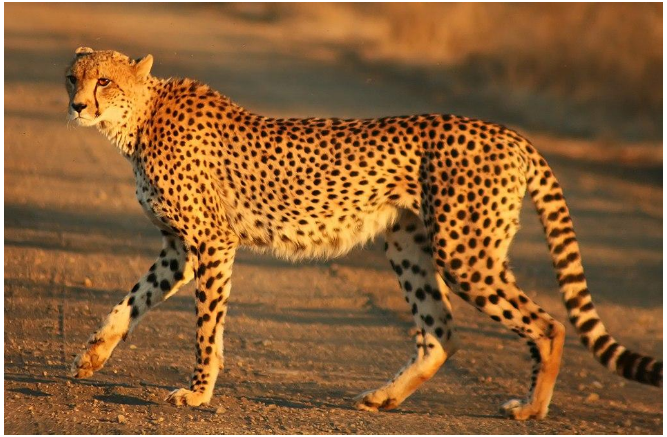
Capstone Project #2

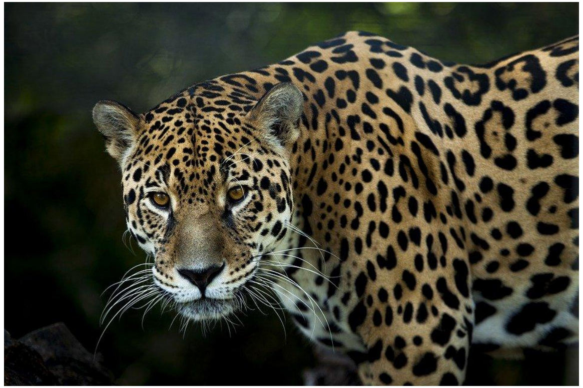
Classifying Images of Spotted Cat Breeds

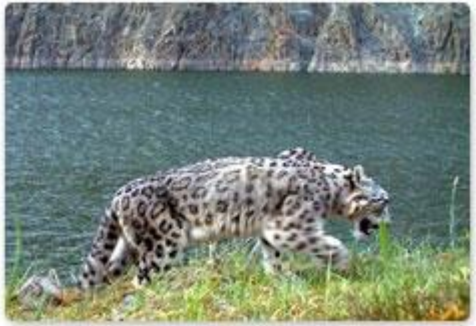
Sangyeol Baek

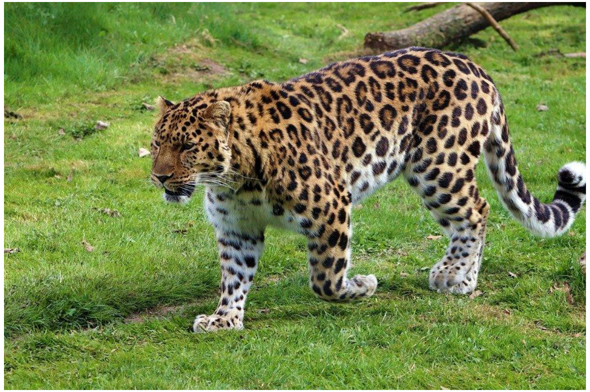
Problem



Can you identify which breed is which for each image?



