Cats VS Dogs Image Classification and Generation Using DCGAN

Warren Quattrocchi
ECS
CSUS
SACRAMENTO, CA, US
warrenquattrocchi@csus.edu

Janus Kwan
ECS
CSUS
SACRAMENTO, CA, US
JKWAN@CSUS.EDU

Andrew Wright
ECS
CSUS
SACRAMENTO, CA, US
andrewwright@csus.edu

ABSTRACT

Motivation:

Image classification, and this problem in particular, is very popular in machine learning. Web services are often protected with a challenge that's supposed to be easy for people to solve, but difficult for computers. Such a challenge is often called a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) or HIP (Human Interactive Proof).

Problem Statement:

Asirra (Animal Species Image Recognition for Restricting Access) is a HIP from 2007 that works by asking users to identify photographs of cats and dogs. This task is difficult for computers, but studies have shown that people can accomplish it quickly and accurately. It contains over three million images of cats and dogs, manually classified by people at thousands of animal shelters across the United States.

While random guessing is the easiest form of attack, various forms of image recognition can allow an attacker to make guesses that are better than random. There is enormous diversity in the photo database (a wide variety of backgrounds, angles, poses, lighting, etc.), making accurate automatic classification difficult. In an informal poll conducted many years ago, computer vision experts posited that a classifier with better than 60% accuracy would be difficult without a major advance in the state of the art.

Approach:

Since we were using models we didn't learn in class for this project, we decided to first make sure we could get a simple DCGAN working. After our initial black and white DCGAN was complete, we started working on the CNN and Transfer Learning models for classification, and then returned to the DCGAN again to implement color images.

Results

For the Image classification portion of the project, we achieved much better results from our first CNN than the Transfer Learning model. (.91 vs .8 f1 score). The results we achieved from the DCGAN were very interesting and while most of the generated images barely resembled a cat/dog, we were able to get a few pretty good examples out of it.

(A color image from the final DCGAN iteration)

(A black and white image from the first DCGAN iteration)

INTRODUCTION

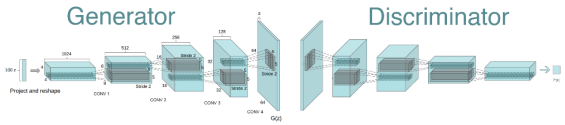
A classification problem was chosen. Since it is one of the most prevalent problem types and also important. Classification is currently a hot topic, with all

the focus on real time recognition, which is all done through classification. The problem here is a simple one, dogs vs cats. Classify it as either a dog, or a cat. The dataset is run through two different CNN networks, one built by the team, and a transfer learning model, VGG.

In addition, a second problem was chosen if time allowed, a CATDOG DCGAN problem. One of the most amazing things about AI is the ability of it to learn and generate. DCGAN is an Adversarial network. The concept is to take two models, one who will keep generating work, and another who will keep examining and testing said work. Initially, since the generator is fresh and generally terrible, the second model, the discriminator, will flag it as fake, and the generator will take that, and generate a new one to hopefully beat the discriminator. The interesting concept is that both models are trained at the same time, while the discriminator is learning to differentiate between fake and real, the generator is also learning to create better fakes. The endless cycle allows the generator, after thousands of epoch to generate a picture of a newsworthy cat.

PROBLEM FOUNDATION

For Our DCGAN, the initial model started with an input comprised of a 'random noise vector' which was turned into an image via the following diagram:



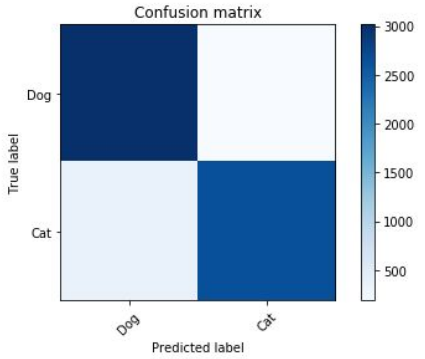
The image generated would then be run through the 'Discriminator' (right side of diagram) which would determine whether or not the generators image could be correctly identified as a cat.

EXPERIMENTAL EVALUATION

The dataset used in the following problems was downloaded from a Cats Vs Dogs competition. (https://www.kaggle.com/c/dogs-vs-cats-redux-kernels-edition) The kaggle dataset was extremely abundant, it offered 30000 pictures, half dog half cat, with a small margin of outliers. The dataset for the first problem, classification was split into a 75-25, with 75 being train and 25 being test data. In the DCGAN, the data was not split, as the dataset included a test archive and that was used instead of splitting the training data.

