Evaluating the Impact of Deep Learning Data Augmentation Methods on a Medical Imagery Classification Task

Quentin Le Roux

Abstract. We evaluate the performance of three Deep Learning data augmentation methods on the performance of a Convolutional Neural Network classifier trained on labelled medical images. In a majority of cases, we find that we yield relatively better results than when using geometric data augmentation or no augmentation at all[1].

1 Introduction

MedMNIST[7] is a collection of labelled, biomedical image datasets standardized to fit the MNIST format[9] (See Table 1). Being labelled, MedMNIST datasets lend themselves to surpervised classification tasks and the comparison of classifier algorithms – one of the key goals of the MedMNIST's creators[7].

The taxonomy of data augmentation techniques is broadly split between two families: image manipulations (e.g. geometric transformations like rotation) and Deep Learning (DL) approaches.

In this study, we explore the use of DL data augmentation techniques – either Variational Autoencoders (VAEs) or Generative Adversarial Networks (GANs) – to assess whether they improve the performance of a classifier compared to using geometric data augmentation or no augmentation at all. To test this, we retain 8 MedMNIST datasets (See Table 1).

Table 1. MedMNIST datasets retained for this study

MNIST Name	Type (Fig. with montage)	Classification	Train size	Val. size	Test size
Path	Colon pathology (Fig. 1)	Multi (9)	89,996	10,004	7,180
Derma	Dermatoscope (Fig. 2)	Multi (7)	7,007	1,003	2,005
Oct	Retinal OCT (Fig. 3)	Multi (4)	97,477	10,832	1,000
Pneumonia	Chest X-ray (Fig. 4)	Binary (2)	4,708	524	624
Breast	Breast Ultrasound (Fig. 5)	Binary (2)	546	78	156
OrganA	Axial Abdo. CT (Fig. 6)	Multi (11)	34,581	6,491	17,778
OrganC	Coronal Abdo. CT (Fig. 7)	Multi (11)	13,000	2,392	8,268
OrganS	Sagittal Abdo. CT (Fig. 8)	Multi (11)	13,940	2,452	8,829

2 Methodology

2.1 A Baseline Classifier

We use a Convolutional Neural Network (CNN) Pytorch architecture[13][14] provided by the MedMNIST team on their GitHub page[8]. The CNN architecture and hyperparameters are fixed to control for each modelization factor besides data augmentation

[1] All code and results used in this document are available on Google Colab: `https://colab.research.google.com/drive/13jawmMq8KPiW5dQ9c3UGsM2b3fMO8Zjl?usp=sharing`

methods. Though the CNN's input and hidden layers are identical across dataset implementations, the output layer differ due to varying numbers of dataset classes (See Fig. 9 for the MedMNIST CNN architecture summary in the PathMNIST case).

For each case, the CNN is trained for 20 epochs with Stochastic Gradient Descent[1] and a 0.001 learning rate. The loss is computed via cross-entropy (binary cross-entropy in a binary classification case). Batch size is set to 64 when DL data augmentation is used, 128 otherwise (the MedMNIST's default). We also reuse the MedMNIST team's custom evaluator object, which provides two output metrics to assess model performance: accuracy and area-under-the-curve (AUC).

2.2 Imbalancedness, Augmentation, and Weighted Random Sampling

The retained datasets display class imbalance (See Fig. 10 to 17) and are unequally populated (e.g. the PathMNIST is an order of magnitude larger than DermaMNIST). This may impact classifier generalization and convergence towards an optimum[15]. In such a case, image data augmentation techniques can help improve models' generalization capability by reducing overfitting to the training data[18].

Besides data augmentation, and to further reduce the effect of class imbalance, we will also add to our process PyTorch's Weighted Random Sampler (WRS). This allows us to deal with class imbalance at the dataloader level. We will evaluate each CNN setup with and without the WRS method.

2.3 Augmenting Dtasets: Geometric Approaches

We implement setups with geometric data augmentation as part of our comparison process. To do so, we rely on Pytorch's Transforms class at the dataloader level.

In the case of a black-and-white image dataset, the corresponding geometric data augmentation is a 3-step pipeline: random inversion, random sharpness adjustment (with a factor of 2), and random horizontal flipping.

In the case of a color image dataset, the geometric data augmentation is a 2-step pipeline: color jittering (with parameters 0.5 brightness and 0.3 hue) and random rotation (up to 180 degrees). Each pipeline ends with a normalization step.

2.4 Augmenting Datasets: VAE and GAN Approaches

The Manifold Hypothesis states that high-dimensional data are representations of lower dimensional objects[4]. VAEs assume that a random process involving a continuous random variable has generated the data of a given dataset. The latent space, or representation of the data, can be learned via an encoder and decoder networks[2]. Meanwhile, GANs generate new data using adversarial training, relying on a generator and discriminator networks to learn how to produce new data[6]. As Deep Convolutional GANs (DCGANs) have been shown to improve medical data classification[5][20], we are interested in implementing both types of methods for our DL data augmentation setups.

We settle on 3 methods: Conditional VAE, Joint VAE, and Conditional DCGAN.

Conditional Variational Autoencoders learn a latent representation of a dataset where the underlying random process is informed by the data classes [16]. Conditional VAEs use data classes as input for both decoder and encoder (See Fig. 18 for the representation of a Conditional VAE, and Fig. 19 and 20 for our implementation of the decoder and encoder in the PathMNIST case). Instead of learning the prior distribution $P_\theta(z)$ of the latent space z (with the data x being generated by the distribution $P_\theta(x|z)$), a Conditional VAE learns $P_\theta(z|y)$ with y the data's labels conditioning the latent space. As such, Conditional VAEs allow us to generate new labelled data.

Joint Variational Autoencoders learn both a continuous and discrete representation of a dataset[3]. As such, a high-dimensional dataset can be represented in a low-dimensional latent space with continuous and discrete elements. Thus, a Joint VAE learns the prior distribution $P_\theta(z, c)$ of the latent continuous space z and discrete/categorical space c with the data x being generated by the distribution $P_\theta(x|z, c)$ (See Fig. 21 for the representation of a Joint VAE, and Fig. 22 and 23 for the decoder and encoder architectures in the PathMNIST case).

A Joint VAE may help capture medical images' characteristics that a purely continuous latent space might not. Indeed, medical images are obtained in strictly controlled environments where some variables are discrete (e.g. the left-right orientation of a CT scan (discrete) versus the contrast of a X-ray (continuous)).

The main drawback is the absence of conditioning on classes. We can either train one Joint VAE per dataset class or train a single Joint VAE for a whole dataset. The latter case, though less computationally expensive, implies that we do not have the ability to categorically condition the underlying generative process.

Conditional Generative Adversarial Networks are similar to Conditional VAEs in that they generate data conditioned on a dataset's classes[10] (See Fig. 24 for the representation of a Conditional GAN, and Fig. 25, and 26 for the discriminator and generator architectures in the PathMNIST case).

Conditional GANs rely on using labels as inputs to guide the generator and discriminator's learning and the subsequent generative process [11][12][17]. The discriminator is evaluated on the similarity between generated and real data, and the match between the generated image's predicted and true labels. The rationale behind using a Conditional GAN is similar to that of the Conditional VAE: we want to generate new labelled data as part of a data augmentation process.

VAE and GAN Training Setup

Our two VAEs rely on existing implementations available online[19]. We use the TorchFusion library for the Conditional GAN [12].

We train the two VAEs for 200 epochs (with early stopping), and the Conditional GAN for 100. The Conditional VAE is trained with a latent space of dimension 150, the Joint VAE with continuous and discrete latent spaces of dimensions 100 and 50, and the Conditional GAN with a latent space of dimension 128.

Data Augmentation Setup

As a result of training (See Fig. 27 for the three methods' loss per epoch in the PathMNIST case), we generate new images for each retained MedMNIST datasets. We only augment the training sets; validation and test sets are untouched.

The Conditional VAE and GAN generate new labelled data in amounts inversely proportional to the scarcity of the class in each original datasets. We add to each dataset (in the order stated in Table 1) the following amounts of images: 4%, 10%, 4%, 10%, 10%, 5%, 10%, 10%, i.e. 3599, 700, 3899, 470, 54, 1729, 1300, and 1394 images (See Fig. 28 for examples of generated data for the OctMNIST dataset).

The Joint VAE augment the MedMNIST datasets by interweaving the original images with their VAE reconstructions. We expect this setup to increase intra-class noise, which may help our CNN reach better results.

In total, 10 setups are implemented for each retained MedMNIST dataset.

3 Results

The obtained accuracies and AUC on the test sets are available in Tables 2, 3, and 4 where the best results obtained for each dataset are highlighted in green.

Weighted Random Sampling seems to improve performance for the cases using either geometric data augmentation or no augmentation at all. We evidence that using DL data augmentation marginally improves performance on 5 out of 8 of the retained MedMNIST datasets Performance is equivalent between geometric and DL on PneumoniaMNIST and geometric outperforms DL on PathMNIST and OctMNIST.

Table 2. Accuracy & AUC obtained with geometric data augmentation

	no process		WRS, no DA		no WRS, DA		WRS & DA	
MNIST Name	Acc.	AUC	Acc.	AUC	Acc.	AUC	Acc.	AUC
PathMNIST	0.96	0.81	0.94	0.73	0.99	0.86	**0.99**	**0.88**
DermaMNIST	0.91	0.7	0.89	0.59	0.85	0.52	0.85	0.51
OctMNIST	0.93	0.74	0.94	0.75	**0.95**	**0.79**	0.94	0.73
PneumoniaMNIST	0.97	0.85	0.96	0.86	0.97	0.87	**0.97**	**0.88**
BreastMNIST	0.77	0.74	0.83	0.79	0.75	0.65	0.72	0.69
OrganAMNIST	0.99	0.88	0.99	0.88	0.98	0.84	0.99	0.85
OrganCMNIST	0.99	0.86	0.99	0.86	0.97	0.79	0.97	0.78
OrganSMNIST	0.96	0.71	0.96	0.71	0.94	0.61	0.94	0.59

Table 3. Accuracy & AUC obtained with Deep Learning data augmentation

	Conditional VAE		Joint VAE		Conditional GAN	
MNIST Name	Acc.	AUC	Acc.	AUC	Acc.	AUC
PathMNIST	0.97	0.82	00.97	0.82	0.97	0.85
DermaMNIST	0.91	0.61	**0.91**	**0.74**	0.93	0.71
OctMNIST	0.93	0.7	0.94	0.74	0.93	0.71
PneumoniaMNIST	**0.97**	**0.88**	0.93	0.88	0.96	0.87
BreastMNIST	0.85	0.79	0.82	0.8	0.86	0.79
OrganAMNIST	0.99	0.88	0.99	0.88	0.99	0.88
OrganCMNIST	0.99	0.86	0.99	0.86	**0.99**	**0.87**
OrganSMNIST	**0.96**	**0.72**	**0.96**	**0.72**	**0.96**	**0.72**

Table 4. Accuracy & AUC obtained with DL data augmentation and WRS

	Conditional VAE		Joint VAE		Conditional GAN	
MNIST Name	Acc.	AUC	Acc.	AUC	Acc.	AUC
PathMNIST	0.97	0.85	0.97	0.83	0.97	0.86
DermaMNIST	0.9	0.72	0.87	0.56	0.89	0.71
OctMNIST	0.91	0.69	0.93	0.72	0.89	0.71
PneumoniaMNIST	0.95	0.87	0.93	0.87	0.97	0.87
BreastMNIST	0.86	0.79	**0.87**	**0.84**	0.86	0.81
OrganAMNIST	**0.99**	**0.89**	**0.99**	**0.89**	0.99	0.87
OrganCMNIST	0.98	0.86	0.98	0.86	0.99	0.86
OrganSMNIST	0.96	0.72	0.96	0.71	0.96	0.7

The training results and subsequent confusion matrices for each of the resulting 80 classification cases can be found in Annex (See Fig. 29 to Fig. 44).

If we find that DL data augmentation techniques generally improve performance (though, marginally), we cannot say that any of the three methods we selected over-performed over the others. Furthermore, if WRS does not seem to help improve performance when DL data augmentation techniques are involved, we can attest that no classifier achieved the best results when neither WRS or data augmentation (whether geometric or DL) was involved, lending credence to the idea that data augmentation does help improve a classifier's performance.

Some other interesting results can be pointed out. For instance, we obtain a higher number of false negatives when training on BreastMNIST data augmented with DL techniques compared to other setups (See Fig.41), i.e. the CNN over-diagnoses malignant images as benign – which is something we definitely want to avoid in a medical context.

4 Conclusion and final words

As such, we have a preliminary evidence that DL-based data augmentation methods (in our example Conditional VAE, Joint VAE, and Conditional GAN) may help improve the performance of a classifier.

However, we must note that such an improvement obtained via data augmentation should be further explored and studied with different metrics. As evidenced with the case of BreastMNIST, even if the overall accuracy was shown to increase with the use of DL data augmentation, we found that such setup led to an increased number of false negatives – an issue in the field of medical diagnoses. As such, other metrics such as the F1 score should be included to evaluate classifiers.

Finally, it is necessary to note that the MedMNIST authors insist that the datasets are not intended for any clinical use. We must be careful with extending any preliminary conclusion especially in the case of possible real-life, critical applications such as cancer detection.

