Revolutionizing Image Synthesis and Classification: A Practical Exploration of DCGANs and CNNs

Yi Li
Halıcıoğlu Data Science Institute
UC San Diego
yil115@ucsd.edu

Weiyue Li
Halıcıoğlu Data Science Institute
UC San Diego
wel019@ucsd.edu

Abstract

Our project aims to revolutionize image synthesis and classification by combining Deep Convolutional Generative Adversarial Networks (DCGANs) with Convolutional Neural Networks (CNNs). We seek to demonstrate the potential of DCGAN techniques in producing highly realistic images and achieving strong performance in image classification by utilizing these synthetic images during training. For this purpose, we select the Intel Natural Scene dataset. Following processing with the DCGANs model, we generate natural scene images that are extremely authentic. To assess the performance of our method, we feed the synthetic images into a pre-trained ResNet-18 model and undertake the classification task using actual testing data. Our approach yields impressive results, achieving a 78.1% 6-class classification accuracy.

1 Introduction

1.1 Understand of the Problem

It is well known that acquiring natural scene images demands considerable time and human resources. For example, if our goal is to obtain 5,000 distinct glacier images, we first need to engage photographers to traverse the globe, identify glaciers, and take photographs from multiple angles. Subsequently, we must discuss with the photographers whether they would be willing to donate their images to our dataset. After securing their permission, we must preprocess and choose suitable images for our dataset. Lastly, it is essential to label these images. This entire process can span several years.

1.2 Motivation of Solving this Problem

Consequently, rather than manually amassing millions of images, we have opted to begin with a few thousand images and utilize DCGANs (Deep Convolutional Generative Adversarial Networks) to produce the remaining images. This method will undoubtedly conserve significant human, financial, and temporal resources.

1.3 Key Parts of the Approach

DCGANs are composed of two main elements: a Generator and a Discriminator. The Generator processes real images as input and extracts various features from them to create realistic images. The Discriminator carries out binary classification tasks to ascertain whether the input image is real or counterfeit. While the artificial images generated by the DCGAN model might seem slightly blurry, each image possesses unique features that can offer additional information to Convolutional Neural Networks (CNNs), enabling them to develop a more effective classifier.

2 Related Work

2.1 Deep Convolutional Generative Adversarial Networks (DCGANs)

A Generative Adversarial Network (GAN) is an unsupervised machine learning model, first introduced by [Ian et al.(2014)Ian, Jean, Mehdi, Bing, David, Sherjil, ..., and Yoshua], consisting of two deep neural networks, the Generator and Discriminator, which collaborate to generate highly realistic images. The Generator aims to enhance the quality of synthetic images, while the Discriminator focuses on improving its ability to differentiate between real and fake images. The adversarial process continues until the fabricated images become virtually indistinguishable from real ones. However, training GANs can be challenging due to stability issues. To address this, [Radford et al.(2015)Radford, Metz, and Chintala] proposed the DCGAN architecture, a GAN variant that replaces deterministic spatial pooling layers with stride convolutional layers in both the Generator and Discriminator, resulting in improved training stability and higher-quality generated images.

2.2 ResNet-18

ResNet18, introduced in 2015 by [He et al.(2016)He, Zhang, Ren, and Sun], is a convolutional neural network (CNN) architecture pre-trained for image classification tasks. The network comprises 18 layers, including 17 convolutional layers and one fully connected layer. What sets ResNet18 apart is its utilization of residual connections, enabling the training of very deep networks without the issue of vanishing gradients. Residual connections bypass one or more layers and incorporate the output of skipped layers with the output of the current layer, facilitating the learning of residual functions instead of learning the entire function from scratch.

