1) introduction: problem definition, motivation, challenges, etc.

2) literature review: existing methods with pros and cons.

3) methodology: your proposed algorithms with detailed description

4) results: data, evaluation metrics, baselines, and performance comparison, etc.

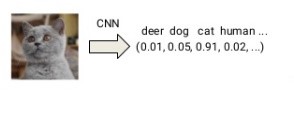
5) implications and limitations: any discussions can be put here.

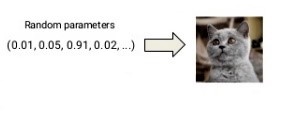
6) conclusions

**Introduction**:

Since the image database may be quite limited if they were all real images, the technology needed to depend on database would be constrained because of the number of images. We were thinking what if we could make some fake images

Our goal is to create images of a new cat face from our real cats image dataset. Our inspiration comes from our daily lives. In this semester, we learned the Convolutional Neural Network(CNN) in Big Data Class. Months ago, our group tried to do cat breeds classification through CNN because recently, one of our friends adopted a cat. The staff from the adoption center told him that the cat was Birman. However, he doubted it since the adopted cat’s outlooking was somehow different from Birman cat photos he searched online, so we wanted to build a CNN model to identify cats’ breed by reading their outlooking photos. We successfully build it, but after that we wanted to know if there is anything else we could do beyond this topic. Then we learnt about that Generative adversarial networks (GANs) are one of the hottest topics in deep learning in recent years. With GAN, for instance, we could create a new model that generates fake images which contained the identities cats have.



But our desired model can generate an image from random parameters: 

**Data Processing:**

We searched more than 9000 images of cat from Kaggle.com. However, we didn’t need the whole image of every cat we got since it would be much harder to generate the a whole cat. What we needed was only the face part of every cat image, so we used a specific classifiers called frontalcatface\_extended.xml to detect frontal cat faces in the images and cut them off to be the new image dataset which contained nearly 5000 images. This was still not enough for our model, so in order to enlarge our dataset, we first flipped every image horizontally and then generated five rotated images for each between -25 degrees and 25 degrees. After all these augmentation, we successfully enlarged our dataset to about 56400 images.

**Challenges:**

The overall flow was to define a generator and a discriminator and train them on numbers of images we had in our dataset first. In the end, the model output the images we generated through process.

We did lots of researches based on how to set up the model, and what model satisfied our desired goal. The first bottleneck we face was that we had a hard time in which model we should choose between Generative Adversarial Network(GAN) and Deep Convolution Generative Adversarial Network(DCGAN). These two are very similar.

* Generative Adversarial Network (GAN) takes the idea of using a generator model to generate the fake images and discriminator model that tried to decide if the image it receives from the generator is a fake or a real sample.
* A Deep Convolution GAN(DCGAN) does something very similar but specifically focuses on using Deep Convolutional networks in place of those fully-connected networks. The generator of the DCGAN uses the transposed convolution technique to perform up-sampling of 2D image size. Generally speaking, DCGAN would likely be more fitting for image data, whereas the general idea of a GAN can be applied to wider domains.

In this case, we selected DCGAN as our model, since we think it was somehow related to the CNN model we learned in class and we would quite familiar with.

**Compare DCGAN with CNN:**

Like what we talked in the previous sections, the DCGAN is similar to CNN architectures, but they still have some differences:

1. In DCGAN, it removed all the fully-connected hidden layers and it does not use max pooling method to reduce its dimensionally.
2. Moreover, in the DCGAN. We used batch normalization on both the generative and discriminative models. Because one cause was that at the intermediate layers process speed slowed due to the size change during training, and normally, we processed and normalized all the data in order to resemble the normal distribution.That was why we needed to use batch normalization here since it effectively helped us overcome the low process speed.
3. In the DCGAN, we used LeakyReLU activation functions in both generator and discriminator models. But, in the CNN, we mostly used ReLU as the activation function. The two major benefits for LeakyReLU were first, in the ReLU function, the problem occured when the downside for being zero for all negative values. ReLu did not work if it was in the range of the negative side and the output was always 0. But in the LeakyRelu, there is no chance it will become zero-slope.
4. The convolutional filters work in the CNN and generators in DCGAN was in the opposite direction. In the CNN, NxN old pixels contributed to generate a new pixel. Each square of old pixels was multiplied by the corresponding filter value, and the results from old pixels are added together to become a new pixel. Generator in DCGAN worked in a transposed way, a single old pixel contributed to generate NxN new pixels, each old pixels times each of NxN filter values, and results from old pixels are formed together to become a new dimension.

