CAT AND DOG CLASSIFICATION

By: Susan Nunez, Madison Yonash, and Claire Robinson

INTRODUCTION

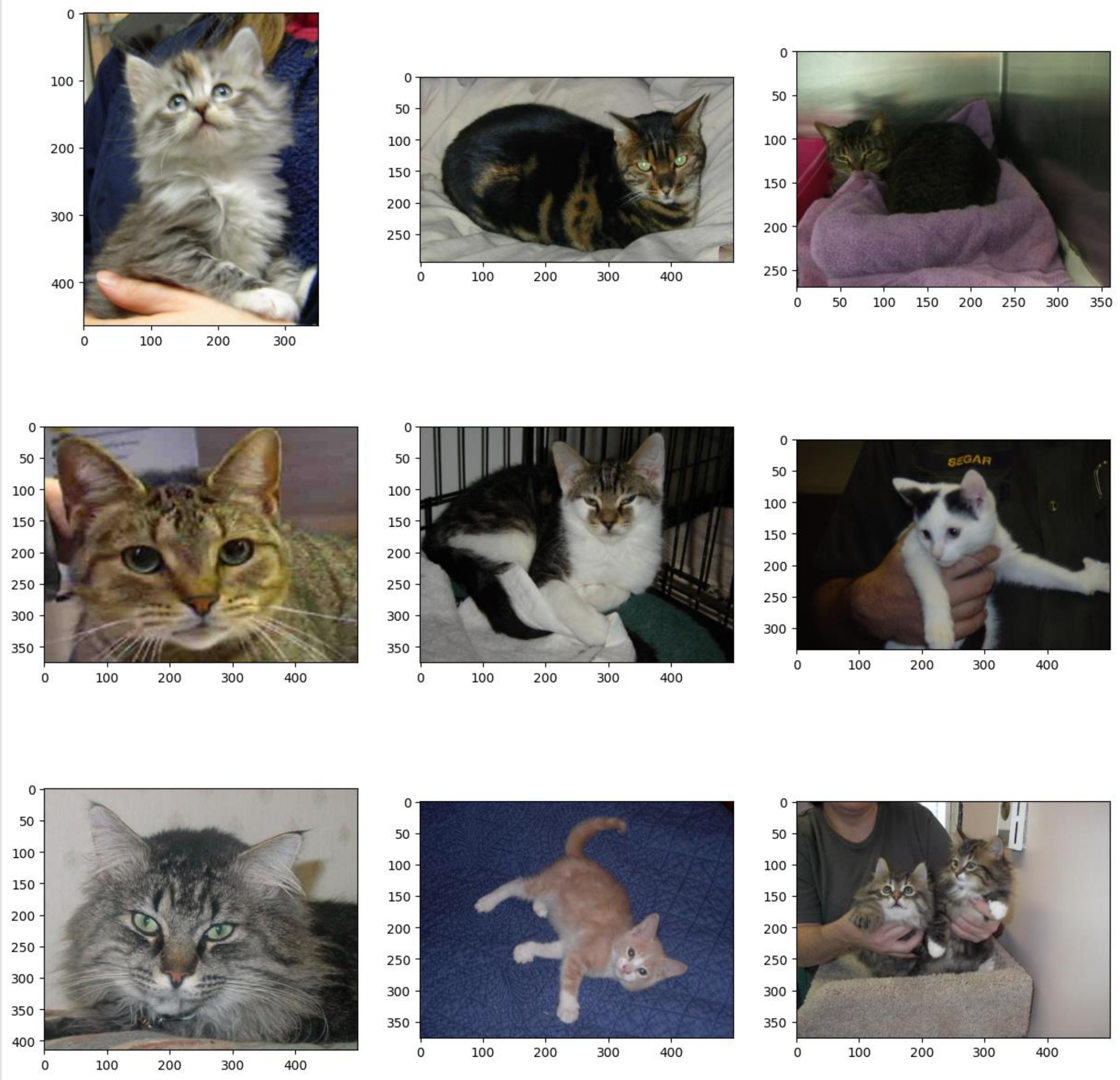
- Image classification using neural networks (NN) allows for the automatic categorization of visual data such as photos or videos
- Supervised learning with the goal of labeling the test image
- Specifically convolutional neural networks (CNN) are good at learning features such as edges, textures, and shapes