3 Method

3.1 Fake Images Generation

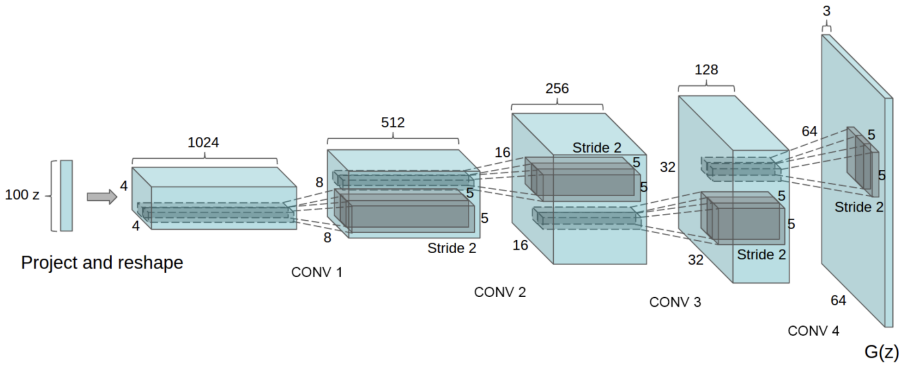


Figure 1: Architecture of DCGANs

3.1.1 Primary Structure (1) of DCGANs

- Generator: The generator is a deep neural network that accepts 100-dimensional latent vectors as input and produces synthetic images with dimensions of (64, 64, 3). It comprises several transposed 2D convolutional layers, batch normalization, and ReLU activation functions, with the exception of the last layer, which employs a tanh activation function.
- Discriminator: The discriminator is a deep convolutional neural network that differentiates between real images and fake images generated by the generator. It features a sequence of convolutional layers, batch normalization, and Leaky ReLU activation functions. The concluding layer utilizes a sigmoid activation function to yield the likelihood of the input image being real.

- The generator and discriminator are trained simultaneously in an adversarial fashion, where the generator tries to produce more realistic images while the discriminator attempts to better distinguish between real and fake images. The process continues until the generator produces high-quality images that are indistinguishable from real ones.

3.1.2 Training Algorithm of DCGANs

Algorithm 1 Training DCGAN for image generation

```
procedure TRAINING PROCEDURE
    roots = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
    for root in roots do
        dataset = create_data(dataroot, image_size)
        dataloader = create_loader(dataset, batch_size, workers)
        device = set_device(ngpu)
        netG = initialize_generator(ngpu, device)
        netD = initialize_discriminator(ngpu, device)
        criterion = BCELoss()
        fixed_noise = random_noise(64, nz, device)
        real_label, fake_label = 1.0, 0.0
        optimizerD, optimizerG = set_optimizers(netD, netG, lr, beta1)
        img_list, G_losses, D_losses = [], [], []
        Epoch_G_loss, Epoch_D_loss, Epoch_D_acc = [], [], []
        all_img = []
        iters = 0
        for epoch in range(num_epochs) do
            E_G_loss, E_D_loss, E_D_acc, batch_count = 0, 0, 0, 0
            for i, data in enumerate(dataloader) do
                Update D network (netD, data, real_label, fake_label)
                Update G network (netG, netD, real_label)
                Save losses (G_losses, D_losses)
                Update epoch losses (E_G_loss, E_D_loss, E_D_acc)
                Update iters and batch_count
                Save generated images (all_img, fixed_noise, netG)
            end for
            Update epoch losses (Epoch_G_loss, Epoch_D_loss, Epoch_D_acc)
        end for
        Save fake images (all_img, root)
    end for
end procedure
```

3.1.3 New Proposed Techniques

Typically, DCGANs are employed to generate artificial images of various natural scenes randomly without labels. However, we take a different approach. We partition our training dataset into six distinct categories based on their labels and feed these images into the training algorithm separately. Upon completing the training process, we can obtain fake images categorized separately. Consequently, each fake image will have its own label. Our method aims to offer fake images that can serve as additional training data for image classification tasks. As previously noted, fake data encompasses numerous unique features, which can supply more information for CNN models to train a superior classifier. Therefore, by providing labels for fake data, we could potentially enhance the performance of CNN models.

3.2 Image Classification

For the image classification task, we decide to use a transfer learning method: using a pre-trained ResNet-18 model to perform the multi-class classification task. The architecture of the ResNet-18 model from the original paper is shown in Figure 2.

