AI-Generated Image Classifier

Thomas Cazort

Introduction

On September 1st, 2022 an AI-Generated piece of artwork won the Colorado State Fair's fine art competition. This was a historic outcome, as it was the first time art generated by a computer had won a competition. This sparked a lot of controversies online, as people argued over the validity of accepting AI-generated art as comparable to human-created art.

This event also proved that AI-generated art (AIGA) has become advanced enough to compete against and even outclass human-created art. Because AIGA has reached this level, it has become difficult to detect whether or not art was actually created by a human or not. In the case of the Colorado State Fair, they knew that the art was AI-generated. There isn't much in the way of someone maliciously submitting an AIGA to an art contest while claiming it was human-generated.

This inspired me to create a Convolutional Neural Network to classify art as either AI-generated or human-created. This will allow users to validate whether or not art is truly human-created or AI-Generated.

Related Work

After searching for an example of another model that classifies AIGA and human-created art, I came up empty handed. However, I have found some examples of similar work that share a few similarities.

The first example is of Generative Adversarial Networks (GANs) which were proposed by Goodfellow et al, in 2014. A GAN works by having two neural networks compete against each other, a Generator G and a Discriminator D. The generator learns to create unique images based on random noise, and the discriminator determines whether or not an image was created from the generator or not. It takes as input both geneorated images, and real images that the generator attempts to replicate. My model works similarly to the discriminator, except its using images generated from an already trained GAN. [Goodfellow, I. J., Mirza, M., Xu, B., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Networks. *arXiv*. https://doi.org/10.48550/arXiv.1406.2661]

The next example is of a deep learning model that attempts to classify videos as a deepfake or not. Deepfake videos are the product of machine-learning models that superimpose images and video clips onto a video. This technology is often used to superimpose a face onto an existing video, making it seem as though it was someone else in the video. Marie-Helen Maras and Alex Alexandrou have created a model to determine wether or not a video has been doctored using deepfake technology. This is similar to my model, as it attempts to determine if an image has been modified by AI. These images start as real pictures that have then been modified, making it much harder to determine authenticity. [Maras, M.-H., & Alexandrou, A. (2019). Determining authenticity of video evidence in the age of artificial intelligence and in the wake of Deepfake videos. The International Journal of Evidence & Proof, 23(3), 255–262. https://doi.org/10.1177/1365712718807226]

Dataset

To train the model, I needed to feed it both AIGI and human-created art. I would need a dataset of comparable AIGI and human-generated art in order to accurately and fairly train a model to determine the difference. I decided to use two different datasets from Kaggle for each type of art. The datasets I used were the 'Midjourney User Prompts & Generated Images' , for AIGA and the 'Wiki-Art : Visual Art Encyclopedia' for the human-generated art.

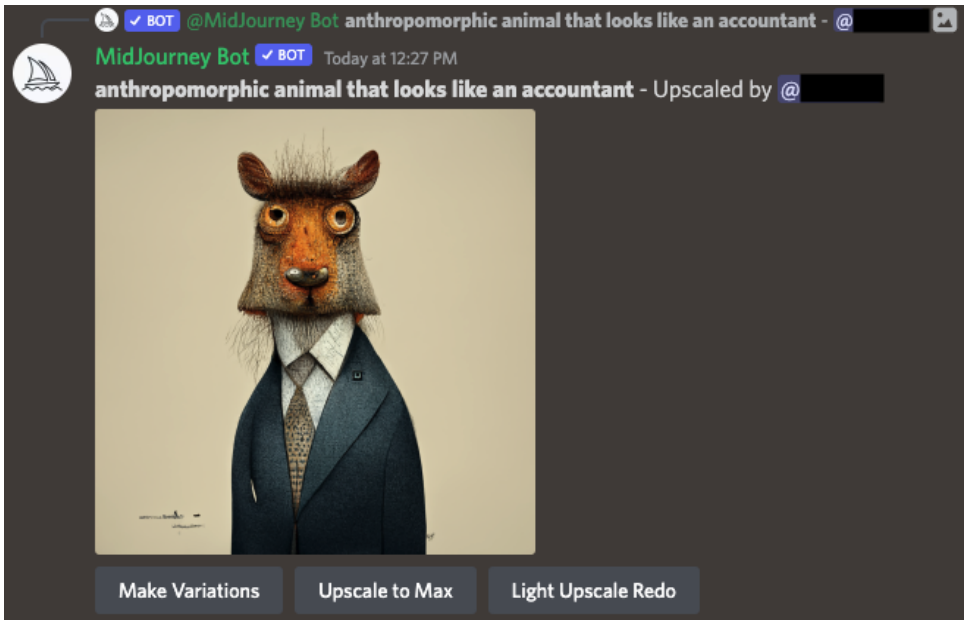
Midjourney

Midjourney is a cutting-edge AI art generator based on the principle of Generative Adversarial Networks (GANs). Midjounrey takes user-inputted text and then generates an image based on the text. This can be used to create beautiful, unique, and complex images quickly and easily.

Midjourney is unique in its accessibility. It has a public discord server in which users can input their prompt to be created by the GAN. This means that thousands of examples of AIGA are created daily in the discord. The Midjourney dataset from Kaggle scrapes these images along with the prompt.

When a user inputs a prompt into the server, Midjourney creates 4 different images based on the same prompt and outputs them into one image with each on all corners.

Users can then choose to upscale one of the images to get a higher-resolution version.



Users can then make different variations based on the upscaled image.

Of the 286k prompts submitted into the scraped discords, 60% were requests for new images (initial or variation requests for a previously-generated image), and 40% were requests for upscaling previously-generated images. I chose to use only the upscaled images for two reasons. One, it would be easy for the model to learn that images that looked split into fours were AI-generated. And two, I made the assumption that upscaled images were likely better quality and thus more likely to potentially be mistaken as human-created.

