This project is a team of two:
(1) Mokshagna Sai Teja Karanam - U1418261
(2) Abu Zahid Bin Aziz - U1410993

This project code is available on GitHub - https://github.com/mahimoksha/CS6190---ProbabilisticMLProject

1 Introduction

Deep Learning is data hungry and data augmentation has proven extremely useful in improving model generalization performance in medical image analysis tasks; it is very difficult to generalize with fewer data; where in our project, we discuss whether a patient has monkeypox or not with very limited data. There are proven methods where increasing amounts of training samples are valuable for the performance of model in generalization such as organ-at-risk (OAR) segmentations. So, it is very important to provide challenging augmentation to the model to increase the data diversity. In this project, we are trying to solve a task that is to classify if the sample has monkeypox or not with very less data by analyzing with different data augmentation techniques. Our objective is to employ variational autoencoder (VAE) and generative adversarial network (GAN), to generate new samples, which will enrich the number of samples in our training strategy.

We have setup the baselines using conventional augmentation techniques and tried to add probabilistic machine learning based methodologies GAN and VAE to generate more diversified samples. We compared the baseline augmentation methods with probabilistic approaches and compare their performance on same model architectures (ResNet34 and ResNet50). We can see significant improvements to the performance on held-out data when probabilisitc models are used for increasing the sample space.

2 Data Collection

The dataset is primarily compiled from publicly available case reports, news portals, and websites through extensive manual searching [1]. The non monkeypox images are mostly from chicken pox and measles. There are 228 images in total in this dataset. Among them 102 are labeled as monkeypox and the rest are labled as non-monkeypox images. All images are screened and resized to (224 x 224).

3 Data preprocessing

For preprocessing the data we are normalizing the images for all splits with mean 0.485 and standard deviation of 0.229 with the following function:

$$Normalize(mean = (0.485), std = (0.229))$$

3.1 Splitting Data

We have a total of 228 samples. Out of which 102(44.73%) samples has Monkey pox and other 126(55.26%) are normal samples. As the data is almost balanced. Currently, we have tried without

any weights.

Train data contains a total of 154 samples, Validation Data -> 35, train Data -> 39.

4 Data Augmentation Techniques

There's a wide range of different data augmentation techniques available which have been used in different deep-learning tasks to increase the diversity of the dataset. So before applying our probabilistic methodology, we want to establish our baseline by applying these augmentations to our dataset. The basline peformance is summarized in Table 1.

4.1 Generic Augmentation

As part of generic data augmentation, we have tried the following transformations:

- Random horizontal flip
- Random crop
- Random auto contrast
- Random rotation

Using these data augmentations we used pre-trained model with ResNet34 and ResNet50 backbones and evaluated their performance on our dataset. The accuracy for the ResNet50 and ResNet34 model is 76.92 % and 70.01% respectively.

4.2 Random Augmentation

We have applied the RandAugment augmentation techniques from this paper [3] which proposes a simplified search space that vastly reduces the computational expense of automated augmentation and tested them on our dataset. The accuracy for the ResNet50 and ResNet34 model is 74.35 % and 67.71% for the ResNet50 and ResNet34 model respectively.

4.3 Auto Augmentation

we have tried automatic data augmentation, which is Auto Augment with some wide variations in the intensity of the image, appearance, and shape. But there are some limitations of autoaugment which has a very limited scope of usability in terms of efficiency [2]. As at first, it will design a useful search space that has a pre-defined set of transformations with boundaries. and it uses bi-level optimization problems to evaluate augmentation performance using validation data. This method has 2 parameters, probability and magnitude, which are not differentiable.

With hyperparameter tuning we have figured that by using default parameters getting better results which are as follows, Accuracy: 71.7% with resnet34 pre-trained model on ImageNet dataset.

Using ResNet50, we got an accuracy of 76.923 which is better compared with ResNet34 model.

