Dog Breed Classifier and Generator

1st Yao Zhong
Khoury College of Computer Science
Northeastern University
San Francisco, United States
zhong.yao@northesatern.edu

1st Hui Hu
Khoury College of Computer Science
Northeastern University
San Francisco, United States
hu.hui1@northeastern.edu

***Abstract*—In this project, we explored several topics in deep network using the Stanford Dogs Dataset. Firstly, we implemented a dog breed classifier. By implementing and comparing different architectures, we found out that the ResNet-50 architecture has the best performance in this task and reached a 57% accuracy for dog breed classification. In addition, we were interested in the image generator. Therefore, we also extended our project to a dog breed generator using the conditional deep convolution generative adversarial network(cDCGAN).**

***Index Terms*—deep network, dog classifier, generative adversarial network, ResNet, DCGAN**

I. Introduction

The Stanford Dog Dataset [1] is a large collection of annotated images of dogs, organized by breed. It was created by researchers at Stanford University and was first introduced in 2011. The dataset consists of 20,580 images of 120 different dog breeds, with about 150-200 images per breed. Each image in the dataset is annotated with the breed of the dog in the image. This makes the dataset a valuable resource for training and testing machine learning algorithms for image recognition and classification tasks. Based on this dataset, we create our own classification network by using transfer learning and compare the effects of layers, learning rates and momentum to optimize model and increase classification accuracy.

After that, we extended our work to the area of generative adversarial network. As s slow start, we decided to experiment with a rather simpler subject, the digits, and use the MNIST digit dataset as the material we can work on. Since the subjects are simple and we are familiar with this dataset, we can be more focused on the concepts of GAN. After we successfully implement the digits generator, we transformed it into a conditional GAN by adding the labels for both input of generator and the discriminator. After all works done with digits, we finally have the whole pip-line of conditional GAN and we moved to dog breed generator. By continuously experimenting with different model architectures and hyper parameters. We had our own conditional deep convolutional generative adversarial network.

II. Related Work

A. Dog Breed Classification

As a subfield of machine learning that aims to leverage knowledge learned from one task or domain to improve performance on a different but related task or domain, transfer learning approaches are developed and proposed for handling the situations where the domains are of the same feature space but differ only in marginal distributions. [2]

Kaiming He et al. [3] proposed a new architecture called the residual network (ResNet) that introduces residual connections, which allows information to be directly passed from one layer to another. By doing so, the ResNet architecture overcomes the problem of vanishing gradients, and enables the training of much deeper neural networks, up to 152 layers, while maintaining high accuracy.

B. Dog Breed Generator

The generative adversarial network was first proposed by Ian Goodfellow [4] in 2014. This ground breaking work introduced a new generative modeling which trains two networks at the same time: one is the generator and the other one is the discriminator. The generator is responsible for creating synthetic data that is similar to the real data. And the discriminator is used to differentiate the real data and the generated data. By training the generator and discriminator simultaneously, the generator will evolve to generate better fake images that fools the discriminator and the discriminator will evolve to better discriminate the real and fake images. Furthermore, in the Goodfellow's paper, it is proposed that a conditional model can be obtained by adding a label in both generator and discriminator input. And this is all the theoretical ground of our project.

III. Methods

A. Image Pre-processing

The Stanford Dogs dataset provides detailed lists and labels of all the training and test images in the dataset, along with associated bounding box annotations for each image.

In order to make the images easier to work with, we cropped each image according to its bounding box file and resized it to 64×64 for universal operations. As we worked with the dataset, we organized the images into different folders based on their labels, such as "training image" or "test image." To make it easier to work with the dataset in the future, we also used a json file to save the number-name pairs for each breed.

B. Dog Breed Classification

In order to determine which model was superior, we conducted experiments on five different ResNet networks.

Followings are the architecture of the five networks, the ResNet architecture enables the training of much deeper neural networks, up to 152 layers, while maintaining high accuracy. We will train each network while keeping other parameters fixed so that we could accurately measure their performance. We proceeded to compare the final accuracy rates of each model to determine which one was superior.