References

1. Bottou, L., Curtis, F.E., Nocedal, J.: "Optimization Methods for Large-Scale Machine Learning." In: SIAM Review, Vol. 60, No.2, pp. 223-311. (2018)
2. Doersch, C.: "Tutorial on Variational Autoencoders." In: eprint arXiv:1606.05908. (2016)
3. Dupont, E.: "Learning Disentangled Joint Continuous and Discrete Representations." arXiv preprint arXiv:1804.00104. (2018)
4. Fefferman, C., Mitter, S., Narayanan, H.: "Testing the manifold hypothesis." In: Journal of the American Mathematical Society, 29(4), pp.983-1049. (2016)
5. Frid-Adar, M., Diamant, I., Klang, E., Amitai, M., Goldberger, J., Greenspan, H.: "GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification." In: Neurocomputing, Volume 321, pp. 321-331. (2018)
6. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y.: "Generative Adversarial Nets." NIPS. (2014)
7. Jiancheng, Y., Rui, S., Donglai, W., Zequan, L., Lin, Z., Bilian, K., Hanspeter, P., Bingbing, N.: "MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification". arXiv preprint arXiv:2110.14795. (2021)
8. Jiancheng, Y., Rui, Bingbing, N.: "MedMNIST Classification Decathlon: A Lightweight AutoML Benchmark for Medical Image Analysis". In: IEEE 18th International Symposium on Biomedical Imaging (ISBI). (2021)
9. LeCun, Y., Cortes, C.: "MNIST handwritten digit database." (2010)
10. Mirza, M., Osindero, S.: "Conditional Generative Adversarial Nets." In: arXiv eprint arXiv:1411.1784. (2014)
11. Nayak, M.: "An Introduction to Conditional GANs." In: Data Drive Investor (blog). `https://medium.datadriveninvestor.com/an-introduction-to-conditional-gans-cgans-727d1f5bb011`. (2019)
12. Olafenwa, J.: "Simple Introduction to Conditional GANs with TorchFusion and PyTorch." In: Towards Data Science (blog). `https://towardsdatascience.com/simple-intro-to-conditional-gans-with-torchfusion-and-pytorch-404264a3267c`. (2018)
13. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Chintala, S.: "Automatic Differentiation in PyTorch." In: Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. (2019)
14. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Chintala, S.: "PyTorch: An Imperative Style, High-Performance DL Library." In: NIPS 2017 Workshop Autodiff Submission. (2017)
15. Quasim, A., Ezhov, I., Suprosanna, S., Schoppe, O., Paetzold, J., Anjany, S., Kofler, F., Lipkova, J., Hongwei, L., Bjoern, M.: "Red-GAN: Attacking class imbalance cia conditioned generation. Yet another medical imaging perspective." In: Proceedings of Machine Learning Research, 1-14. (2020)
16. Rahman, M.A.: "Understanding Conditional Variational Autoencoders." `https://theaiacademy.blogspot.com/2020/05/understanding-conditional-variational.html`. (2020)
17. Sayak, P.: "Conditional GAN." In: Keras Documentation. `https://keras.io/examples/generative/conditional_gan/`. (2021)
18. Shorten, C., Khoshgoftaar, T.M.: "A survey on Image Data Augmentation for DL." In: J Big Data 6, 60. (2019)
19. Subramanian, A.K.: "PyTorch-VAE." GitHub repository, `https://github.com/AntixK/PyTorch-VAE`. (2020)
20. Yi, X., Walia, E., Babyn, P.: "Generative Adversarial Network in Medical Imaging: A Review." In: Medical Image Analysis. 58. 101552. (2019)

Annex

Of note: We see in the relevant Figures (See Fig. 29 to 36 that training accuracies recorded when training on datasets augmented with a Joint VAE are low compared to their respective validation accuracies. This indicates that, though the CNN learn, there is an issue with the computation of the training accuracy metric. The reason has not been identified as at time of this reporting. It is supposed that there is a conflict between the MedMNIST team's evaluator object and the custom dataloader used in the Joint VAE case (See the previously reference Google Colab document).

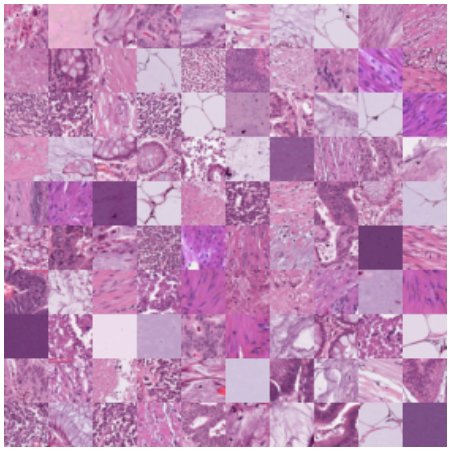

Fig. 1. Montage of training samples from the PathMNIST dataset.

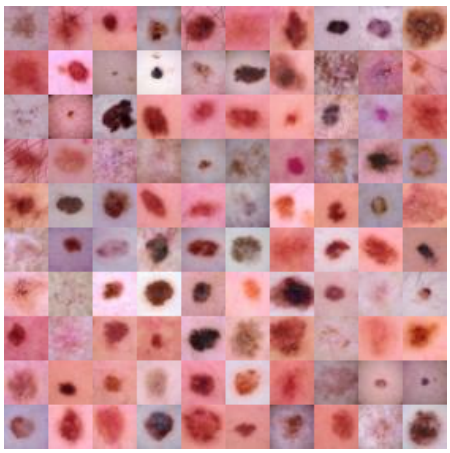

Fig. 2. Montage of training samples from the DermaMNIST dataset.

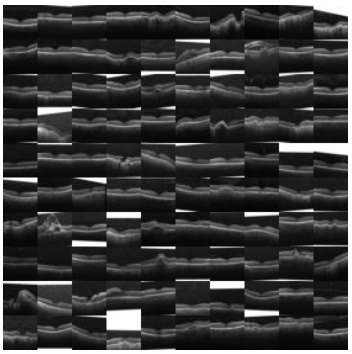

Fig. 3. Montage of training samples from the OctMNIST dataset.

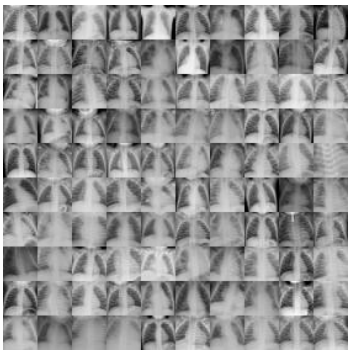

Fig. 4. Montage of training samples from the PneumoniaMNIST dataset.

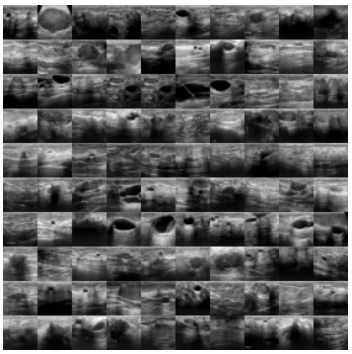

Fig. 5. Montage of training samples from the BreastMNIST dataset.

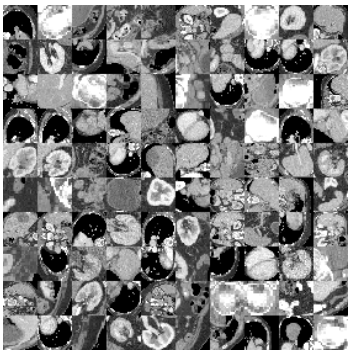

Fig. 6. Montage of training samples from the OrganMNIST_Axial dataset.

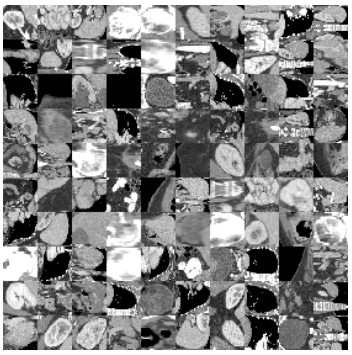

Fig. 7. Montage of training samples from the OrganMNIST_Coronal dataset.

