Pet Identification with Machine Learning

A Convolutional Neural Network (CNN) model used for image recognition

GROUP 3: Christopher Yang, Sameer Zubairi,
Alan Deng and Richard Malolepszy

Objective

Create a Convolutional Neural Network model that can categorize images of the following types of pets:

Cats

Dogs

Fish

Birds

Hamsters

Bearded Dragons

Project Outline

1. Collection and processing of image files
2. Creation and optimization of the CNN model
3. Deployment of the model for end-user interaction

Collecting Image Files: Flickr vs. Imgur

- Web-scraped 500 images for each type of animal using various search terms to get more accurate results
  - “Pet bird”, “small bird”, “house bird”, etc.
  - Images were manually reviewed to ensure:
    - The images were of singular animals, rather than groups
    - Various orientations of the animals were collected
    - Any images where the animal is difficult to see were scrapped
- Image sources: Flickr, Imgur, Kaggle, Bing Images, Google Images:
  - Flickr had much better results than Imgur. Bing and Google had to be done manually.

Flickr


image_1.jpg
image_2.jpg
image_3.jpg
image_4.jpg
image_5.jpg
image_6.jpg
image_7.jpg
image_8.jpg
image_9.jpg
image_10.jpg
image_11.jpg
image_12.jpg
image_13.jpg
image_14.jpg
image_15.jpg
image_16.jpg
image_17.jpg
image_18.jpg
image_19.jpg
image_20.jpg
image_21.jpg
image_22.jpg
image_23.jpg
image_24.jpg
image_25.jpg
image_26.jpg
image_27.jpg
image_28.jpg
image_29.jpg
image_30.jpg
image_31.jpg
image_32.jpg
image_33.jpg
image_34.jpg
image_35.jpg
image_36.jpg
image_37.jpg
image_38.jpg
image_39.jpg
image_40.jpg
image_41.jpg
image_42.jpg
image_43.jpg
image_44.jpg
image_45.jpg
image_46.jpg
image_47.jpg
image_48.jpg
image_49.jpg
image_50.jpg
image_51.jpg
image_52.jpg
image_53.jpg
image_54.jpg
image_55.jpg
image_56.jpg
image_57.jpg
image_58.jpg
image_59.jpg
image_60.jpg