In order to determine which ResNet model was the superior one, we conducted experiments on five different ResNet networks. The five ResNet networks that we experimented on were ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152. The ResNet architecture is a deep convolutional neural network architecture that enables the training of much deeper neural networks, up to 152 layers, while maintaining high accuracy.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Fig. 1. ResNet Architecture.

We trained each network while keeping other parameters fixed so that we could accurately measure their performance. To ensure fairness in our experiments, we used the same dataset to train each network. The dataset we used was the Stanford dogs dataset, which consists of 12,000 training images and 8,580 testing images. After training each network, we compared the final accuracy rates of each model to determine which one was superior.

C. Dog Breed Generator

The dog generator part is roughly two parts. The first part is using the MINIST digits dataset as the material, we followed the work of Neeraj Varshney' tutorial [5]. In his work, the generator and discriminator consist of 4 hidden layers and each is a linear layer followed by an activation function (Leaky ReLU). And the training process is consisted of two main steps: 1. updating the discriminator with the loss on both true inputs and fake(generated) inputs; 2. updating the generator with the loss on the generated data in discriminator. After this, we added label in both the input of the generator and discriminator to make the network able to produce digit by condition.

In the second part of the dog breed generator, we modified the network we obtained in the first part in digits and transformed it to generate breeds of dogs. By experimenting on different generator and discriminator architectures and hyper parameters. We found the deep convolution generative adversarial network(DCGAN) [6] which mainly uses the convolution layers in the model a good choice. In this network, the generator and discriminator consist of 4 convolution layers and each is a convolution layer followed by a normalization and an activation function (ReLU in generator and Leaky ReLU in discriminator). The training process is similar with that of the digits generator. And we also changed the models slightly so that it supports the conditional labels.

IV. Experiments and Results

A. Experiment 1: Optimal ResNet architecture

After conducting the experiments, we found that each model achieved relatively stable accuracy rates after only 10 epochs. We then proceeded to compare the final accuracy rates of each model to determine which one was superior. The results of our experiments showed that ResNet-50 had the highest accuracy rate, followed by ResNet-152, ResNet-101, ResNet-34, and ResNet-18, in that order.

To provide a visual representation of our findings, we've included the negative log likelihood loss graph below.

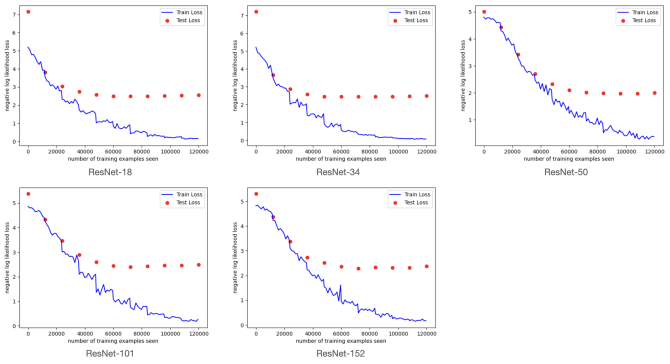

Fig. 2. Training Loss of different ResNet Architecture.

ResNet	Accuracy	Percent	Avg. loss
18-layer	3069	36%	2.5575
34-layer	3247	38%	2.4934
50-layer	4020	47%	1.999
101-layer	3467	40%	2.4964
152-layer	3573	42%	2.3849

Fig. 3. Accuracy of different ResNet Architecture.

After carefully examining the results table presented above, we can confidently state that ResNet50 has exhibited superior performance when compared to other models. Based on this observation, we can take measures to further optimize its accuracy by modifying some of its parameters, such as the learning rate and momentum. This will allow us to achieve even better results and make the most of the potential offered by this model.

B. Experiment 2: Learning Rate Optimization

The learning rate is a hyperparameter in machine learning that determines how much the model parameters are updated in response to the estimated error between predicted and actual values during training. It will effect convergence speed, accuracy and stability of models, therefore, and the optimal learning rate may vary depending on the specific problem, data, and model architecture. Therefore, we tried different learning rates and select the one that provides the best results.

learning rate	Accuracy	Percent	Avg. loss
0.001	585	7%	4.5633
0.005	3192	37%	2.5268
0.01	4020	47%	1.999
0.05	4726	55%	1.8171
0.07	4878	57%	1.8085
0.08	4924	57%	1.8416
0.1	4784	56%	1.9550
0.5	3899	45%	2.6964

Fig. 4. Accuracy under different Learning Rate.

