriminator also serves as auxiliary classifier for GOLD score. The network architecture of the 3D CNN used for supervised training is shown in VIII.

TABLE VIII. Architecture of the 3D CNN Network

Layer	Filter size, stride	Output size(C, D, H, W)
Input	-	$1\times128\times128\times128$
Conv3D	$3\times3\times3$, 1	$8\times128\times128\times128$
BatchNorm+ELU	-	$8\times128\times128\times128$
Conv3D	$3\times3\times3$, 2	$8\times64\times64\times64$
BatchNorm+ELU	-	$8\times64\times64\times64$
Conv3D	$3\times3\times3$, 1	$16\times64\times64\times64$
BatchNorm+ELU	-	$16\times64\times64\times64$
Conv3D	$3\times3\times3$, 1	$16\times64\times64\times64$
BatchNorm+ELU	-	$16\times64\times64\times64$
Conv3D	$3\times3\times3$, 2	$16\times32\times32\times32$
BatchNorm+ELU	-	$16\times32\times32\times32$
Conv3D	$3\times3\times3$, 1	$32\times32\times32\times32$
BatchNorm+ELU	-	$32\times32\times32\times32$
Conv3D	$3\times3\times3$, 1	$32\times32\times32\times32$
BatchNorm+ELU	-	$32\times32\times32\times32$
Conv3D	$3\times3\times3$, 2	$32\times16\times16\times16$
BatchNorm+ELU	-	$32\times16\times16\times16$
Conv3D	$3\times3\times3$, 1	$64\times16\times16\times16$
BatchNorm+ELU	-	$64\times16\times16\times16$
Conv3D	$3\times3\times3$, 1	$64\times16\times16\times16$
BatchNorm+ELU	-	$64\times16\times16\times16$
Conv3D	$3\times3\times3$, 2	$64\times8\times8\times8$
BatchNorm+ELU	-	$64\times8\times8\times8$
Conv3D	$3\times3\times3$, 1	$128\times8\times8\times8$
BatchNorm+ELU	-	$128\times8\times8\times8$
Conv3D	$3\times3\times3$, 2	$128\times4\times4\times4$
BatchNorm+ELU	-	$128\times4\times4\times4$
AvgPool	-	$128\times1\times1\times1$
Reshape	-	128
Dense	-	5

TABLE IX. Testing the impact of sub-volume multiplier

Multiplier Factor	Memory Usage (MB)
1/8	5961
1/4	10689
1/2	13185

Effect of Sub-volume selection on memory usage We split the baseline models by applying our proposed sub-volume selection method to the baseline models, and measured memory usage during training. The output size of the original generator for each model is set as 128^3, and we set the size of sub-volume as 32×128^2. The batch size is set as 2. The results are shown in Fig. 4. We can see that our proposed sub-volume selection method drastically reduces the memory demand during training for baseline methods.

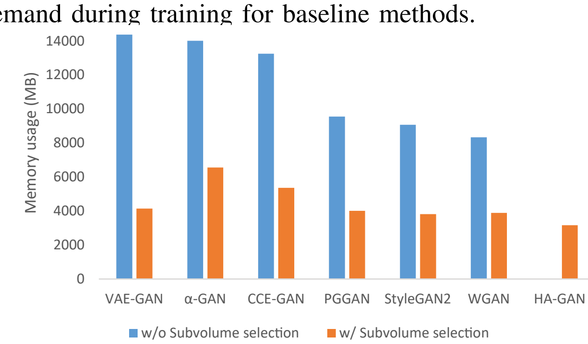

Fig. 4. Effect of Sub-volume selection on memory usage