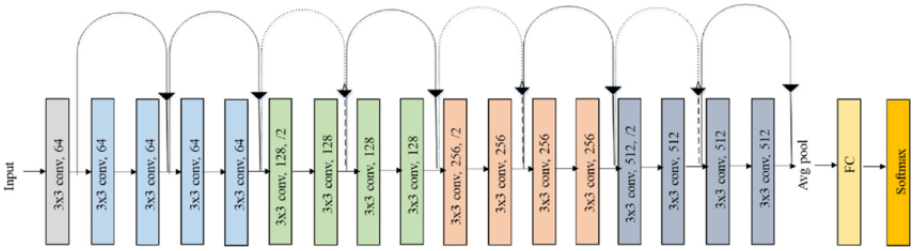


Figure 2: Architecture of ResNet-18

3.2.1 Training Algorithm of ResNet-18

Algorithm 2 Training ResNet-18 for multiclass classification

```
procedure RESNET-18 TRAINING PROCEDURE
    epochs ← hyperparameters[1]
    val_loss ← [ ]
    tn_loss ← [ ]
    for epoch in range(1, epochs+1) do
        train_loss ← 0.0
        valid_loss ← 0.0
        model.train()
        for batch_idx, (data, target) in enumerate(train_loader) do
            if train_on_gpu then
                data, target ← data.to(device), target.to(device)
            end if
            optimizer.zero_grad()
            output ← model(data)
            loss ← criterion(output, target)
            loss.backward()
            optimizer.step()
            train_loss += loss.item()*data.size(0)
        end for
        model.eval()
        for batch_idx, (data, target) in enumerate(valid_loader) do
            if train_on_gpu then
                data, target ← data.to(device), target.to(device)
            end if
            output ← model(data)
            loss ← criterion(output, target)
            valid_loss += loss.item()*data.size(0)
        end for
        train_loss ← train_loss/len(train_loader.sampler)
        valid_loss ← valid_loss/len(valid_loader.sampler)
        val_loss.append(valid_loss)
        tn_loss.append(train_loss)
        scheduler.step()
    end for
end procedure
```

3.2.2 New Proposed Techniques

The objective of multi-class classification tasks is to improve accuracy on actual testing data. To avoid overfitting to the real training data, we suggest training ResNet-18 models using a combination of real and artificial images. We believe that the fake images produced by the DCGANs algorithm have captured essential features and can outperform a ResNet-18 model trained solely on actual data.

Thus, we propose three methods for training ResNet-18 models: (1) training on artificial images only, (2) training on actual images only, and (3) training on both actual and artificial images.

4 Experiments

4.1 Dataset

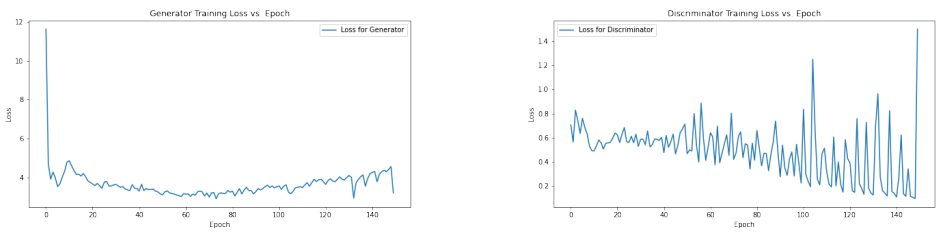
For the experiments, we first used the Intel Natural Scene to generate the fake data by using the DCGANs. The benefit of this dataset is that it is used for supervised learning, which contains labels for different classes. Therefore, we can still inherit the labels when generating fake images so that we can use those fake images in supervised learning later. This Data contains around 25k images of size 150x150 distributed under 6 categories (buildings, forest, glacier, mountain, sea, and street). The Train, Test, and Prediction data is separated in each zip file. There are around 14k images in Train, 3k in Test, and 7k in Prediction. For the purpose of our analysis, we only utilized the Train and Test data, where we divide the Train into Train(0.85) and Validation(0.15) when feeding into the pipeline. 1 included detailed information on the dataset. Each image in the original dataset is 150x150. However, for the sake of our analysis, we resized it to 64x64.

Table 1: Original dataset description.

Class	Number of Training Images	Number of Testing Images	Label
Buildings	2191	438	0
Forest	2271	474	1
Glacier	2404	553	2
Mountain	2512	525	3
Sea	2274	510	4
Street	2382	501	5

4.2 Fake Images Generation

In the first experiment, we run the DCGANs model for 150 epochs. We observe that the generator loss decreases as the number of epochs increases (see Fig. 3), indicating that the generator can produce increasingly realistic images with more training. However, when examining the loss plot (Fig. 3) and accuracy plot (Fig. 4) for the discriminator, we make some intriguing observations. Significant oscillations occur, revealing that the discriminator's performance is unstable. We believe this result is expected. The reason is that DCGANs train the generator and discriminator simultaneously. While the discriminator's performance improves during the training process, the enhanced generator model counteracts the progress of the discriminator's performance.