According to above table, 0.08 is the optimal learning rate.

C. Experiment 3: Momentum Optimization

In machine learning, momentum is a technique used to accelerate the gradient descent algorithm by adding a fraction of the previous update to the current update. Momentum can have a significant impact on the optimization process, and it can offer several benefits, including faster convergence and reduced oscillations.

Also, we tried different momentum and selected the one that provides the best results based on the learning rate 0.08.

Momentum	Accuracy	Percent	Avg. loss
0.1	4712	55%	1.8100
0.3	4830	56%	1.8290
0.5	4924	57%	1.8416
0.7	4781	56%	2.0370
0.8	4605	54%	2.2586
0.9	4073	47%	2.5736

Fig. 5. Accuracy under different Momentum.

D. Experiment 4: Digits Generator

In this experiment, we implemented the generative adversarial network for digits following the tutorial of Neeraj Varshney. The generator and discriminator are consist of 4 hidden layers and each is a linear layer followed by an activation function (Leaky ReLU). Fig. 6 and Fig. 7 are the graphic model of the generator and discriminator.

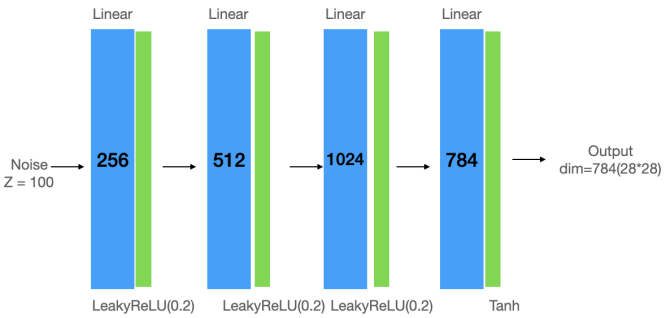

Fig. 6. The digits generator model.

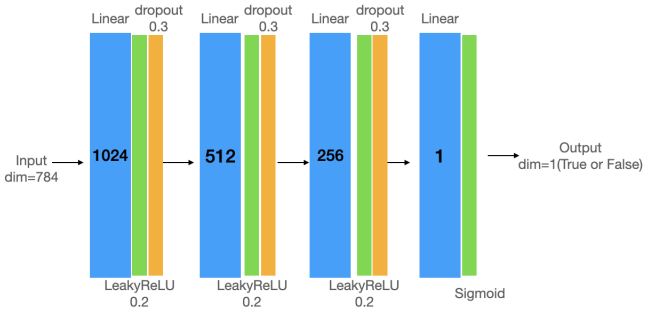

Fig. 7. The digits discriminator model

The training of the generator and discriminator was trained by the following process: first, update the discriminator with the loss on both true inputs and fake(generated) inputs, where the loss is calculated by:

$$loss = \frac{D(TrueImages) + D(1 - G(Z))}{2} \tag{1}$$

Then, updating the generator with the loss on the generated data in discriminator which is:

$$loss = D(G(Z)) \tag{2}$$

As for the hyper parameters, we set the learning rate being 0.0001, batch size being 64 and number of epochs being 50.

As shown in Fig. 8, in the first a few epochs, the generator can only generate a blurred dust of white dots. Starting from epoch 14, it is able to form some shape, but it is still not a number. At epoch 26, we can see that a digit 3 is formed but there are a lot of noise around. After epoch 44, the result of the generator became more clearer with less noise.

As shown in Fig. 9, this is the result of the digits GAN after 50 epochs of training. The results are 9 random numbers generated. We can easily find out that the digits 9, 5, 7, 6, and 0 were clearly generated. However, there were still some result with a lot of noise, for example the right-top image.

