Advanced Deep Learning Mini Project: Data Augmentation

Nicolas E. Fricker[a]

[a]*Université Côte d'Azur, Sophia Antipolis 06902, Alpes-Maritimes, France*

Abstract

For this project we are going to evaluate the quality of generative models for data augmentation in medical domain. You will augment data using both geometric data augmentation, AND a Variational Auto-Encoder AND/OR an Auto-Encoder (Stacked Denoising Auto-Encoder, Denoising Auto-Encoder, or any kind of Auto-Encoder you will decide to use) AND/OR a Generative Adversarial Network (here again, you will choose the GAN model (GAN, DCGAN, Wasserstein GAN, etc).

You will use these 3 kind of models to generate new data and evaluate the (possible) improvement of the final classification score compared with augmenting the training set and without.

Email address: fricker@etu.univ-cotedazur.fr (Nicolas E. Fricker)

Contents

1. Introduction

Machine learning and deep learning models depend heavily on the quality, quantity and relevancy of training data. The most common bottle neck is insufficient data due to its high cost both monetarly and time wise. Data augmentation is a set of techniques to artificially increase the amount of data by generating new data using existing datasets. These techniques include geometric augmentations or deep learning models to generate new data. Geometric augmentation performes small changes in orientation, zoom and shift of the original data. Various deep learning models can be implemented for augmenting data each with their benefits and drawbacks. In this paper we will compare the performance of Convolutional Variational AutoEncoders (CVAE), Deep Convolutional Generative Adversarial Networks (DCGAN), and an combination of the two (CVAEDCGAN) where we combine the decoder from the CVAE and the generator of the DCGAN. Each model is then trained using tensorflow (Abadi et al., 2015) with keras (Chollet and others, 2015) on the original data, and then using random noise we generate a total of 100 new images, which we used to predict the classification with a confidance percentage. We performed augmentation of the MedMNIST 2D datasets (Yang et al., 2021) in Table 1.

Dataset	**Description**	**Label Type (#)**	**Training**	**Validation**	**Test**
PathMNIST	Colon Pathology	Multi-Class (9)	89,996	10,004	7,180
DermaMNIST	Dermatoscope	Multi-Class (7)	7,007	1,003	2,005
OCTMNIST	Retinal OCT	Multi-Class (4)	97,477	10,832	1,000
PneumoniaMNIST	Chest X-ray	Binary-Class (2)	4,708	524	624
BreastMNIST	Breast Ultrasound	Binary-Class (2)	546	78	156
OrganMNIST_Axial	Abdominal CT	Multi-Class (11)	34,581	6,491	17,778
OragnMNIST_Coronal	Abdominal CT	Multi-Class (11)	13,000	2,392	8,268
OrganMNIST_Sagittal	Abdominal CT	Multi-Class (11)	13,940	2,452	8,829

