NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

GENERATIVE AI PROJECT REPORT

ON

PASCAL CHALLENGE ON CLASSIFICATION

SUBMITTED BY

MAHAD HAMEED

20K-0338

ALISHBA SUBHANI

20K-0351

MANNAHIL MIFTAH

20K-0234

SUBMITTED TO

Dr. Muhammad Atif Tahir

Contents

0.1 PROBLEM STATEMENT

The Pascal VOC2008 challenge focuses on the task of object detection and classification. The problem is to classify the objects in the dataset of images into the following twenty classes.

1. person
2. bird, cat, cow, dog, horse, sheep
3. aeroplane, bicycle, boat, bus, car, motorbike, train
4. bottle, chair, dining table, potted plant, sofa, tv/monitor

0.2 DATA

The VOC2008 database contains a total of 10,057 annotated images of which the train set has 2111 samples, and the validation set has 2221 samples. For each image in the sets, there is an annotation file which contains the labels for each object in that image. This also means that there can be multiple objects in an image.

Following is an example of an image with the labels of 'dining table', and 'person' signifying their presence in the image.

Figure 1: Sample Image

0.3 EVALUATION

A precision/recall(PR) curve will be used for evaluation purposes. It provides a visual representation of the trade-off between precision and recall. The principal quantitative measure used will be the average precision (AP). Average Precision is a common metric

for image classification, detections tasks, and it summarizes a PR curve as its defined as the “area under the PR curve”.

0.4 CLASSIFICATION

0.4.1 Pseudocode

Download and extract PASCAL VOC 2008 dataset
Identify labels
Read training and validation file lists
Define function to extract labels from XML annotations
Extract labels for training and validation sets
Encode labels into one-hot format
Create DataFrames for training and validation sets
Concatenate one-hot encoded DataFrame with main DataFrame
Append ’.jpg’ extension to ’path’ column values
Write DataFrame to CSV file
Read CSV file into DataFrame and display it

Define the VGG16 Model as the base model
    Set constants for optimizer, loss function, and input shape
    Load VGG16 model with pre-trained ImageNet weights, excluding the top layer
    Freeze all layers in the base model
    Add additional layers to the model
    Compile the model with specified optimizer, loss, and metrics
    Return the compiled model

Define data generator for training and validation
Set constants for number of epochs, batch size, patience, and minimum delta
Create the output models directory if it does not exist
Iterate over each label in the dataset
    Check if the model for the current label has already been trained or not
    Create a model using the Model function
    Create data generators for training and validation
    Set up early stopping callback with specified parameters
    Train the model using the fit method
    Save the trained model

0.4.2 Precision Recall Curve

For each class, a binary VGG model was trained.For each class, the Precision-Recall curve, and average precision is given below:

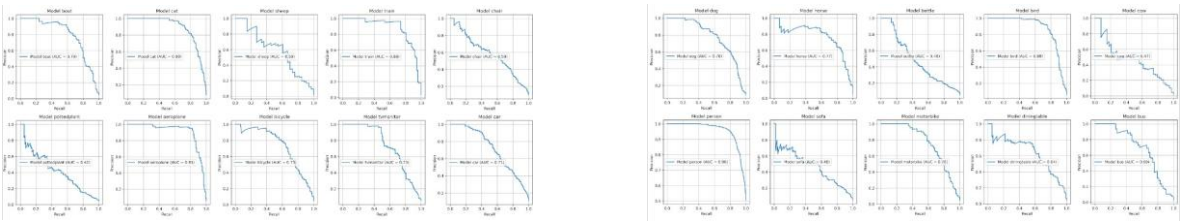


Figure 2: Precision-Recall Curve for each class

0.4.3 Highest Ranked Images

Best Model

The highest performing model was for the class Person due to its higher support comparatively, with an AP of 0.96. Following are the highest confidence images belonging to class Person.

Figure 3: Class Person

Worst Model

The lowest performing model was for the class Potted Plant due to a low support, with an AP of 0.42. Following are the highest confidence images belonging to class Potted Plant.

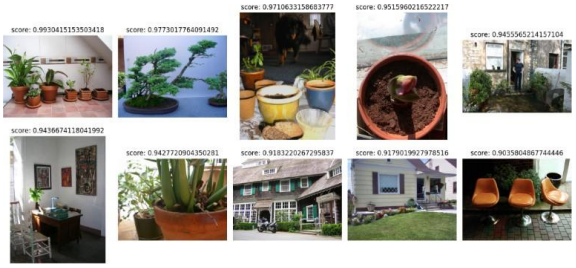


Figure 4: Class Potted Plant

0.4.4 Mean Average Precision

The mean average precision is 0.706. There is a very obvious correlation between the support of each class, with the average precision for it. For example, the person class has 1022 examples in the training set, and has the best AP of 0.96, whereas the classes such as sheep, cow, potted plant have less than 100 examples, and it shows with a low AP.