As shown in Fig. 10, this is the loss of generator and discriminator during the training process. Theoretically, when the generator creates digits that is more and more like the real digits, the generator loss will be decreasing, and the

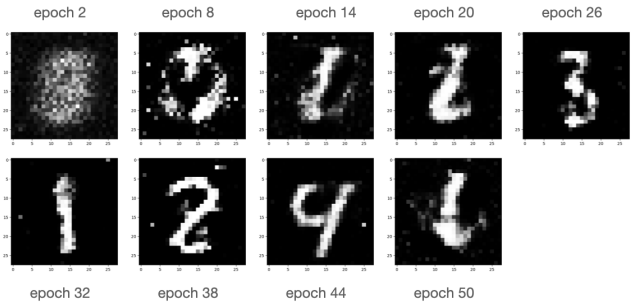

Fig. 8. Digits Generator Training Process.

Fig. 9. Digits Generator Example Results.

discriminator loss will be increasing since it is more trickier to differentiate generated digits and real digits.

E. Experiment 5: Conditional Digits Generator

In this experiment, we we added label in both the input of the generator and discriminator to make the network able to produce digit by condition. Here is the graph of the updated generator and discriminator. The main change was the input dimensions of the models. The training process and hyper parameters stays the same with the previous experiment except that we now set the number of epochs to be 100.

As shown in Fig. 11, in the first a few epochs, the generator can only generate a blurred dust of white dots. Starting from epoch 4, it is able to form some shape, but it is still not a number. At epoch 6, we can see that a digit 0 is formed but there are a lot of noise around. After epoch 48, the result of the generator became more clearer with less noise.

As shown in Fig. 12, this is the result of the digits conditional GAN after 100 epochs of training. The results are all the 10 digits from 0 to 9. From the result, we can see that almost all digits are generated correctly and clearly according to the given label, only the numbers 2, 5 and 8 is not clear enough, but the shape of them are still distinguishable.

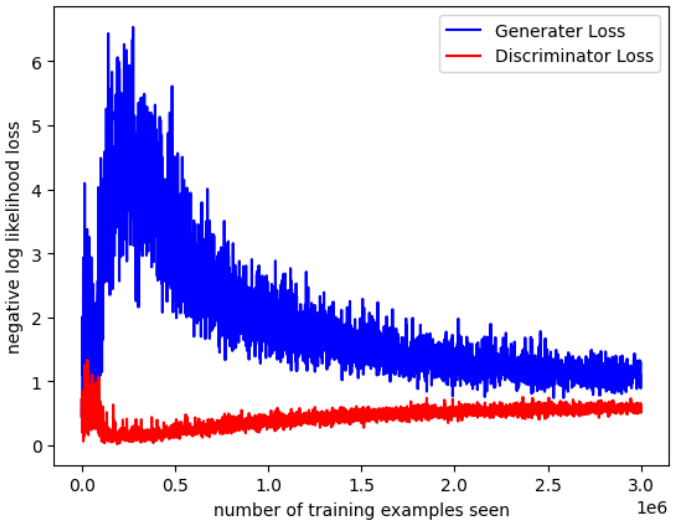

Fig. 10. Digits Generator Training Loss.

As shown in Fig. 13, this is the loss of generator and discriminator during the training process. We can see that after the 100 epochs of training, the loss of the generator and discriminator converges to the same value.

F. Experiment 6: Transfer to Dog Images

In this experiment, we finally moved to the dog generator. As the first model, we decided to continue with the former model of generator and discriminator in Experiment 5. The only change down by us was to change the dimensions in each layer, since the input of the images now have changed to colored images.

As shown in Fig. 14, this is the process output of the first iteration of the dog breed generator. Although after 80 epochs of training, the out put of the generator was always heavily blurred there is no object can be seen from those images.

G. Experiment 7: Dog Breed Conditional Generator with Deep Convolution Network

Due to the bad behavior of the last experiment, we found out that the linear model that is a good for the digit images is now no longer a fit for the dogs images. Therefore, we decided to adopt a better but more complex model for the dog images generator.