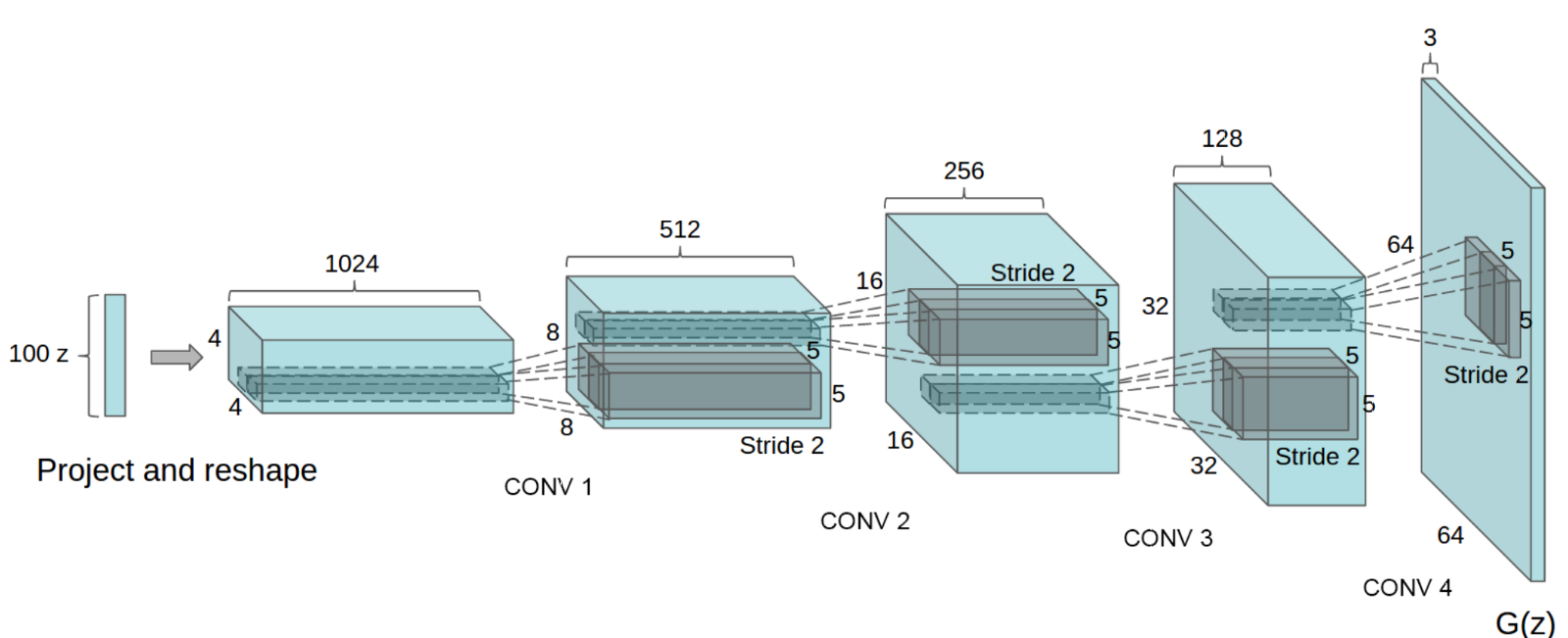
**Pros of DCGAN:**

* DCGAN could create an image we never saw before and this technology can help us with enlarge datasets if one’s datasets are not enough to meet the demands. DCGAN was asymptotically consistent as we saw it formed images in the output. Also, DCGAN was considered a better image generator than the method like PixelCNN.
* DCGAN combines GAN and CNN

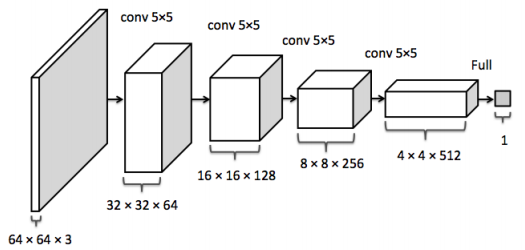
**Cons of DCGAN:**

* The major limit for DCGAN was mode collapse. The generator collapsed which produced limit varieties of samples. When we started to run the DCGAN model, it did not mean to cause mode collapse immediately. During the time of the training process. Mode collapse could sometime happen or get the chance not to happen at all. The discriminator ended up would not really take focus on diversity, they would only generate the sole mode. In that case, where the generator had collapsed to a single point, it could not get out.
* In addition, as we said, DCGAN is good at training and generating images, but for some discrete data like text, it is hard to learn to generate.
* If you only observed these fake images generated by DCGAN, you felt these images were quite realistic. However, when you compared the original image in the dataset with those fake images, you would still feel somehow different since in the DCGAN, it filled something else to the images which did not exist in the original images.