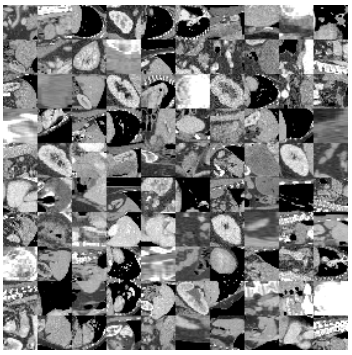

Fig. 8. Montage of training samples from the OrganMNIST_Sagittal dataset.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 26, 26]             448
       BatchNorm2d-2          [-1, 16, 26, 26]              32
              ReLU-3          [-1, 16, 26, 26]               0
            Conv2d-4          [-1, 16, 24, 24]           2,320
       BatchNorm2d-5          [-1, 16, 24, 24]              32
              ReLU-6          [-1, 16, 24, 24]               0
         MaxPool2d-7          [-1, 16, 12, 12]               0
            Conv2d-8          [-1, 64, 10, 10]           9,280
       BatchNorm2d-9          [-1, 64, 10, 10]             128
             ReLU-10          [-1, 64, 10, 10]               0
           Conv2d-11            [-1, 64, 8, 8]          36,928
      BatchNorm2d-12            [-1, 64, 8, 8]             128
             ReLU-13            [-1, 64, 8, 8]               0
           Conv2d-14            [-1, 64, 8, 8]          36,928
      BatchNorm2d-15            [-1, 64, 8, 8]             128
             ReLU-16            [-1, 64, 8, 8]               0
        MaxPool2d-17            [-1, 64, 4, 4]               0
           Linear-18                 [-1, 128]         131,200
             ReLU-19                 [-1, 128]               0
           Linear-20                 [-1, 128]          16,512
             ReLU-21                 [-1, 128]               0
           Linear-22                   [-1, 9]           1,161
================================================================
Total params: 235,225
Trainable params: 235,225
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 0.82
Params size (MB): 0.90
Estimated Total Size (MB): 1.73
----------------------------------------------------------------
```

Fig. 9. Convolutional Neural Network classifier used on the PathMNIST dataset.

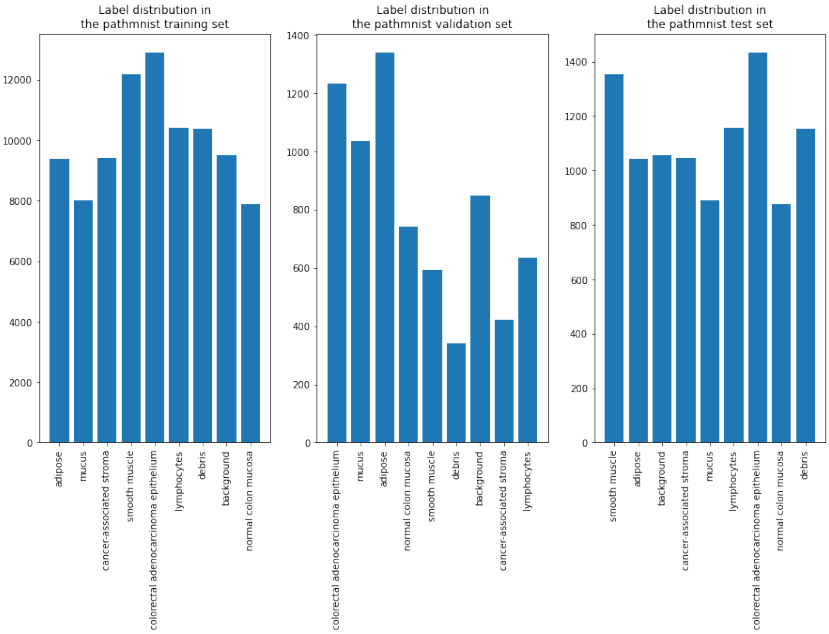

Fig. 10. Class distribution of the PathMNIST dataset.

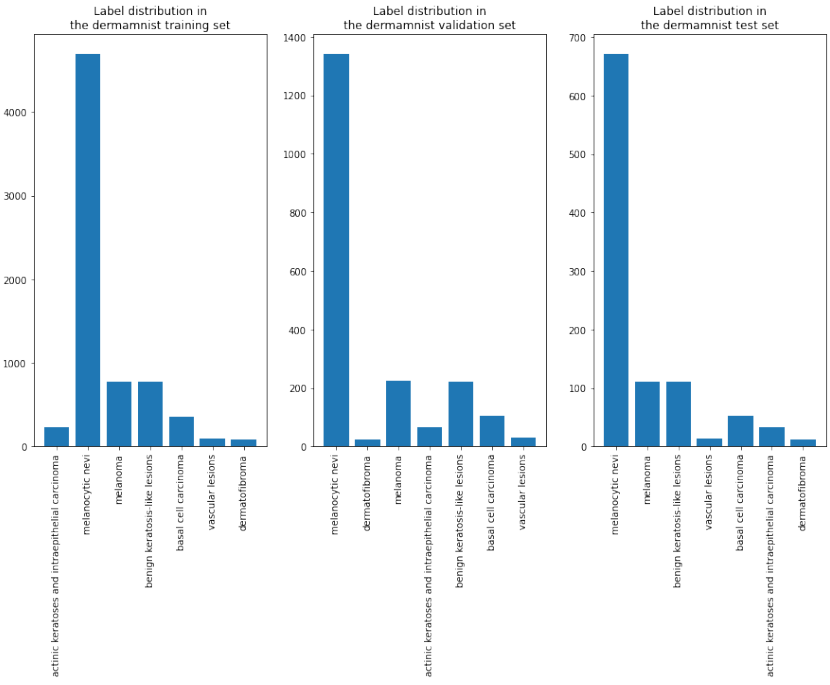

Fig. 11. Class distribution of the DermaMNIST dataset.

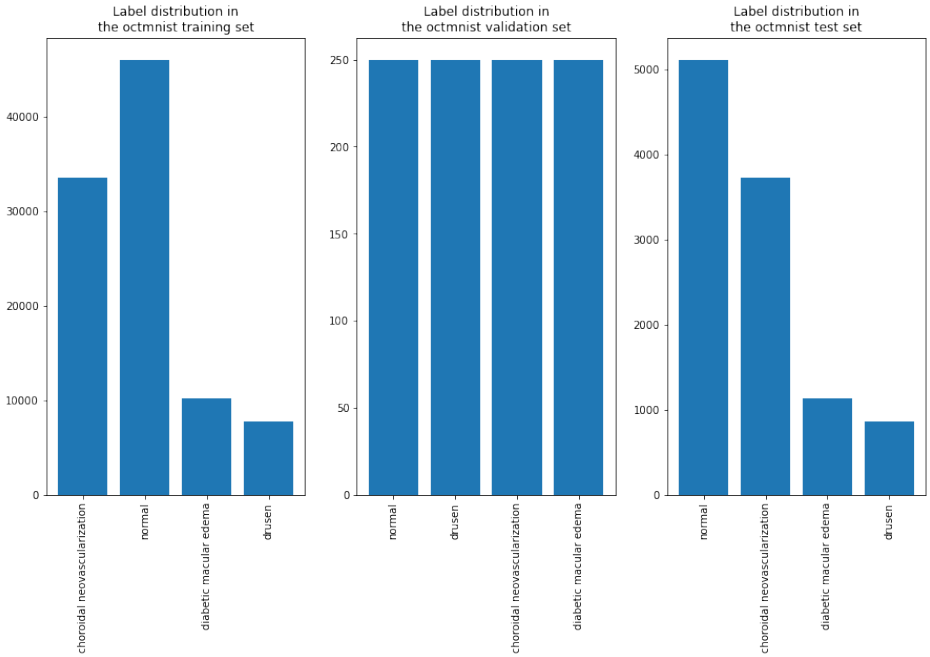

Fig. 12. Montage of training samples from the OctMNIST dataset.

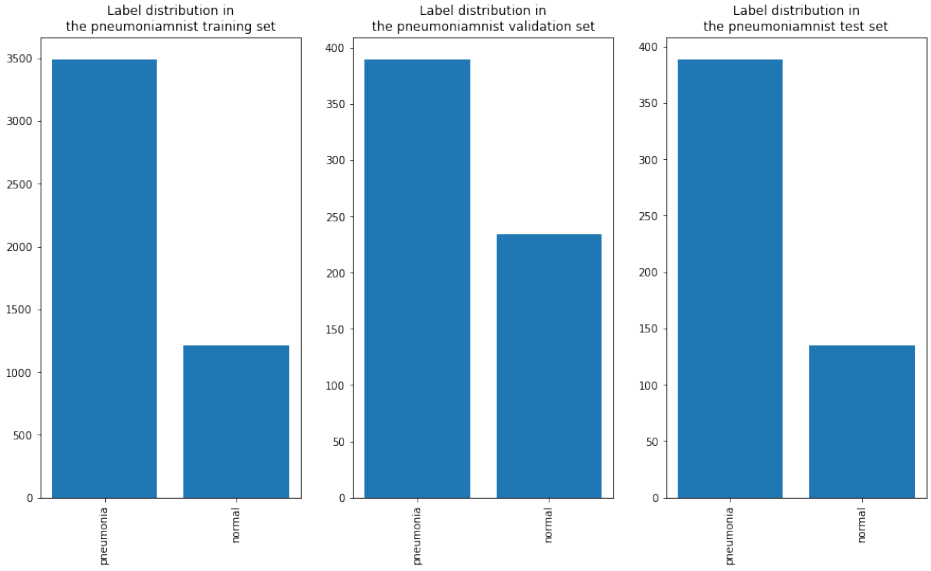

Fig. 13. Montage of training samples from the PneumoniaMNIST dataset.

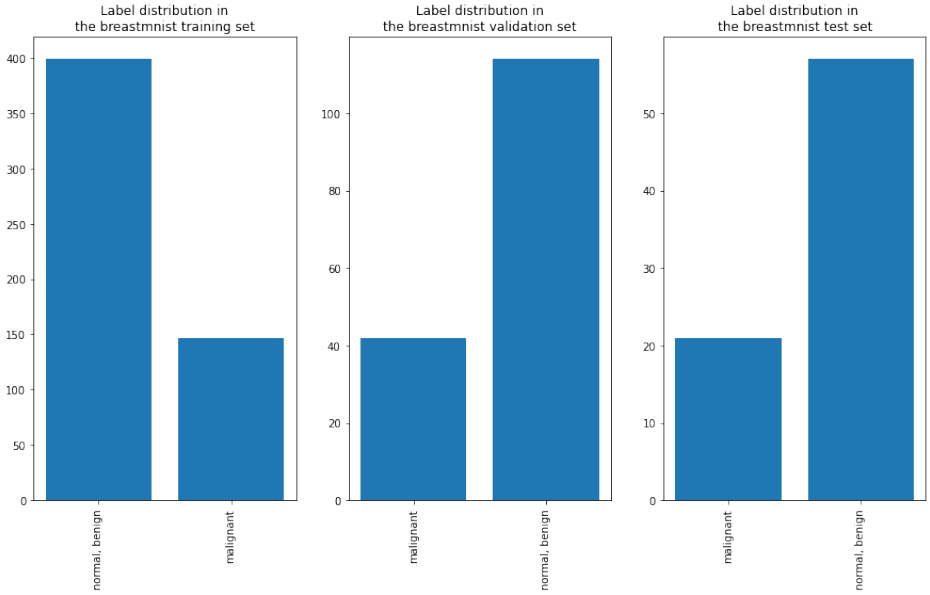

Fig. 14. Class distribution of the BreastMNIST dataset.

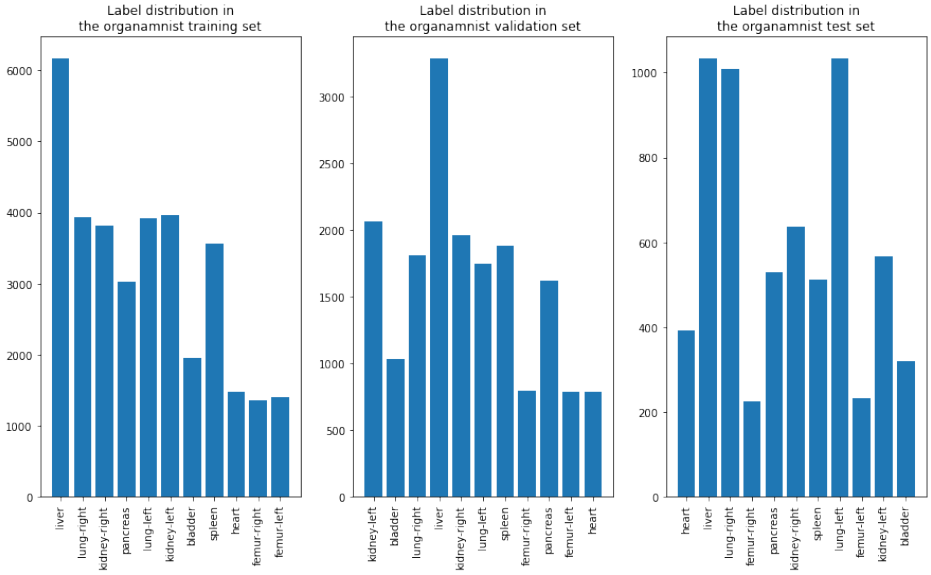

Fig. 15. Class distribution of the OrganMNIST_Axial dataset.

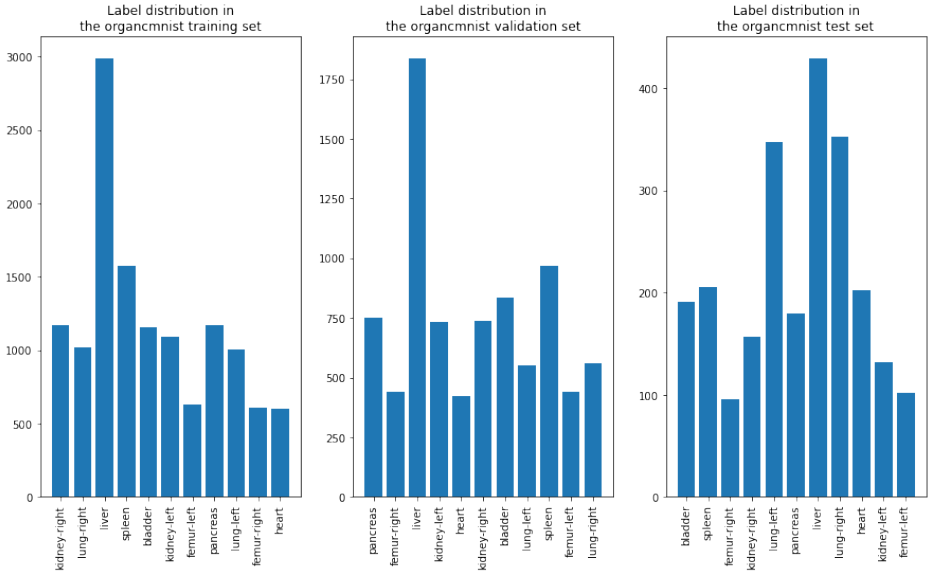

Fig. 16. Class distribution of the OrganMNIST_Coronal dataset.

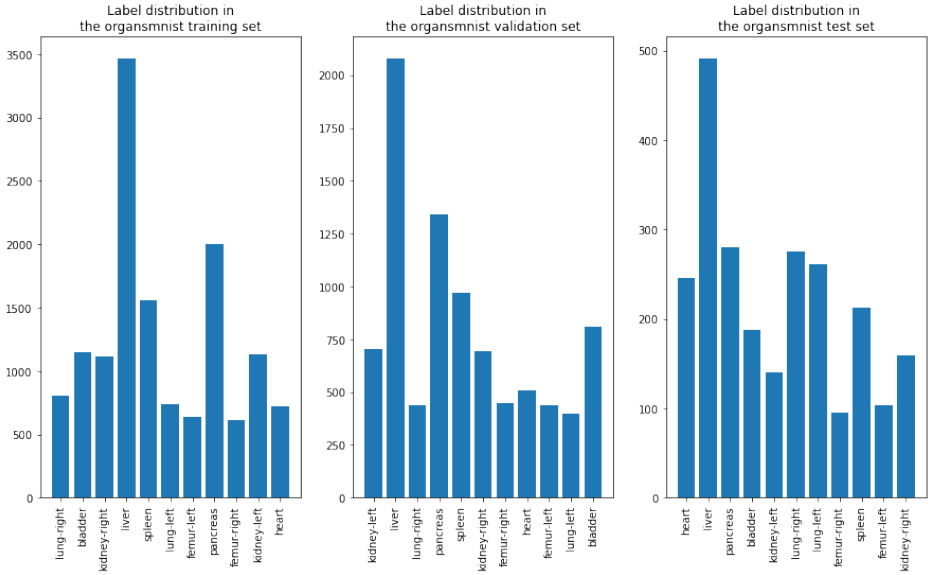

Fig. 17. Class distribution of the OrganMNIST_Sagittal dataset.

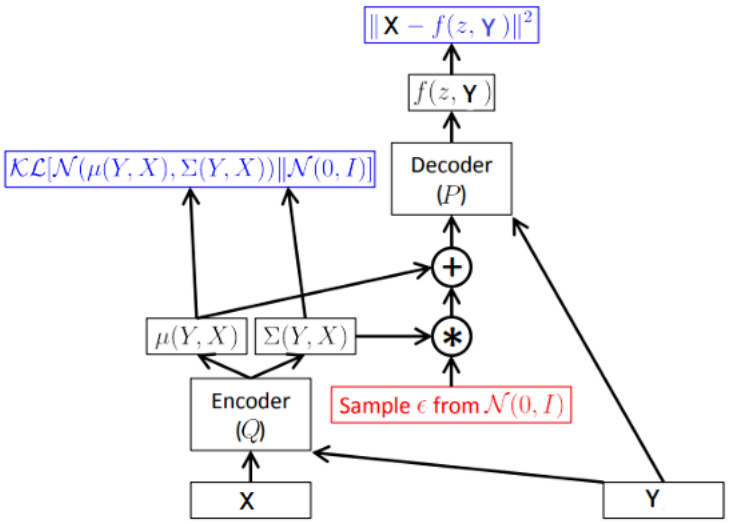

Fig. 18. Reproduction of the graphical representation of a Conditional Variational Auto-Encoder[2].

```
Decoder input layer:
 Linear(in_features=109, out_features=2048, bias=True)

Decoder hidden layers:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 256, 4, 4]       1,179,904
       BatchNorm2d-2            [-1, 256, 4, 4]             512
         LeakyReLU-3            [-1, 256, 4, 4]               0
   ConvTranspose2d-4            [-1, 128, 8, 8]         295,040
       BatchNorm2d-5            [-1, 128, 8, 8]             256
         LeakyReLU-6            [-1, 128, 8, 8]               0
   ConvTranspose2d-7           [-1, 64, 16, 16]          73,792
       BatchNorm2d-8           [-1, 64, 16, 16]             128
         LeakyReLU-9           [-1, 64, 16, 16]               0
  ConvTranspose2d-10           [-1, 32, 32, 32]          18,464
      BatchNorm2d-11           [-1, 32, 32, 32]              64
        LeakyReLU-12           [-1, 32, 32, 32]               0
================================================================
Total params: 1,568,160
Trainable params: 1,568,160
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.41
Params size (MB): 5.98
Estimated Total Size (MB): 7.40
----------------------------------------------------------------

Decoder output layer:

----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1           [-1, 32, 61, 61]           9,248
       BatchNorm2d-2           [-1, 32, 61, 61]              64
         LeakyReLU-3           [-1, 32, 61, 61]               0
            Conv2d-4           [-1, 32, 30, 30]           9,248
       BatchNorm2d-5           [-1, 32, 30, 30]              64
         LeakyReLU-6           [-1, 32, 30, 30]               0
            Conv2d-7            [-1, 3, 28, 28]             867
              Tanh-8            [-1, 3, 28, 28]               0
================================================================
Total params: 19,491
Trainable params: 19,491
Non-trainable params: 0
```

Fig. 19. Decoder architecture of the Conditional VAE in the case of the PathMNIST dataset.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 14, 14]           1,184
       BatchNorm2d-2          [-1, 32, 14, 14]              64
         LeakyReLU-3          [-1, 32, 14, 14]               0
            Conv2d-4            [-1, 64, 7, 7]          18,496
       BatchNorm2d-5            [-1, 64, 7, 7]             128
         LeakyReLU-6            [-1, 64, 7, 7]               0
            Conv2d-7           [-1, 128, 4, 4]          73,856
       BatchNorm2d-8           [-1, 128, 4, 4]             256
         LeakyReLU-9           [-1, 128, 4, 4]               0
           Conv2d-10           [-1, 256, 2, 2]         295,168
      BatchNorm2d-11           [-1, 256, 2, 2]             512
        LeakyReLU-12           [-1, 256, 2, 2]               0
           Conv2d-13           [-1, 512, 1, 1]       1,180,160
      BatchNorm2d-14           [-1, 512, 1, 1]           1,024
        LeakyReLU-15           [-1, 512, 1, 1]               0
================================================================
Total params: 1,570,848
Trainable params: 1,570,848
Non-trainable params: 0
```

Fig. 20. Encoder architecture of the Conditional VAE in the case of the PathMNIST dataset.

