Naruto Face GAN

Noah Buchanan
CS 4143: Deep Learning
University of Arkansas - Fort Smith
Spring 2022

May 4, 2022

Introduction

Naruto Face GAN is a deep convolutional Generative Adversarial Network. The network was trained to replicate the art style of various characters from the show Naruto. The networks only input is a seed that is randomized for all intents and purpose for this project and the output is a generated face that should resemble the art style of Naruto characters.

Background

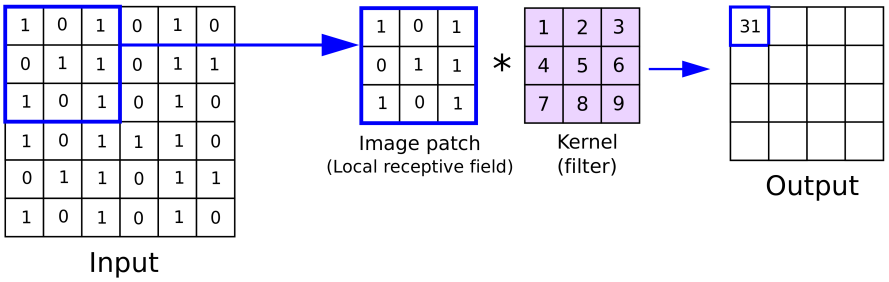
Generative Adversarial Networks

GANs consist of two main components: a Generator and Discriminator. The names are fairly intuitive as this is exactly what they do. The generator and Discriminator are opposing each other as the word "adversarial" would suggest: the Generator is trying to fool the Discriminator into thinking that the outputs it is creating are in fact actual outputs that have not been generated and the Discriminator's job is, simply put, to not be fooled by the Generator. The Generator and Discriminator network architectures mirror each other. As each network learns they both improve at their respective tasks and the idea here is to achieve indistinguishably generated outputs. GANs can be somewhat unstable when training if either the Discriminator or Generator learns to fast in relation to the other. Both networks are also trained from a single back propagation originating for the discriminator so the network is very sensitive to change.

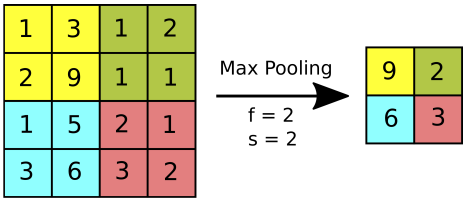
Convolutional Neural Networks

Convolutional Neural Networks are typically the go to type of neural network for image-based tasks. CNNs reduce the number of input nodes and parameters and are also capable of taking advantage of the correlations that are observed in complex images. CNNs typically follow this pattern: First, a CNN applies a

filter to an image. Filters are also sometimes called kernels. A filter is a smaller matrix of pixels typically from about 1×1 to 5×5 and the intensity of each pixel(floating point value between 0 - 1) is initialized randomly at the beginning and is later determined by back propagation. The filter is overlaid onto the image and the dot product of a filter sized part of the image and the filter itself is calculated and added into a feature map.



We can say that the filter is "convolved" with the input after performing this which i what gives CNNs their name. At this point an activation function is typically applied to the feature map. Once the feature map has been run through and activation function we can use a technique called pooling on the feature map. Pooling reduces the dimensionality of the input by a factor of 2 or more. Max pooling, which is typically the most used, simply takes the max value from a block of n × n and that is the new output for that block of size n × n in the original input.



As you can see the dimensionality is reduced quite a bit. These steps do not always have to be performed in this order. The output is also up to your goals and specific task. Sometimes the output is flattened and sent to a regular neural network after being convolved and other times the output is left as feature maps typically for heatmap classification tasks.