(a) Training Loss for Generator

(b) Training Loss for Discriminator

Figure 3: Generator/Discriminator Training Loss vs. Epoch

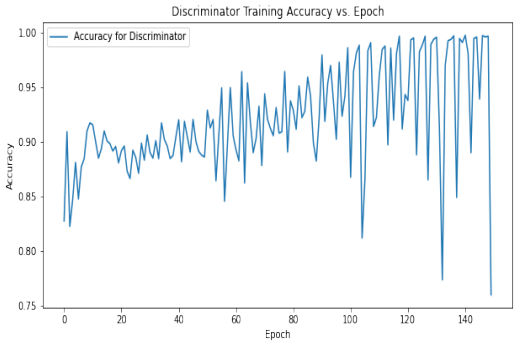
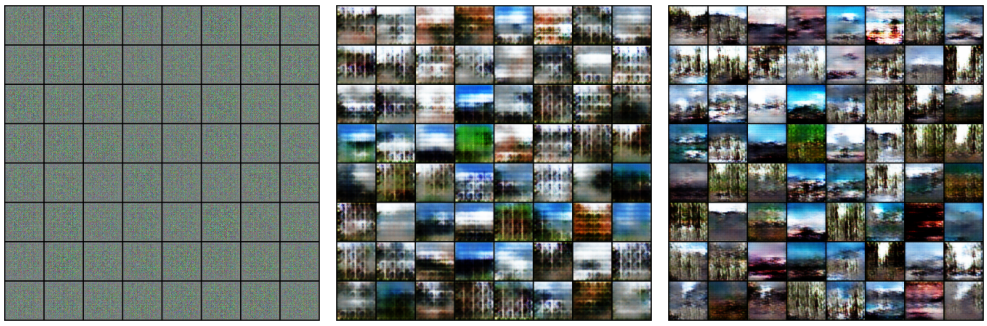


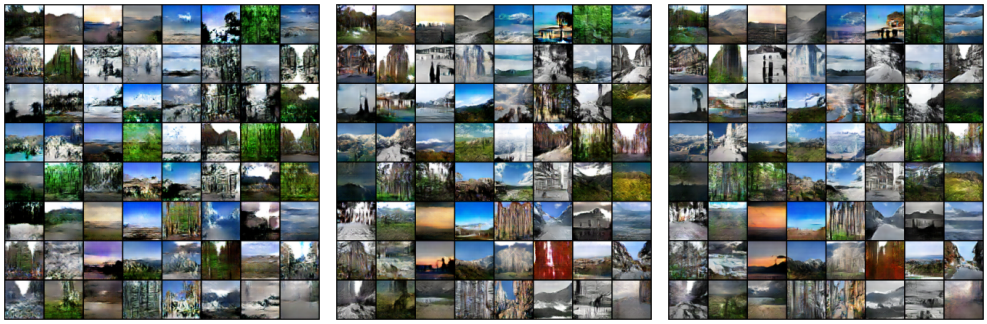
Figure 4: Discriminator Training Accuracy vs. Epoch

In the second experiment, we collect the generated fake images at various checkpoints to visually verify if the fake image quality improves as the generator loss decreases. Upon examining these checkpoint fake images from Figure 5 and 6, it becomes evident that the fake images become more realistic as the number of epochs increases. We can conclude that the quality of fake images improves as the model is trained for more epochs.



(a) Fake images at stage 1. (b) Fake images at stage 2. (c) Fake images at stage 3.

Figure 5: Fake images at state 1, 2, and 3.

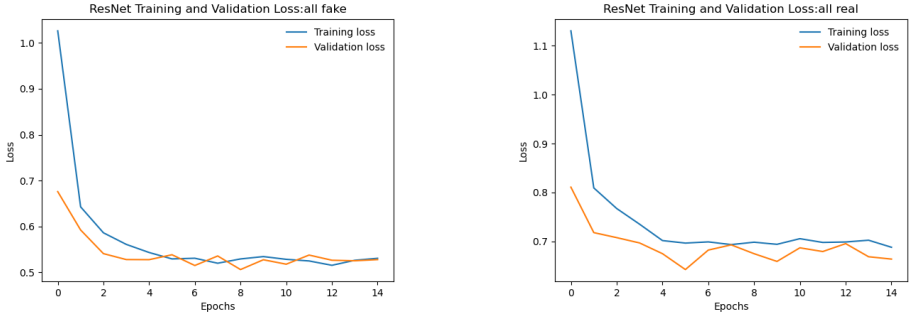


(a) Fake images at stage 4. (b) Fake images at stage 5. (c) Fake images at stage 6.

