Improving Nonalcoholic Fatty Liver Disease Classification Performance With Latent Diffusion Models

Romain Hardy[1,*], Ryan Mitchell[1], Joe Klepich[1], Steve Hall[1], Jericho Villareal[1], and Cornelia Ilin[1,†]

[1]School of Information, U.C. Berkeley

[*]First author
[†]Corresponding author: cornelia.ilin@berkeley.edu
[1]School of Information, U.C. Berkeley

July 18, 2023

Appendices

A Data Summary

Although the preprocessing transformations are the same for the diffusion and classification pipelines, there is a slight difference between the two approaches. In the diffusion pipeline, image preprocessing is applied dynamically during training, thus if a raw image appears in n mini-batches it will generate n cropped and resized images. In the classification pipeline, the preprocessing is fixed beforehand, such that each raw image corresponds to a single preprocessed image. The preprocessed dataset is saved and reused across all classification experiments so that we can objectively evaluate the impact of synthetic images on model performance.

B Methods Summary

B.1 Latent Diffusion Models

Latent diffusion models (LDMs) are variants of diffusion models that first encode images to a low-dimensional latent space representation using a pretrained encoder E, thus avoiding costly computations in pixel space. A visual diagram of a LDM is shown in Rombarch et al., 2021.[?] During training, an image x is fed through E to produce a latent vector z. The model then adds Gaussian noise to z according to a fixed Markov chain, producing z_T. The training objective of the model is to learn the reverse process, that is, how to denoise z_T back to its original state z. Mathematically, this objective can be expressed as:

$$L_{\text{LDM}} := \mathbb{E}_{E(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta(z_t, t, \tau_\theta(y))\|_2^2\right], \tag{1}$$

where ϵ_θ is a denoising autoencoder (shown in the figure as a UNet model) and τ_θ is an encoder that feeds conditioning inputs y (i.e. text, semantic maps, and class labels) to ϵ_θ via concatenation and cross-attention layers. Finally, the recovered latent vector $\tilde{z}$ can be projected back to pixel space through the use of a pretrained decoder D.

C Supplementary Figures

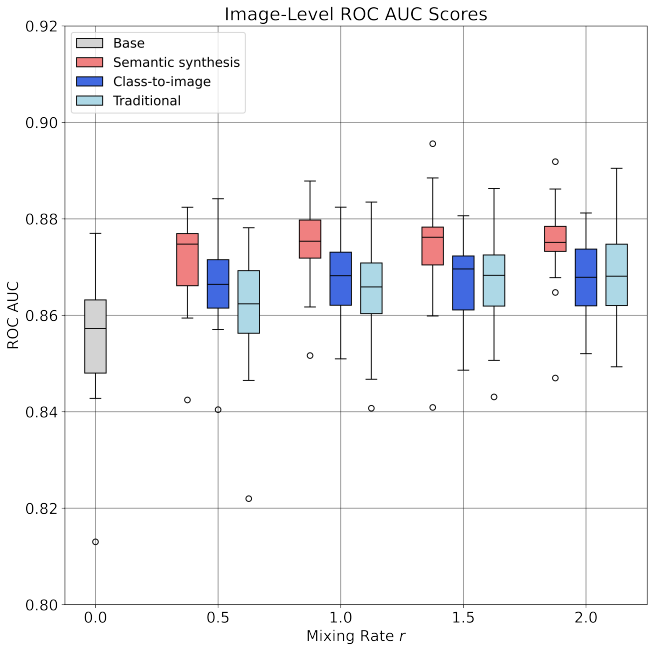

Figure 1: **Image-level out-of-sample NAFLD classification performance – Sensitivity test 1.** These results are generated using the same ResNet-50 model as in Figure **??**A. However, instead of concatenating the out-of-fold predictions and computing the ROC AUC, we average the ROC AUC across the five folds to produce a final fold-averaged score.

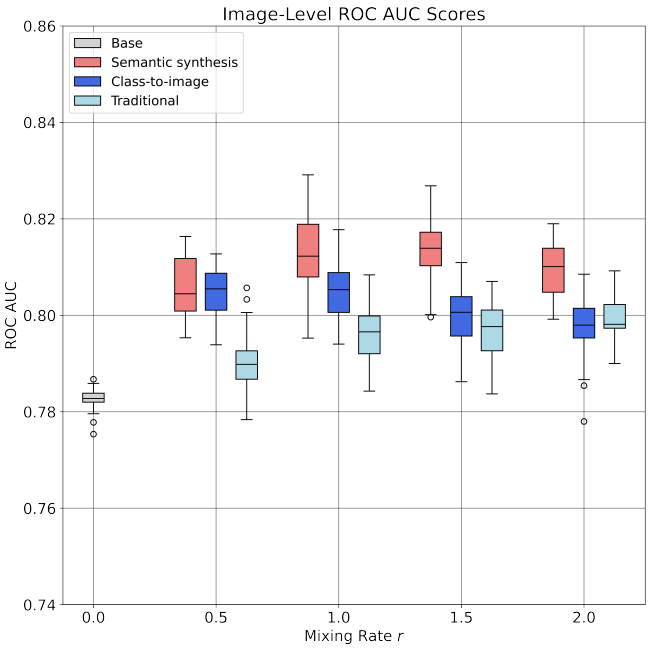

Figure 2: **Image-level out-of-sample NAFLD classification performance – Sensitivity test 2.** This plot shows results for the same data input scenarios as in Figure **??**A, except that we freeze every hidden layer in the ResNet-50 backbone.