The recall for this technique is TN -> 21, TP-> 9, FN-> 8, FP -> 1:

$$Recall = \frac{TP}{TP + FN} = \frac{9}{17} = 0.52$$

The confusion matrix is as follows: We can observe from the above confusion matrices that ResNet50 is handling False negatives well. improving the performance on True negatives. But still lacking in converging on True positives.

The recall is very low even though the accuracy is 76.923.

4.4 Pre-trained Auto Augmentations

There are some pretrained policies on datasets like CIFAR-10, ImageNet etc.,. torchvision provides these augmentation. out of other pretrained augmentations. Transfer learning on CIFAR-10 dataset augmentation policy on this dataset works better. For ResNet34 Model with no augmentation we got an Accuracy of 69.230%, For the specified augmentation policy on CIFAR-10 dataset with model ResNet34 we got an Accuracy of 74.35%. For ResNet50 Model with no augmentation we got an Accuracy of 79.48%, For the specified augmentation policy on CIFAR-10 dataset with model ResNet50 we got an Accuracy of 84.615%.

We have observed that ResNet50 with AutoAugmentation policy on CIFAR-10 dataset , we are getting very high accuracy of 84.61%. So this value and model will act as a baseline to beat with the future works.

5 Proposed Methodology

5.1 Variational Auto Encoder - VAE

Variational Autoencoders (VAEs) can be used for data augmentation by learning a compact and continuous latent representation of the input data [5]. This latent space can be sampled to generate new data points that have similar characteristics to the original data, thus providing augmented data. The chosen VAE has the following components:

- Encoder (Recognition Model): The encoder is a neural network that maps the input data (e.g., images or text) to a latent space. It takes the input data and outputs two vectors, the mean (σ) and the log-variance ($\log(\mu)$) of the approximate posterior distribution over the latent space, typically a multivariate Gaussian distribution. The purpose of the encoder is to learn an efficient encoding of the input data in the lower-dimensional latent space.
- Reparameterization Trick: This is a technique that enables the backpropagation algorithm to work through the stochastic sampling operation in VAEs. Instead of directly sampling from the approximate posterior distribution, we sample from a standard Gaussian distribution (mean = 0, variance = 1), and then compute the latent vector z using the following equation: $z = \mu + \sigma \otimes \epsilon$. This makes the sampling operation deterministic and differentiable with respect to the encoder's parameters.
- Decoder (Generative Model): The decoder is a neural network that maps the latent vector z back to the original data space. It takes the sampled latent vector z and outputs a reconstruction of the input data. The purpose of the decoder is to learn efficient decoding of the latent space back into the original data space, effectively "reconstructing" the input data.