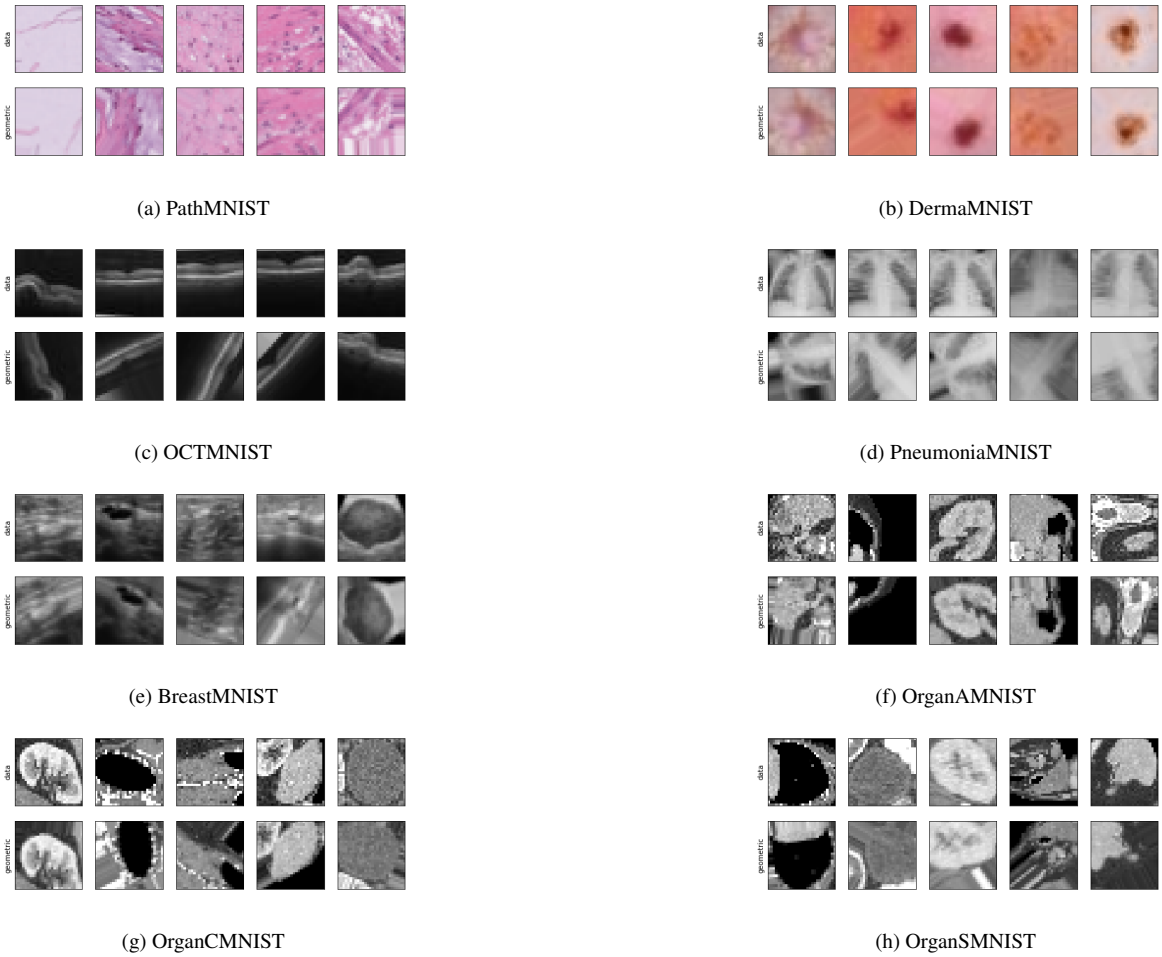
Table 1: **MedMNIST 2D Datasets** Yang et al. (2021)

2. Methods

2.1. Geometric

Geometric augmentation was performed using Keras´ ImageDataGenerator class (*tensorflow.keras.preprocessing.image.ImageDataGenerator*, Chollet and others (2015)). This class allows us to perform a variety of geometric alterations on the original data, we performed random

rotation up to 90°, random horizontally and/or vertical shifts up to 0.2 and randomly flip the image horizontally and/or vertically. In Figure 1, we geometrically augmented each dataset using the first 5 images of each training set with the original on the top row and the geometrically altered version on the bottom row. This method of data augmentation is the fastest since it requires no training.



(a) PathMNIST

(b) DermaMNIST

(c) OCTMNIST

(d) PneumoniaMNIST

(e) BreastMNIST

(f) OrganAMNIST

(g) OrganCMNIST

(h) OrganSMNIST

Figure 1: **Geometric Augmentation**

2.2. CVAE

A Convolutional Variational AutoEncoders (CVAEs) consist of an encoder (Figure 6) and a decoder (Figure 7). The encoder (Figure 6) encodes the data sample x into a latent representation z while the decoder (Figure 7) tries to reconstruct x from the latent vector z. The encoder is a stack of convolutional 2D layers followed by flattening the image and then a fully connected dense layer. We use a reparameterization trick to address the backpropagation bottleneck by approximating

