Mushroom Identification

"Is it poisonous?"

**Ahmed Alweheiby, Joseph Santiago,
Paul Smith and Haowen Yong**

Project Goals

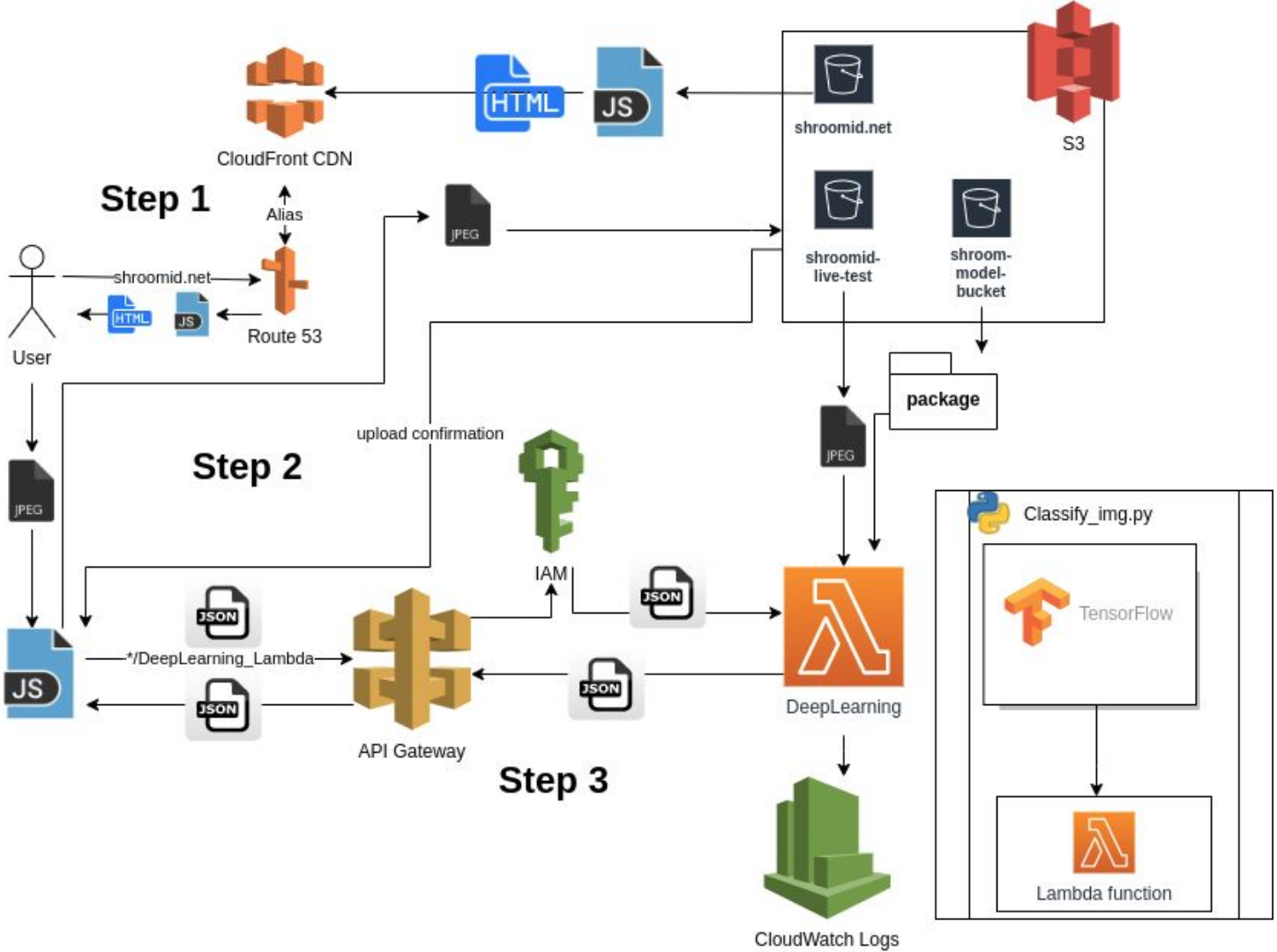
1. Identify a mushroom based on its image
2. Determine if the mushroom is poisonous or non-poisonous.
    a. Currently lacking species identification needed to determine poisonous or non-poisonous

Project Overview

- **Domain:** AI / Machine Learning
- **Programming language:** Python & Javascript
- **Deep learning framework:** TensorFlow & Keras
- **Algorithm:** Convolutional Neural Network, Inception Model
- **Data Set:** Natural History Museum of Denmark: Fungi Classification Challenge
- **Tools:** Google Colab, AWS

