Conditional generative adversarial network for generating cat breed images

Veronika Kalousková
ČVUT - FIT
kalouver@fit.cvut.cz

December 30, 2021

1 Introduction

The aim of this semestral project is to create a conditional generative adversarial network (CGAN), capable of generating images of cat breeds. A typical generative adversarial network (GAN) consists of two models, the discriminator and generator, playing a zero-sum-game, where the generator is trying to fool the discriminator into thinking the generated images are in fact real. [6] The conditional variation makes use of image labels, in order to be able to generate images corresponding to a certain class. [9] These networks are notoriously hard to train, because a proper balance needs to be kept between both of the models in order to achieve stable training.

2 Data

The biggest problem I encountered working on the project was getting appropriate data. For the checkpoint I implemented a GAN and used the Cats faces 64x64 data set from Kaggle. [5] The data set I chose for training the final CGAN implementation is the Cat Breeds Dataset from Kaggle [8], which contains images of cats sorted into folders based on the cat breed. However, the images do not contain only cat faces and are so variable, that the model was not able to converge to generating plausible cat images without preprocessing the images first. After encountering this problem I used the OpenCV Cascade Classifier [1] for identifying and cropping the cat faces.

At first I wanted the model to work with at least 128x128 images, but many of the identified faces were smaller than this so I resized the final cat faces images to 64x64. As a result of preprocessing I ended up with much less data, and was able to choose only six breeds represented by more than 1000 images. Unfortunately, because of this I had to omit distinct looking breeds such as the British Shorthair, which would serve well for identifying traits in the generated images. Overall, after image preprocessing I ended up with 10568 images of cats, labeled by their breeds, each breed containing between about 1200 and 2000 images.

3 Method

During the modelling phase of the project I followed these two tutorials [3] [4], using a similar model architecture, but personalizing it for my project, and trying to include as many tips as possible from this list [11]. Since the implemented model is a conditional deep convolutional generative adversarial network (Conditional-DCGAN), the discriminator architecture comprises of concatenated embedded label input, and image input, followed by downsampling layers with LeakyReLU activation functions, at the end outputting a single prediction. The downsampling is performed from 64x64 to 4x4, using strided convolution instead of pooling, each layer containing double the amount of filters as the previous one, sampled from a Gaussian distribution. The used optimizer is Adam, with learning rate set to 0.00008.

The generator also takes embedded label as an imput, together with points from the latent space, also making use of upsampling layers from 4x4 to 64x64, using strided convolution, followed by LeakyReLU activation functions. The used optimizer is Adam, with learning rate set to 0.0002. In both of the models I tried adding batch normalization and dropout layers, which should help stabilize the training, but was not able to reach better results than with current model architecture. The biggest aspect of stabilizing the model was finding a correct balance between the learning rates of both discriminator and generator. This article [2] inspired me to set a bit higher learning rate for generator than discriminator and it worked wonders.

During the training process the discriminator is trained separately on true and false images, and the generator is trained with inverted class labels. The positive class labels are also smoothed to include a range from 0.9 to 1.0, because GANs often suf-

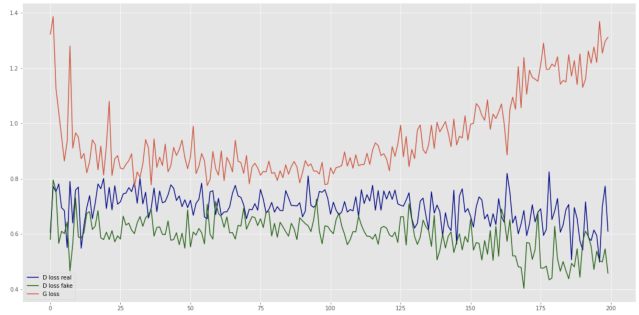

Figure 1: Training process loss

fer from overconfidence, which leads to producing images of lower quality. [7] The latent points are sampled from a space of size 128, and the batch size used is 64 [10]. Setting higher and lower numbers for both latent space and batch size resulted in convergence failure, or impaired image quality.