Imgur

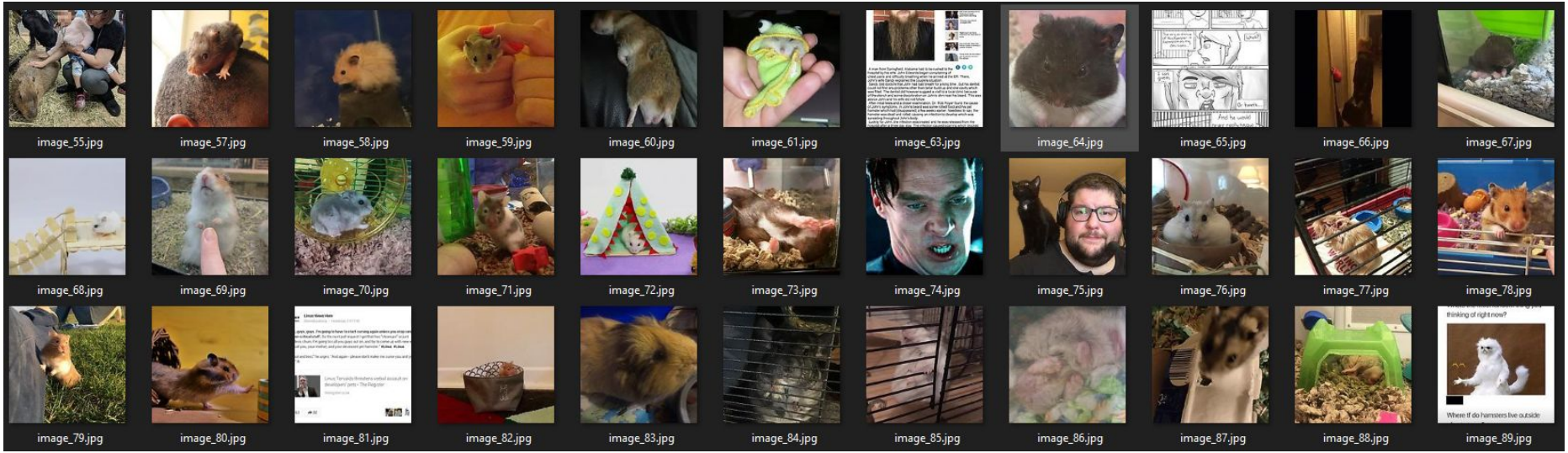


image_55.jpg
image_57.jpg
image_58.jpg
image_59.jpg
image_60.jpg
image_61.jpg
image_63.jpg
image_64.jpg
image_65.jpg
image_66.jpg
image_67.jpg
image_68.jpg
image_69.jpg
image_70.jpg
image_71.jpg
image_72.jpg
image_73.jpg
image_74.jpg
image_75.jpg
image_76.jpg
image_77.jpg
image_78.jpg
image_79.jpg
image_80.jpg
image_81.jpg
image_82.jpg
image_83.jpg
image_84.jpg
image_85.jpg
image_86.jpg
image_87.jpg
image_88.jpg
image_89.jpg
thinking of right now?
Where if do hamsters live outside

Image Processing

- Any .jpg or .jpeg format images converted to .png using the Pillow (PIL) library
- Images then converted to fit a 240x240 pixel frame
    - Aspect Ratio was maintained by filling the rest of the 240x240 pixel frame with a clear background
- Image files renamed to start with the animal name, e.g: dog1.png

Separating Test and Training Datasets

- Created test and train directories with folders for each animal:
    - Test:
        - Cat
        - Dog
        - Etc.
    - Train:
        - Cat
        - Dog
        - Etc.
- Used the Random library to randomly sort 25% of the processed images into the test directory and 75% into the train directory for each animal respectively

Building the Model in Tensorflow

- Input Layer: 32 nodes, 3x3 filter, activation: ReLu
- 3 Hidden Layers:
  - 64 nodes, 3x3 filter, activation: ReLu
  - 128 nodes, 3x3 filter, activation: ReLu
  - 512 nodes, activation: ReLu
- Output Layer: 6 nodes (for the classes of animals), activation: softmax

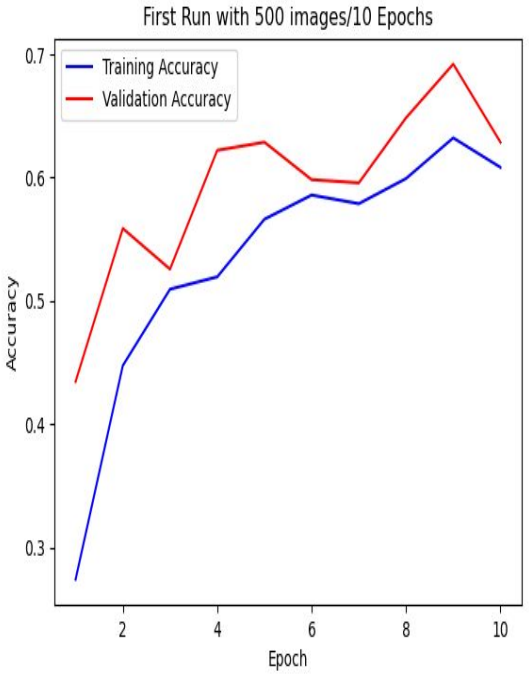
```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_HEIGHT, IMG_WIDTH, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax')  # 6 classes of animals
])
```