DATA

- This data set includes 25,000 images that are a subset of three million annotated images compiled by Petfinder.com and Microsoft
- There are 12,500 images for each class
- The dataset is not split into training and testing sets, but separated based on if they are a cat or dog
- Many of the images have additional noise or obstructions in the images
- The dataset is not uniform

	Cat	Dog
Normal		
Duplicated		
Obstructed		

EDA- RANDOM SAMPLING OF PHOTOS



DATA PREPROCESSING

Normalize Colors

- Dividing the pixel colors by 255
  - Common technique when dealing with images in Red, Green, and Blue color space
  - Range in color intensity from 0-255 but will be 0-1 after normalization

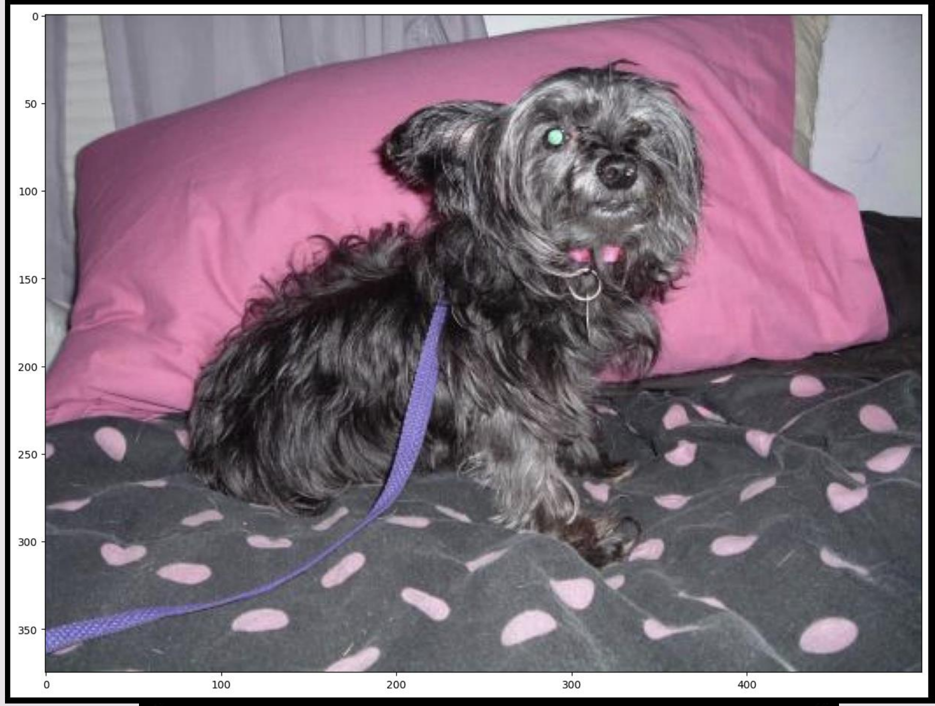
Resize Images

- Change all pictures to be 128x128 pixels

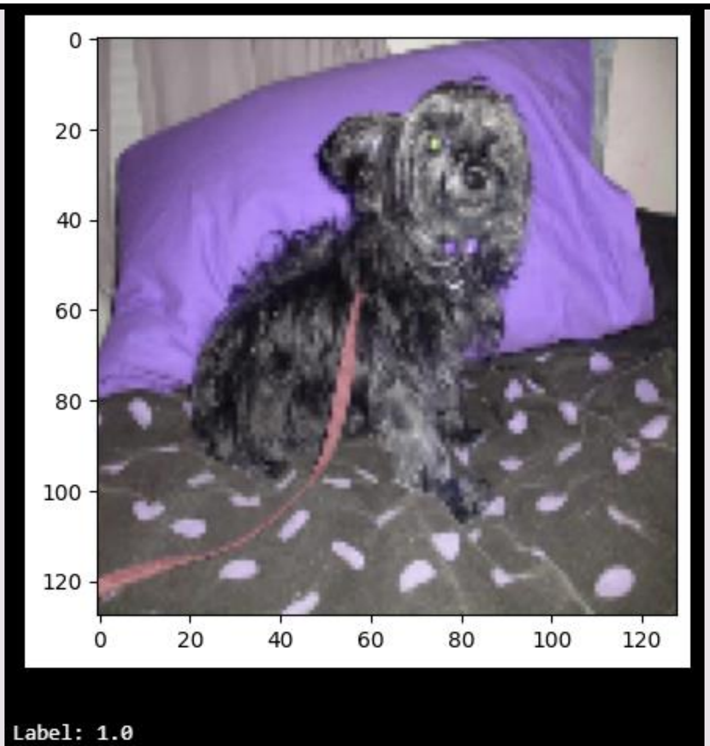
Add Data Labels

- 0 for cats and 1 for dogs

BEFORE:



AFTER:



MODEL

- Used pre-built architecture from Abdullah Al Asif on Kaggle
- The neural network roughly follows an AlexNet architecture
  - A type of CNN architecture that in 2012 won the LSVRC (Large Scale Visual Recognition Challenge) by a large margin
- Roughly 84% accuracy at 100 epochs

Layers	Explanation
Convolutional	Extract features such as edges, texture, and patterns
Max Pooling	Reduce dimensions by selecting max value from feature map regions
Batch Normalization	Normalize input from previous layer through recentering or rescaling (makes NN more stable and faster)
Flatten	Flatten input to 1D vector in order to put through Dense layers (convolutional to dense)
Dense	A layer of neurons that connects to other layers of neurons (Activation Function: ReLU aka Rectified Linear Unit)
Dropout	Randomly sets inputs to zero- aka it randomly drops neurons (helps prevent overfitting)

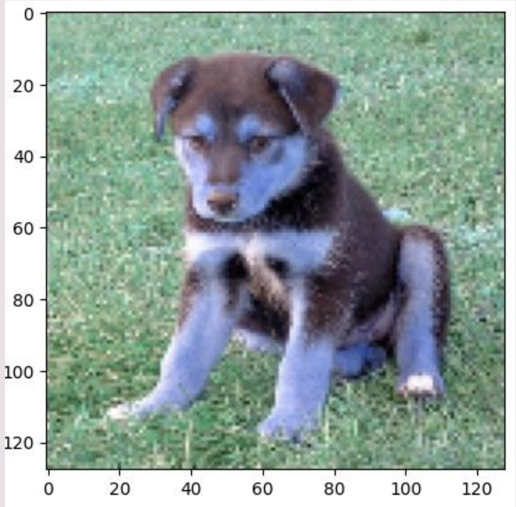
MODEL

Totals:

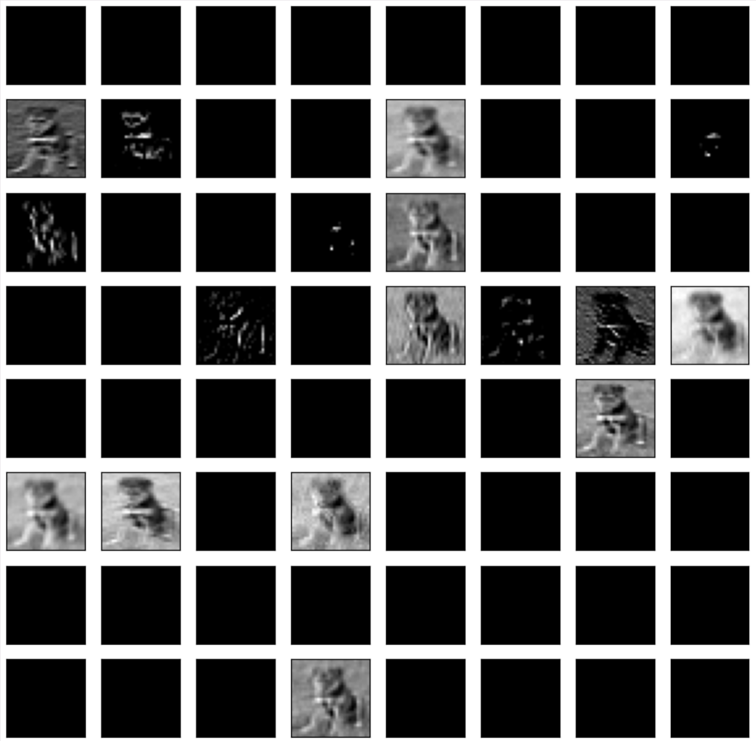
- Convolutional: 5 layers
- Batch Normalization: 3 layers
  - Used after Convolutional
- Max Pooling: 3 layers
  - Used after Convolutional
- Flatten: 1 layer
- Dense: 3 layers
- Dropout: 2 layers