I chose to use the Midjourney dataset for the AIGA for multiple reasons. The first reason was that the AIGA that won the Colorado State Fair was generated by Midjourney. This means that it is capable of creating quality images. Another reason was that the best images were already somewhat determined based on whether or not they were upscaled.

Wiki-Art

Wiki-Art is a collection of paintings in multiple different styles and genres. These include abstract, animal-painting, cityscape, figurative, flower-painting, genre-painting, landscape, marina, mythological-painting, nude-painting-nu, portrait, religious-painting, still-life, and symbolic-painting.

I chose to use the figurative subgroup. This is because it looked the most similar to the kinds of images produced by Midjourney. This was a collection of 4500 images of different sizes and styles.

Wiki art was a good choice because it included art from a variety of different styles, techniques, and time periods. The specific dataset I chose was actually created to be used to

train GANs to create realistic art. I figured this would cause the images to be even more likely to be similar to that of what a GAN can create.

Preprocessing

In total, I collected around 4500 images from each dataset. One issue that arose was that all of the images needed to be the same size. I originally chose to resize all of the images to 512x512 using the python package PIL. This proved to be problematic, as it would cause my kernel to crash whenever I tried to convert the image into a form readable by the model. I opted to halve the sizes to 256x256 instead which solved the issue.

Once I had resized the images, I stored them all in a single file. I then created a CSV file that contained the file name, and whether or not it was AIGA (0 or 1). This CSV was then read into the main Image Classifier python file. All of the images were then read and converted into a 256x256x3 NumPy array using the Keras function 'img_to_array'. This put the images into a form that was readable to the model. I then normalized these values by dividing them by 255.

For some reason, about 20 Wiki-Art images were a different size and after multiple attempts of resizing, I still couldn't get them to the correct proportions. I chose to simply remove these images from the training. In total there were 4529 Midjourney images (Class 1) and 4480 Wiki-Art images (Class 0).

Methodology

I chose to use a Convolutional Neural Network (CNN) because it has historically proven to do very well in image classification problems. I started with the CNN example we were given for assignment 4. I experimented with many different variations of sequences, such as having multiple dense layers, multiple or few convolutional layers, and taking out the dropout layers altogether.

I also experimented with many different values within the sequence. I tried with different filter sizes, different amounts of dense layer nodes, and different percentages of dropout. I was limited to the amount I could try and change because of the amount of time it took to train and test each variation.

I also changed the amounts of epochs trained as well as the batch size. I found that increasing the epochs wasn't always beneficial as it tended to overfit the more added. I chose to use a 90/10 train test split. The training took about 35 minutes.

Experiments and Results

After experimenting and hyperparameter tuning, I came to the conclusion that the original model was the best fit. I did make a few changes to make it work for my use case. I changed the input shape to 256x256x3 to fit my input images. I also changed the output to a dense layer with 1 node and a sigmoid activation. This is so that the model can predict a number between 0 and 1, with anything under 0.5 being classified as human-created, and anything over being classified as AIGA.

```
model = Sequential()

model.add(Conv2D(filters=32,
                 kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)            (None, 254, 254, 32)      896

 conv2d_3 (Conv2D)            (None, 252, 252, 64)      18496

 max_pooling2d_1 (MaxPooling   (None, 126, 126, 64)     0
 2D)

 dropout_2 (Dropout)          (None, 126, 126, 64)      0

 flatten_1 (Flatten)          (None, 1016064)           0

 dense_2 (Dense)              (None, 128)               130056320

 dropout_3 (Dropout)          (None, 128)               0

 dense_3 (Dense)              (None, 1)                 129

=================================================================
Total params: 130,075,841
Trainable params: 130,075,841
Non-trainable params: 0
_________________________________________________________________
```

I chose the loss function as binary cross entropy because it works well for binary classification problems.

```
model.compile(loss=tf.keras.losses.binary_crossentropy,
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
```

After multiple rounds of testing, I found that 3 was the perfect amount of epochs. With anything more, the model would begin to overfit and have a lower testing accuracy. With anything less, it would be under fitted and have a low training and testing accuracy.

```
model.fit(X_train, y_train, batch_size=64, epochs=3, verbose=1)
```

It took a total of around 35 minutes to train the model with the specifications above. It ended training with an accuracy of 0.94 and a loss of 0.16.

```
Epoch 1/3
127/127 [==============================] - 713s 6s/step - loss: 1.5321 - accuracy: 0.7828
Epoch 2/3
127/127 [==============================] - 692s 5s/step - loss: 0.2690 - accuracy: 0.9018
Epoch 3/3
127/127 [==============================] - 648s 5s/step - loss: 0.1644 - accuracy: 0.9414
```

The testing took a total of 15 seconds and had an accuracy of 0.91 and a loss of 0.28.

```
results = model.evaluate(X_test, y_test, batch_size=64)
15/15 [==============================] - 15s 940ms/step - loss: 0.2825 - accuracy: 0.9134
```

Conclusion and Future Directions

Summarize and discuss possible extensions of the project

All in all, I am very happy with the results. A 91% testing accuracy is significant. This proves that while a human can be fooled by AIGA, there are still ways for a neural network to tell the difference.

One extension could be to feed the network more data. It would be interesting to give it AIGA from multiple different GANs, as opposed to just Midjourney. It would also be interesting to see how it would fair with different groups of art and not just the figurative section.

Another extension would be to use a model that doesn't need to change the image size. A big limitation of the CNN I used was that all of the images had to be 256x256. There are models that can take a variable size input. I would be interested in seeing the differences that keeping the images the same size would create.