The model that we used is called the deep convolutional network. In which the convolution transpose are explicitly used. In the deep convolution networks, the generator and discriminator are consist of 4 convolution layers and each is a

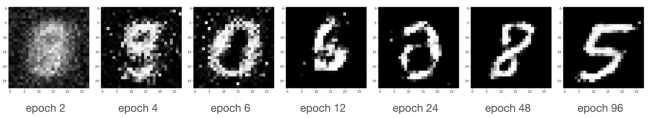

Fig. 11. Conditional Digits Generator Training Process.

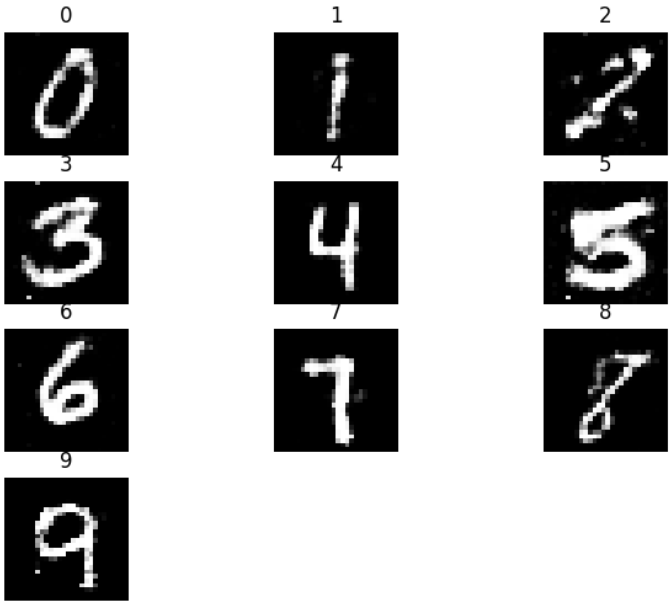

Fig. 12. Conditional Digits Generator Results

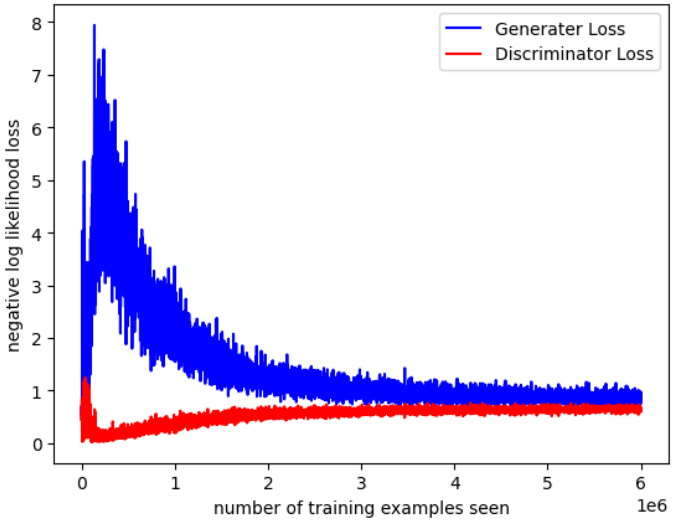

Fig. 13. Conditional Digits Generator Training Loss.

convolution transpose layer followed by a normalization and an activation function (ReLU in generator and Leaky ReLU in discriminator).

The main changes we made to this model was that we added the label as the input for both generator and discriminator.

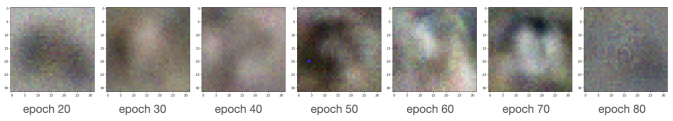

Fig. 14. Linear Dog Breed Generator Process.