Naruto Face GAN

Training Data

The data used to train my model consisted of roughly 3600 images of varying sizes. For training purposes all data is scaled to 64x64x3, all images are RGB

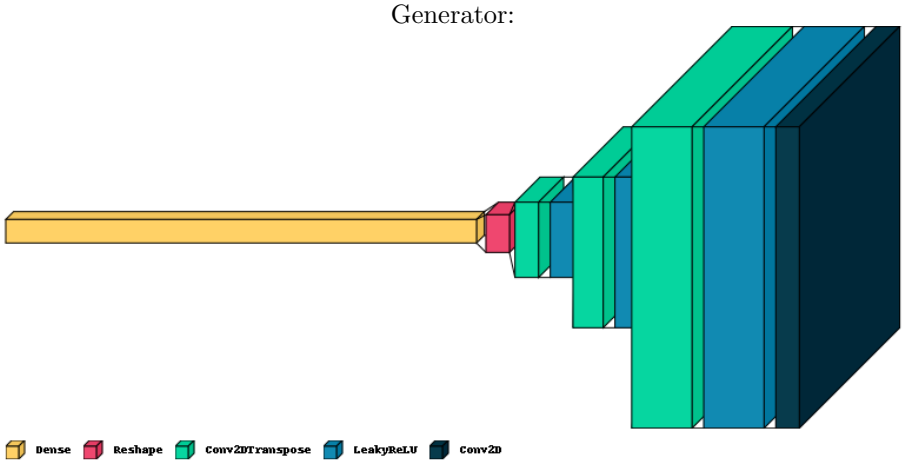
hence the 3.
Some examples of training data are as follows:



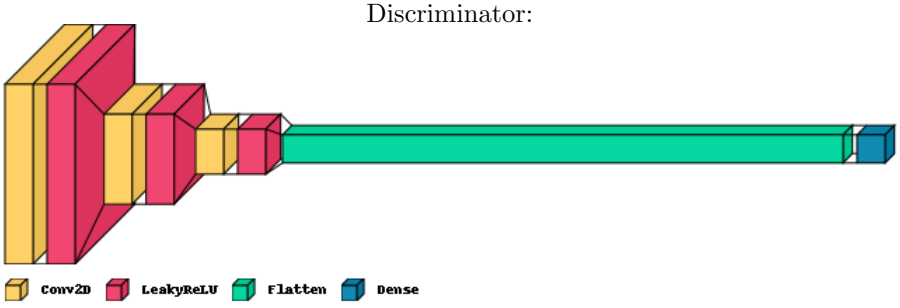
Model Architecture

The entire architecture consists of roughly 4.4 million trainable parameters, 4 million of those belonging to the generator. As I specified before, the Generator and Discriminator networks mirror each other as follows:

Generator:



Discriminator:



The parameters of each part is representative of the difficulty of the task and I tried my best to balance their capabilities so that one would not eclipse the other during training. Being that the discriminator's task of simply classifying real or fake images is drastically easier in comparison to generating fake images, the parameter count reflects this.

Model Performance

Below are some outputs from the trained GAN:

The faces are not always as good as I would hope however you can see that it learned some nuances of the characters specifically the fact that it almost always attempts to add a headband when generating a new image.

Model Details

Dropout and leaky ReLU were both utilized throughout the network. Binary Cross Entropy was the loss function used to train the network. Both the Generator and Discriminator utilized a Adam optimizer at a learning rate of $1e^{-3}$.

The learning rates could have been tuned and the size of the architectures corresponding to those changes but ultimately I just decided to keep both of the learning rates the same.

Future Work

As good as this turned out in the time frame that I was working on it, there are many things I would change given more time. The dataset used was quite limited in its actual variance of characters and it is noticeable in the output considering the disproportionate amount of yellow and pink haired characters it generates. I would have preferred a dataset with more characters including less important ones so that it could learn more of the art style rather than the characters themselves. The main characters inhabit the majority of all the images. I think given more time as well for training the model could have performed leagues above where it is now. Unfortunately my resources were limited and I do think it would have continued to learn. I also would have used a larger architecture with more parameters and possibly a larger output value, somewhere around 128×128 for clearer pictures.