SignBot

Sam Thurman

Using computer vision to interpret American Sign Language

Table of Contents

About Me

Libraries and Packages

Data

- Numpy
- Sklearn
- ImageDataGenerator
    - Keras 2.3.1

Image

- Skimage
- Imageio
- OpenCV 4.2.0.34
- PIL 6.2.0 (pillow)

Modeling / Evaluation

- Keras 2.3.1
    - Tensorflow 2.2.0rc3
    - Python 3.7.3
- Matplotlib
- Seaborn
- LIME 0.2.0

Processing

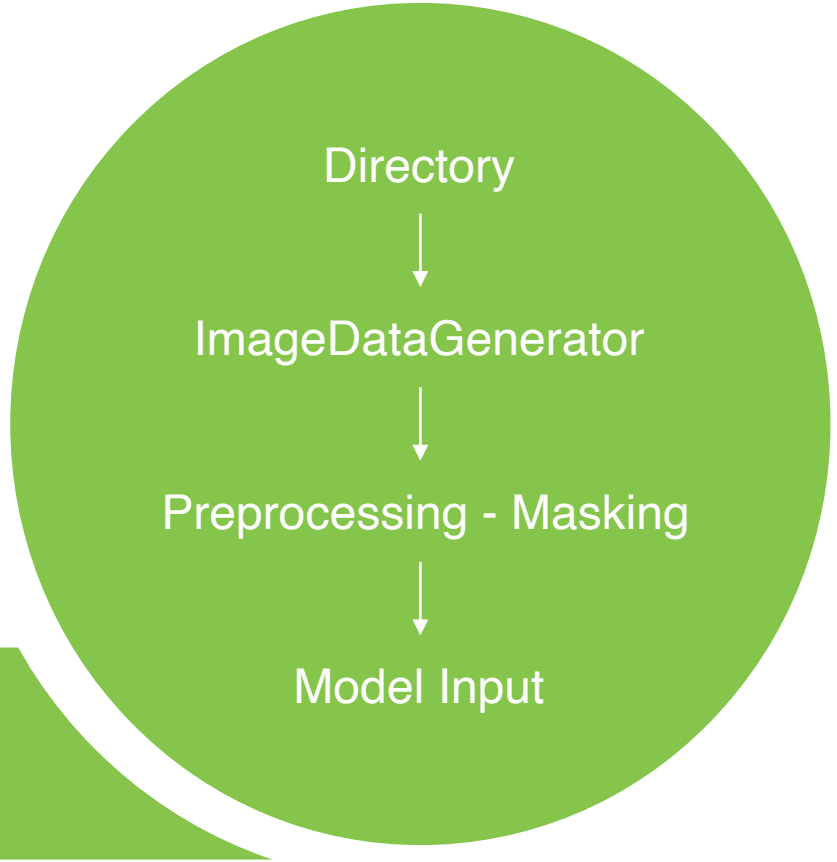
- Google Colabobratory's mystery GPU

"Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world." — Atul Butte

About the Data

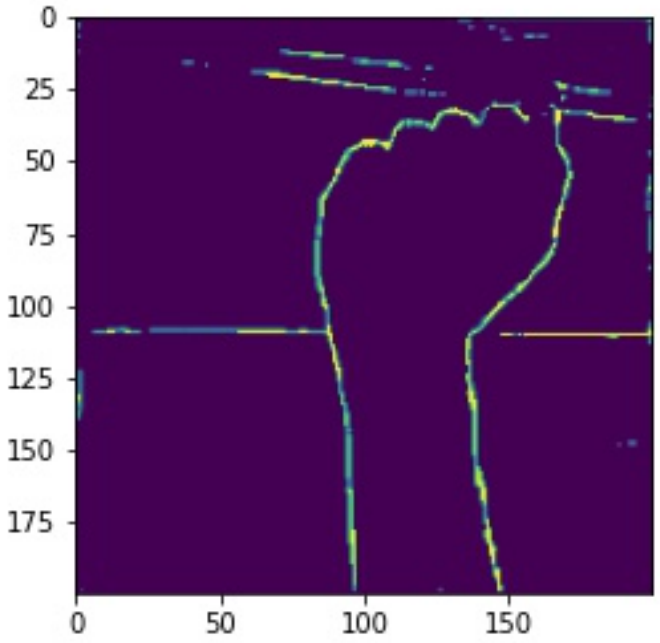
- 87,000 jpg files (3,000 ea. A-Z, space, del, nothing)
- Data Issues
  - Saturation - low/inconsistent
  - Background environment - too consistent
- ImageDataGenerator —> Keras
  - Grayscale
  - Preprocessing function

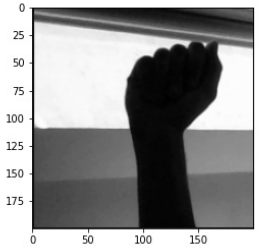
1.11GB data
Google Colab mystery GPU



Data Flow

Final shape (200,200,1)





Transforming the Data

- Semantic Segmentation
- Unet (Olaf Ronneberger) architecture tweaked to produce lines instead of full object shapes
- Format for Unet
  - Grayscale —> RGB
  - Resize/rescale pixels
- Format for Keras
  - RGB —> Grayscale
  - Add batch/color channels