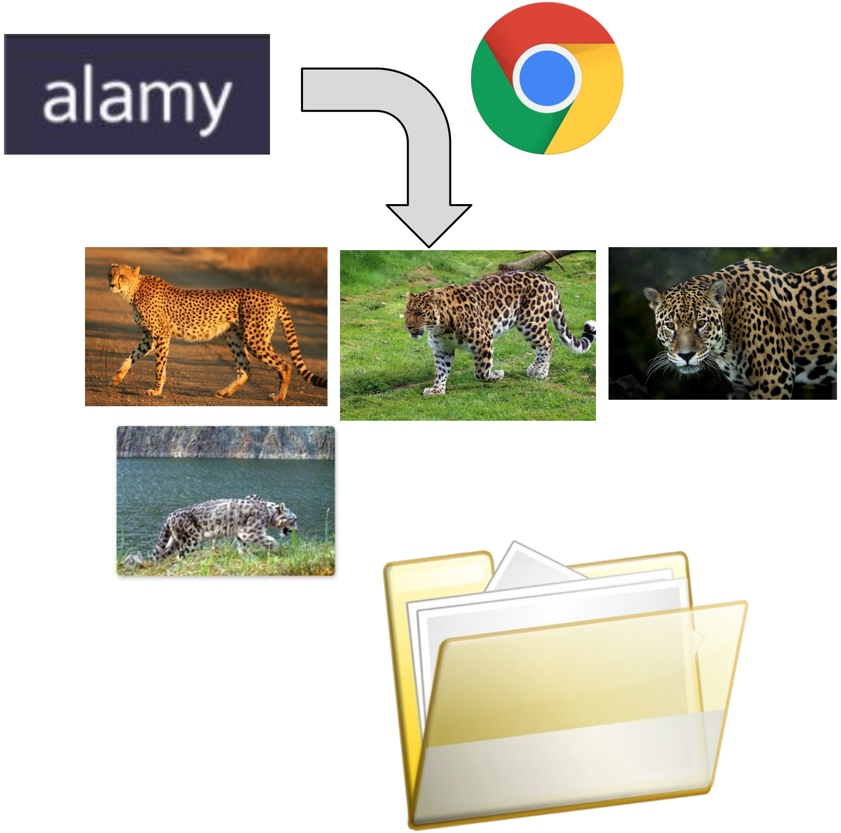




Getting the Data

As with all image classification projects, they would obtain images from image databases such as from Google Images, Bing Images, etc.

- Images obtained from: www.alamy.com
- Tool used?
    - A Google Chrome extension that scraps all images from a webpage: https://chrome.google.com/webstore/detail/download-all-images/ifipmflagepipjokmbdecpmjbibjnakm



Getting the Data (continued)

- The resulting dataset consists of **4 folders representing one of the 4 spotted cat breeds**, each with 1000 images:
  - **Cheetah**
  - **Jaguar**
  - **Leopard**
  - **Snow leopard**

Building the Model

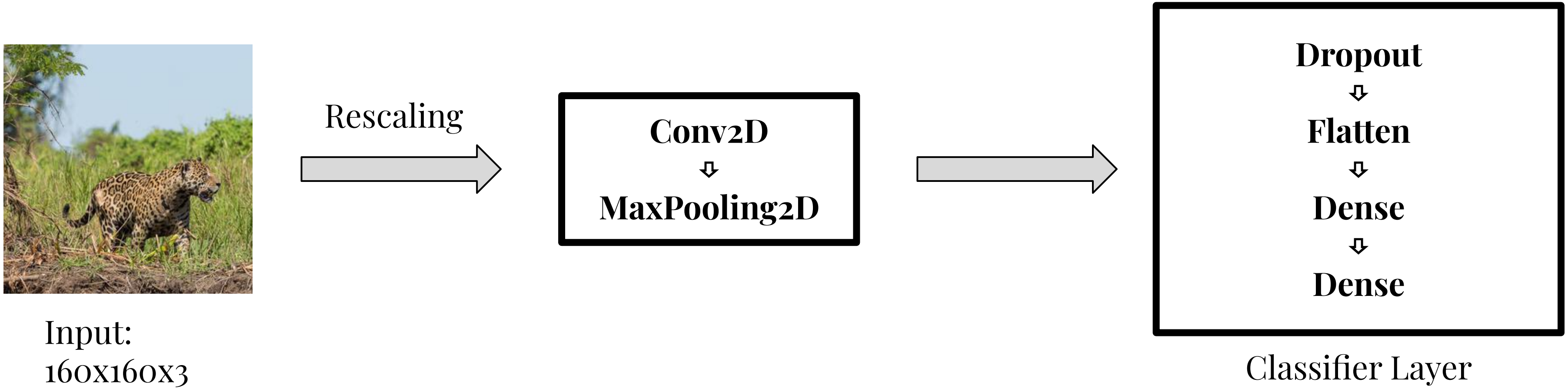
Which framework to choose?

Building the Model

Which framework to choose?

For this project, I chose Tensorflow since one of its APIs, **Keras**, due to its powerful yet more beginner-friendly for those new to building Deep Learning models.