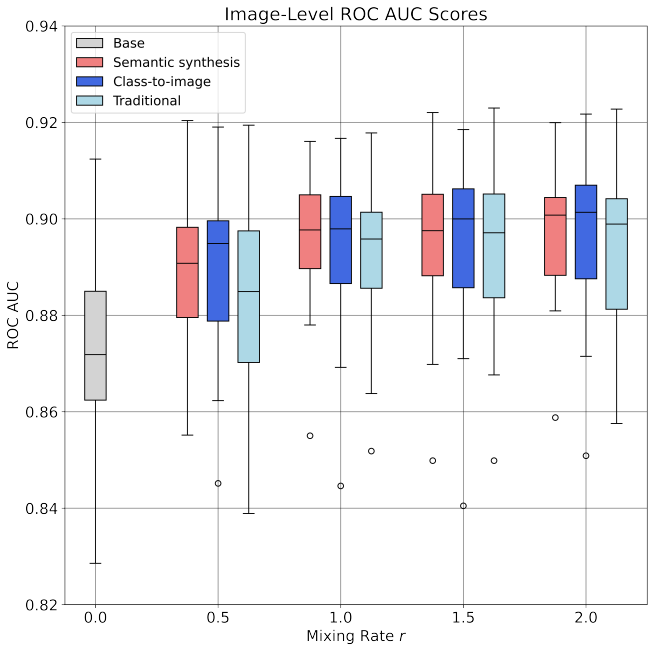

Figure 3: **Image-level out-of-sample NAFLD classification performance – Sensitivity test 3.** This plot shows results for the same data input scenarios as in Figure **??**A, except that we unfreeze every hidden layer in the ResNet-50 backbone.

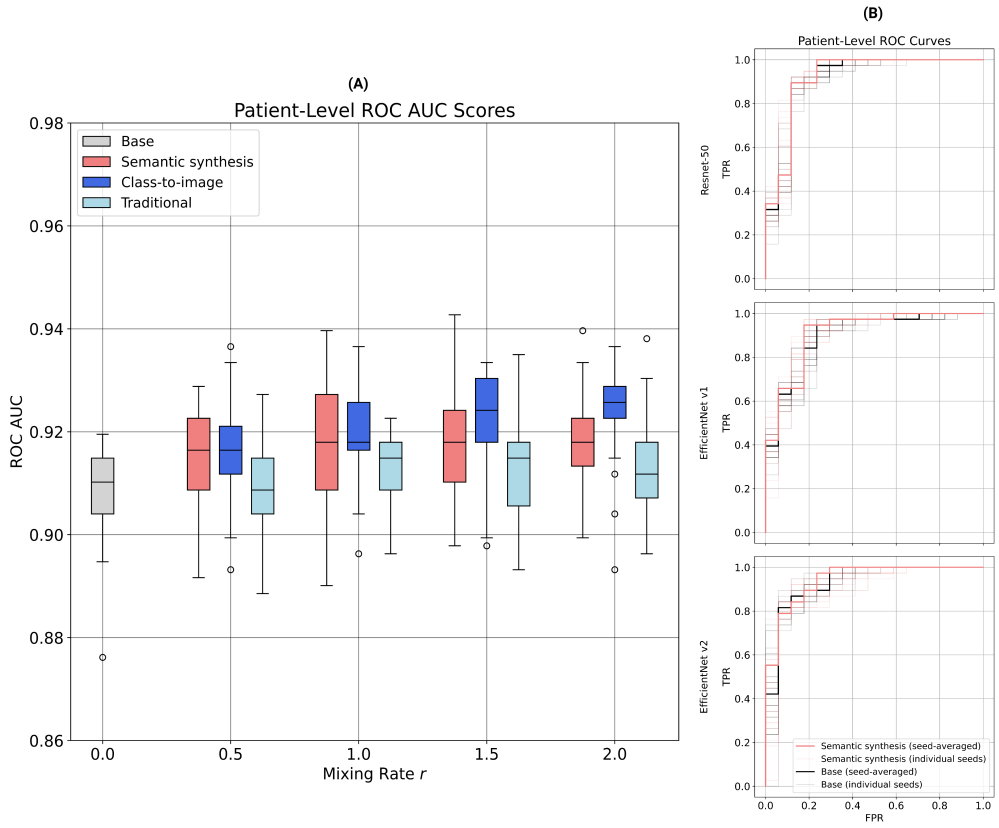

Figure 4: **Patient-level out-of-sample NAFLD classification performance – Sensitivity test 4**. **(A)** Box plots of the patient-level ROC AUC as a function of the mixing rate r. **(B)** Patient-level ROC curves for three families of CNN classifiers: ResNet-50 (top), EfficientNet v1 (middle), and EfficientNet v2 (bottom). The average ROC AUC for the base and synthetically augmented models are as follows: ResNet-50, 0.907 versus 0.918; EfficientNet v1, 0.896 versus 0.908; EfficientNet v2, 0.932 versus 0.927.