For the two CNN models, the metrics used to compare the models would be the standard, Real Loss, Fake Loss, Total Loss and Accuracy. The CNN we created had 18 layers total. With one being input, 16 being hidden layers and one output layer. Our hidden layers consisted of 12 Conv2d layers with 4 Max Pooling layers approximately every 3 layers. The second CNN model, the transfer learning model is a pre-built model called VGG. It consists of 18 layers, with one input, 16 hidden, one output. 12 Conv2D layers and 4 Max Pooling layers. Since it is a pre-built model, on top of it we also added 2 dense layers and 2 dropout layers.

(CNN Confusion Matrix and Metrics)



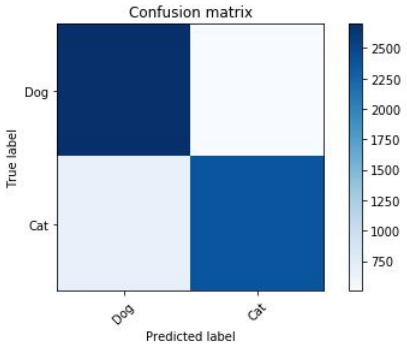

```
Accuracy: 0.90816
Averaged F1: 0.9079956160391608
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      3202
           1       0.93      0.87      0.90      3048

    accuracy                           0.91      6250
   macro avg       0.91      0.91      0.91      6250
weighted avg       0.91      0.91      0.91      6250
```

To compare the two models, the CNNs we ran them both with a max of 100 epochs, and early stopping and checkpointing to ensure it does not over train. The results were obvious, our personal model achieved an average accuracy of 91% whereas the VGG model, achieved an accuracy of 81%.

(Transfer Learning Confusion Matrix and Metrics)



```
Accuracy: 0.81344
Averaged F1: 0.8132039772888372
              precision    recall  f1-score   support

           0       0.80      0.84      0.82      3202
           1       0.83      0.78      0.80      3048

    accuracy                           0.81      6250
   macro avg       0.81      0.81      0.81      6250
weighted avg       0.81      0.81      0.81      6250
```

The same dataset was used for the DCGAN, you may have noticed above, it was called a CATDOG DCGAN, that is because we did not add in a filter to allow the generator model to train on only one, we allowed it to learn both. The end result was hilarious to say the least. The main experimentation performed on the DCGAN was related to learning rate. According to an article we found by someone with a lot of experience in DCGAN's [1] "The learning rate is one of the most important hyperparameter, if not the most, as even small changes of it can lead to **radical changes** during training. Usually you can allow higher learning rates when using larger batch sizes, but being on the **conservative side** was almost always a safe choice in my experience.". While we found the results achieved from using a smaller learning rate to be more desirable than that of the higher one, it increased the training time to a point that we would need multiple days for a decent output. Most of the 'passable' images we generated came from a learning rate of around .0001

RELATED WORK

A large portion of our DCGAN came from a tutorial on tensorflow's website (https://www.tensorflow.org/tutorials/generative/dcgan). In this tutorial, they used the MNIST dataset to generate black and white digits that were 28x28.

While this was a good starting point, we were using a different dataset, different input size, and we wanted to add color. We improved this method by adding the ability to do color and supporting our dataset.

CONCLUSION

The results we got for the image classification portion of the project were much better from the first CNN model - The transfer learning model couldn't keep up with the accuracy - possibly due to our data set being large enough to not see the benefits of transfer learning. The results we achieved from the DCGAN were, even if not the most perfect images of cats, very interesting and better than expected in some cases. We went into this project not really knowing what to expect from Generative Adversarial Networks, and what we got was satisfactory.

WORK DIVISION

This was a long and difficult project, We had 4 models to work on, 2 CNN and 2 GAN. Andrew worked with Janus on the CNN models, and Warren followed the tutorial to create the GAN mode along with helping code both the CNN models when there was trouble. As for most of the training, Andrew was unable to get tensorflow 2.0 working on his computer so he unfortunately was unable to help train the DCGAN model. Most of the training was done on Warren's computer. We were unable to get checkpointing working, so our image results for GAN were all single run through generations limiting the max epoch to around 1000-2000..

LEARNING EXPERIENCE

Stepping into this class, the expectations that were held for this class, was well the last 3-4 weeks of the semester. The Tensorflow assignments and lectures were amazing. Learning about things AI can do through models. DCGAN in particular is an amazing idea, an AI training another AI whilst learning itself. DCGAN in particular was a very interesting subject to study and create.

It took around 100 epochs before the DCGAN model we used to start producing silhouettes, each epoch on a GTX1080 took around 18-24 seconds. We also had to

install tensorflow 2.0 which took us a few days to figure out. Originally we had reshaped the images in our dataset to 128x128 but due to the large picture size and slow speed of running it through the training, we opted to change it to 64x64, lowering the quality of the pictures by half.

REFERENCES

[1]https://towardsdatascience.com/10-lessons-i-learned-training-generative-adversarial-networks-gans-for-a-year-c9071159628