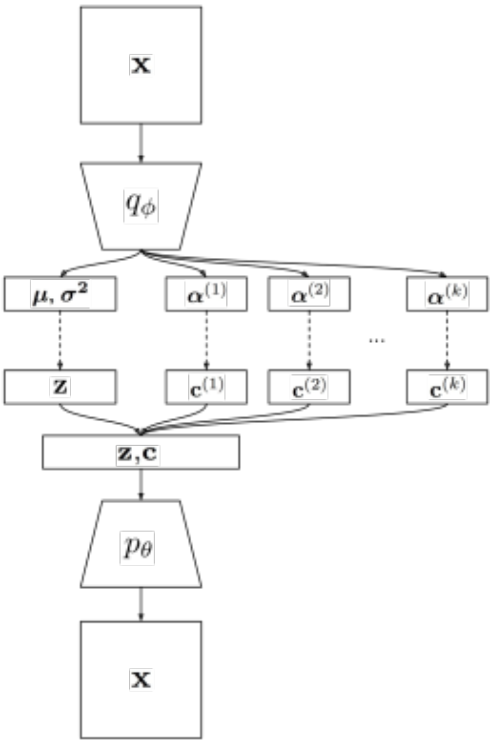

Fig. 21. Reproduction of the graphical representation of a Joint Variational Auto-Encoder[3].

```
Decoder input layer:
 Linear(in_features=110, out_features=2048, bias=True)

Decoder hidden layers:
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 256, 4, 4]       1,179,904
       BatchNorm2d-2            [-1, 256, 4, 4]             512
         LeakyReLU-3            [-1, 256, 4, 4]               0
   ConvTranspose2d-4            [-1, 128, 8, 8]         295,040
       BatchNorm2d-5            [-1, 128, 8, 8]             256
         LeakyReLU-6            [-1, 128, 8, 8]               0
   ConvTranspose2d-7           [-1, 64, 16, 16]          73,792
       BatchNorm2d-8           [-1, 64, 16, 16]             128
         LeakyReLU-9           [-1, 64, 16, 16]               0
  ConvTranspose2d-10           [-1, 32, 32, 32]          18,464
      BatchNorm2d-11           [-1, 32, 32, 32]              64
        LeakyReLU-12           [-1, 32, 32, 32]               0
================================================================
Total params: 1,568,160
Trainable params: 1,568,160
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.41
Params size (MB): 5.98
Estimated Total Size (MB): 7.40
----------------------------------------------------------------

Decoder output layer:

----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1           [-1, 32, 61, 61]           9,248
       BatchNorm2d-2           [-1, 32, 61, 61]              64
         LeakyReLU-3           [-1, 32, 61, 61]               0
            Conv2d-4           [-1, 32, 30, 30]           9,248
       BatchNorm2d-5           [-1, 32, 30, 30]              64
         LeakyReLU-6           [-1, 32, 30, 30]               0
            Conv2d-7            [-1, 3, 28, 28]             867
              Tanh-8            [-1, 3, 28, 28]               0
================================================================
Total params: 19,491
Trainable params: 19,491
Non-trainable params: 0
```

Fig. 22. Decoder architecture of the Joint VAE in the case of the PathMNIST dataset.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 14, 14]             896
       BatchNorm2d-2          [-1, 32, 14, 14]              64
         LeakyReLU-3          [-1, 32, 14, 14]               0
            Conv2d-4            [-1, 64, 7, 7]          18,496
       BatchNorm2d-5            [-1, 64, 7, 7]             128
         LeakyReLU-6            [-1, 64, 7, 7]               0
            Conv2d-7           [-1, 128, 4, 4]          73,856
       BatchNorm2d-8           [-1, 128, 4, 4]             256
         LeakyReLU-9           [-1, 128, 4, 4]               0
           Conv2d-10           [-1, 256, 2, 2]         295,168
      BatchNorm2d-11           [-1, 256, 2, 2]             512
        LeakyReLU-12           [-1, 256, 2, 2]               0
           Conv2d-13           [-1, 512, 1, 1]       1,180,160
      BatchNorm2d-14           [-1, 512, 1, 1]           1,024
        LeakyReLU-15           [-1, 512, 1, 1]               0
================================================================
Total params: 1,570,560
Trainable params: 1,570,560
Non-trainable params: 0
```

Fig. 23. Encoder architecture of the Joint VAE in the case of the PathMNIST dataset.

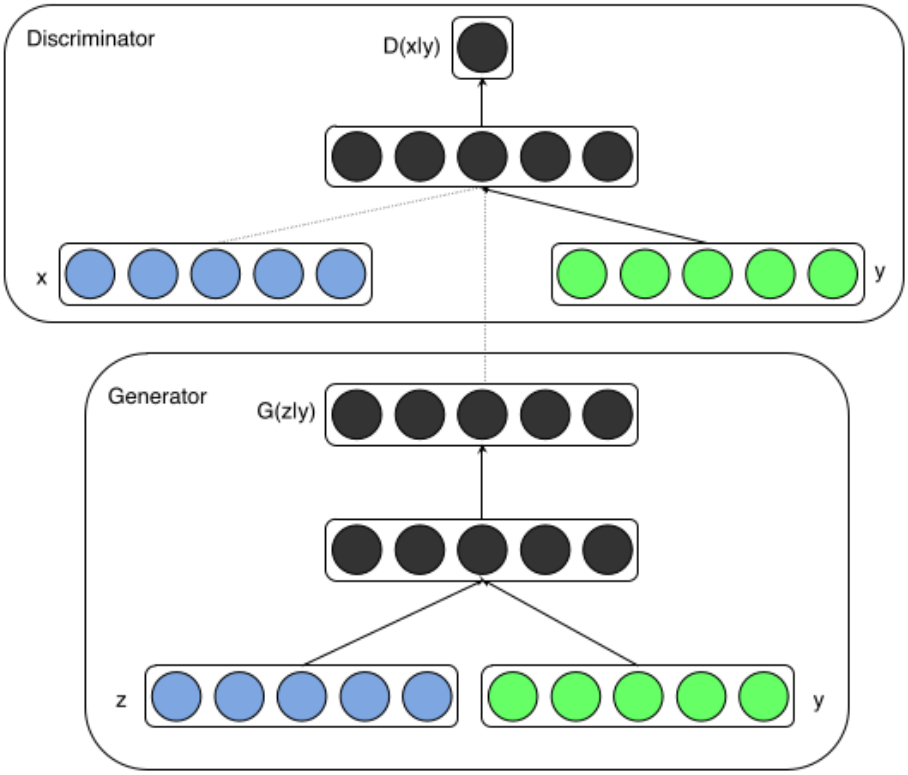

Fig. 24. Reproduction of the graphical representation of a Conditional Generative Adversarial Network[2].

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 32, 32]             896
         LeakyReLU-2          [-1, 32, 32, 32]               0
StandardDiscriminatorBlock-3          [-1, 32, 32, 32]
            Conv2d-4          [-1, 64, 16, 16]          32,832
         LeakyReLU-5          [-1, 64, 16, 16]               0
         LeakyReLU-6          [-1, 64, 16, 16]               0
         LeakyReLU-7          [-1, 64, 16, 16]               0
StandardDiscriminatorBlock-8          [-1, 64, 16, 16]
           Dropout-9          [-1, 64, 16, 16]               0
           Conv2d-10           [-1, 8, 16, 16]             512
           Conv2d-11           [-1, 8, 16, 16]             512
           Conv2d-12          [-1, 64, 16, 16]           4,096
          Softmax-13            [-1, 256, 256]               0
    SelfAttention-14          [-1, 64, 16, 16]               0
           Conv2d-15           [-1, 128, 8, 8]         131,200
        LeakyReLU-16           [-1, 128, 8, 8]               0
        LeakyReLU-17           [-1, 128, 8, 8]               0
        LeakyReLU-18           [-1, 128, 8, 8]               0
StandardDiscriminatorBlock-19           [-1, 128, 8, 8]
          Dropout-20           [-1, 128, 8, 8]               0
           Conv2d-21           [-1, 256, 4, 4]         524,544
        LeakyReLU-22           [-1, 256, 4, 4]               0
        LeakyReLU-23           [-1, 256, 4, 4]               0
        LeakyReLU-24           [-1, 256, 4, 4]               0
StandardDiscriminatorBlock-25           [-1, 256, 4, 4]
          Dropout-26           [-1, 256, 4, 4]               0
          Flatten-27                [-1, 4096]               0
           Linear-28                   [-1, 1]           4,097
        Embedding-29             [-1, 1, 4096]          36,864
           Linear-28                   [-1, 9]          36,873
================================================================
Total params: 772,426
Trainable params: 772,426
Non-trainable params: 0
```

Fig. 25. Discriminator architecture of the Conditional GAN† in the case of the PathMNIST dataset.

†To work with the TorchFusion library, the width and height of the datasets' images are upscaled from 28 to 32