**Methodology:**

The cores of DCGAN are generator and discriminator. Generator is a process that we input noise and output an image randomly. We used deconvolution as our major frame to form generator. Deconvolution is an algorithm-based process used to reverse the effects of convolution on recorded data which means that we will transform small pixels to complete image pixels. On the other hand, discriminator is a process that we used to recognize whether the image that generator out is real or fake. In this process, we used convolution algorithm as we have discussed in class. We used three convolution with different number of filters and one flatten in discriminator to make it as a thin but longer dimensions. For these two conflict process, we needed to use the images generator generates to fool the discriminator. For example, the first image that the generator generates cannot fool discriminator and discriminator will identify this image as a fake one. After the first round, the generator will learn some coefficients to improve its generate power and generate the second image. However, the second image may not fool discriminator again, and generator learns more and improves its image-maker power much stronger. Learn again and again and identify round and round, the generator can finally output an image that the discriminator cannot identify whether it is fake or real. Then the image will output. The following two images show the complete reflection of generator and discriminator.

Generator: Deconvolution process



Discriminator: Convolution process

Also, we needed to use one function to link these two processes. The most popular method is to calculate the total loss of discriminator( unsuccessfully identify the real image and fake image). More, we needed to set the optimization operations and image output process. Then we can train the model with the tons of images in our database and set different batch sizes, learning rates and epochs to have a better view of the output of the images.

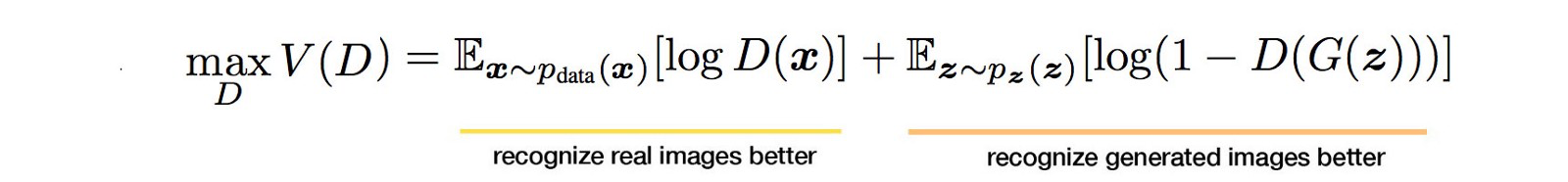
**Results:**

All the images of cats we used are from Kaggle. We used around ten thousand images (regardless of the breeds of cats) to train DCGAN model to generate the new image of the cat which is learned by the computer itself. However, the pictures are not enough and the cats are not uniform. We used cat face classifier to cut the whole images into only cat face. Then, we used flip to create the mirror images and used rotation to random generate 5 pictures between range from -25 degree to 25 degree. First of all, we needed to preprocess those images for future use. We needed to make resize the images into the same size and transform them into Numpy array. For those Numpy arrays, if they have less than 4 dimensions, we need to add one dimension to them to make all the images have the same dimension. Also, we needed to set batches for images to separate them into different batches.

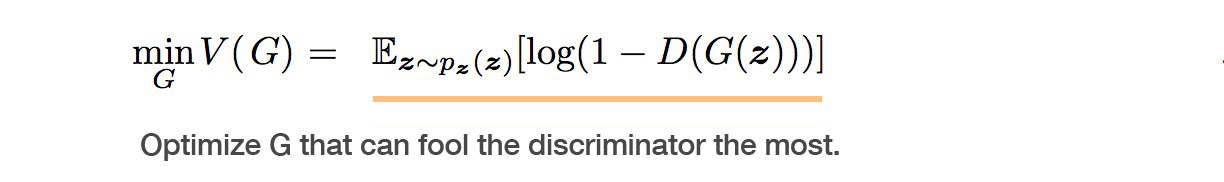
For discriminator, we used 4 convolution layers to simplify the images and flatten them as a long dimension and use sigmoid function to output a number between 0 and 1. In this first convolution layer, we apply 64 5\*5 filters with 2 strides and zero paddings on the image. Also, we used the maximum method under tensorflow to choose convolution or the convolution multiply by alpha (leaky ReLU). In the second convolution layer, we apply 128 5\*5 filters with 2 strides and zero paddings on the output from convolution 1. In this step, we needed to use batch normalization which reduces the amount by what the hidden unit values shift around and improve the training speed. Also, we used maximum method under tensorflow to choose convolution 2 or the convolution 2 multiply by alpha. In the third convolution layer, we used 256 5\*5 filters with 1 stride and zero padding on the output from convolution 2. Also, this layer contains batch normalization and maximum selection. Finally, we used flatten and dense function to generate one logit and use sigmoid function to get a number.