The blue and green lines on figure 1 display the discriminator loss, and the red line generator loss during training. The generator loss was expected to hover somewhere between 1.0 and 2.0, and the discriminator loss was supposed to go down to around 0.5 for both real and fake samples. It begun to get there after about 160 epochs, because of which I believe that longer training than 200 epochs might help stabilize the process, but this was hard to reach with standard Google Colab setup. I kept track of these during the whole training process to be able to identify failure modes early on, for example discriminator or generator loss falling sharply to zero meant that there was convergence failure present.

4 Results

For the checkpoint I implemented a GAN, which I turned into a conditional GAN for the final submission, by adding class label input for the models, and passing the corresponding labels during training. The results I obtained in both phases are different though, which I believe is due to the data sets used. The data set containing only cat faces contains 15700 images of cats faces, perfectly cropped and centered, as compared to the cat breeds data set, which contains various images of cats, automatically cropped during preprocessing. Using visual inspection about 1 to 2 images in 25 were mistakenly identified objects as cats, and about 5 pictures contain cats not looking directly into the camera. As a result of this, using the same parameters, the CGAN model using the second data set was not able to converge to images as good as the GAN trained on more precise images.

The cat faces generated by GAN can be seen in figures 2 and 3. After training for 100 epochs there


Figure 2: GAN generated cat


Figure 3: GAN generated cat

is a nice face symmetry present, with distinct face features and color of fur. On the other hand, the cat faces generated by CGAN, as seen on figures 4 and 5, after 200 epochs, are not as symmetrical, but still contain distinct eyes, noses, and fur patterns. The selected images are handpicked, not all of the cats were this good looking, but overall all of the generated images could be clearly identified as cats.

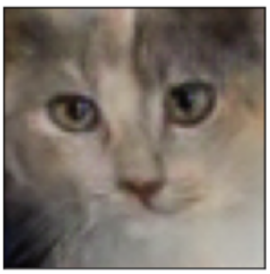
Figure 4: CGAN generated Calico cat


Figure 5: CGAN generated Bengal cat

Even though the cat images generated by CGAN are not as as nice looking as the ones generated by GAN, it is important that there is no mode collapse present, and since it is a conditional GAN, its ability

to generate distinct classes. For testing this I chose two different color variations - Bengal and Calico. The image on figure 4 is generated with Calico class label, it contains the typical fur pattern of dilute grey, white and beige spots. Figure 5 represents an image generated with Bengal class label, showing typical striped face pattern in brown color.

5 Conclusion

The main goals of this project were reached - the Conditional-DCGAN model was able to converge to realistic looking cat images, and was able to generate images belonging to selected classes. In the future this project could be expanded to use images of at least 128x128 resolution, as was the original plan, but it would require better data. Building a CGAN rather than GAN made it harder to find a good data set. In order to have better input data it would also be beneficial to preprocess the data by visual inspection, removing incorrectly identified objects by the OpenCV Cascade Classifier. [1] Finally, it would definitely be beneficial to be able to train the model for more than 200 epochs, which would require more computational power or at least significantly more time.

References

[1] Cascade classifier.

[2] Meow generator, Jul 2017.

[3] Jason Brownlee. How to develop a conditional gan (cgan) from scratch, Sep 2020.

[4] Jason Brownlee. How to develop a gan to generate cifar10 small color photographs, Sep 2020.

[5] Spandan Ghosh. Cats faces 64x64 (for generative models), Feb 2019.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, October 2020.

[7] Jonathan Hui. Gan - ways to improve gan performance, Mar 2020.

[8] ma7555. Cat breeds dataset, Dec 2019.

[9] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[10] Tushar Mittal. Tips on training your gans faster and achieve better results, Aug 2019.

[11] Soumith. soumith/ganhacks: starter from "how to train a gan?" at nips2016.