First Simple Model



Rescaling

Conv2D ⇩ MaxPooling2D

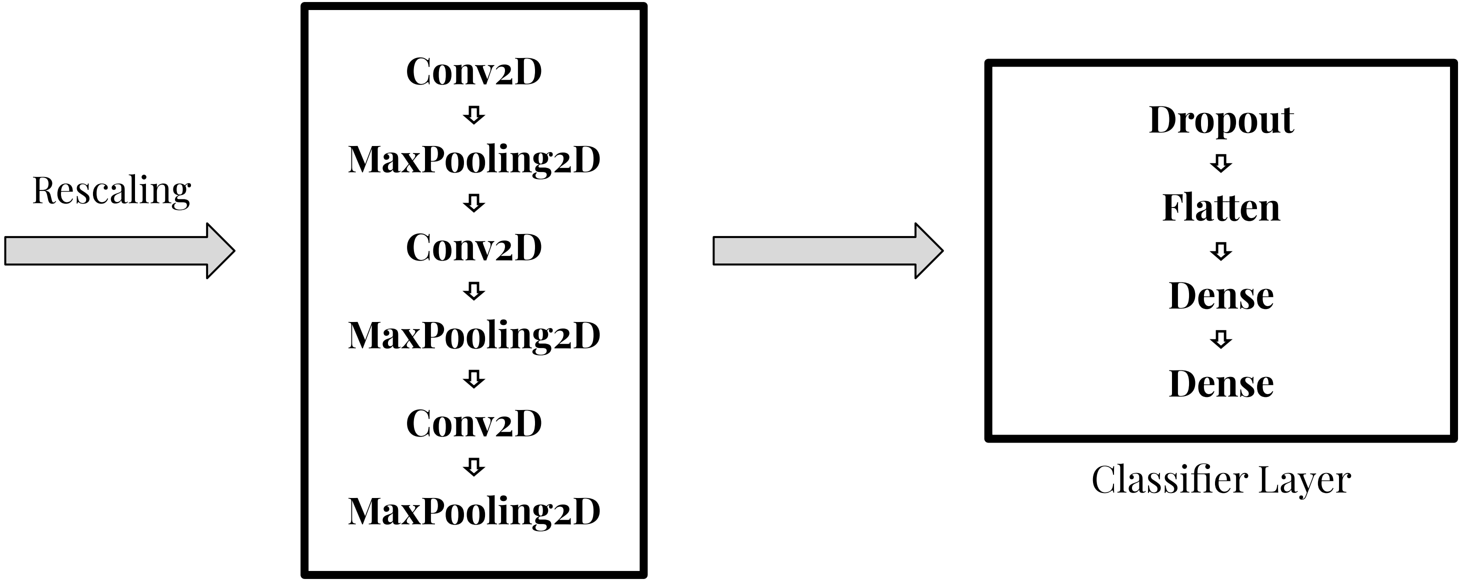
Dropout ⇩ Flatten ⇩ Dense ⇩ Dense

Input:
160x160x3

Classifier Layer

With More Convolutional Layers

Input:
160x160x3



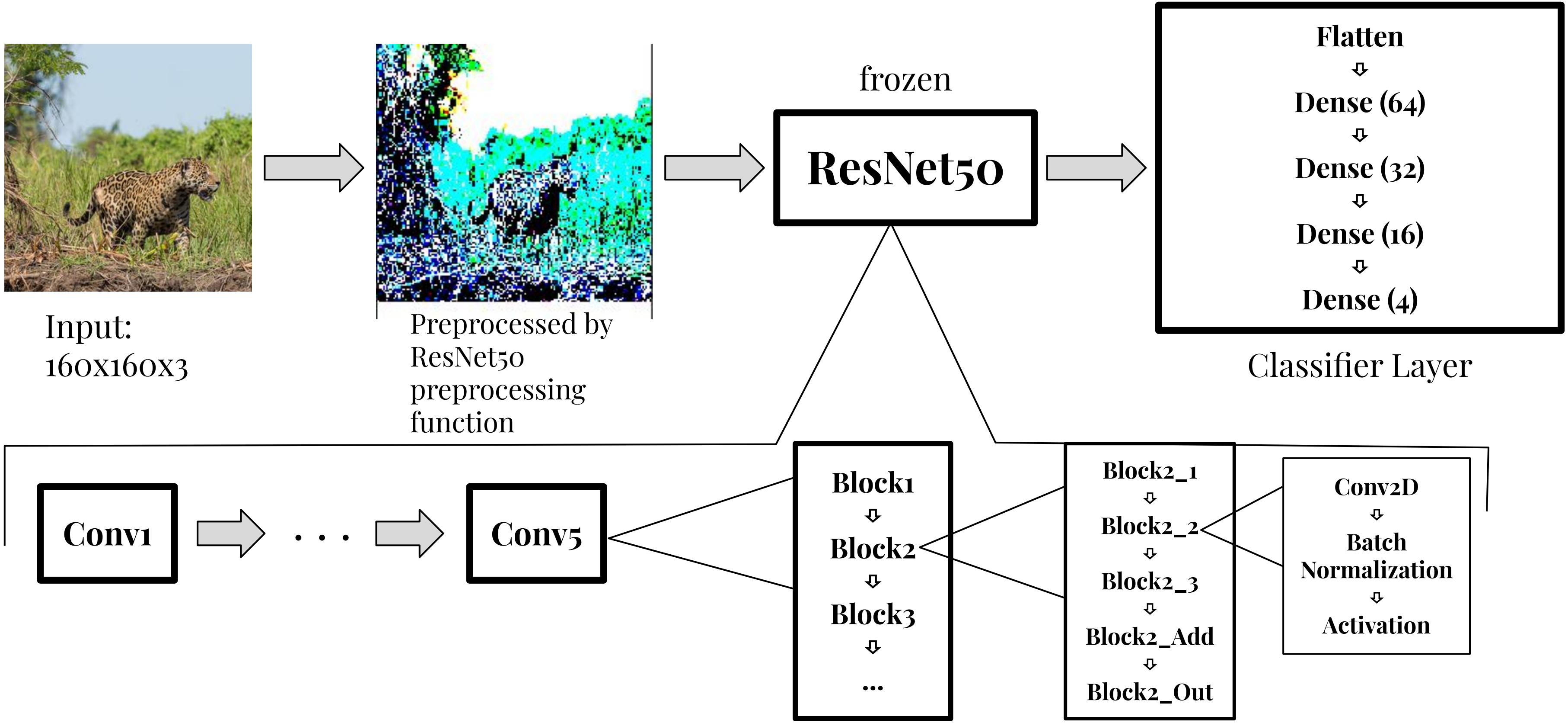
Rescaling

Conv2D
⇩
MaxPooling2D
⇩
Conv2D
⇩
MaxPooling2D
⇩
Conv2D
⇩
MaxPooling2D

Dropout
⇩
Flatten
⇩
Dense
⇩
Dense

Classifier Layer

Applying Transfer Learning with ResNet50 Architecture



Input:
160x160x3

Preprocessed by
ResNet50
preprocessing
function

frozen

ResNet50

Flatten
⇩
Dense (64)
⇩
Dense (32)
⇩
Dense (16)
⇩
Dense (4)

Classifier Layer

Conv1 ⇨ . . . ⇨ Conv5

Block1
⇩
Block2
⇩
Block3
⇩
...

Block2_1
⇩
Block2_2
⇩
Block2_3
⇩
Block2_Add
⇩
Block2_Out

Conv2D
⇩
Batch
Normalization
⇩
Activation

Performance Comparison

Compare the models' performances (in this order):

1. Simple Model
2. Model with additional Convolutional Layers
3. Model with frozen ResNet50 architecture

Training Acc	Training Loss	Validation Acc	Validation Loss
24.3%	8.72	24.5%	8.82
24.7%	5.01	25.5%	4.98
96.0%	0.142	84.6%	0.492

Making the Model Better?

- The model with the ResNet50 architecture already has a decent performance, but are there ways of fine-tuning it to boost its performance?

- Some possible changes I have tried:
    - Train model with rescaling (**rescaler**=**1./255.**) in the ImageDataGenerator
    - Change target size: **160x160** => **200x200**
    - Change learning rate: **0.001** => **0.01**
    - More epochs: **5** => **8**

Performance Comparison

Compare the models' performances (in this order):

1. Original
2. With Rescaler (Rescaler=1./255.)
3. Change Image Size (160x160 => 200x200) => improved accuracy!!
4. Change learning rate (lr=0.001 => lr=0.01), in addition to #3
5. More epochs (5 => 8), in addition to #3

Training Acc	Training Loss	Validation Acc	Validation Loss
96.0%	0.142	84.6%	0.492
43.1%	1.28	42.9%	1.27
97.7%	0.0915	87.0%	0.420
96.9%	0.131	85.4%	0.828
97.7%	0.0871	88.3%	0.544

Interpreting the Model's Predictions with LIME

- Used an API called LIME, which creates visualizations of how the model predicted the way it did
- The following slides will demonstrate the API for one image of each category, then for a couple of images in which the model mispredicted.