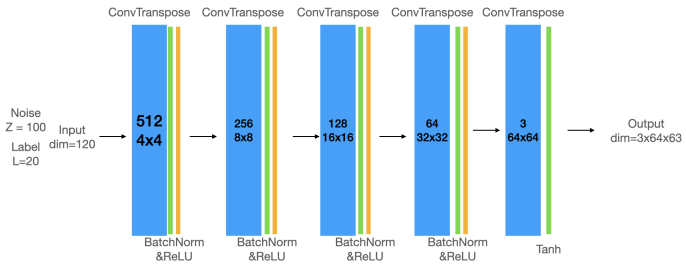

Fig. 15. Conditional Dog Breed Generator Model

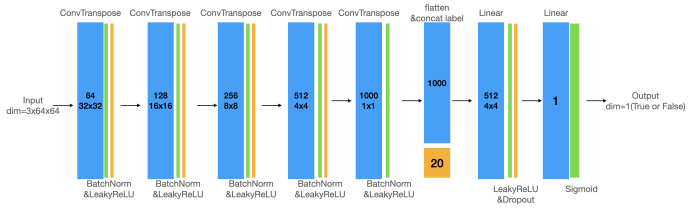

Fig. 16. Conditional Dog Breed Discrininator Model

After a few small changes in the iterations, we finally decided on the following structure of generator and discriminator. We tried our best to keep the original structure not being disturbed.

As for the training process, we followed the same training loop structure and loss metrics in the digits GAN. As for the hyper parameters, we set the learning rate being 0.0002, batch size being 128, beta value for optimizer was 0.5, and number of epochs being 50.

Here is the output of the training process (Fig. 17). To obtain these pictures, we maintained a noise vector during the training process. Therefore, we are able to see how the did the generator learn to generate images during the process of training. The number of each sub-image was the breed number of that dog. From epoch 2 to epoch 44, we can find out that the generated images changed from random noises to color chunks and then the features of dogs became more and more clear, especially the front leg, back legs and tails.

Fig. 18 is some examples of the generated images of the model after 50 epochs of training. The left most column are the pictures of the dogs in the dataset, and the right 3 columns are 3 examples of that breed generated images. From the result, we can tell that the generated images are dogs, they have captured the basic features of the dog shape. However, the generated images can hardly relate to the breed which it should belong to.

As shown in Fig. 19, this is the loss of generator and discriminator during the training process. We can observe that the loss curve for the generator and discriminator are different with that of the conditional digits generator. The loss of the generator were increasing during the process of training and the discriminator loss were decreasing during the process. Which indicates that the generator are generating dog images that are less and less look like a dog.

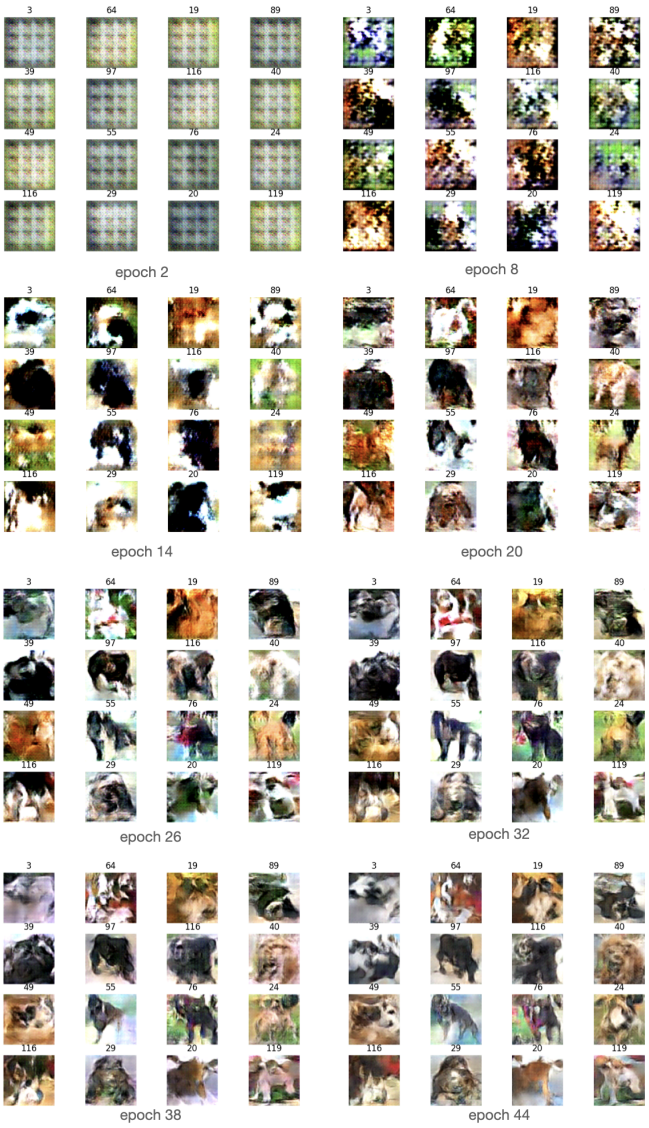

Fig. 17. Conditional Dog Breed DCGAN Training Process.