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
   ConvTranspose2d-1            [-1, 256, 4, 4]         409,856
       BatchNorm2d-2            [-1, 256, 4, 4]               0
         Embedding-3               [-1, 1, 256]           2,304
         Embedding-4               [-1, 1, 256]           2,304
ConditionalBatchNorm2d-5            [-1, 256, 4, 4]
         LeakyReLU-6            [-1, 256, 4, 4]               0
         LeakyReLU-7            [-1, 256, 4, 4]               0
         LeakyReLU-8            [-1, 256, 4, 4]               0
StandardGeneratorBlock-9            [-1, 256, 4, 4]
  ConvTranspose2d-10            [-1, 128, 8, 8]         524,416
      BatchNorm2d-11            [-1, 128, 8, 8]               0
        Embedding-12               [-1, 1, 128]           1,152
        Embedding-13               [-1, 1, 128]           1,152
ConditionalBatchNorm2d-14            [-1, 128, 8, 8]
        LeakyReLU-15            [-1, 128, 8, 8]               0
        LeakyReLU-16            [-1, 128, 8, 8]               0
        LeakyReLU-17            [-1, 128, 8, 8]               0
StandardGeneratorBlock-18            [-1, 128, 8, 8]
         Dropout-19            [-1, 128, 8, 8]               0
  ConvTranspose2d-20           [-1, 64, 16, 16]         131,136
      BatchNorm2d-21           [-1, 64, 16, 16]               0
        Embedding-22                [-1, 1, 64]             576