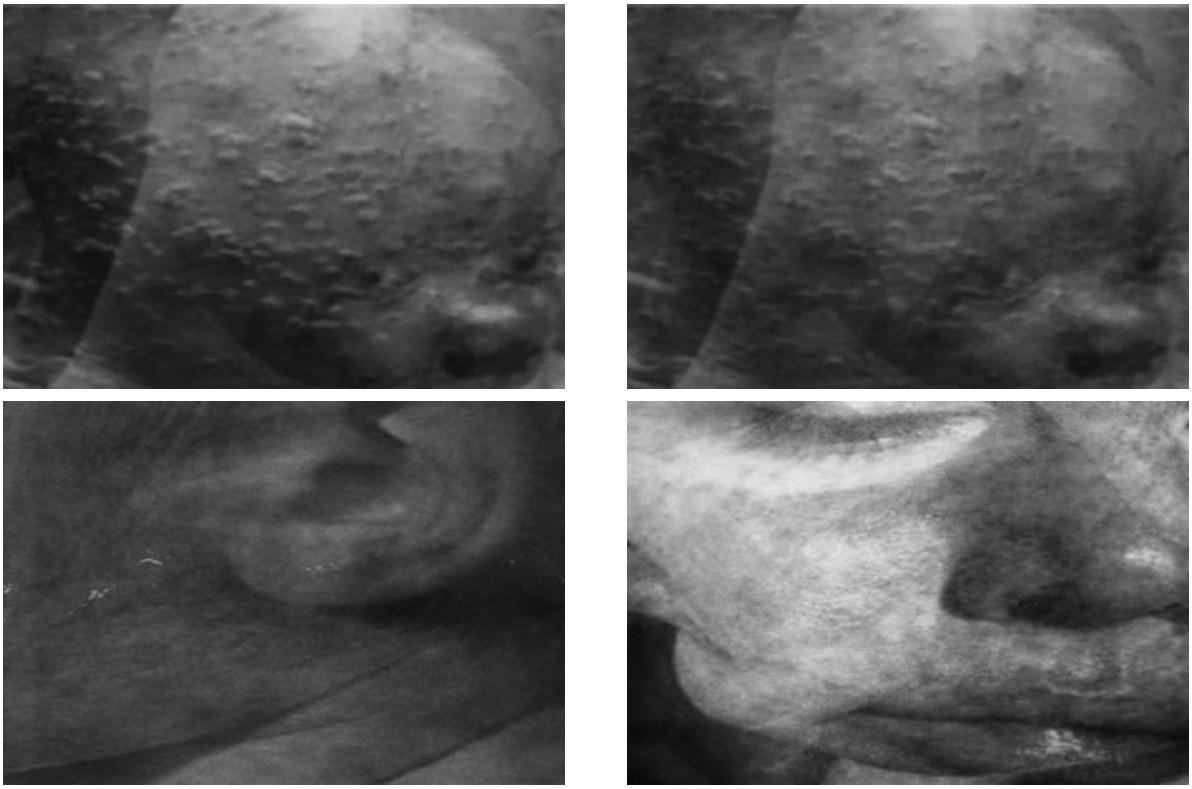
Once the VAE is trained, we can perform data augmentation by sampling new latent vectors z from the prior distribution and passing them through the decoder. This generates new data points that resemble the original data but with variations, effectively augmenting the dataset.

The first row images are positive samples, second row images are negative samples

To see the results: please run the following command python vae.py

The samples are saved in generated_images/.

Following are some of the samples generated from the VAE:



5.2 Generative Adversarial Networks - GAN

So The proposed technique is adapted from the [4] where we have tried to implement intensity generator because implementing deformation generator will change the course of the domain in an image it will be very difficult to provide ground truth samples for generated ones, The data given to Intensity generator structure as follows:

The data is split into 80%-20%, where 80% goes to intensity generator and discriminator will take the remaining images as input.

- **Generator** : The Intensity generator generates a random noise field (Additive Mask), we transform the images by pixel-wise addition. To control intensities we add a TV loss of the generated intensity which will control the pertubation magnitude.

- **Discriminator**: This is a 2D discriminator main goal is to check if the generated images are

Table 1: Performance comparison for different augmentation techniques

Augmentation	Model	Accuracy(%)	TP	TN	FP	FN	Recall
Generic	ResNet50	76.92	12	18	4	5	0.70588
	ResNet34	70.01	11	16	6	6	0.64705
RandAugmentation	ResNet50	74.35	9	20	2	8	0.52941
	ResNet34	67.10	8	18	4	9	0.47058
AutoAugmentation	ResNet50	76.92	9	21	1	8	0.52941
	ResNet34	71.7	10	18	4	7	0.58823
Pretrained Augmentation	ResNet50	84.61	12	21	1	5	0.70588
	ResNet34	74.35	9	20	2	8	52.941
Proposed VAE	ResNet50	**71.12**	**14**	**14**	**3**	**8**	**82.35**
	ResNet34	66.66	9	17	8	5	52.9
Proposed GAN	ResNet50	**82.0513**	**13**	**19**	**3**	**4**	**81.4171**
	ResNet34	79.487	13	18	4	4	79.14438