V. Discussion

A. The Classifier

Transfer learning could reduce the need for large amounts of labeled data by leveraging knowledge from related tasks or domains and apply it to a new task or domain, allowing the model to learn from a smaller dataset; it can also improve the performance of a model on a new task or domain by providing the model with prior knowledge learned from related tasks or domains which results in better generalization and faster convergence of the model; more importantly, transfer learning can leverage pre-trained models that have already learned useful features, and significantly reduces the time required to train a new model.

By incorporating the ResNet architecture, which is widely used and highly regarded in the field of deep learning, we are able to train our classification model more efficiently and in a shorter amount of time. However, it is important to note that the highest accuracy achieved by the ResNet architecture is only 80%. While transfer learning can certainly help in improving the accuracy of the model, it is not sufficient to achieve a higher rate. Thus, in order to adequately address an unfamiliar domain, it is necessary to create a custom model from scratch that is specifically tailored to the problem at hand. This custom model can be trained on a large dataset and fine-tuned to achieve a higher accuracy in the target domain. Additionally, one can use ensemble methods such as bagging or boosting to further improve the performance of the model. Overall, while transfer learning can be a useful tool, it is important to consider the limitations of pre-trained models and to explore other strategies that can help to achieve a more accurate and robust model.

B. The Generator

During experiment 4 and 5, we implemented the GAN for digits and then converted it to a conditional GAN for digits. We found these pilot experiment very useful because we were not very familiar with the concepts of generative adversarial networks. By implementing them, we learned the whole process of writing the generator model and discriminator model, create the metrics for updating the generator and discriminator and the process of updating them. And the result of these two experiment turns out to be very good, we successfully generated clear digits by their labels and also the loss curve are the same with theory assumption. The following experiments were benefited from these pilots, and the result of the two experiments also guided our next steps.

For the dog breed generator, we first tried to use the same structure of the digit GAN. However, although this model which consists of only linear layers and active functions have a very good performance on generating binary digit images, it is not a good structure for more complex images. Actually, before we moved to another model, we also tried to increase

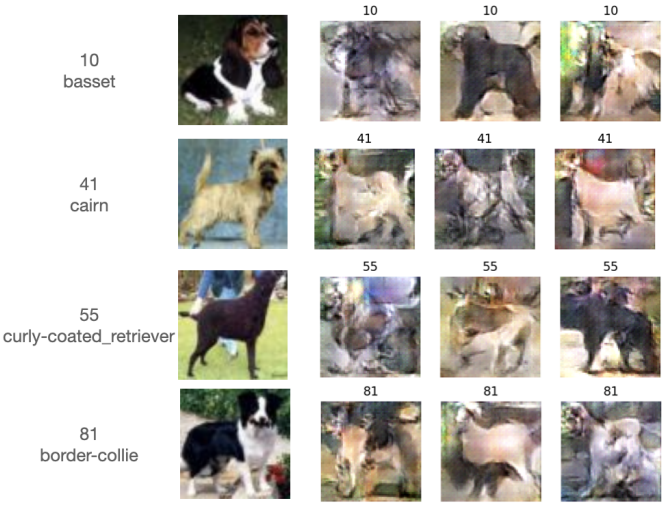

Fig. 18. Conditional Dog Breed DCGAN Result.