        Embedding-23                [-1, 1, 64]             576
ConditionalBatchNorm2d-24           [-1, 64, 16, 16]
        LeakyReLU-25           [-1, 64, 16, 16]               0
        LeakyReLU-26           [-1, 64, 16, 16]               0
        LeakyReLU-27           [-1, 64, 16, 16]               0
StandardGeneratorBlock-28           [-1, 64, 16, 16]
         Dropout-29           [-1, 64, 16, 16]               0
  ConvTranspose2d-30            [-1, 3, 32, 32]           3,075
================================================================
Total params: 1,076,547
Trainable params: 1,076,547
Non-trainable params: 0
```

Fig. 26. Generator architecture of the Conditional GAN† in the case of the PathMNIST dataset.

† To work with the TorchFusion library, the width and height of the generated images are set to 32, and are subsequently downscaled to 28 to work with the baseline classifier

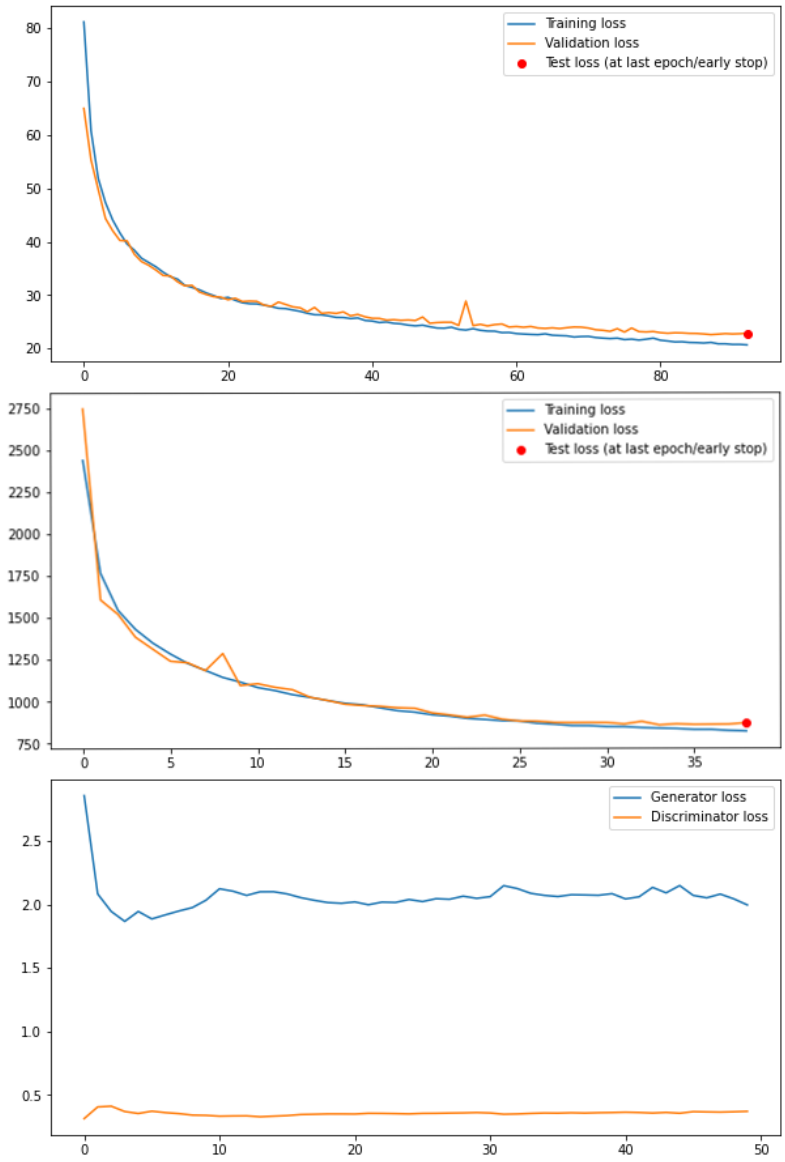

Fig. 27. Training loss of (in order) the conditional VAE, joint VAE, and conditional GAN on the PathMNIST dataset

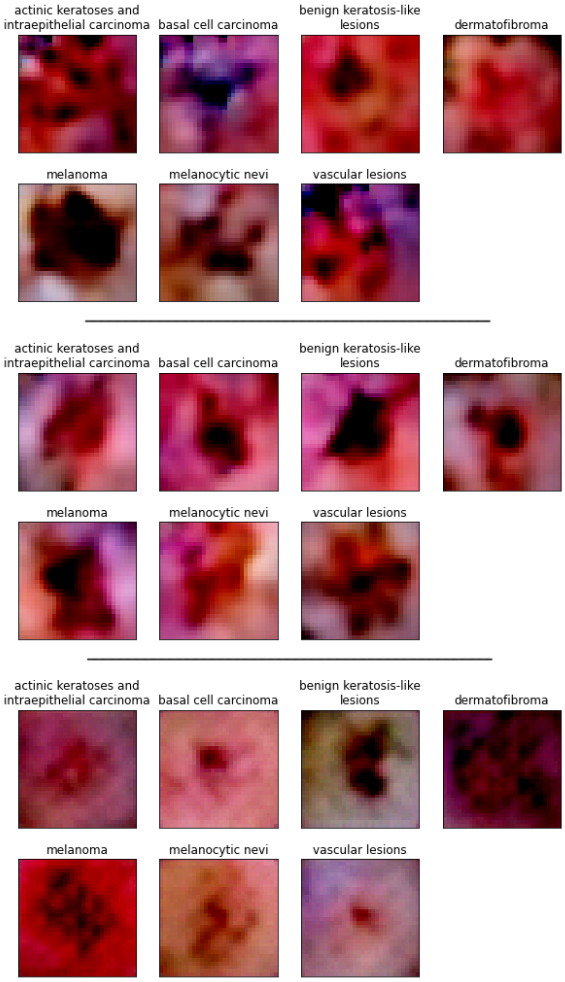

Fig. 28. Example of generated data for the DermaMNIST dataset for each data augmentation models in order: Conditional VAE, Joint VAE, Conditional GAN

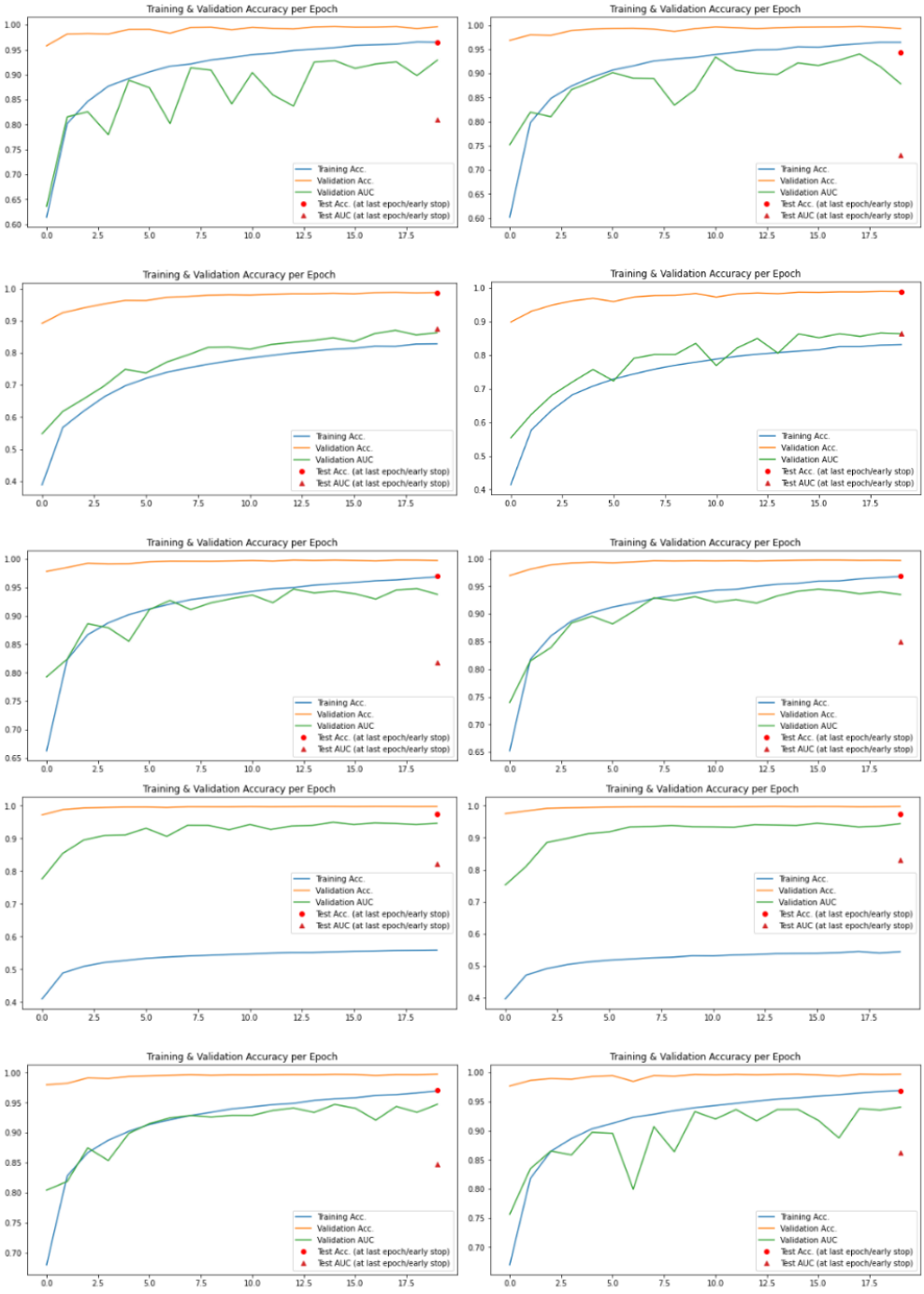