$z = \mu + \sigma \odot \epsilon$ where μ and σ represent the mean and standard deviation of a Gaussian distribution respectively. They can be derived from the decoder output. The ϵ can be thought of as a random noise used to maintain stochasticity of z. Generate ϵ from a standard normal distribution to backpropagate gradients in the encoder and maintain stochasticity through ϵ. This was implemented using a custom keras layer called *Sampling*, by subclassing *tf.keras.layers.Layer*. The decoder uses Convolutional 2D Transpose layers to reconstruct an image from the latent layer. CVAEs train by maximizing the evidence lower bound (ELBO) on the marignal log-likelihood:

$$\log p(x) \geq ELBO = \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] \tag{1}$$

We optimize the single sample Monte Carlo estimate of this expectation:

$$\log p(x|z) + \log p(z) - \log q(z|x), \ \textit{where } z \textit{ is sampled } \textit{from } q(z|x) \tag{2}$$

Although this model is quite fast compared to DCGAN and CVAEDCGAN, since the encoder model is reletavely small compared to the decoder or the discriminator, the quality of the reconstructed images is quite low. Most of the more pronounced features are learned, the finer deatails are very noisy (See Figure 2).

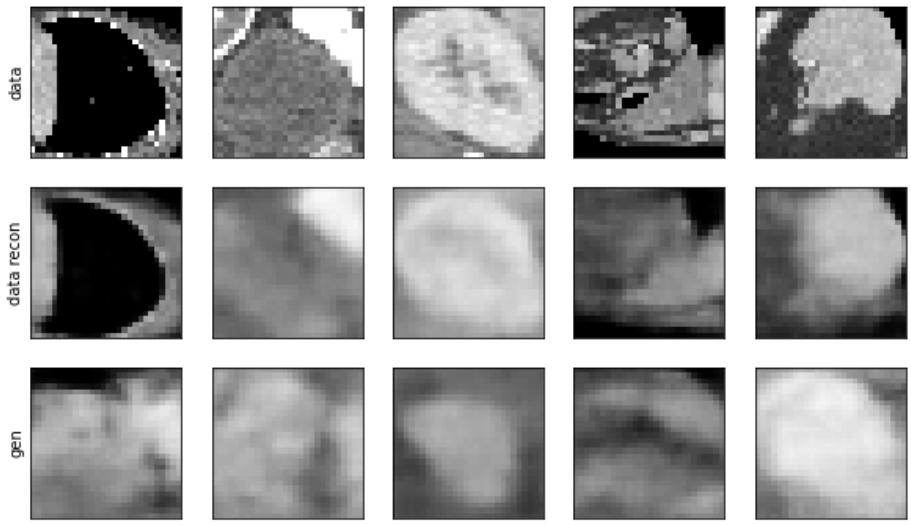


Figure 2: **CVAE OrganSMNIST**

2.3. *DCGAN*

Deep Convolutional Generative Adversarial Networks (DCGANs) are composed of a generator (Figure 7) which learns to create images that look real, and a discriminator (Figure 8) learns to tell real images apart from fakes. The generator is identical to the decoder of the CVAE model, the generator (Figure 7) constructs an image from the latent vector z which is fed into the discriminator (Figure 8). The generator's loss quantifies how well it was able to trick the discriminator. If the generator is performing well, the discriminator will classify the fake images as real (or 1). It compares the discriminator's predictions on real images to an array of 1s, and the discriminator's predictions on fake (generated) images to an array of 0s.

The DCGAN model is slower than the CVAE but faster than the CVAEDCGAN, since it does not have the encoder part, the increased complexity compared to the CVAE introduces greater fine details (See Figure 3).