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 96)	34,944
batch_normalization (BatchNormalization)	(None, 30, 30, 96)	384
max_pooling2d (MaxPooling2D)	(None, 14, 14, 96)	0
conv2d_1 (Conv2D)	(None, 14, 14, 256)	614,656
batch_normalization_1 (BatchNormalization)	(None, 14, 14, 256)	1,024
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_2 (Conv2D)	(None, 6, 6, 384)	885,120
conv2d_3 (Conv2D)	(None, 6, 6, 384)	1,327,488
conv2d_4 (Conv2D)	(None, 6, 6, 256)	884,992
batch_normalization_2 (BatchNormalization)	(None, 6, 6, 256)	1,024
max_pooling2d_2 (MaxPooling2D)	(None, 2, 2, 256)	0
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 4096)	4,198,400
dropout (Dropout)	(None, 4096)	0
dense_1 (Dense)	(None, 4096)	16,781,312
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 2)	8,194

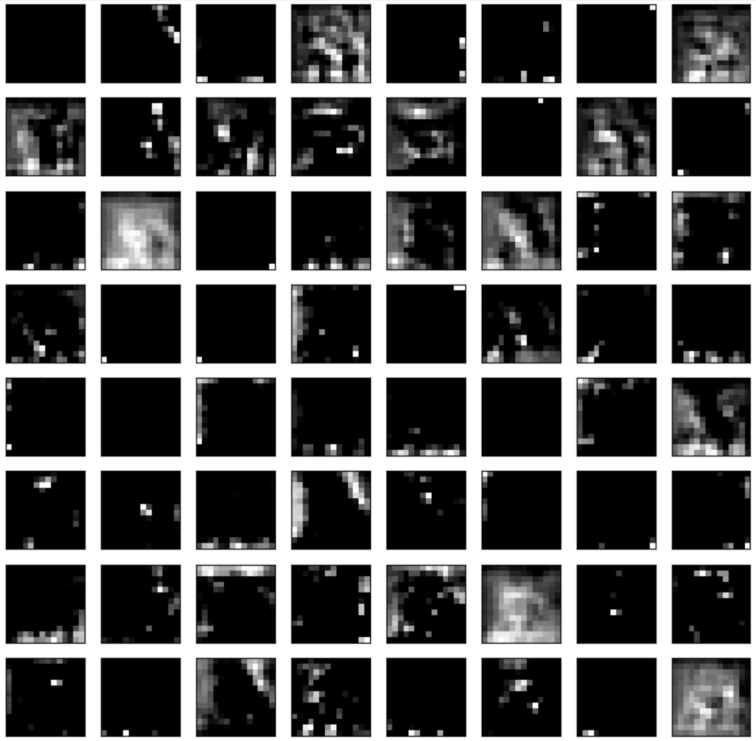
FEATURE MAPS



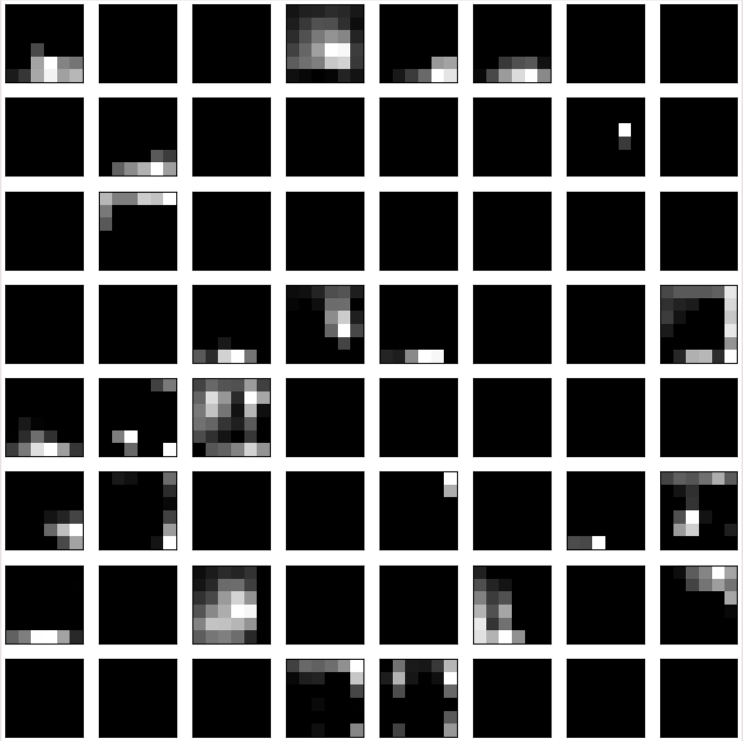
1st Convolutional Layer



2nd Convolutional Layer

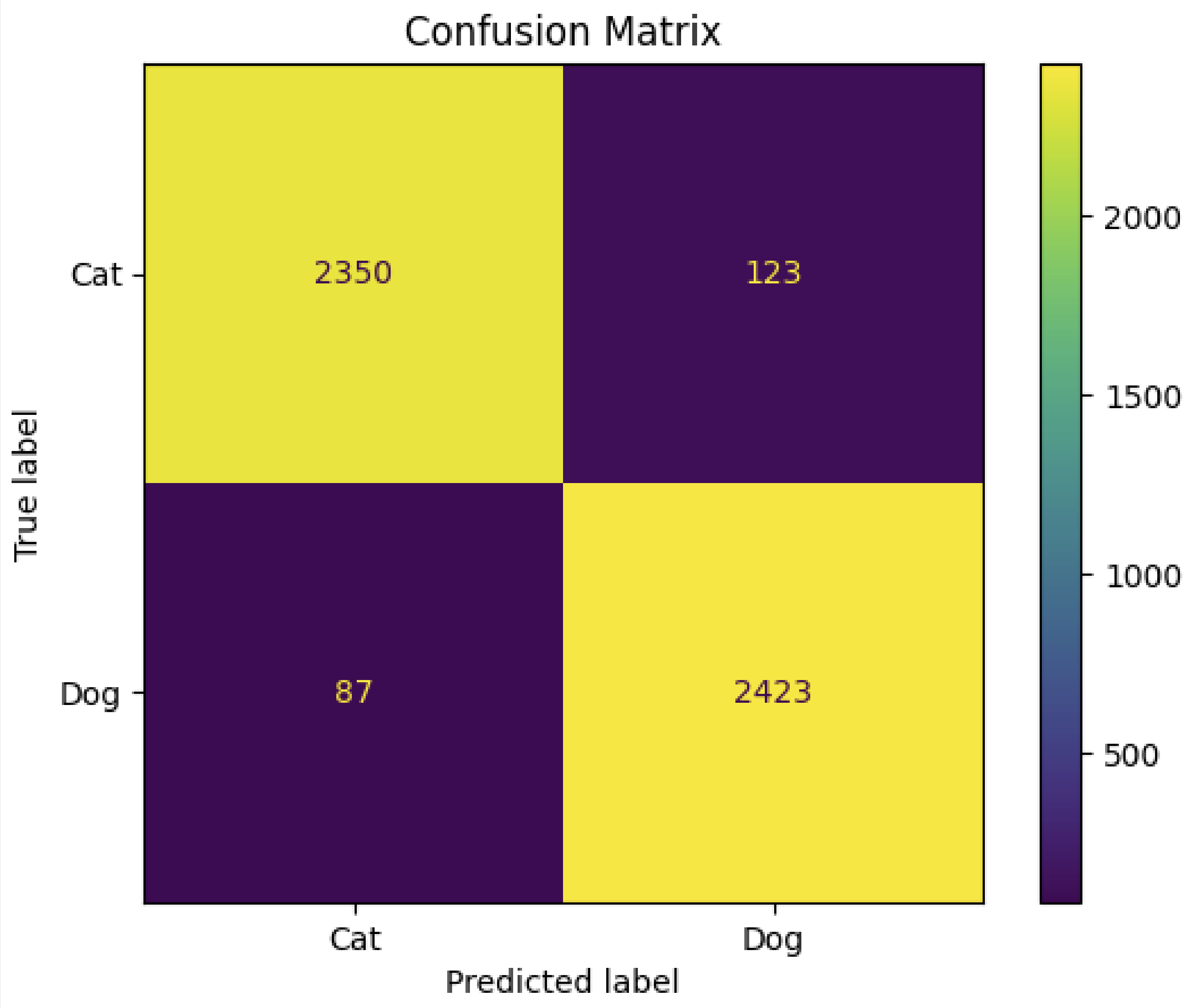


5th Convolutional Layer



MODEL RESULTS

- Accuracy: 84.06%
- Loss: 1.5003
- Epochs: 100
  - Note: the original model had 1,000 epochs and achieved a higher accuracy
  - On desktop it took roughly ~2minutes per epoch or 3.33 hours per train. Therefore, 1,000 was not feasible as it would take us roughly 1.3 days to run :(