0.5 DATA AUGMENTATION

The current hypothesis is that by applying data augmentation, and generating more samples, the performance of the models of classes with really low support can be improved. To address this, both a Variational Autoencoder (VAE) and a Deep Convolutional Generative Adversarial Network (DCGAN) are employed. The aim is to mitigate the challenge of class imbalance, thereby aiding in the enhancement of model performance, particularly for classes with low support.

0.6 VARIATIONAL AUTOENCODERS

0.6.1 Pseudocode

Define the VAE model class
Create dictionaries to store models, optimizers, and training statistics for each label
Iterate over each label and train loader pair
    Initialize the model and optimizer
    Initialize training statistics
    Iterate over each epoch
        Initialize lists to store mu and logVar for the epoch
        Initialize total loss for the epoch
        Iterate over each batch in the train loader
            Get the input images and labels
            Move images to the appropriate device
            Forward pass through the model to get the output, mu, and logVar
            Resize the output images
            Compute the loss using binary cross-entropy and KL divergence
            Perform backpropagation
            Update the model parameters
            Collect mu and logVar for the epoch
            Update the total loss for the epoch
            Append the resized output images to improvements

Compute the average loss for the epoch
Collect training statistics for the epoch
Print label, epoch, and average loss

0.6.2 Training Process

Following are the images of training process for class person. During the training process of the Variational Autoencoder (VAE), the model exhibited progressive learning as evidenced by its gradual improvement in reconstructing the displayed images over time.

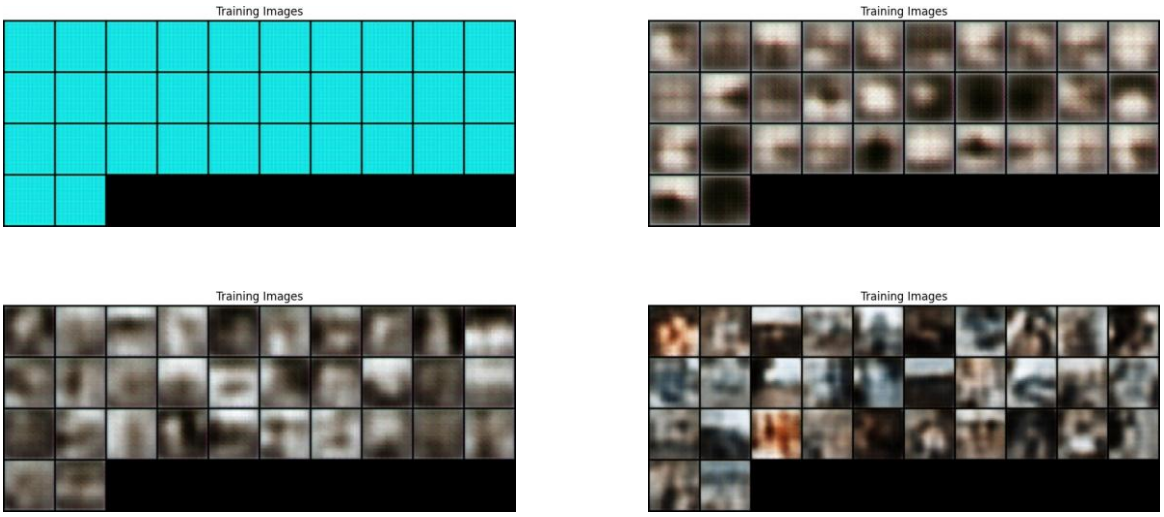
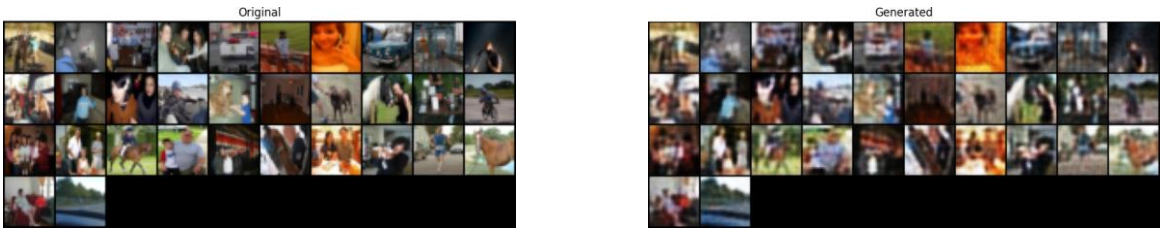


Figure 5: Images during training process

0.6.3 Results

Showing results of VAE for person class. Total 32 images have been reconstructed for each class.