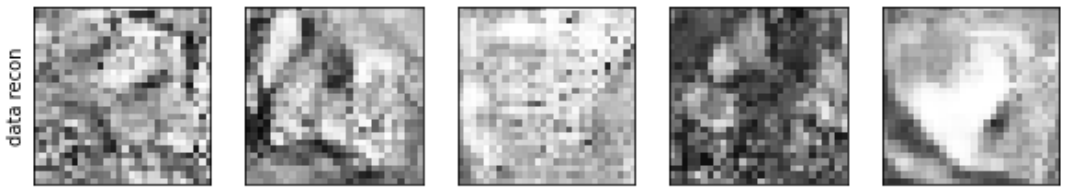


Figure 3: **DCGAN OrganSMNIST**

2.4. *CVAEDCGAN*

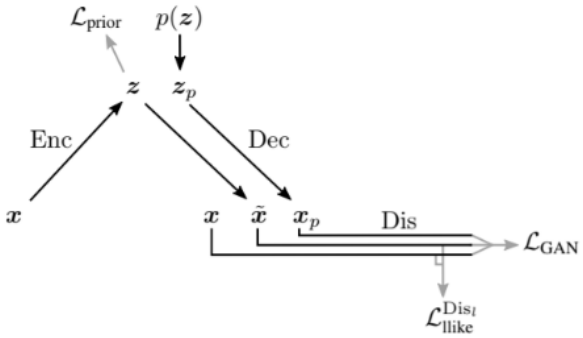


Figure 4: **VAEGAN Diagram** Boesen Lindbo Larsen et al. (2015)

Convolutional Variational AutoEncoder Deep Convolutional Generative Adversial Networks (CVAEDCGAN) previously described (Figure 4) by Boesen Lindbo Larsen et al. (2015) are a

combination of the two previously defined models, by treating the decoder of the CVAE as the generator of the DCGAN (See Figure 7). The encoder (Figure 6) encodes the data sample x into a latent representation z while the decoder (Figure 7) tries to reconstruct x from the latent vector z. This reconstruction is fed into the discriminator (Figure 8) of the DCGAN allowing the model to learn the higher-level sample similarity. Both the CVAE and the DCGAN are trained simultaneously using the loss function:

$$\mathcal{L} = \mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l} + \mathcal{L}_{DCGAN} \tag{3}$$

which is composed of the prior regularization term from the encoder ($\mathcal{L}_{prior}$), the reconstruction error ($\mathcal{L}_{llike}^{Dis_l}$) and the error from the DCGAN ($\mathcal{L}_{DCGAN}$). Each loss during training is computed as described in algorithm 1. With this increase in computation, this model is slower than the two previous but it makes up for the slower training speed by having a better reconstruction which we can visually see the big and fine details (See Figure 5).