Figure 6: Fake images at state 4, 5, and 6.

4.3 Image Classification

After getting the fake images generated, we performed image classification tasks by using the transfer learning technique. In particular, we used a pre-trained ResNet-18 model to perform these tasks. We modified the last FC layer for this ResNet-18 model so that it outputs 6 features instead of 1000 features in the original paper. Three modes of experiments were run. In particular, we have mode 0 was to train all fake images; mode 1 was to train all real images; mode 2 was to train both real and fake images. We have tried different types of optimizers and chose Adam as our final optimizer. After limited fine-tuning due to the time constraint, our combination of both real and fake testing accuracy outperforms the traditional real images as the only training source by a tiny fraction in a fraction of experiments in terms of testing accuracy.
Figure 7 shows the comparison between using all fake and all real as the training set in terms of training and validation loss; Figure 8 shows the training and validation loss of our proposed fake + real method. Because we utilized the transfer learning technique, the model converges in just a few epochs.
Table 2 shows the results of the three methods. The Fake + Real method proposed outperforms the rest of the two methods.



(a) All fake images in training (b) All real images in training

Figure 7: Training and validation loss of all fake or real images

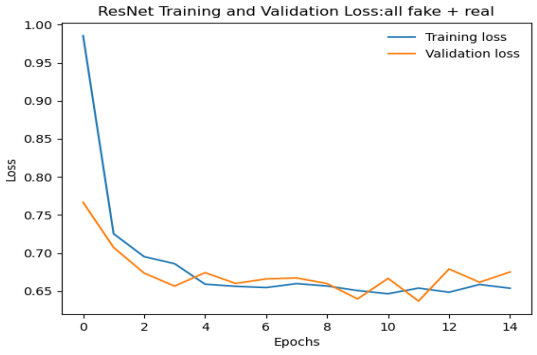


Figure 8: Training and validation loss of both fake and real images

Table 2: Training and testing results for three methods of training.

Training Data	Training Loss	Validation Loss	Testing Accuracy
All Fake	0.526	0.525	68.033
All Real	0.684	0.684	77.8
Fake + Real	**0.65**	**0.658**	**78.1**

5 Discussion

From the aforementioned findings, it is apparent that DCGANs can produce steady synthetic images that capture the latent attributes of the initial images. By incorporating them into training sets for multiclass classification models, it is possible to enhance accuracy on identifying previously unseen data. However, the difference in outcomes between training on entirely authentic images and training on a combination of half-real and half-fake images is not significantly significant. This suggests that the superior outcome may have been the result of chance. Further refinement and study are therefore necessary. Although this novel training method cannot be systematically verified to always produce better outcomes, it is possible to infer that when researchers need more data for analysis, they can employ DCGANs or similar algorithms to create stable synthetic data, thus conserving time and resources.

6 Supplementary Material

Here is a video presentation of this report.

6.1 Weights

The first convolutional layer of the ResNet model has filter shape of 3x64x7x7. For the purpose of better visualization, we have combined these three R, G, and B filters. Figure 9 shows the first convolutional weight map for our ResNet model. We know that learned filters exhibit structure and are uncorrelated. From Figure 9, we can clearly see that each filter is uncorrelated to others but they somehow have some interesting structures as follows.

1. Filters such as (1,1), (2,3), (4,1), (4,7), and (6,3) are focusing on detecting the horizontal edges.
2. Filters such as (2,1), (3,5), (6,2), and (8,4) are focusing on detecting the vertical edges.
3. Filters such as (1,7), (3,1), (4,4), (4,5), (5,1), (5,2), (6,6), and (7,5) are focusing on detecting images with similar colors.
4. Filters such as (2,8) and (3,8) are focusing on detecting texture and color brightness.