(a) Original Images **(b)** Reconstructed Images

Figure 6: VAE Results

0.6.4 Classification

Before training each class-specific binary VGG model, reconstructed images were augmented to the original dataset (training set only) for that class. Specifically, experiments were conducted with augmenting 10, 25, and 32 samples, respectively. For each sample, the Precision-Recall curve and average precision are presented below.

10 samples per class

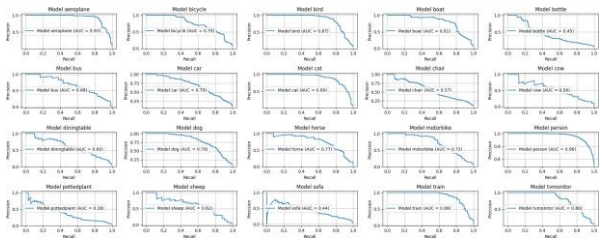


Figure 7: Precision-Recall Curve of 10 samples

25 samples per class

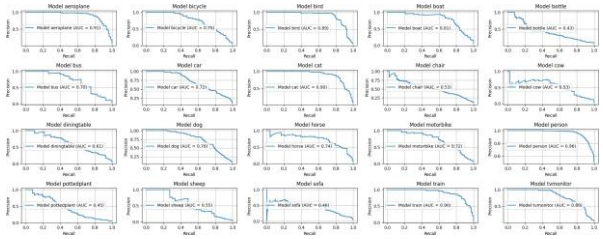


Figure 8: Precision-Recall Curve of 25 samples

32 samples per class

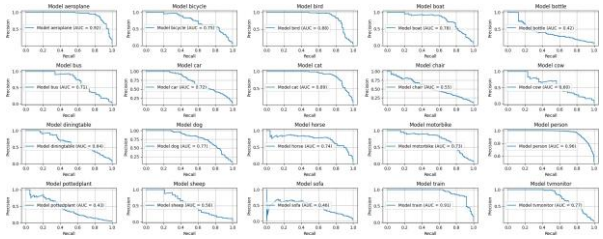


Figure 9: Precision-Recall Curve of 32 samples

0.6.5 Mean Average Precision

10 samples per class

The mean average precision is 0.70854. The person class has the best AP of 0.96, whereas, the potted plant has the lowest AP of 0.38.

25 samples per class

The mean average precision is 0.70596. The person class has the best AP of 0.96, whereas, the bottle class has the lowest AP of 0.43.

32 samples per class

The mean average precision is 0.70975. The person class has the best AP of 0.96, whereas, the bottle class has the lowest AP of 0.42.

Analysis

The initial VGG training on the Pascal VOC dataset achieved a mean average precision (mAP) of 0.706. However, upon observing class imbalances, particularly with fewer images for some classes, a Variational Autoencoder (VAE) was employed to reconstruct additional images to address this issue.

Augmenting the dataset with 10 reconstructed images per class led to a slight improvement in mAP to 0.7085. However, increasing the number of reconstructed images to 25 resulted in a minor decrease in mAP to 0.7059. This outcome suggests that while additional data augmentation can benefit model performance, there may be diminishing returns beyond a certain threshold, possibly due to the introduction of noise or irrelevant variations. Adding 32 reconstructed images per class yielded a slight improvement in mAP to 0.7097.

Hence, it can be concluded that while data augmentation can enhance model performance, there's a balance to strike between augmentation efforts and maintaining dataset quality for optimal results.

0.7 DEEP CONVOLUTIONAL GENERATIVE ADVERSIAL NETWORK

0.7.1 Pseudocode

Discriminator:

Convolutional Layer: 3–64, 4x4, stride=2, padding=1, BN, LeakyReLU(0.2)
Convolutional Layer: 64–128, 4x4, stride=2, padding=1, BN, LeakyReLU(0.2)
Convolutional Layer: 128–256, 4x4, stride=2, padding=1, BN, LeakyReLU(0.2)
Convolutional Layer: 256–512, 4x4, stride=2, padding=1, BN, LeakyReLU(0.2)
Convolutional Layer: 512–1, 4x4, stride=1, padding=0, Sigmoid Flatten

Generator:

Transposed Convolutional Layer: latent–512, 4x4, stride=1, padding=0, BN, ReLU

Transposed Convolutional Layer: 512–256, 4x4, stride=2, padding=1, BN, ReLU
Transposed Convolutional Layer: 256–128, 4x4, stride=2, padding=1, BN, ReLU
Transposed Convolutional Layer: 128–64, 4x4, stride=2, padding=1, BN, ReLU
Transposed Convolutional Layer: 64–3, 4x4, stride=2, padding=1, Tanh

Training:

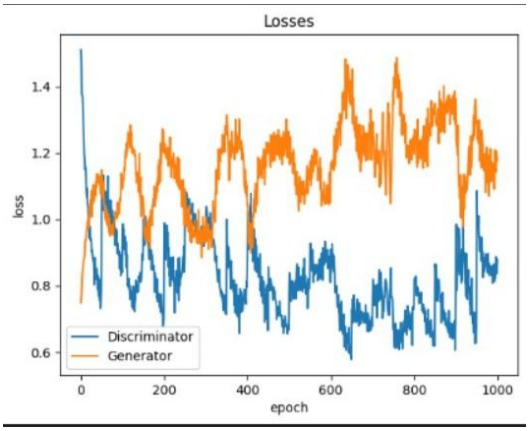
```
Fit(epochs, lr, start_idx=1):
    Empty GPU cache
    Initialize lists for losses_g, losses_d, real_scores, fake_scores
    Create Adam optimizers for discriminator and generator
    For each class:
        Filter dataset
        For each epoch:
            For each batch:
                Train discriminator
                Train generator
            Record losses and scores
            If epoch divisible by 1: Save samples
        Save models
    Return losses_g, losses_d, real_scores, fake_scores
```

0.7.2 Training Process

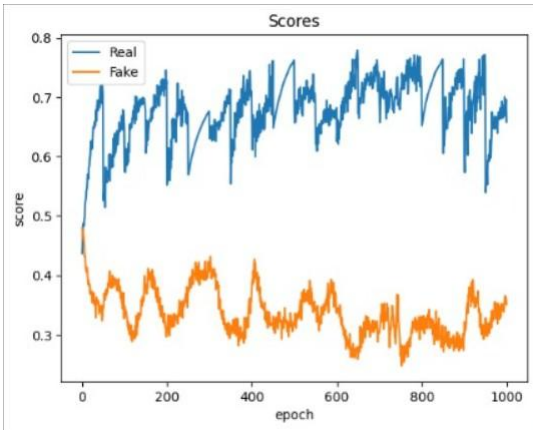
We initially trained the model for 25, 100, and 500 epochs, but the results weren't satisfactory; the generator loss kept increasing, and there were significant fluctuations in losses. Upon reassessing our data, we adopted a different approach. We trained the model for 25 epochs first, then used the output as a pretrained model for subsequent runs.

We observed improvements in image generation with this approach. We repeated the process, loading the current model output for the next run. Notably, we only loaded the generator model as a pretrained model, keeping the discriminator model stock.

After repeating this process 3 to 4 times, we noticed considerable improvements in image quality. We concluded that repeating this process would likely lead to even better images after a few more runs.



(a) Loss during training



(b) Real & Fake scores

Real and fakes scores generated by discriminator represents the image generated by generator is either real or fake. If the fake score gets higher than real score in earlier stages then the generator is too strong to fool the discriminator. According to our results, it has started to converge and at some point there will trade off scores and better image generation.

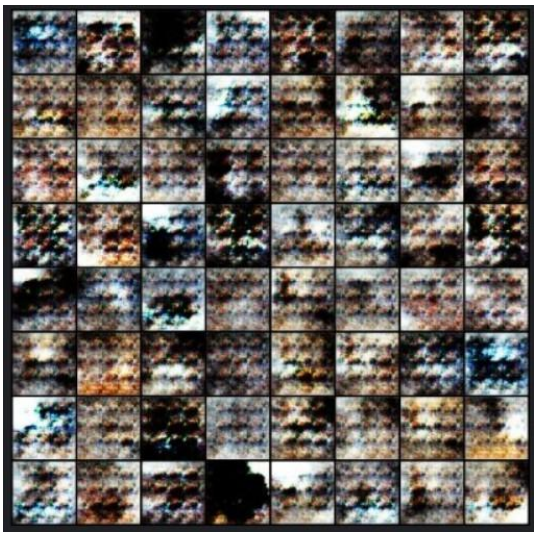
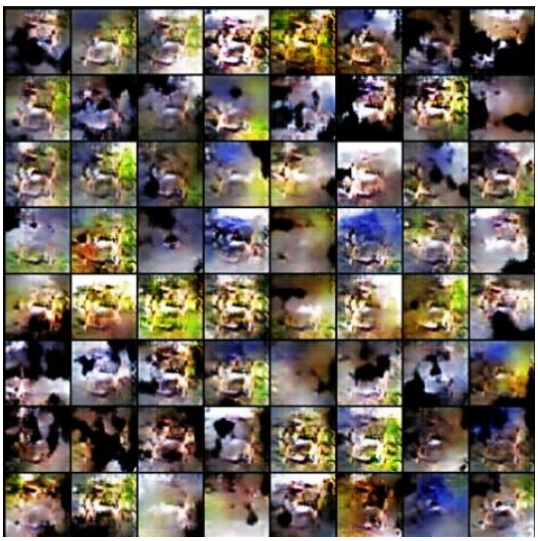