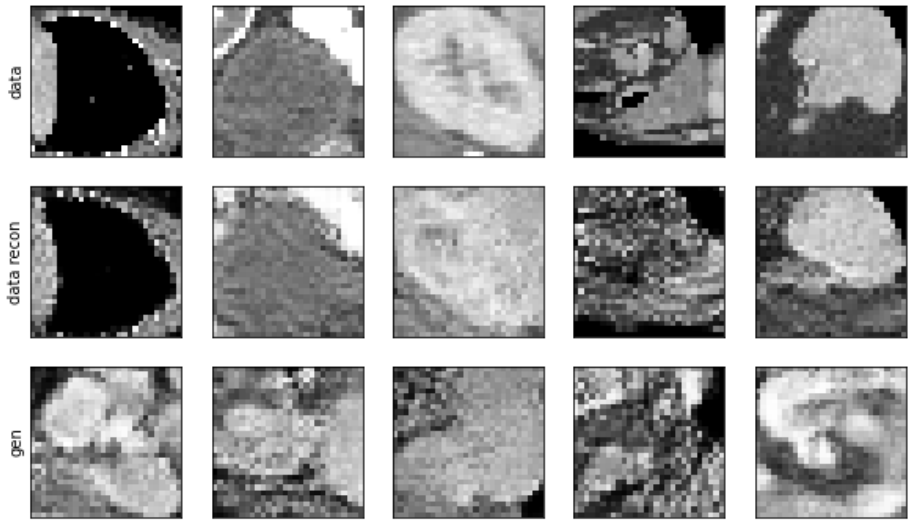


Figure 5: **CVAEDCGAN OrganSMNIST**

Algorithm 1: Training CVAEDCGAN

$\theta_{Enc}, \theta_{Dec}, \theta_{Dis} \leftarrow$ initialize the network parameters
while *dataset is not empty* **do**
  $X \leftarrow$ mini-batch from dataset
  $Z = Enc(X)$
  $\mathcal{L}_{prior} \leftarrow D_{KL}(q(Z|X)\|p(Z))$
  $X \leftarrow Dec(Z)$
  $\mathcal{L}_{llike}^{Dis_l} \leftarrow -\mathbb{E}_{q(Z|X)}\left[p(Dis_l|Z)\right]$
  $Z_p \leftarrow$ samples from prior $\mathcal{N}(0, I)$
  $X_p \leftarrow Dec(Z_p)$
  $\mathcal{L}_{DCGAN} \leftarrow \log(Dis(X)) + \log(1 - Dis(X)) + \log(1 - Dis(X_p))$
  // Update parameters according to gradients
  $\theta_{Enc} \overset{+}{\leftarrow} -\nabla_{\theta_{Enc}}(\mathcal{L}_{prior} + \mathcal{L}_{llike}^{Dis_l})$
  $\theta_{Dec} \overset{+}{\leftarrow} -\nabla_{\theta_{Dec}}(\gamma\mathcal{L}_{llike}^{Dis_l} - \mathcal{L}_{DCGAN})$
  $\theta_{Dis} \overset{+}{\leftarrow} -\nabla_{\theta_{Dis}}\mathcal{L}_{DCGAN}$
end

2.5. Classification

The classifier (Figure 9) is a series of convolutions 2D acivated by LeakyReLU and implements a Dropout layer to allow which randomly sets inputs units to 0 with a certain frequency this helps prevent overfitting. The output layer is the size of the number of labels (classes) for example the PathMNIST has 9 different labels, hence the output layer will have 9 nodes, we use a softmax to transform into values between 0 and 1 with a sum of each node equal to 1.

3. Results

The classifier was trained using 100 epochs and tested on the repsective testing datasets BreastMNIST: 81.41%, DermaMNIST: 66.93%, OCTMNIST: 63.20%, OrganAMNIST: 84.13%, OrganCMNIST: 81.74%, OrganSMNIST: 71.25%, PathMNIST: 80.71%, and PneumoniaMNIST: 85.09%. These percentage is quite low for datasets with less training samples (DermalMNIST & OCTMNIST).

In Table 2, we compared the mean confidance percentage using 100 generated images with each model, we observed, the geometric augmentations had the overall lowest confidance percentage between all 4 models. However supprisingly the DCGAN had a higher confidance level than the CVAEDCGAN even though it is a slight difference. The CVAE performed worse that the DCGAN and the CVAEDCGAN.

Dataset	Geometric	CVAE	DCGAN	CVAEDCGAN
PathMNIST	100.00%	87.07%	88.33%	93.63%
DermaMNIST	46.28%	100.00%	100.00%	100.00%
OCTMNIST	97.98%	94.73%	97.32%	97.25%
PneumoniaMNIST	99.74%	98.46%	99.59%	98.54%
BreastMNIST	57.48%	97.34%	95.37%	92.21%
OrganMNIST_Axial	100.00%	91.02%	92.98%	94.86%
OrganMNIST_Coronal	99.85%	88.41%	86.93%	87.54%
OrganMNIST_Sagital	99.97%	85.78%	90.44%	83.95%
Mean of Mean confidance	87.66%	92.85%	93.87%	93.50%

Table 2: **Mean Confidance**

4. Discussion

Overall, we were able to augment the data for every dataset in Table 1. We can conclude that a DCGAN or a CVAEDCGAN are the best for data augmentations. Potential issues could arrise from the encoder, we suspect there is some overfitting happening. Reworking the Classifier or training it until it gets to about 90% during testing could provide more accurate results. Intoducing more dropout layers or finetuing the rate could help reduce the overfitting. Implementing a validation test analysis could be useful to finetune the model, due to the lack of time/resources it was not viably possible.