Transforming the Data: the code

```
train_path = '../../data/asl_alphabet_training'
image_size = 200
batch_size = 32
datagen = keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255, preprocessing_function=predict_custom_image)
train_generator = datagen.flow_from_directory(
    validation_path,  # directory for validation images
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical',
    color_mode='grayscale',
    shuffle=False,
    subset='validation')
```

```
target_size = model.input.__dict__['_keras_shape'][1:-1]
im_resize = resize(image, target_size)
gray = gray2rgb(im_resize[:, :, 0])
im = np.expand_dims(gray, axis=0)
preds = model.predict(im)
pred = np.float_(preds[:, :, :, 0][0])
pred = resize(pred, (200, 200))
pred = np.expand_dims(pred, axis=2)
pred = np.expand_dims(pred, axis=0)
```

Model Architecture & Evaluation

Model achieved 96.67% accuracy on validation data, however further examination of the algorithm's focal points during classification showed that the model was using the absence of the hand against the image background to make a prediction.

Classifier pre-Unet

4 Convolutional Layers

Conv2D (3 X 3)
Batch Normalization
Activation - relu
MaxPooling2D (2 X 2)
Dropout (0.25)

2 Dense Layers

Dense (128/64)
Batch Normalization
Activation - relu
Dropout (0.3/0.4)

1 Layer

Conv2D 128 (3 X 3)

Dropout (0.2)

Flatten

Dense (29)
L1 regularization (0.02)

Final Classifier

Final model achieved 93.33% accuracy on validation data, and while the algorithm sometimes had difficulty distinguishing between fist-like shapes (letters A, E, M etc.), it was focusing on the actual shape and features of the hand in order make predictions. Addition of new data in attempt to increase generalizability is the next step for modeling.

Transferability

What Works

- Notebooks and model evaluation **running in Google Colab**
- OpenCV masking feature, no predictions

What's Going On

- Model loses memory of training (R/I deprecated model performance) in certain environments
- Model weight reset upon session switch
  - Sticky hardware specific errors
  - Recognized and unanimously aggravating issue in GitHub community

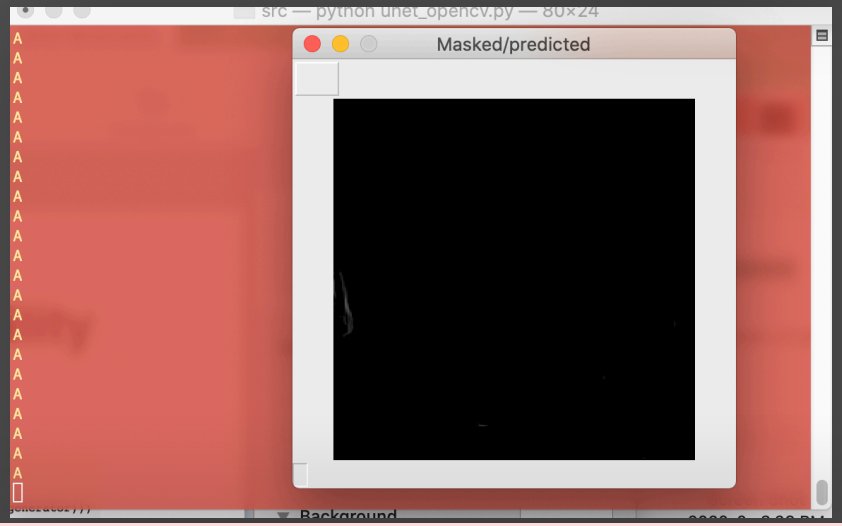
Correct Performance

```
model19 = load_model('model19.keras')
```

```
dict(zip(model19.metrics_names, model19.evaluate_generator(val_generator)))
```

```
{'accuracy': 0.9333333373069763,
 'loss': 73.62454223632812,
```

Faulty Performance

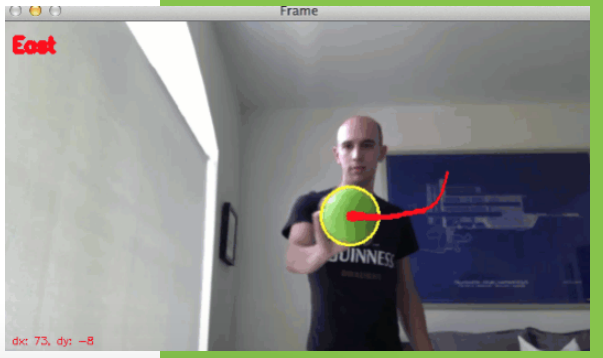



```
InvalidArgumentError: Tensor input_1:0, specified in either feed_devices or fetch_devices was not found in the Graph
```


OpenCV Demo

Next Steps

Although immediate next steps will be developing a new modeling pipeline to allow for model weight transferability, the following is an outline for the development of the rest of this project post-software fixes



1 More Data

Current data doesn't has issues, new data will include…

- Arm with clothing/watches etc.
- Different environments (new backgrounds w/ more noise, pics outside)
- Quality exposure/coloration of images

2 Object Detection

Object path tracking and bounding box would allow for…

- Groomed square frame from video cap
    - Reduce preprocessing done to whole video cap for model input
- Algorithm analysis of motion in movement

3 Expand Vocabulary

Requires object detection…

- Classic CNN modified to account for temporal dependencies in video

Would allow…

- Meaningful information transfer
- Addition of algorithm response (chatbot)

LinkedIn Github @sam-thurman

Fin.