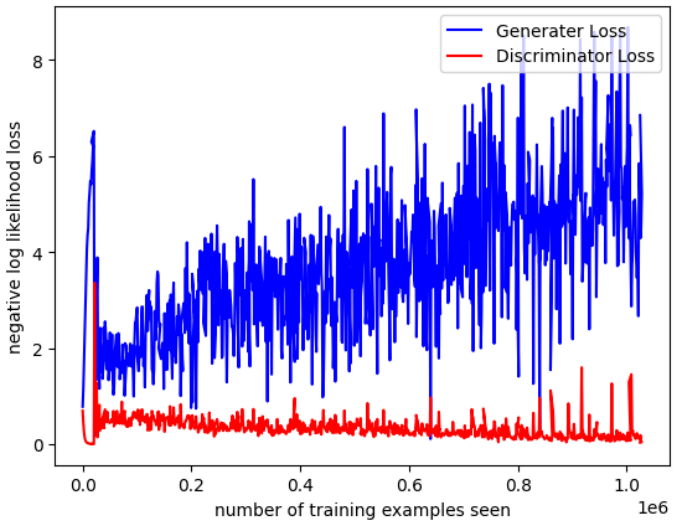

Fig. 19. Conditional Dog Breed DCGAN Training Loss.

the size of the linear layers, since the images been generated now is 3x64x64 images which requires far more information than 28x28 digit images. However, the when the model has too many high dimensional fully connected layer, it got too large to run, we had to reduce the number of dimensions of each linear layer so that it will not consume all the memory space.

The deep convolutional network(DCGAN) we adopted had a very good performance in generating human face images, and the images they generated was also 3x64x64 images. Comparing to our linear models, it did do better in generating the image. As early as the 6th epoch, it starts to have different colors in the generated images, although it is still all noise, but comparing to the gray and heavy blur images generated by the linear model, it is more close to the real life colors. As early as epoch 20, the noise in the images start to decrease and we can easily distinguish the foreground object and the back ground. As early as the 32 epoch, some of the objects in the images start to have the features like a dog, the front legs, back legs and tails are showing, and the color scheme of the dogs are also making sense.

However, the results we got from the dog breed generator is not ideal. Firstly, the generated images are hardly a dog. Only very few of generated images looks like a dog. Not to mention if we can tell if this dog is of which breed. Comparing to the images generated by the A-C-Ra-LS-DC-GAN (Auxiliary Classifier Conditional Relativistic Average Least Squares Deep Convolutional Generative Adversarial Network) [7], which can easily find out the features of a breed. We still have a long way to go. In addition to the shortage of the model, we have a more serious problem revealed in the result, which is the loss curve. We expected to see the similar loss curve of the digits GAN, in which the generator loss gradually decrease and the discriminator loss increases and they converge to the same value. However, during training, the loss of the generator increases. Which indicates that it is generating images that are less and less look like a dog. Actually, this issue gets worse after 50 epochs, our model collapse after 60 epochs of training and starts to produce images like epoch 2. We believe that this problem is due to the hyper parameters we choose, but we tried many different combinations, this problem still exists.

As for the future improvement, we may need to consider using a better model for the generator and discriminator, for example the A-C-Ra-LS-DC-GA. In addition, we need to also figure out where to properly add the label into the model, to make the regular GAN into a conditional GAN. Last but not least, we need to conduct more experiment to research for the best combinations of hyper parameters.

ACKNOWLEDGMENT

The original data source is found on http://vision.stanford.edu/aditya86/ImageNetDogs/.

REFERENCES

[1] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao and Li Fei-Fei. Novel dataset for Fine-Grained Image Categorization. First Workshop on Fine-Grained Visual Categorization (FGVC), IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.
[2] F. Zhuang et al., "A Comprehensive Survey on Transfer Learning," in Proceedings of the IEEE, vol. 109, no. 1, pp. 43-76, Jan. 2021
[3] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition, 2016.
[4] I. Goodfellow, J. Pouget-Abadie, and M. Mirza, et al., "Generative Adversarial Networks", Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680, 2014
[5] N. Varshney, "Step by Step Implementation of Conditional Generative Adversarial Networks," Medium.com, 06-Jun-2020.
[6] N. Inkawhich, "DCGAN TUTORIAL," pytorch.org.
[7] C. Deotte, "Dog Breed ACGAN", Kaggle.com, 2019.