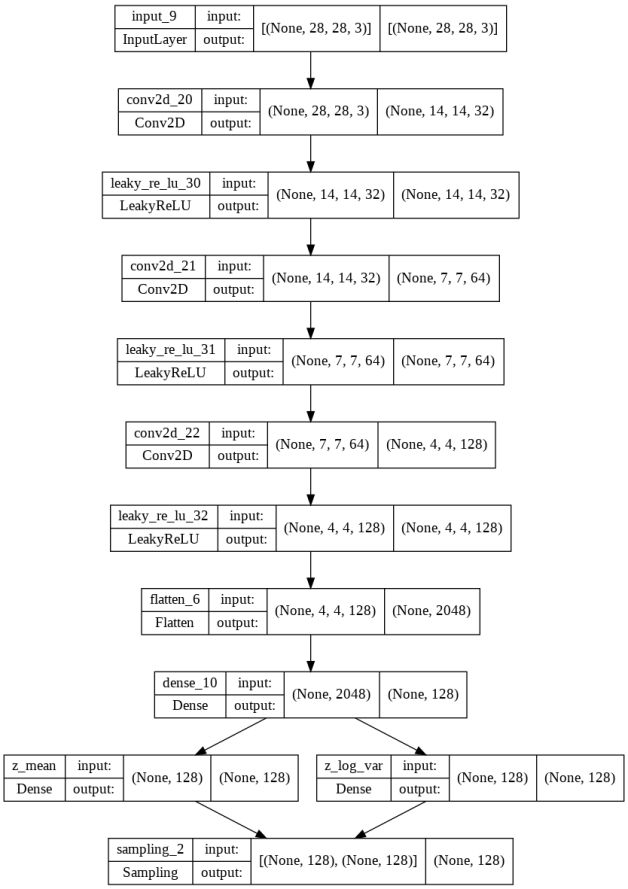


Figure 6: **Encoder:** Example using PathMNIST

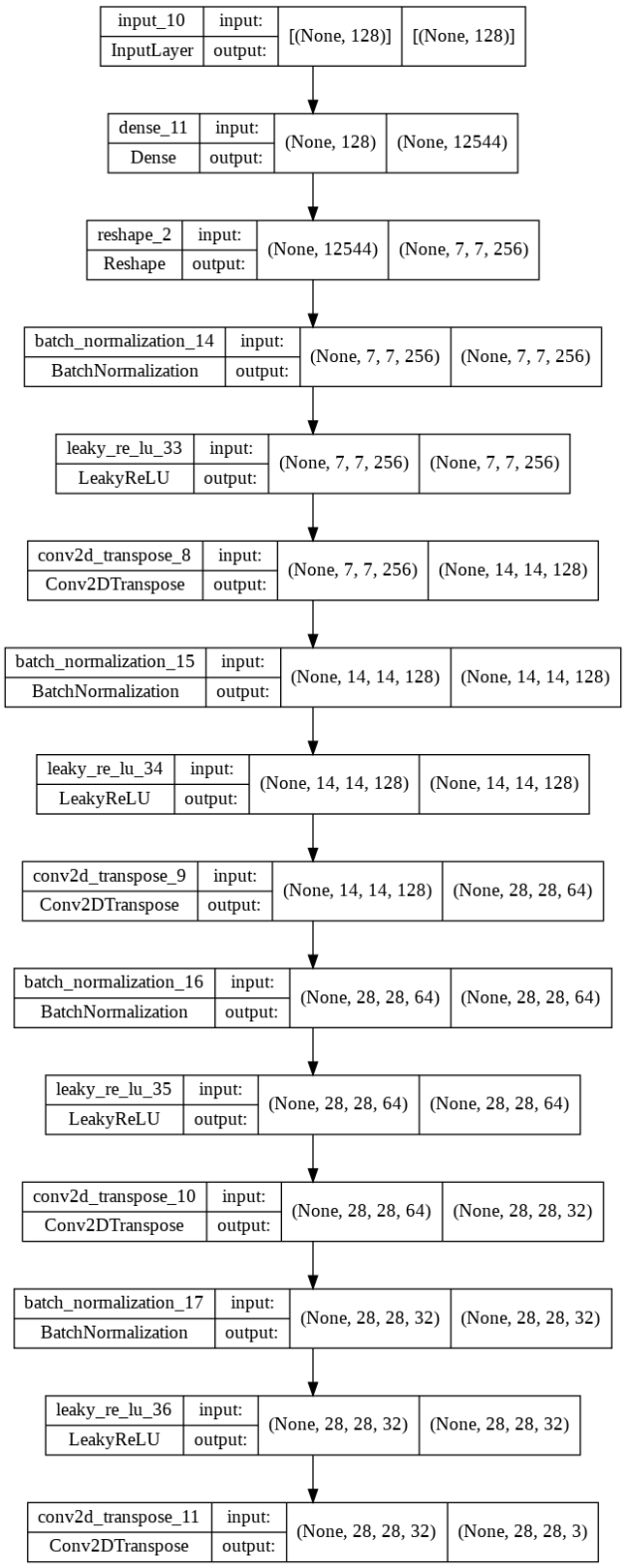


Figure 7: **Decoder / Generator:** Example using PathMNIST

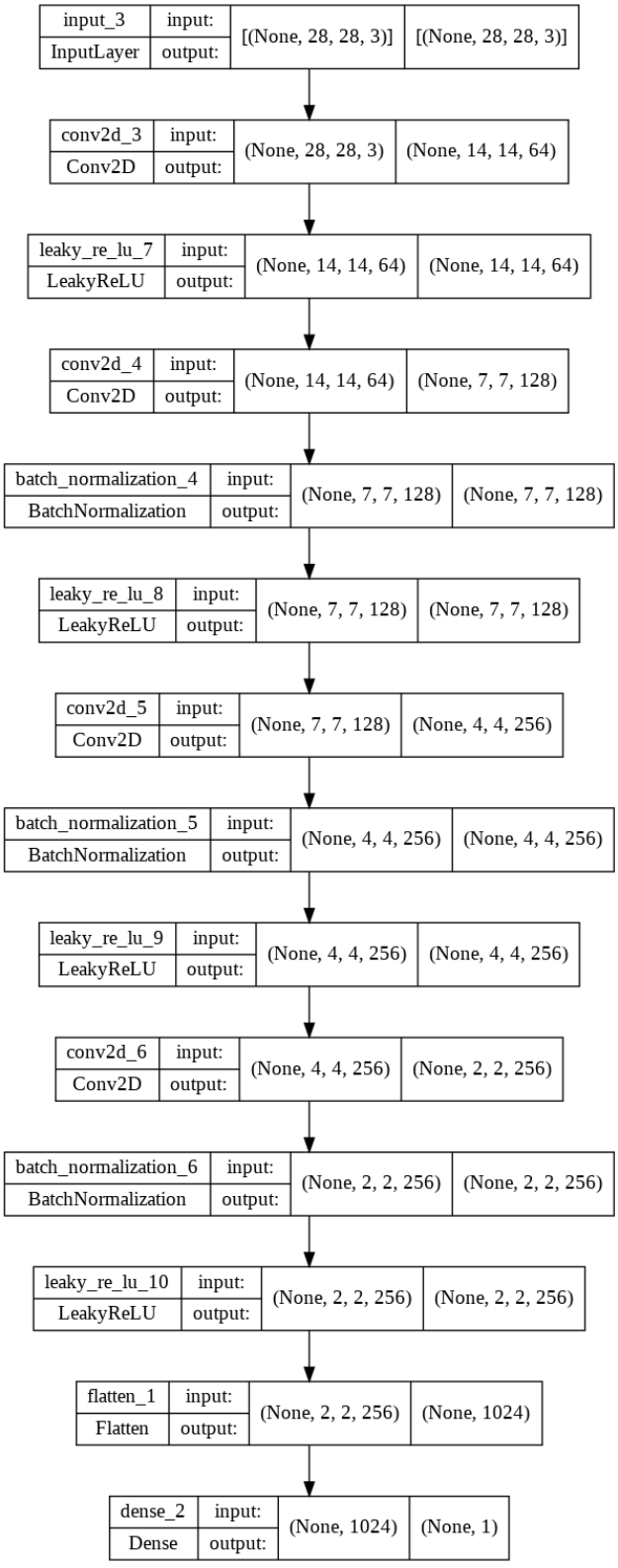


Figure 8: **Discriminator:** Example using PathMNIST