Fitting the model

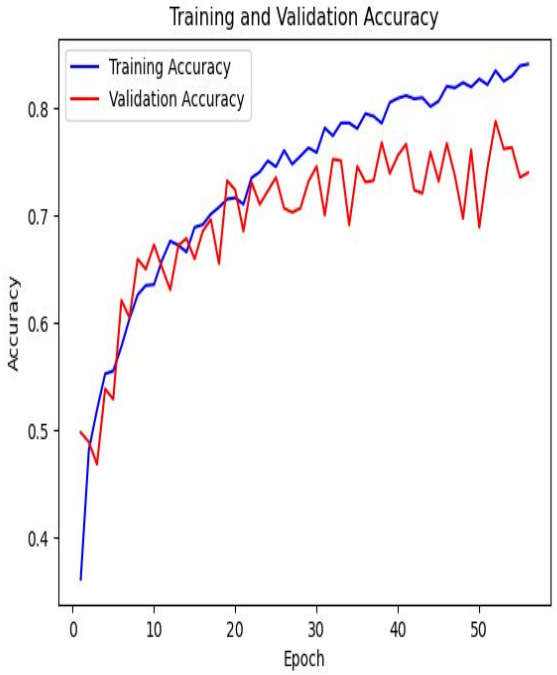
- ImageDataGenerator library from tensorflow used to generate batches of tensor image data
  - The test and train data was processed using the ImageDataGenerator library
  - The model was fit to the train tensor image data and validated using the test tensor image data
- Model was trained locally:
  - Training the data was extremely slow on Free Google Colab and needed to be trained locally over the course of MANY hours as it needed more computing power.
  - Local Machine was 4-5x times faster.
  - Maximum epochs ran was 100. This was too long of a train but we learned when overfitting occurs from the data to further refine our H5 model.
  - Overfitting noticed at around 25 - 30 epochs with most runs. Ultimately this is where we found the best result. We built both 25 and 30 models at the end.

Accuracy Plots:

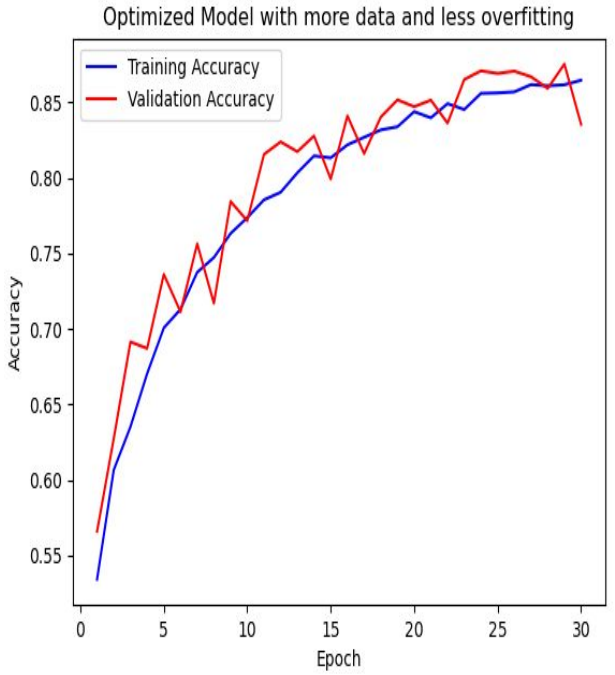
How it started:



Mid Journey:



Where we ended up:



DEMO TIME

Deploying the model using Flask

- Flask was used to deploy the model. Two main libraries were used:
  - Render template: Used to deploy a upload.html and a result.html
  - Tensorflow import_model and image: Used to access the .h5 model, and preprocess images
- 3 Main functions in the flask app:
  - Upload file: produces a GET request to show the upload.html page, and when user uploads an image (POST request), saves the file to temporary storage
  - Image Processing: Converts the image to 240X240 and .PNG format (same parameters used when training the model)
  - Classify Image: Accesses the .h5 model to return a class label for a pet
  - The image is deleted from temporary storage, and the animal class is returned on result.html

Model Limitations

- Through optimization techniques, achieved just over 85% validated accuracy
- Works best with clear view of pet that fills frame. Struggles with certain pet shapes and colorings, and when noise is introduced (other objects in photos, poor framing, multiple pets, pets in odd positions)

Takeaways

- Optimization: The larger the data set (more images) the better the learning. Additional training epochs can help but risks overfitting
- You get what you pay for as far as computing power.
- GitHub needs special plugins to work with files larger than 100MB (our H5 file was ~600MB)
- Image data takes tremendous amount of time to process, and this compounds when more data is added to the dataset with every step: preprocessing, cleaning, resizing etc.

Questions?