WEB APP, USING FLASK

User uploads image

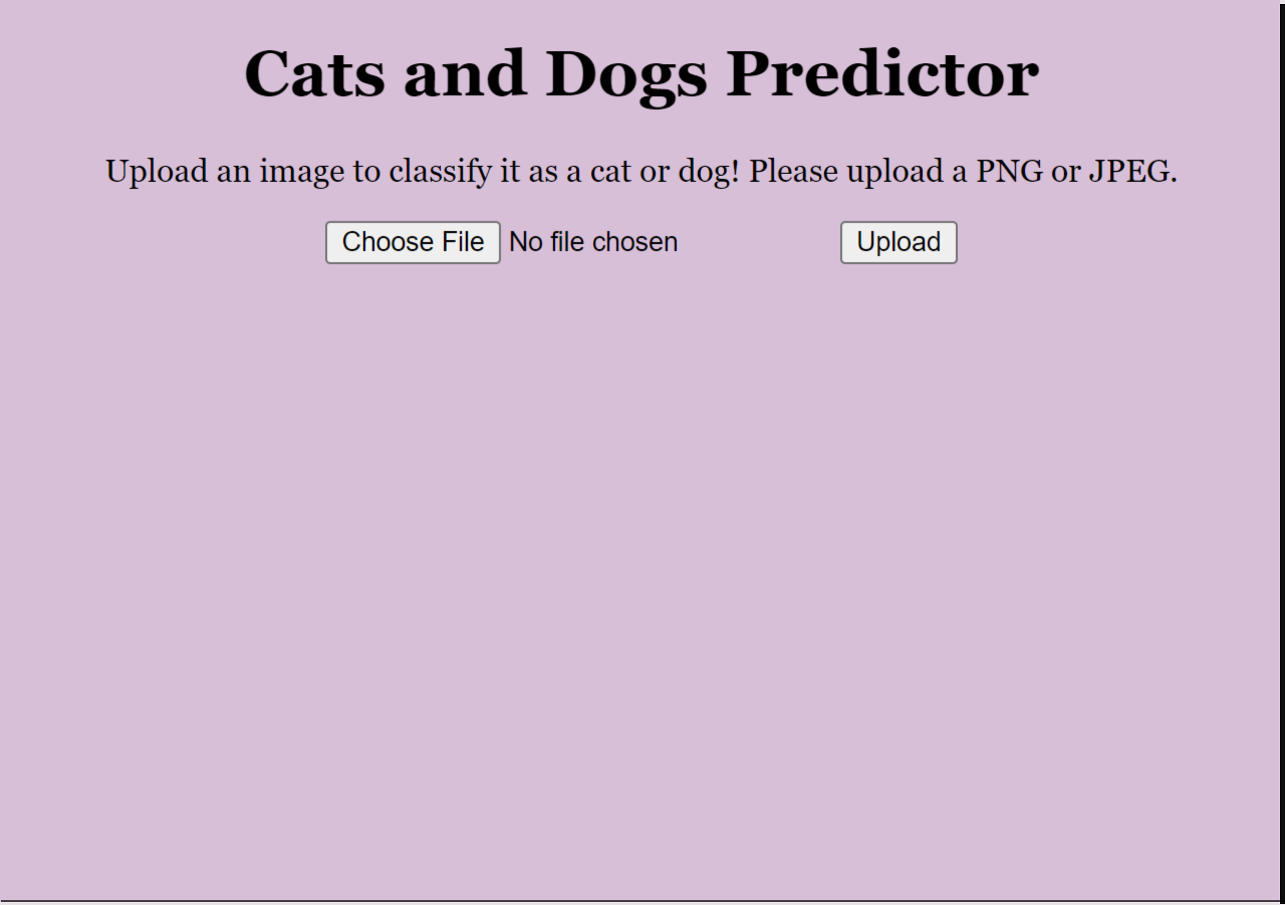
Image undergoes preprocessing

Image is tested using model

Image is classified

Result, photo, and probability are output to user

```python
@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files['file']
        if file:
            # Load the image and convert to RGB
            img = Image.open(file.stream).convert('RGB')
            # Resize the image to match the model's expected input
            img = img.resize((128, 128))
            img_array = np.array(img)
            img_array = img_array / 255.0  # Normalize the pixel values
            img_array = img_array.reshape((1, 128, 128, 3))  # Reshape for model

            prediction = model.predict(img_array)
            img_array = tf.squeeze(img_array)
            prediction = tf.squeeze(prediction)

            # Get the prediction probabilities for both classes
            prediction_probs = tf.squeeze(prediction)

            # Determine the predicted class index (0 or 1)
            predicted_class_index = tf.argmax(prediction_probs).numpy()

            # Get the corresponding probability for the predicted class
            predicted_probability = float(prediction_probs[predicted_class_index])

            # Determine title based on prediction
            if predicted_class_index == 0:
                title = f"It's a Cat!! \n Probability: {predicted_probability}"
            else:
                title = f"It's a Dog!! \n Probability: {predicted_probability}"

            # Display the image with prediction title
            plt.imshow(img)
            plt.title(title)
            plt.axis('off')  # Turn off axis numbers and ticks


            # Save plot to a bytes buffer
            buf = io.BytesIO()
            plt.savefig(buf, format='png')
            buf.seek(0)
            return send_file(buf, mimetype='image/png')
```


Cats and Dogs Predictor
Upload an image to classify it as a cat or dog! Please upload a PNG or JPEG.
Choose File
No file chosen
Upload


It's a Cat!!
Probability: 1.0

WEB APP

DEMONSTRATION

127.0.0.1:5000

Cats and Dogs Predictor

Upload an image to classify it as a cat or dog! Please upload a PNG or JPEG.

Choose File No file chosen Upload

FUTURE WORKS

- Try different kinds of models and neural networks to compare the results
- Continue to improve upon and adjust the current model
- Add in more data as this is only a SMALL subset of the full dataset