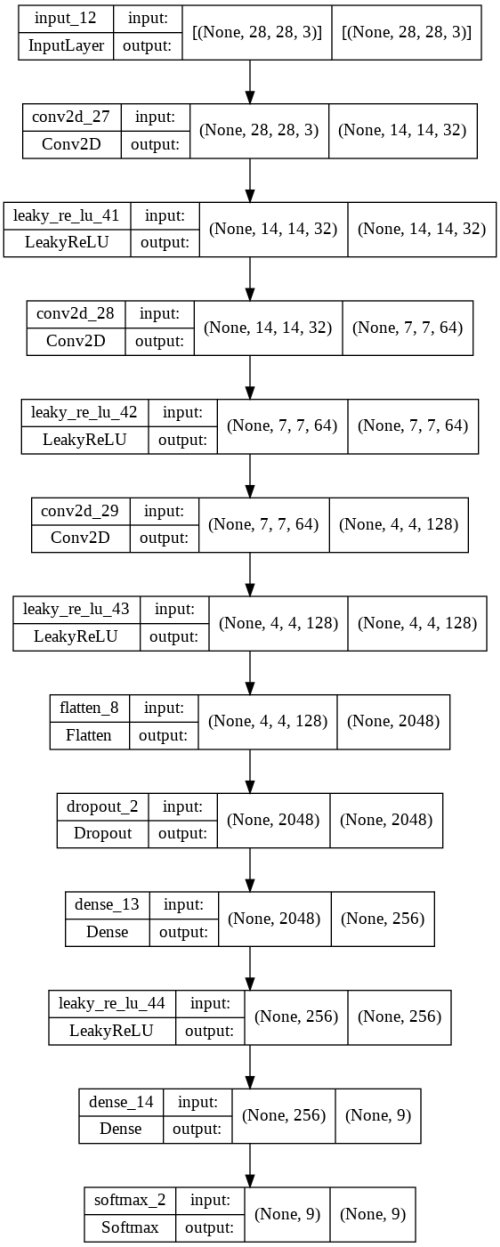


Figure 9: **Classifier:** Example using PathMNIST

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: Large-scale machine learning on heterogeneous systems Software available from tensorflow.org.

Boesen Lindbo Larsen A, Kaae Sønderby S, Larochelle H, Winther O (2015) Autoencoding beyond pixels using a learned similarity metric. *arXiv e-prints* pp. arXiv–1512.

Chollet F et al. (2015) Keras https://keras.io.

Yang J, Shi R, Wei D, Liu Z, Zhao L, Ke B, Pfister H, Ni B (2021) Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795* .