Fig. 29. Accuracies and Areas under the Curve obtained during the classifier training on the PathMNIST dataset given (from top to bottom, left to right) w/o WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

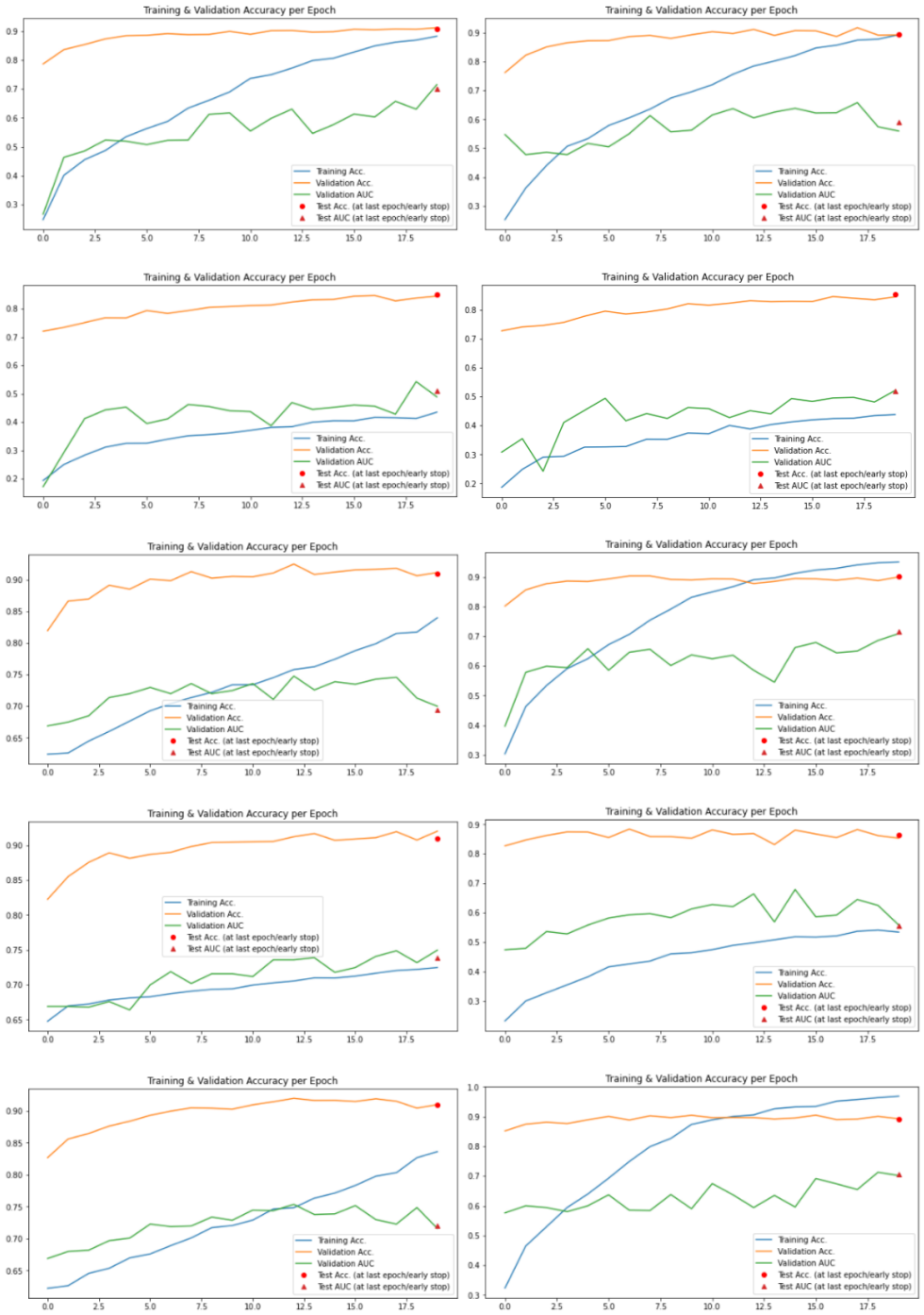

Fig. 30. Accuracies and Areas under the Curve obtained during the classifier training on the DermaMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

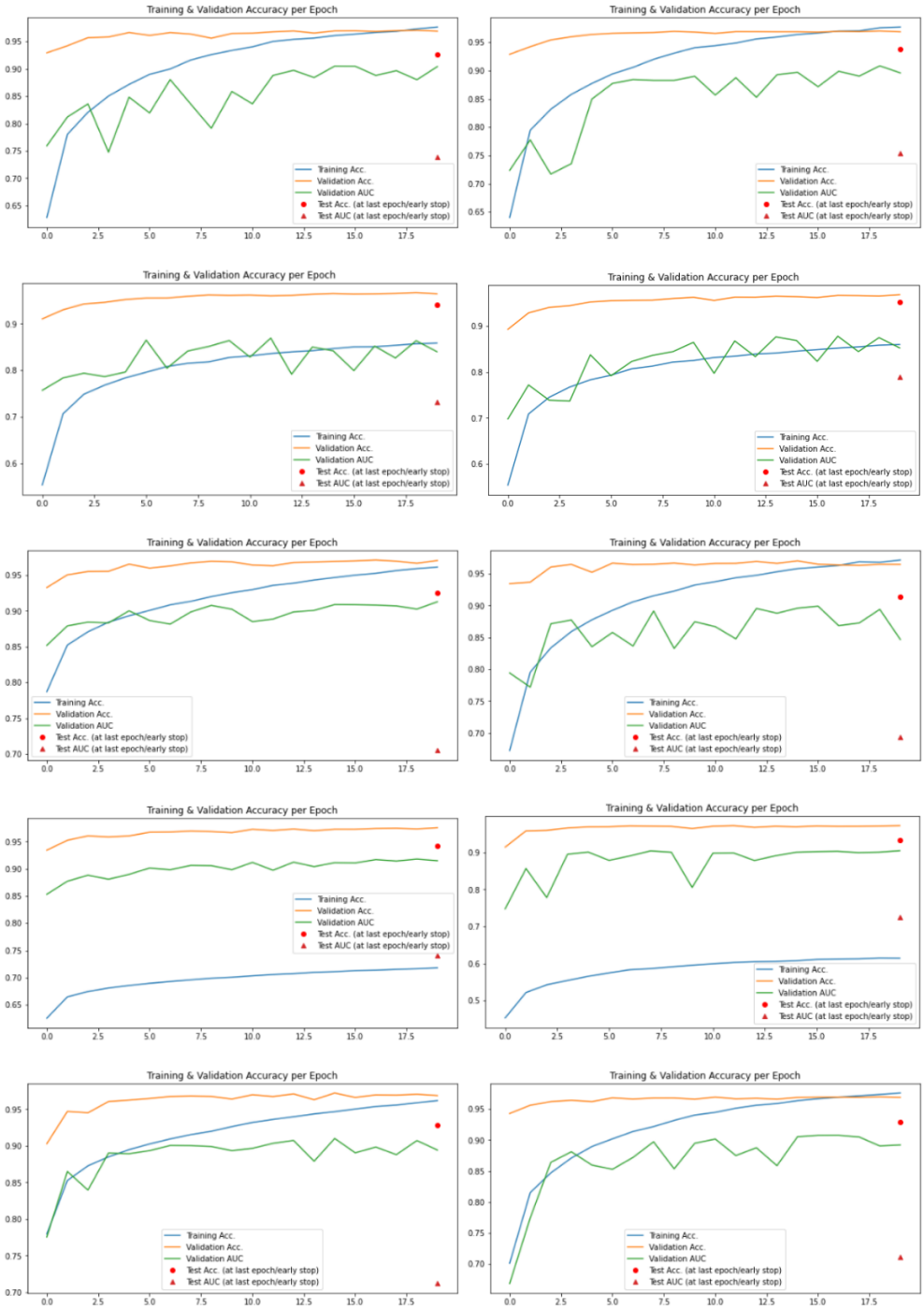

Fig. 31. Accuracies and Areas under the Curve obtained during the classifier training on the OctMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

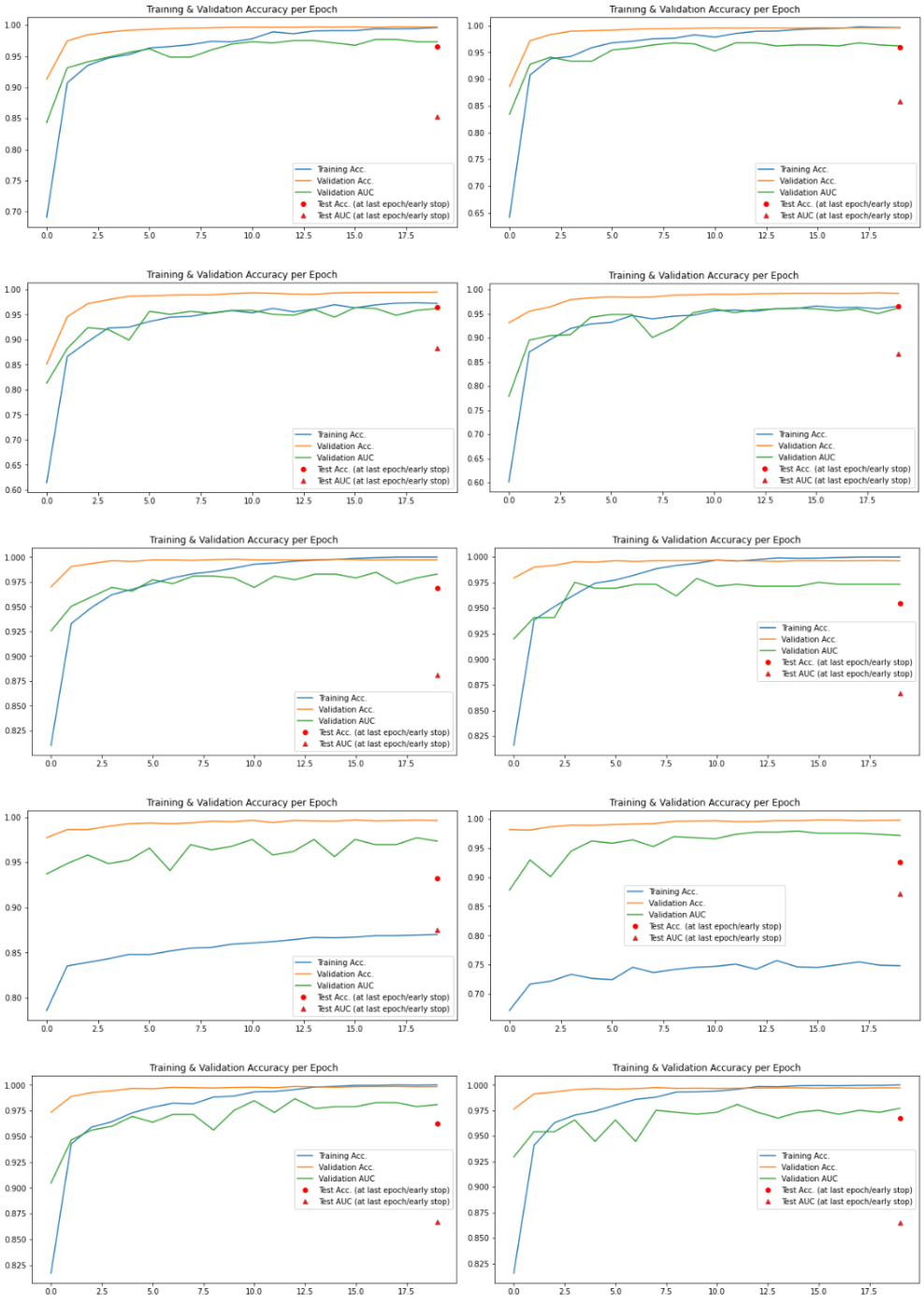

Fig. 32. Accuracies and Areas under the Curve obtained during the classifier training on the PneumoniaMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

Fig. 33. Accuracies and Areas under the Curve obtained during the classifier training on the BreastMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

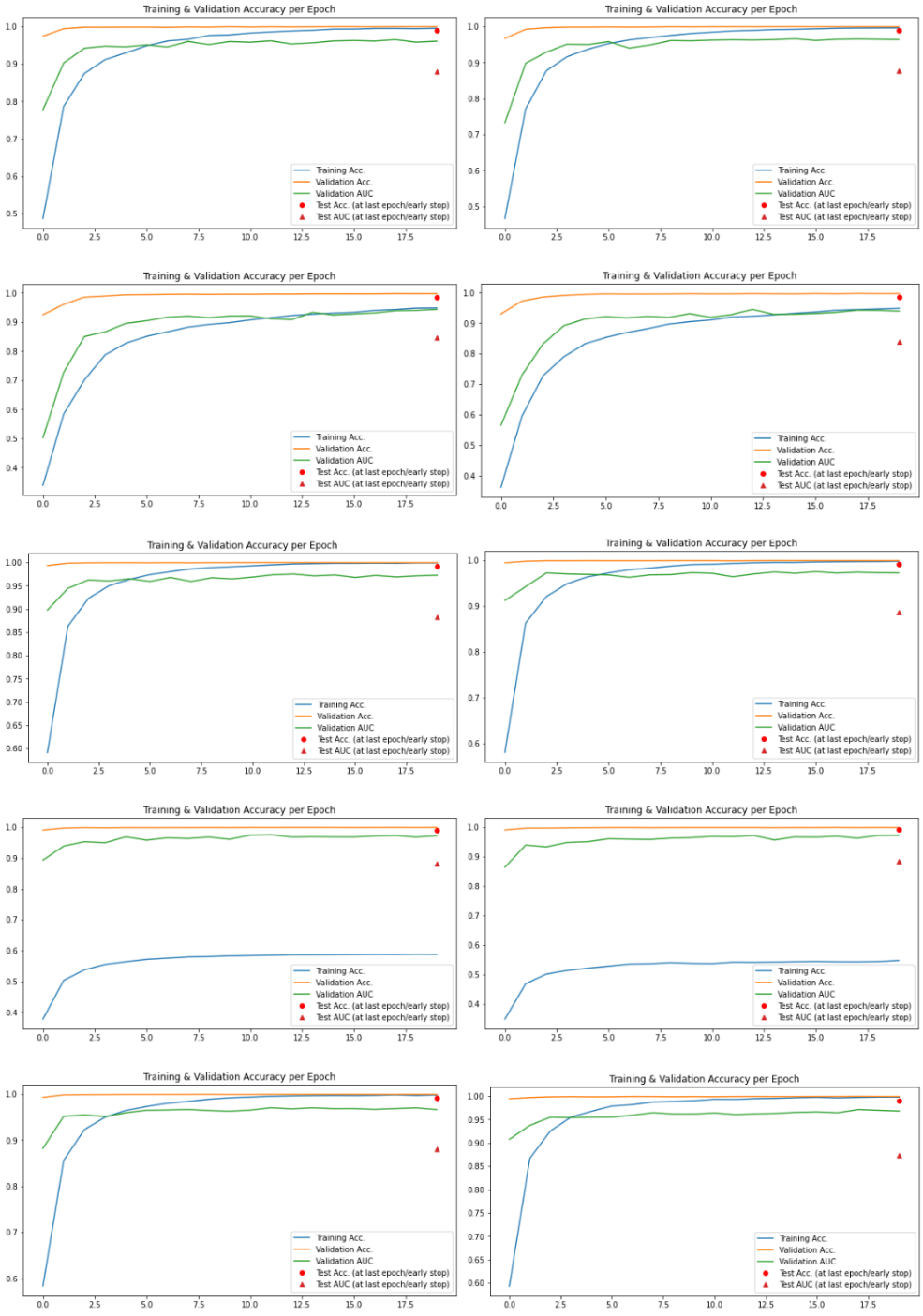

Fig. 34. Accuracies and Areas under the Curve obtained during the classifier training on the OrganAMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

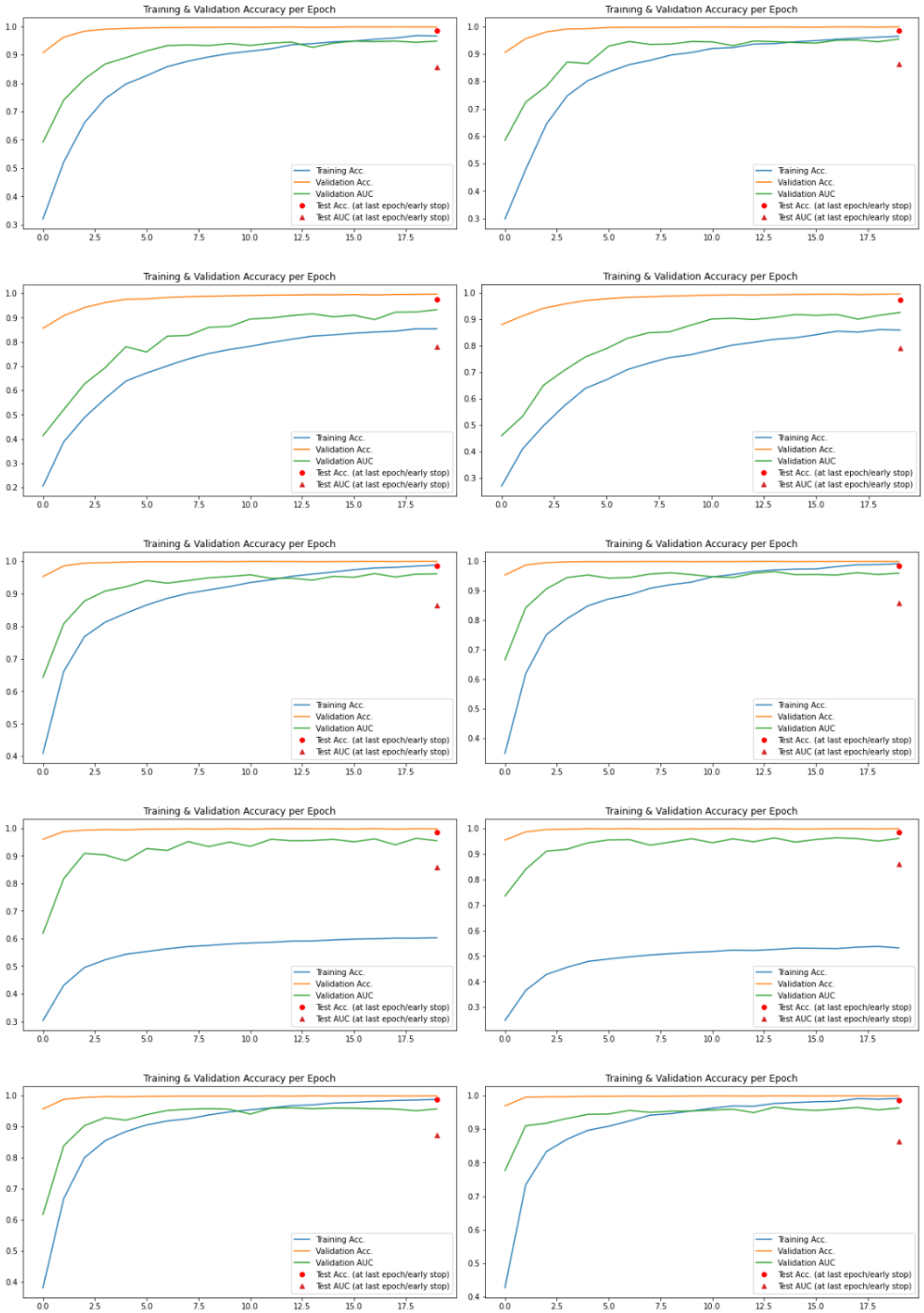

Fig. 35. Accuracies and Areas under the Curve obtained during the classifier training on the OrganCMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

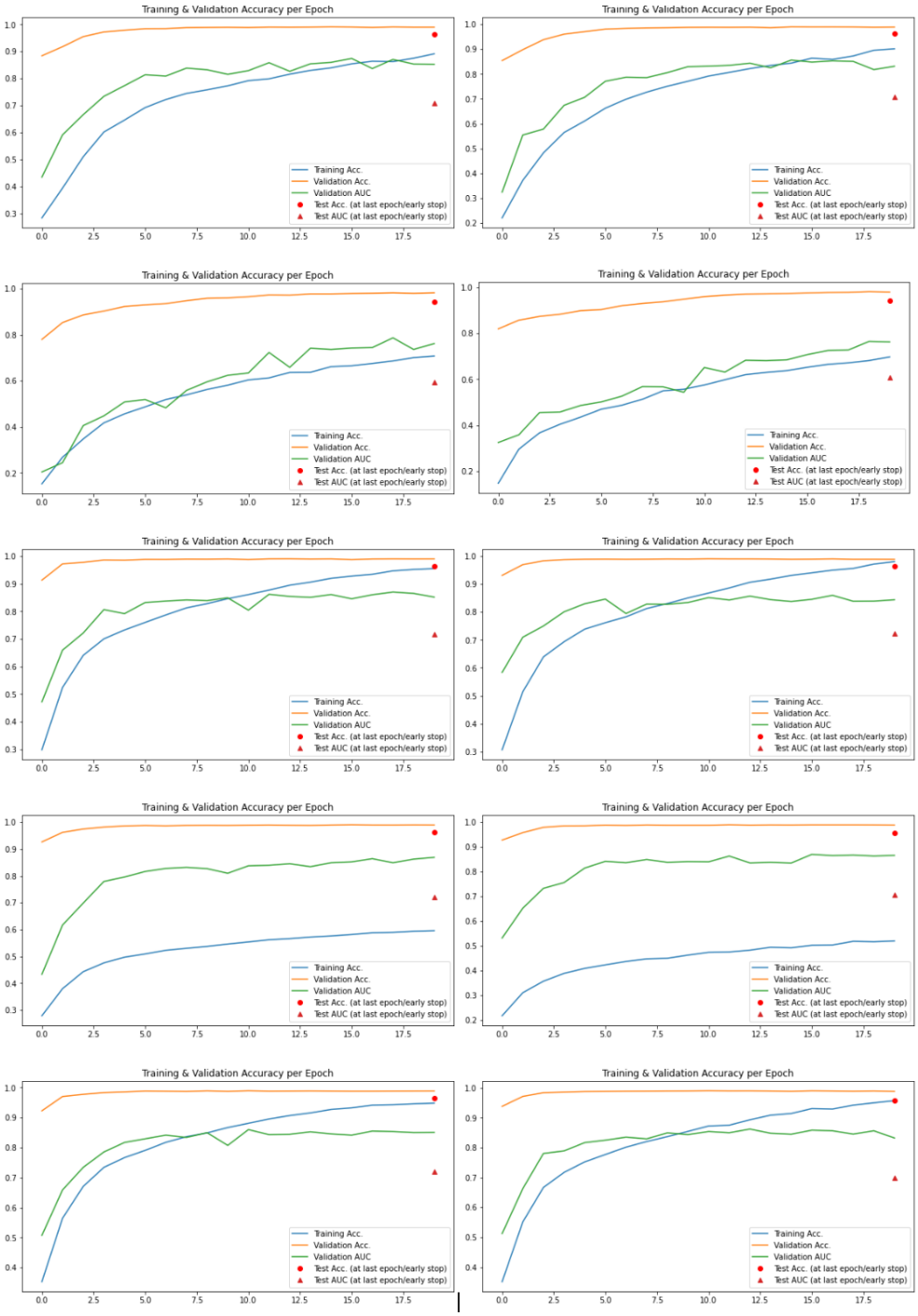

Fig. 36. Accuracies and Areas under the Curve obtained during the classifier training on the OrganSMNIST dataset given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