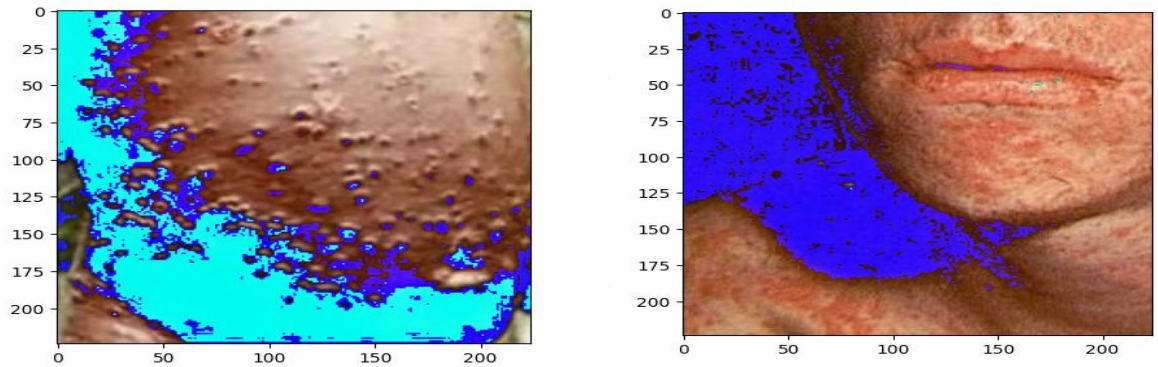
in distribution or out of distribution. This will ensure the generated samples are not far from the diversity of the input dataset.

- **Training**: This method is a multistage training where first we will train the model with the generated samples. and fine tune the model with real data. First the images will be generated from Intensity Generator (IG) and send it to the classifier Target Model (T) (resnet34 or resnet50). To keep the generated images in distirbution Discriminator (D) will come into picture. This type of architecture will also have an objective of generating more challenging augmentations to let the target model learn robust feature and will reduce overfitting issue as well. The adversary will takes place when the gradients are returning from the target model where grad reverse is performed to convert the objective from minimize to maximize that's where challenging samples will be generated in the further epochs.

To see the results: please run the following command python monkey_pox_noise.py

The samples will be saved in results/Generator_samples_intensity/iteration_*/.

Following are some of the samples generated from the GAN:



First is the positive sample, the second sample is the negative sample

6 Discussions

For the C-GAN and DC-GAN models, we observe that the models are not able to generalize because of less number of training samples. After training with different hyperparameters the models are giving random gaussian noise samples as outputs with very little information about the region of interest (disease). So we came up with the proposed approach which has resulted in better performance compared to other augmentation strategies. Even with ResNet34 architecture the model is giving significant variation (stand out performance) in Recall Score compared to other strategies. Although the Accuracy is not as good as the baseline pretrained augmentation results we can see a significant increase in Recall Score for the proposed GAN model. This indicates the completeness of positive predictions of monkeypox which is very important in medical imaging domains.

For the VAE models, we have tried two different approaches interms of model architectures the CNN based VAE was generating repetitive and noisy samples where as the linear layer based VAE provided much better and realistic samples (as seen in proposed methodologies). as a result we can see good performance for the ResNet50 model where this approach achieves the best Recall score.

We can extend the model architectures in proposed GAN generator by introducing the Affine generator, Intensity and Affine generator together, as affine adds more variations to the final target model and adding noise to it makes it generate more challenging augmentations. So adding them in future can result in good improvements in the performance.

Our proposed methodologies are trained on ResNet34 and ResNet50 models. Trying more pretrained models may result in better performance.

References

[1] Shams Nafisa Ali et al. "Monkeypox Skin Lesion Detection Using Deep Learning Models: A Preliminary Feasibility Study". In: *arXiv preprint arXiv:2207.03342* (2022).

[2] Ekin D Cubuk et al. "Autoaugment: Learning augmentation policies from data". In: *arXiv preprint arXiv:1805.09501* (2018).

[3] Ekin D Cubuk et al. "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 702–703.

[4] Yunhe Gao et al. "Enabling data diversity: efficient automatic augmentation via regularized adversarial training". In: *Information Processing in Medical Imaging: 27th International Conference, IPMI 2021, Virtual Event, June 28–June 30, 2021, Proceedings*. Springer. 2021, pp. 85–97.

[5] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).