Mighty Anonymize User Guide

Introduction

Protect the privacy of individuals in your image data with Mighty Anonymize. Mighty Anonymize runs state-of-the-art convolutional neural networks to automatically detect and localize human faces in digital images, and applies blurring to tight, surgical areas to obscure the faces with minimal impact on the surrounding image. Optimized for street scenes and dashcam imagery, Mighty Anonymize can redact faces even in profile, in the background, or in crowds.

Use Mighty Anonymize (GPU) to get started quickly. If you need to be able to configure the size and intensity of the blurring, use Mighty Anonymize (GPU - Advanced) for more fine-grained controls.

Introduction **1**

Running in Batch Mode **1**

Create a model 1

Prepare your inputs(S3 Prefix option) 2

Create a batch transform job 2

Creating Input JSON files for Mighty Anonymize (GPU - Advanced): **3**

About Mighty AI **5**

Running in Batch Mode

Create a model

Assuming you have subscribed to Mighty Anonymize(GPU)/ Mighty Anonymize(GPU Advanced), you should see the product on your AWS Marketplace Subscriptions.

1. Select Mighty Anonymize(GPU) / Mighty Anonymize(GPU - Advanced), click on the “Actions” button and select “Create Model”.

2. Specify a “Model Name”, IAM role and Network options. In the container definition section, for the container input options, select “Use a model package subscription from AWS Marketplace”.
3. Click on “Create Model”. A model with the model name you specified in Step 1 will be created.

Prepare your inputs(S3 Prefix option)

Mighty Anonymize(GPU)

1. Store the images you want to anonymize in a folder in a S3 bucket. (Note that larger images can only be processed in batch mode, not in inference mode.)

Mighty Anonymize(GPU Advanced)

1. Check the User Guide section below on creating the input json file that includes customizable blur size and intensity
2. For every image, create a corresponding json file and store the json files in a folder in a S3 bucket. (Note that larger files can only be processed in batch mode, not in inference mode)

Make sure that the model, input and output S3 bucket are all in the same AWS region.
You can also use provide your inputs using a Manifest file. The Sagemaker documents explain the input methods in detail.

Create a batch transform job

1. Specify a job name.
2. For the model, select the model you created.
3. Select ml.p2.xlarge or ml.p3.2xlarge for the instance type. To run a small test, specify 1 for instance count.
4. If the input files are larger than 6MB, in the additional configuration, specify the maximum size of your input files in the Max Payload size(MB) field.
5. “Input data configuration” :
    a. “S3 data type” - “S3 Prefix”,
    b. “Split Type” -“None”,
    c. “Compression” - “None”,
    d. ”Content type” : “application/x-image” for Mighty Anonymize(GPU) or “application/json” for Mighty Anonymize(GPU Advanced).
    e. “S3 location” : Path to the folder in the bucket where the input files are stored.
6. Output data configuration:
    a. S3 output path: S3 bucket or folder in a S3 bucket where you want to save the anonymized images. Ensure that this folder/bucket is in the same AWS region as the input bucket and the sagemaker model.

b. Click on the “create job” button. You can monitor the job by clicking on the batch transform option on the Sagemaker dashboard screen.

Creating Input JSON files for Mighty Anonymize (GPU - Advanced):

Mighty Anonymize (GPU - Advanced) requires JSON input. The example below shows how to generate the required json input from your image set. This allows us to support “scale” and “sigma” input parameters that let you control the size and intensity of the blurred area. See below for a Python-based serialization example.

If you prefer to use our built-in defaults for the blur settings, you can use Mighty Anonymize (GPU), which takes images as input and does not require a serialization step.

```
import json
import os
import base64
from skimage.io import imsave
import numpy as np
from imageio import imread
from skimage.color import gray2rgb
from skimage.util import img_as_ubyte

def load_image(image_path):
    """
    Inputs
    ------
    image_path: string
        path to image file on local filesystem
    Outputs
    -------
    image: numpy array
        Coerces image into 3-channel RGB image as Numpy array
    """
    image = imread(image_path).view(type=np.ndarray)
    if image.shape == (2,):
        # Strip off weird metadata
        image = image[0]
    if len(image.shape) == 2:
        # Duplicate channels
        image = gray2rgb(image)
    if image.shape[2] == 4:
        # Strip off alpha channel
        image = image[:,:,:3]
    if image.dtype != np.uint8:
        image = img_as_ubyte(image)
    return image


def convert_image_to_str(image):
    tmp_img_file_name = str(os.getpid()) + "." + "png"
    imsave(tmp_img_file_name, image)
    with open(tmp_img_file_name, 'rb') as image_file:
        byte_content = image_file.read()
    base64_bytes = base64.b64encode(byte_content)
    base64_string = base64_bytes.decode("utf-8")
    os.remove(tmp_img_file_name)
    return base64_string

def convert_str_to_image(image_str, img_ext="png"):
    base64_bytes_retr = bytes(image_str, "utf-8")
    img_bytes = base64.b64decode(base64_bytes_retr)
    tmp_img_file_name = str(os.getpid()) + "." + img_ext
    with open(tmp_img_file_name, "wb") as image_file_tmp:
        image_file_tmp.write(img_bytes)
    image = load_image(tmp_img_file_name)
    os.remove(tmp_img_file_name)
    return image


if __name__ = "__main__":
    f = load_image("image_path") # Enter your image path
    img_as_string = convert_image_to_str(f)
    # Higher values of sigma means stronger blurring.
    sigma = str(3) # Can be changed
    # Scale factor for the area to be blurred. A value greater than 1 will
# blur an area larger than the detected face
    scale = str(1) # Can be changed
    data = {"image":img_as_string, "sigma":sigma, "scale":scale}
    with open("input_json_file_path","w") as s:
        json.dump(data,s)
```

About Mighty AI

Mighty AI generates high quality training and validation datasets for teams building computer vision models for autonomous systems. Our fully managed approach to image labeling delivers highly complex ground truth data and model validation at enterprise scale. We consult with every client to configure annotation workflows based on unique project specifications and flexibly tune each project as requirements evolve. Our industry-leading cloud-based annotation tools (including segmentation, bounding boxes, keypoints, polylines, metadata attribution and more), global community of trained annotators, and machine learning-based quality assurance systems enable us to generate millions of complex ground truth labels and validate detection model output with guaranteed accuracy.