For generator, The generator is comprised of [convolutional-transpose](https://pytorch.org/docs/stable/nn.html#torch.nn.ConvTranspose2d) layers, batch norm layers, and [ReLU](https://pytorch.org/docs/stable/nn.html#relu) activations. The input is a noise vector which is drawn from a standard normal distribution. The stridden conv-transpose layers allow the latent vector to be transformed into a volume with the same shape as an image. we used 2 deconvolutional layers and one output layer in this step. First, we apply 256 5\*5 filters with 2 strides and zero padding on the image. Then, we applied batch normalization on the deconvolution 1. Also, we used maximum method under tensorflow to choose convolution or the convolution multiply by alpha (leaky ReLU). Second deconvolution, we used the same method as the first one but changed filter numbers to 128. Finally, in the output layer, we used the filter numbers same as the dimension of output (we want).

We also need to define the model loss. The loss is calculated as the loss of real images and the loss on fake images(generated images). We used the cross-entropy method here to calculate the loss. For real image, it calculated as p\*log(q) and p\*log(1-q) for fake images. And, we want to it max. The equation is listed below.



Also, in the generator side, we want the chance to be minimum.



Since the generators and discriminators always process at the same time, we combined them together. It can be the equation below.



After defining generator, discriminator and loss function (and also the optimization function as Adam), we can train the model and plot the output. We ran several epochs and the following are some results from our model.

**Implication and limitation:**

With this model we designed, we can form a brand new image of cat never exists before. In other field, people can form human’s faces which can grab the features you want to contain and represent. Also, we can falsify some images to do legal things( although not recommend to do so). For those models need to use many pictures to do classification, if they do not have enough pictures to train the models, they can use DCGAN to help them generate more and more pictures.

However, we have a lot of limitations in this model and the project. First of all, we do not have enough and uniform images. From the resource, we only have 9000 thousands images and a lot of them present cats in different background and different motion. Although we have use classifier to catch the cat faces, we still do not have uniform ones which directly caused our generated images very obscure like mosaic. Moreover, the DCGAN model needs very powerful computers to deal with huge amounts of computations. So, we can only run few epochs(100 epochs takes almost 24 hours.) which make the model does not learn as much as they mean to do. In other words, if we can have more powerful computer with more epochs, the output images will be much better than the ones we have now. And, the parameters we set before we train the model are the optimal numbers we searched on the internet. We do not know the real optimal parameters for our own model and this problem really cause the images not as nicely as we want them to be.

**Conclusion:**

In conclusion, we used generator and discriminator function to form our major frame of DCGAN. In generator, we input noise to generate images to be checked by discriminator and after the images were classified as fake ones, the generator would learn some parameters to change the older ones. Again and again, the generator could produce images very close to the real images and output the images. We set several epochs to make the whole model learn much better and the results are much better after epochs.

DCGAN was very powerful in generate images based on the similar image database. If we could apply this technology into our career life, it could make a huge difference. DCGAN can make our data much power than we could think and larger database may have the brighter guide on the project.

**Bibliography**

Hitawala, S. (2018, January 12). *Comparative Study on Generative Adversarial Networks*. Retrieved May 12, 2019, from <https://arxiv.org/pdf/1801.04271.pdf>

Liu,D. (2017, November 30). A *Practical Guide to ReLU.* Retrieved May 12, 2019, from <https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7>

Hui, J. (2018, June 18). *GAN — DCGAN (Deep convolutional generative adversarial networks).* Retrieved May 12, 2019, from https://medium.com/@jonathan\_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

Hui, J. (2018, June 21).GAN — *Why it is so hard to train Generative Adversarial Networks!* Retrieved May 12, 2019, from <https://medium.com/@jonathan_hui/gan-dcgan-deep-convolutional-generative-adversarial-networks-df855c438f>

[Nguyen Canh](https://www.researchgate.net/profile/Thuong_Nguyen_Canh), T. (2018, June 8). *Advantages and Limitations of Generative Adversarial Networks (GANs)?* Retrieved May 12, 2019, from <https://www.researchgate.net/post/Can_anyone_inform_me_please_about_the_advantages_and_limitations_of_Generative_Adversarial_Networks_GANs>

Goodfellow, I. (2016, October 25). *I invented generative adversarial networks*. Retrieved May 12, 2019, from <https://www.quora.com/What-are-the-pros-and-cons-of-using-generative-adversarial-networks-a-type-of-neural-network-Could-they-be-applied-to-things-like-audio-waveform-via-RNN-Why-or-why-not>

Castleman, K. (1996). *Digital Image Processing*. Retrieved May 12, 2019, from http://www.owlnet.rice.edu/~elec539/Projects99/BACH/proj2/refs.html