Functional Requirements

1. The user enters shroomid.net into the address bar and the system shall display our home page.
2. Successful uploads returns the following four fields:
    a. The top three predictions with scores
    b. The name of top prediction
    c. Whether the top prediction is poisonous
    d. The image that was submitted
3. The user receives an error when trying to submit an empty field
4. The user can only make one submission per image loaded
5. If a submitted image has low results than a failure image is displayed
6. A user can exit the browser and thereby exit the application


CloudFront CDN
HTML
JS
shroomid.net
S3
Step 1
Alias
JPEG
shroomid-
live-test
shroom-
model-
bucket
shroomid.net
HTML
JS
Route 53
User
package
upload confirmation
Step 2
JPEG
JPEG
IAM
Classify_img.py
JSON
JSON
JS
*/DeepLearning_Lambda
JSON
JSON
TensorFlow
DeepLearning
API Gateway
Step 3
Lambda function
CloudWatch Logs

Colab + Model Training

Initial Approach

- Google drive mounted to Google colab
- images uploaded in zipped folder and extracted in colab
- Keras library: Sequential Model
- Loss function: Categorical Cross Entropy
- Activation layer: Softmax
- Dropout and train_test_split

Colab + model training

```
# X_Data=X_Data/255

model = Sequential()

model.add(   Conv2D(150, (3,3), input_shape = X_train.shape[1:])   )
# model.add(   Conv2D(150, (3,3), input_shape = X.shape[1:])   )
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Dropout(0.4))
model.add(Conv2D(75, (3,3)))
model.add(Activation("relu"))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dense(128))

model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(64))

model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(64))

model.add(Dropout(0.4))
model.add(Dense(256))
model.add(Activation('softmax'))

model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=['accuracy'])

model.fit(X_Data, Y_Data, batch_size=64, epochs=20, validation_data=(X_test, y_test))
```

Colab + Model Training

Initial Approach Challenges

- Datasets is significantly large (13 GB)
- Datasets have 1407 classes
- Relatively few examples for each class
- Google Server Crashes due to memory overload and/or session timeout.
- Accuracy is low

Colab + Model Training

Second Approach

- Build a Dataset pipeline and producing a standard tensorflow dataset
- Utilize transfer knowledge using inception v3
- Produce higher accuracy .
- Training time is proportional to the size of the dataset.
- Was able to produce 92% accuracy for 200 classes.

Colab + Model Training

Datasets images
Testing images
json files for training
json file for testing
(16.2 GB)

mushroom.py
mushroom_test.py
mushroom.txt
Mushroom.py (testing)

Tensorflow
Inception V3

Mushroom_weights.hdf5
Mushroom.json

(total size 94 MB) for 200 classes

AWS

Colab + Model Training

Second Approach Challenges

- Very sophisticated to build the dataset pipeline
- Debugging is extremely time consuming. Lack of resources of the errors.
- Working on a colab platform is frustrating due to session timeout.
- Network bandwidth issues due to Internet overload.
- All images must be preprocessed to 299 x 299 size.

Colab + Model Training

```
_URL = "https://github.com/alweheiby/yy/blob/master/train.rar"
_NAMES=["A1000", "B10025", "C10052", "D10056", "F10057"]
_IMAGE_SHAPE =(None, None,3)

class Mushroom(tfds.core.GeneratorBasedBuilder):
  """Mushrooms small train dataset."""
```

```
def _info(self):
  """Mshroom dataset Images Dataset Class."""

  return tfds.core.DatasetInfo(
      builder=self,
      description=_DESCRIPTION,
      features=tfds.features.FeaturesDict({
          "image": tfds.features.Image(shape=_IMAGE_SHAPE),
          "label": tfds.features.ClassLabel(names=_NAMES),
      }),
      supervised_keys=("image", "label"),
      homepage= "https://github.com/alweheiby/Mushroom1.git",
      citation=_CITATION,
  )
```

```
def _split_generators(self, dl_manager):
  """Define Splits."""

  path = dl_manager.download_and_extract(_URL)

  return [
      tfds.core.SplitGenerator(
          name=tfds.Split.TRAIN,
          gen_kwargs={
              "data_dir_path": os.path.join(path, "train"),
          },
      ),
  ]
```

```
def _generate_examples(self, data_dir_path):
  """Generate images and labels for splits."""
  folder_names = ["A1000", "B10025", "C10052", "D10056", "F10057"]

  for folder in folder_names:
    folder_path = os.path.join(data_dir_path, folder)
    for file_name in tf.io.gfile.listdir(folder_path):
      if fnmatch.fnmatch(file_name, "*.JPG"):
        image = os.path.join(folder_path, file_name)
        label = folder.lower()
        image_id = "%s_%s" % (folder, file_name)
        yield image_id, {"image": image, "label": label}
```