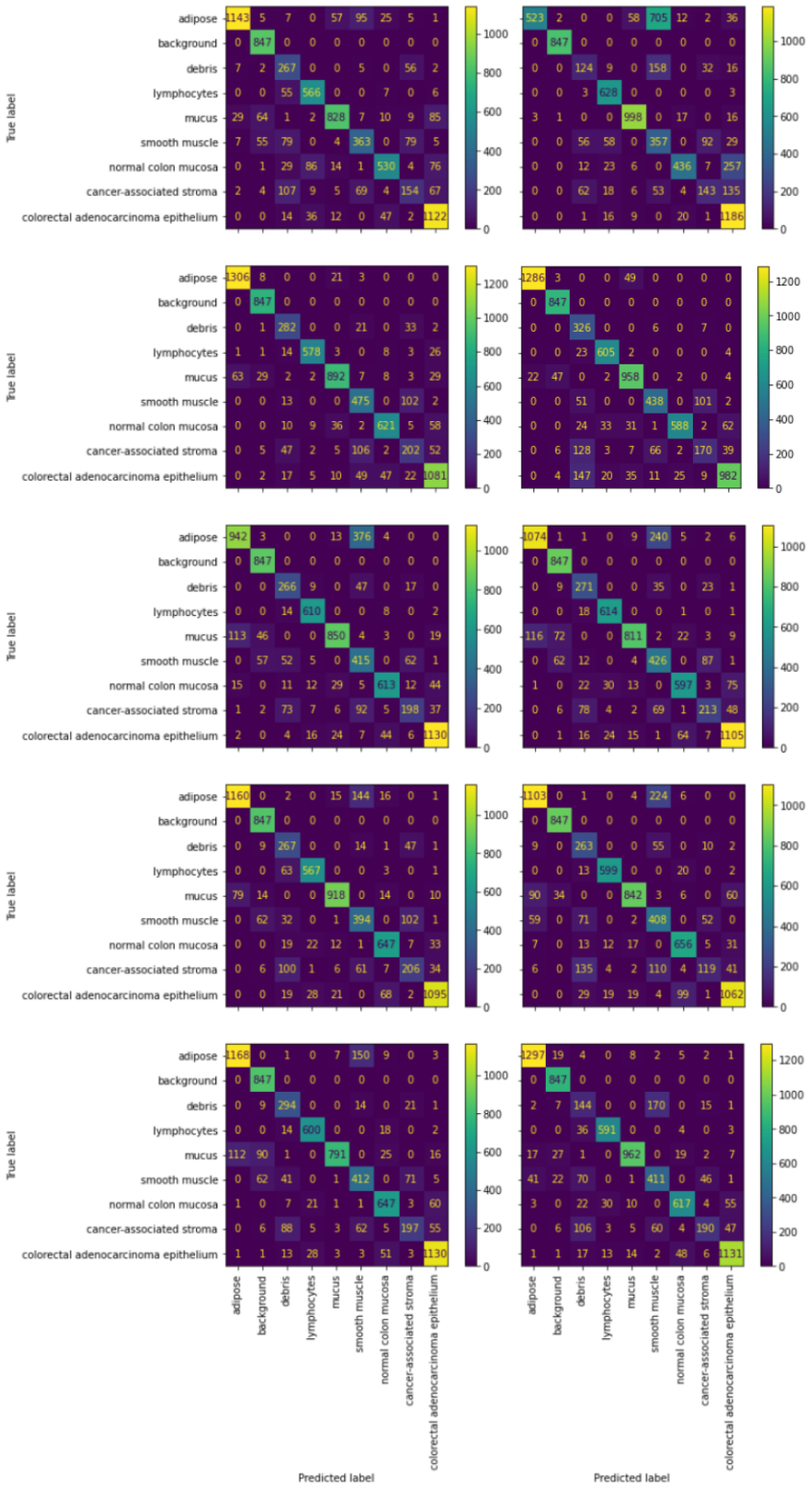

Fig. 37. Confusion matrices on the PathMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

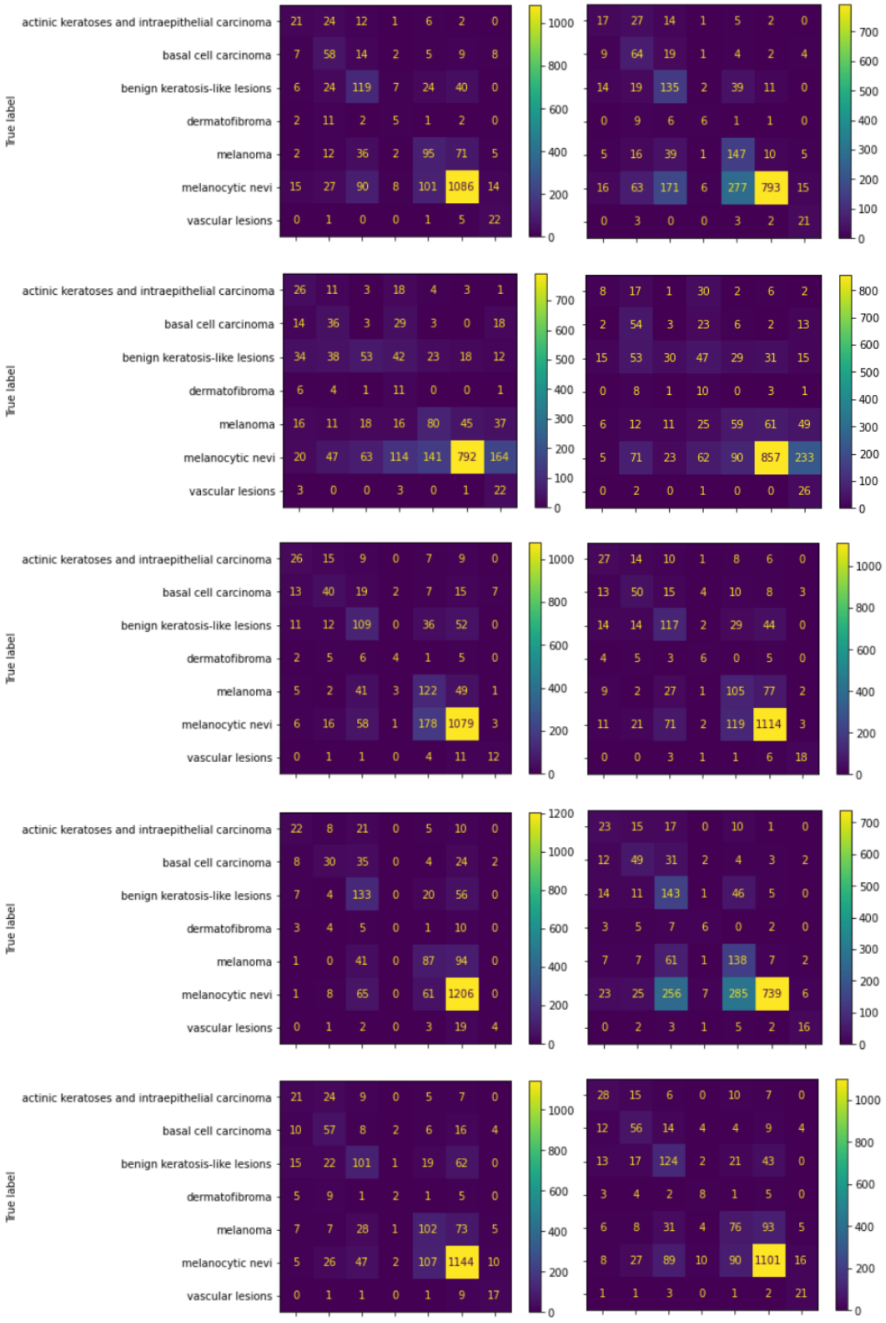

Fig. 38. Confusion matrices on the DermaMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

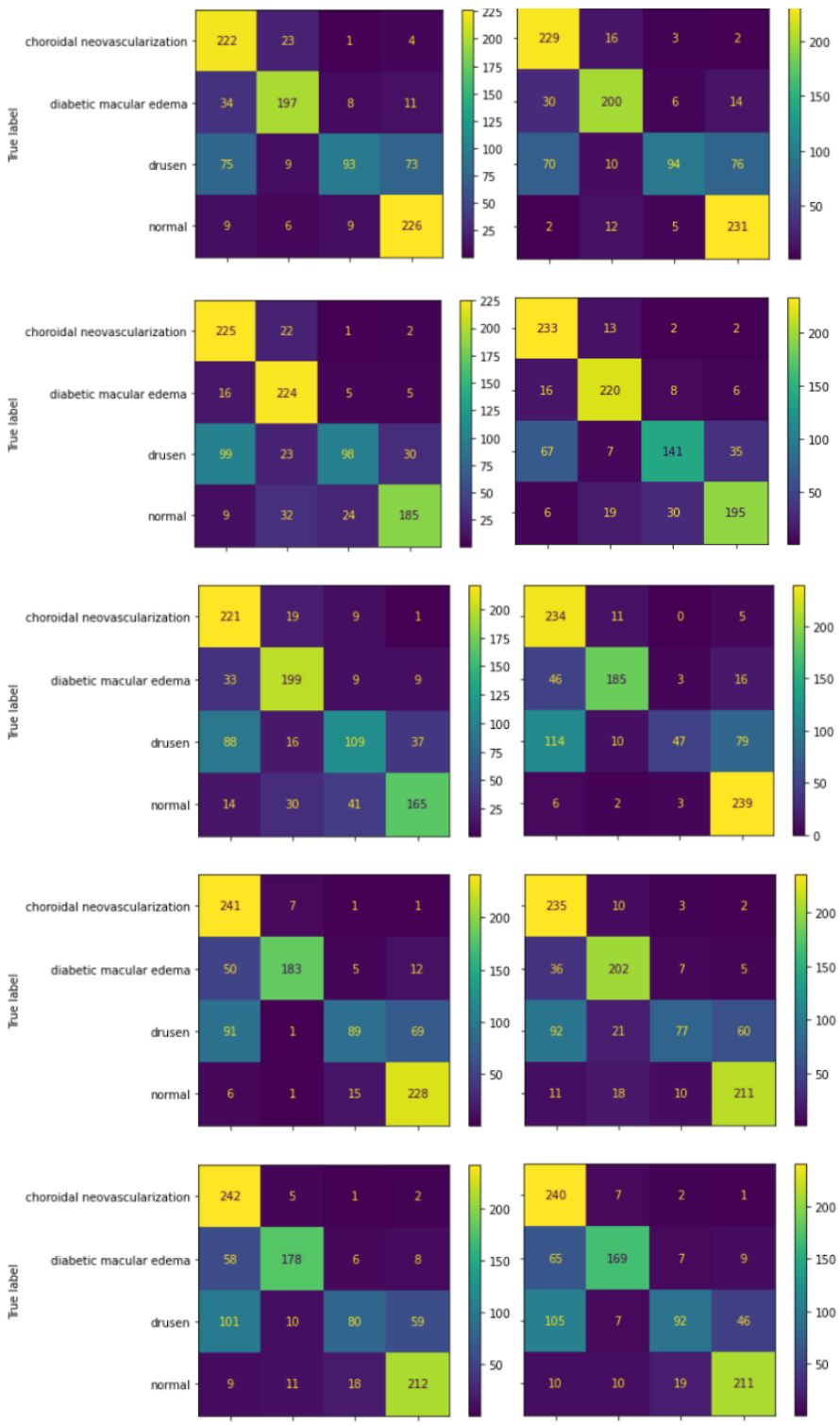

Fig. 39. Confusion matrices on the OctMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

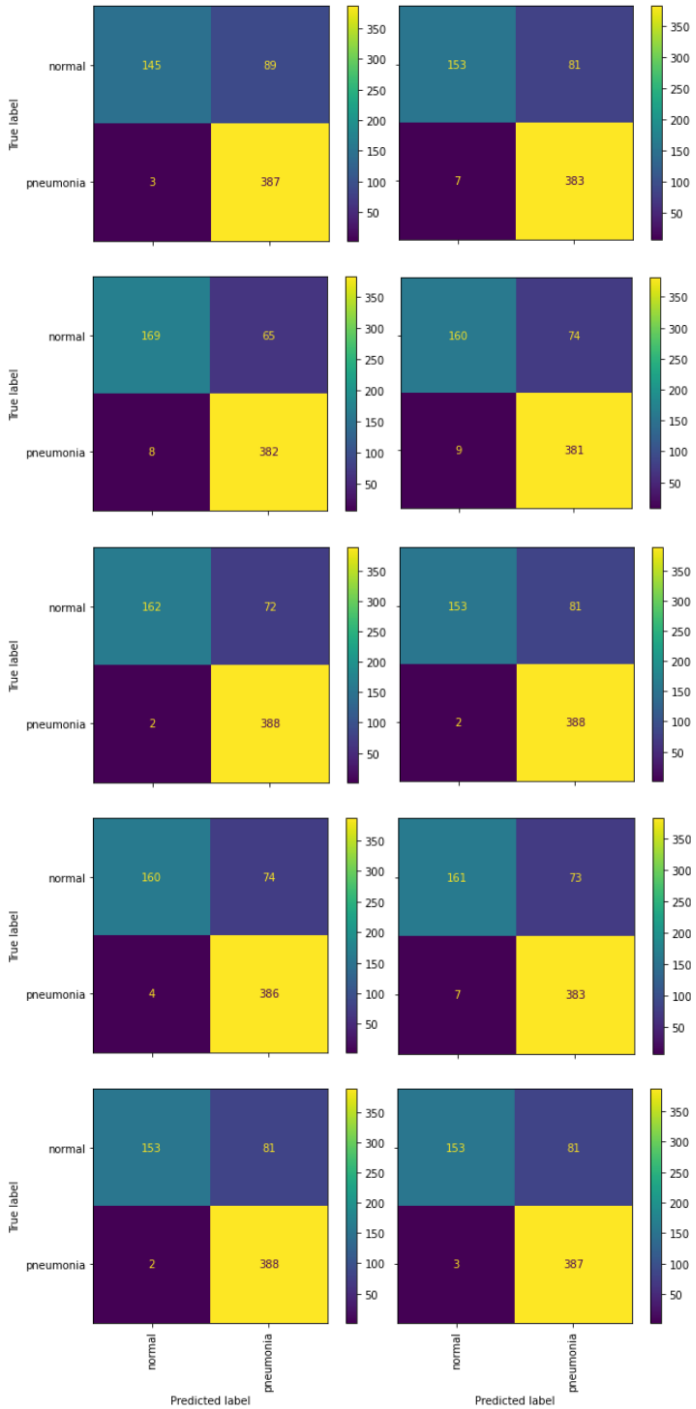

Fig. 40. Confusion matrices on the PneumoniaMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

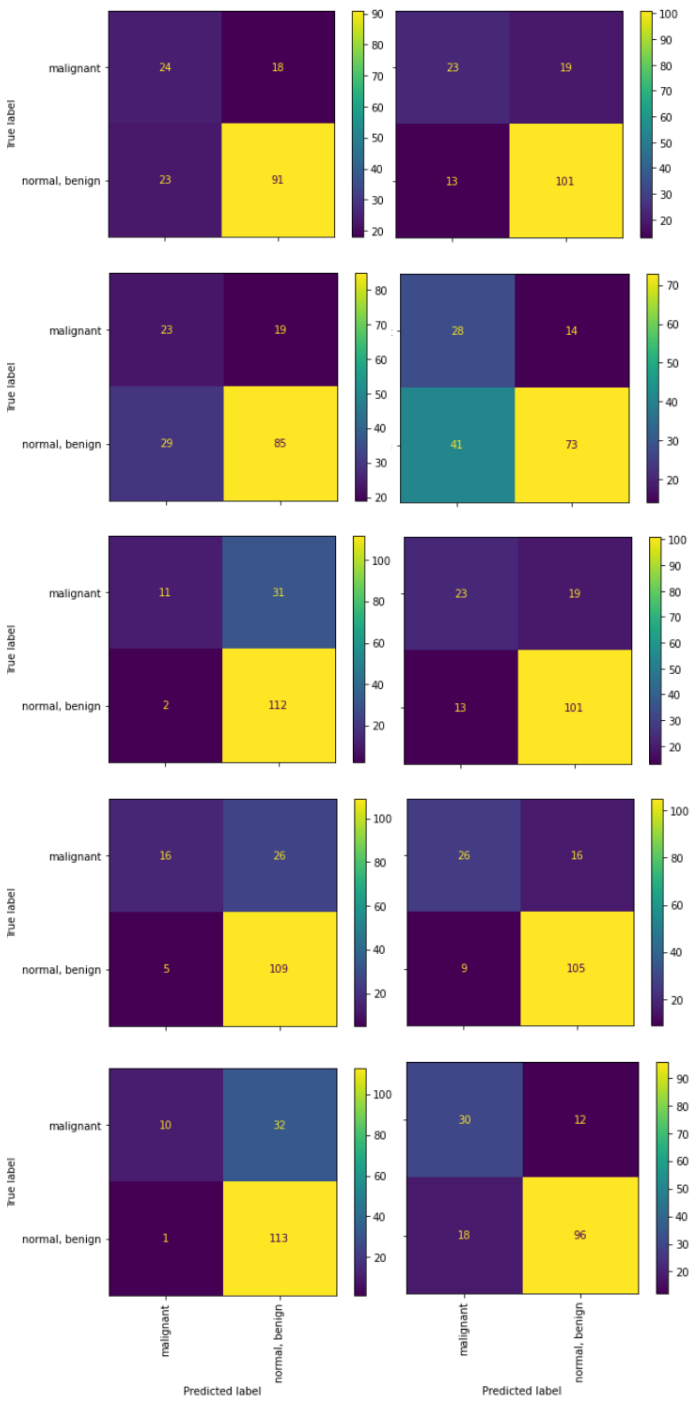

Fig. 41. Confusion matrices on the BreastMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

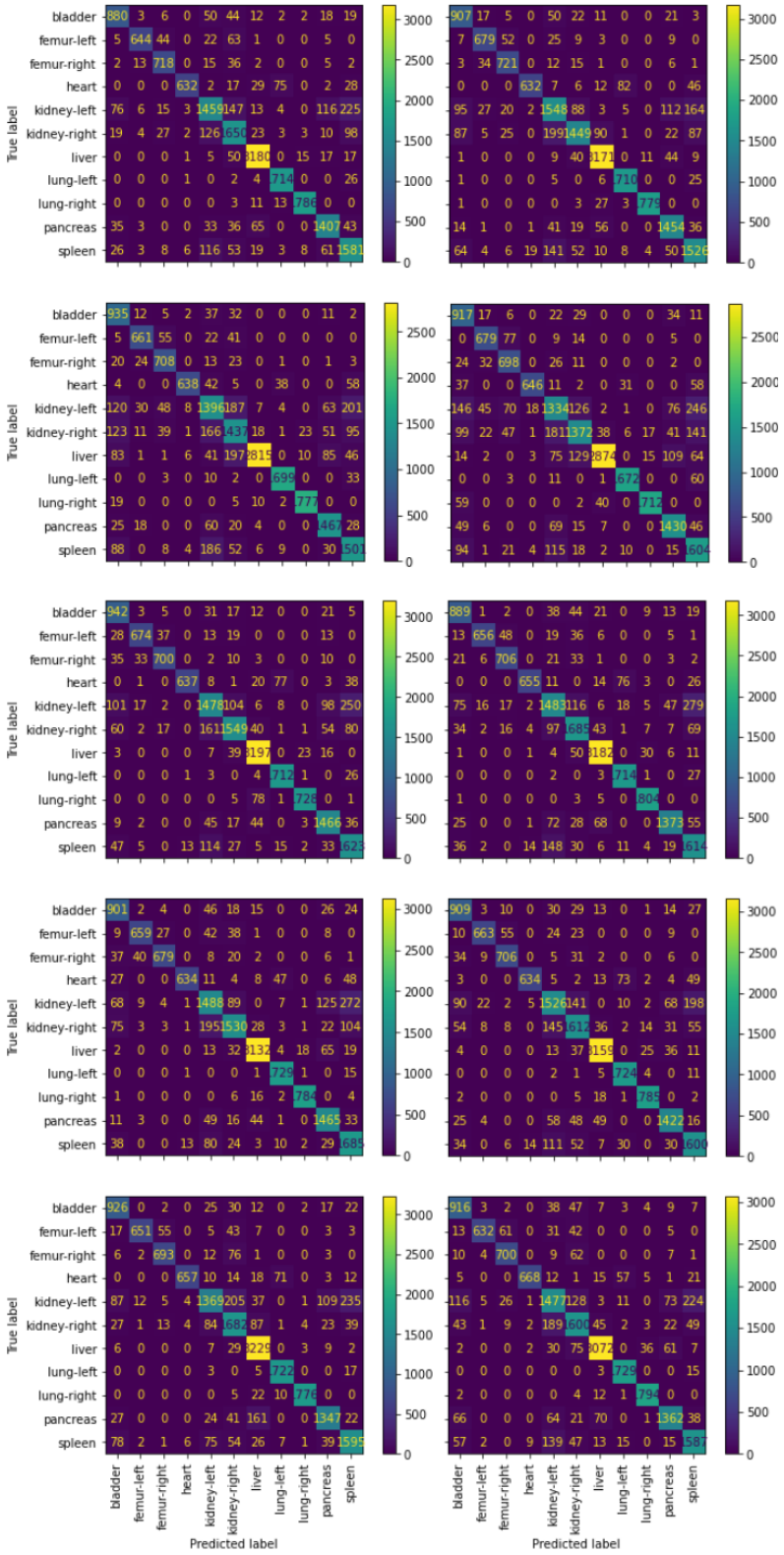

Fig. 42. Confusion matrices on the OrganAMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

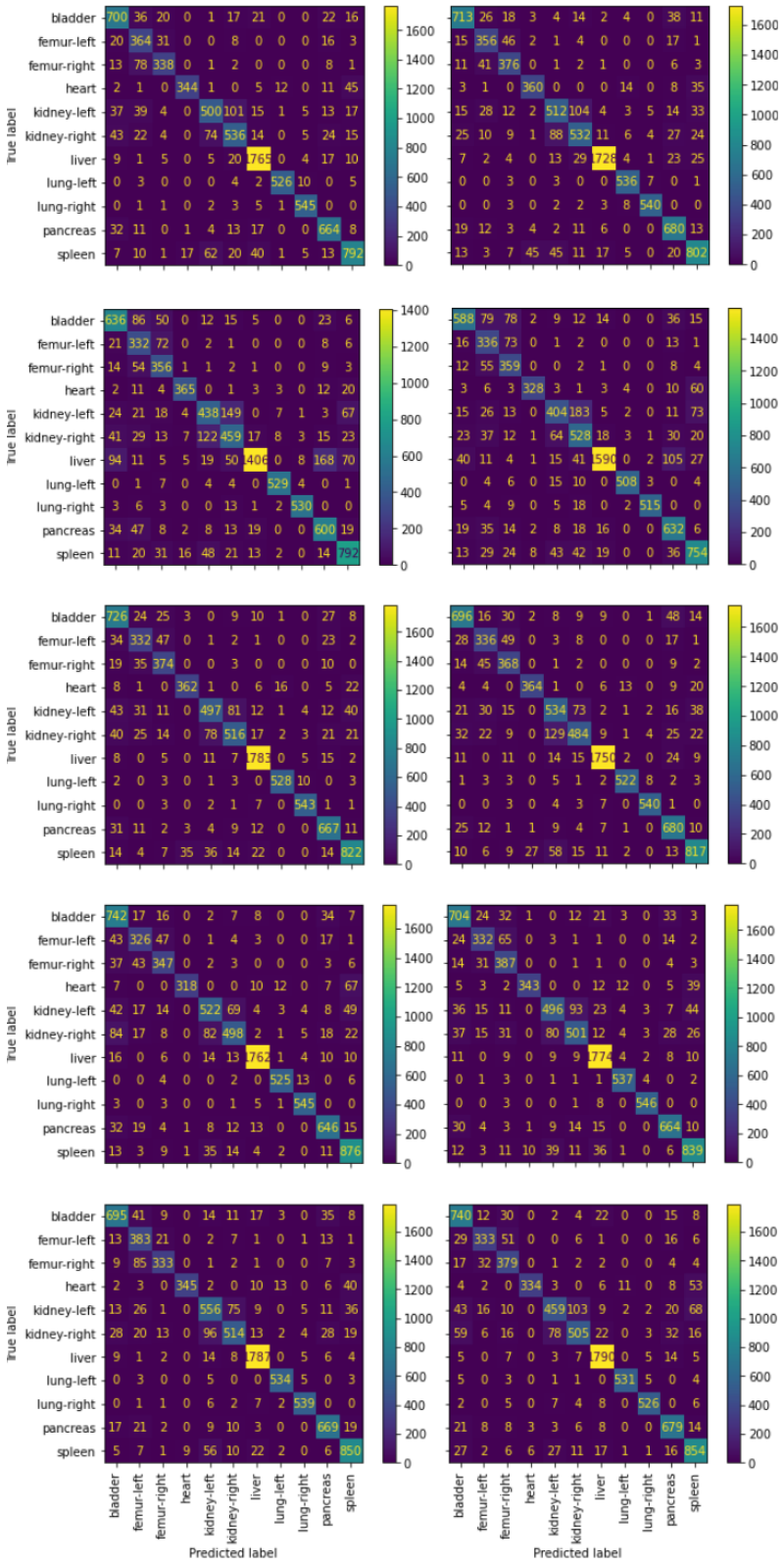

Fig. 43. Confusion matrices on the OrganCMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)

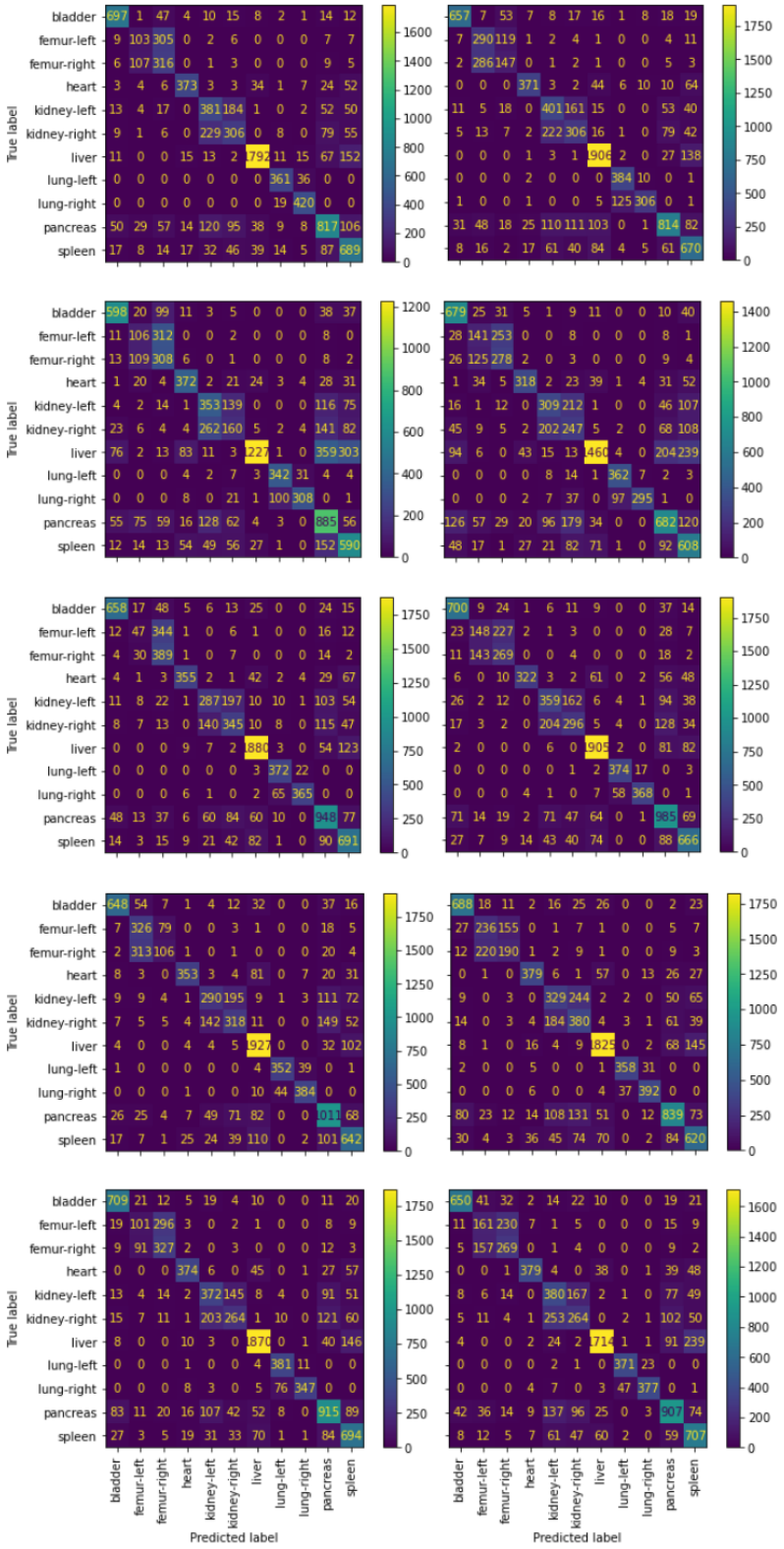

Fig. 44. Confusion matrices on the OrganSMNIST test set given (from top to bottom, left to right) without WRS and geometric DA, with WRS and w/o geom. DA, with WRS and Geom. DA, Geom. DA and w/o WRS, with Conditional VAE augmentation (no WRS then with WRS), with Joint VAE (no WRS then with WRS), Conditional GAN (no WRS then with WRS)