- Explore more image preprocessing techniques to see if it works better with the model

WORKS CITED

1. ASIF , A. A. (2023). Exploring CNN models: Cats and dogs classification. https://www.kaggle.com/code/asif00/exploring-cnn-models-cats-and-dogs-classification
    a. *An example notebook of how to make an image classifier for cats and dogs found on kaggle. This notebook also discusses the use of AlexNet models for image classification and is what we are basing the preliminary model off of.*
2. Acsany, Philipp. (2023, December 13). Build a Scalable Flask Web Project From Scratch. Real Python. https://realpython.com/flask-project/
    a. *Information on using Flask for web app development.*
3. Brownlee, J. (2021, December 7). How to classify photos of dogs and cats (with 97% accuracy). MachineLearningMastery.com. https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/
    a. *An example using a similar cat and dog dataset of photos to build neural networks for classification using Python. It will be beneficial to reference in terms of how to start the modeling process and look for different ways we can make the project our own.*
4. Callens, A. (n.d.). Shiny_Classifier: Shiny application to manually classify images by pushing buttons. GitHub. https://github.com/AurelienCallens/Shiny_Classifier
    a. *Gives an example of how to create a Shiny app for a classification task, will be useful in determining the UI we are creating and how to implement the server versus UI.*
5. Deploying a shiny app with a tensorflow model. TensorFlow for R. (n.d.). https://tensorflow.rstudio.com/guides/deploy/shiny.html
    a. *This is an example of implementing Tensorflow into a Shiny App. It can be useful because it uses a Keras API as the model. It also provides another example of creating a Shiny App and other advanced models that can be applied into the app.*
6. Lendio, L. (2023, September 6). Cats vs dogs image classification model. Kaggle. https://www.kaggle.com/code/orensa/cats-vs-dogs-image-classification-model
    a. *An image classification model that shows the different steps when creating and implementing the model. This was an entry in the Kaggle competition that is where we are getting our data set.*
7. Padhiar, K. (2020, May 1). Cat vs dog dataset. Kaggle. https://www.kaggle.com/datasets/karakaggle/kaggle-cat-vs-dog-dataset
    a. *This is the dataset that contains the cat and dog photos that we will using for model training.*

THANK YOU!

QUESTIONS?