Figure 11: Images during training process

0.7.3 Results

Showing results of VAE for person class. Total 32 images have been reconstructed for each class.

(a) Reconstructed Images of Class Boat



(b) Reconstructed Images of Class Dog

Figure 12: DCGAN Results

0.7.4 Classification

Before training each class-specific binary VGG model, reconstructed images were augmented to the original dataset (training set only) for that class. Specifically, experiments were conducted with augmenting 25 samples per class. The Precision-Recall curve and average precision are presented below:

25 samples per class

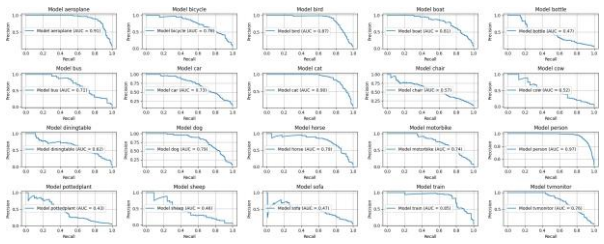


Figure 13: Precision-Recall Curve of 25 samples

0.7.5 Mean Average Precision

The mean average precision is 0.70619. The person class has the best AP of 0.91, whereas, the potted plant has the lowest AP of 0.43.

0.7.6 Analysis

The consistent mAP value of 0.70619 after the addition of DCGAN-generated images suggests that while the DCGAN augmentation helped to mitigate class imbalance, the quality of the generated images might not have been sufficient to significantly impact

the overall classification performance. Despite adding 25 sample images per class, the noise and imperfections in the generated images may have introduced confusion or uncertainty in the VGG model's predictions, leading to little improvement in mAP. Although the augmentation addressed class imbalance, the noise introduced by DCGAN could have offset the benefits, resulting in minimal change in mAP. To improve the mAP, further refinement of the DCGAN model to generate more realistic and less noisy images which requires more resouces.

0.8 CONCLUSION

In conclusion, this project addressed class imbalance in a dataset of 20 classes through data augmentation techniques using Variational Autoencoder (VAE) and Deep Convolutional Generative Adversarial Network (DCGAN). The VGG model was initially trained for image classification, and then, to handle the class imbalance problem, VAE and DCGAN were used to generate additional images for each class, which were incorporated into the training set. Despite the noise and imperfections in the generated images, this approach aimed to improve the classification performance. However, further refinement of the generated images or exploring alternative data augmentation methods may be necessary for more significant improvements in classification accuracy.

0.9 REFERENCES

Goodfellow, Ian, et al. ”Generative adversarial nets.” In Proceedings of the 27th International Conference on Neural Information Processing Systems, pp. 2672-2680. 2014.

Kingma, Diederik P., and Max Welling. ”Auto-encoding variational bayes.” In Proceedings of the 2nd International Conference on Learning Representations, San Diego, CA, USA, 2014.

Radford, Alec, Luke Metz, and Soumith Chintala. ”Unsupervised representation learning with deep convolutional generative adversarial networks.” In Proceedings of the 4th International Conference on Learning Representations, San Juan, Puerto Rico, 2016.

Simonyan, Karen, and Andrew Zisserman. ”Very deep convolutional networks for large-scale image recognition.” In Proceedings of the 3rd International Conference on Learning Representations, San Diego, CA, USA, 2015.

Radford, Alec, Luke Metz, and Soumith Chintala. ”Unsupervised representation learn-

ing with deep convolutional generative adversarial networks." Proceedings of the 4th International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2016.

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In Proceedings of the 34th International Conference on Machine Learning (ICML), Sydney, Australia,2017.