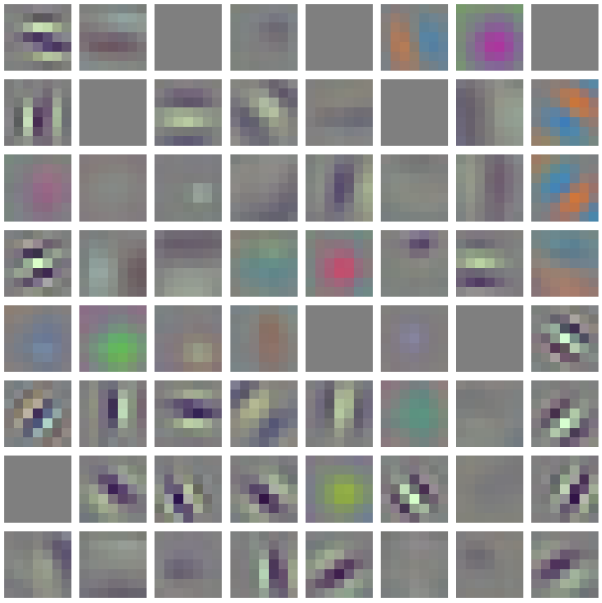


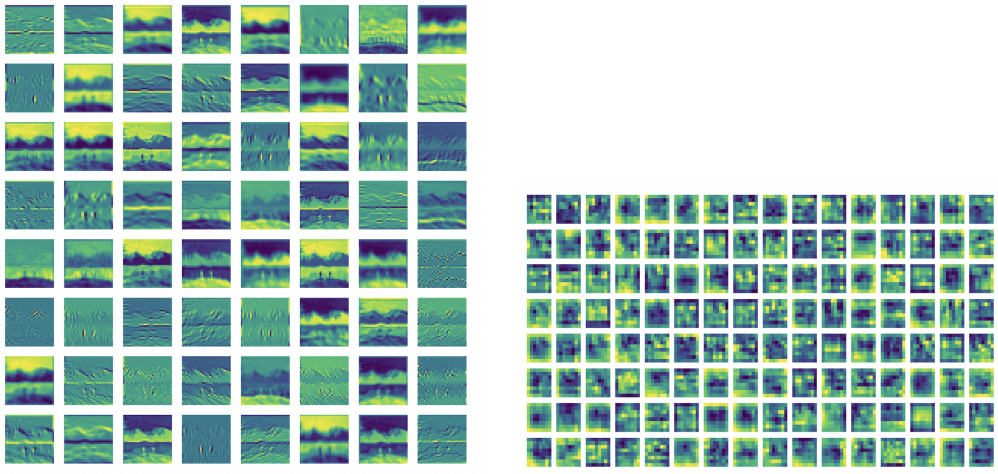
Figure 9: Weight map of ResNet model.

6.2 Feature Maps

We have used a random image from the testing dataset to plot the feature maps of the initial, middle, and final layers. The original image is shown in Figure 10.

Figure 10: Image used for all feature maps

Figure 11 shows the feature maps for the initial and middle convolutional layers in our ResNet model. It is clear that the channels for the initial layer focus on the shape, texture of the mountain, as well as the color differences between the mountain, ice, and land. As introduced in the previous sections, an important feature of ResNet is that it passes down residuals from previous layers to the current layer. Therefore, in the middle layer, we can still see that some of the features from the initial layer are preserved.
On the other hand, Figure 12 shows the feature map for the last convolutional layer in our ResNet model. This last layer has much more diversity than the previously shown layers. However, these extracted features may already be beyond human understanding and may be hard to perceive through visual interpretation alone.



(a) First layer.

(b) Middle layer.

Figure 11: ResNet model first and middle layer feature maps on a random image in the test set.

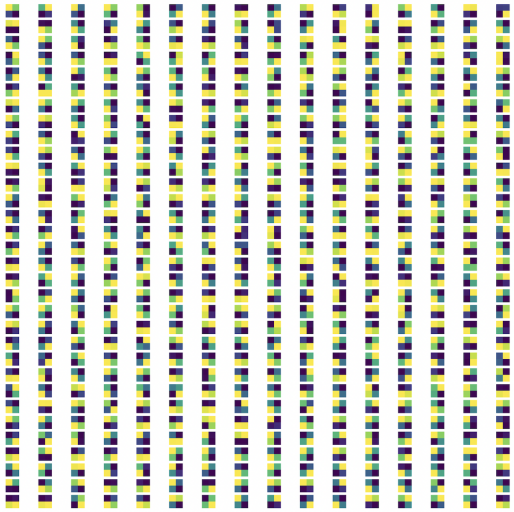


Figure 12: ResNet model final layer feature map on a random image in the test set.

References

[He et al.(2016)He, Zhang, Ren, and Sun] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Ian et al.(2014)Ian, Jean, Mehdi, Bing, David, Sherjil, ..., and Yoshua] Goodfellow Ian, Pouget-Abadie Jean, Mirza Mehdi, Xu Bing, Warde-Farley David, Ozair Sherjil, ..., and Bengio Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Radford et al.(2015)Radford, Metz, and Chintala] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.