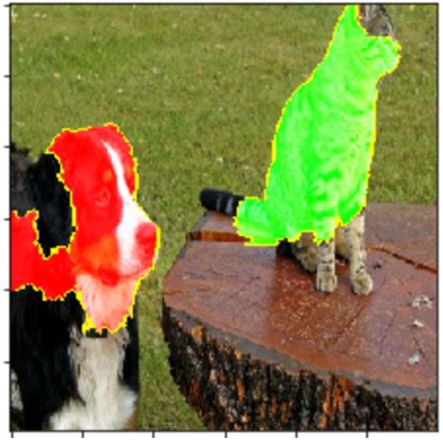


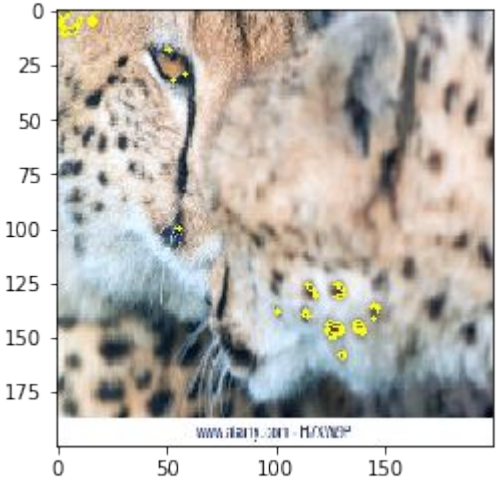
Image source: https://github.com/marcotcr/lime

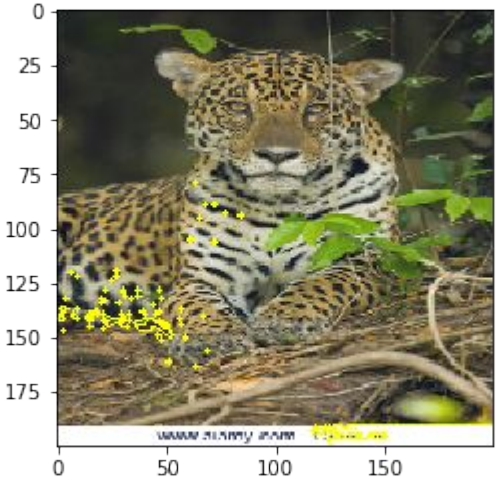
LIME Demo

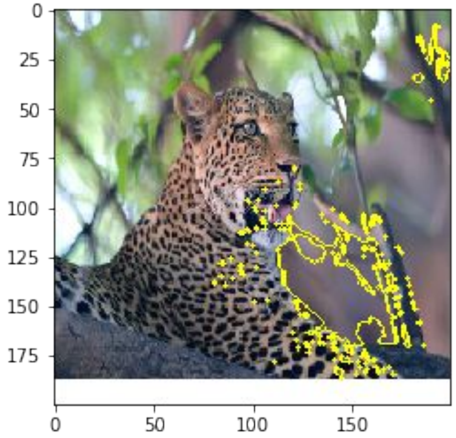
The images on the right were some in which the model predicted correctly with “100% accuracy” or close to it.

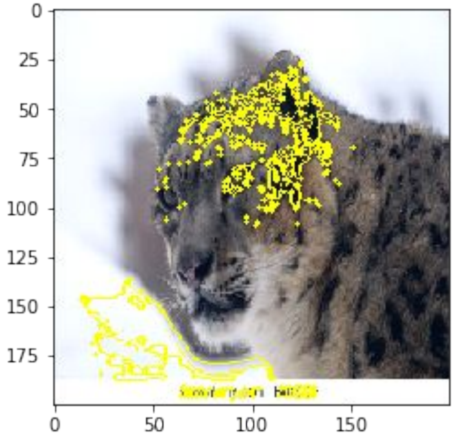
It seems that LIME mainly highlights the creatures’ spots on their skin, which coincidentally are their main distinguishing features.

Most interestingly, the fourth image (snow leopard) also highlighted parts of the white background due to the snow leopards’ skin being white.







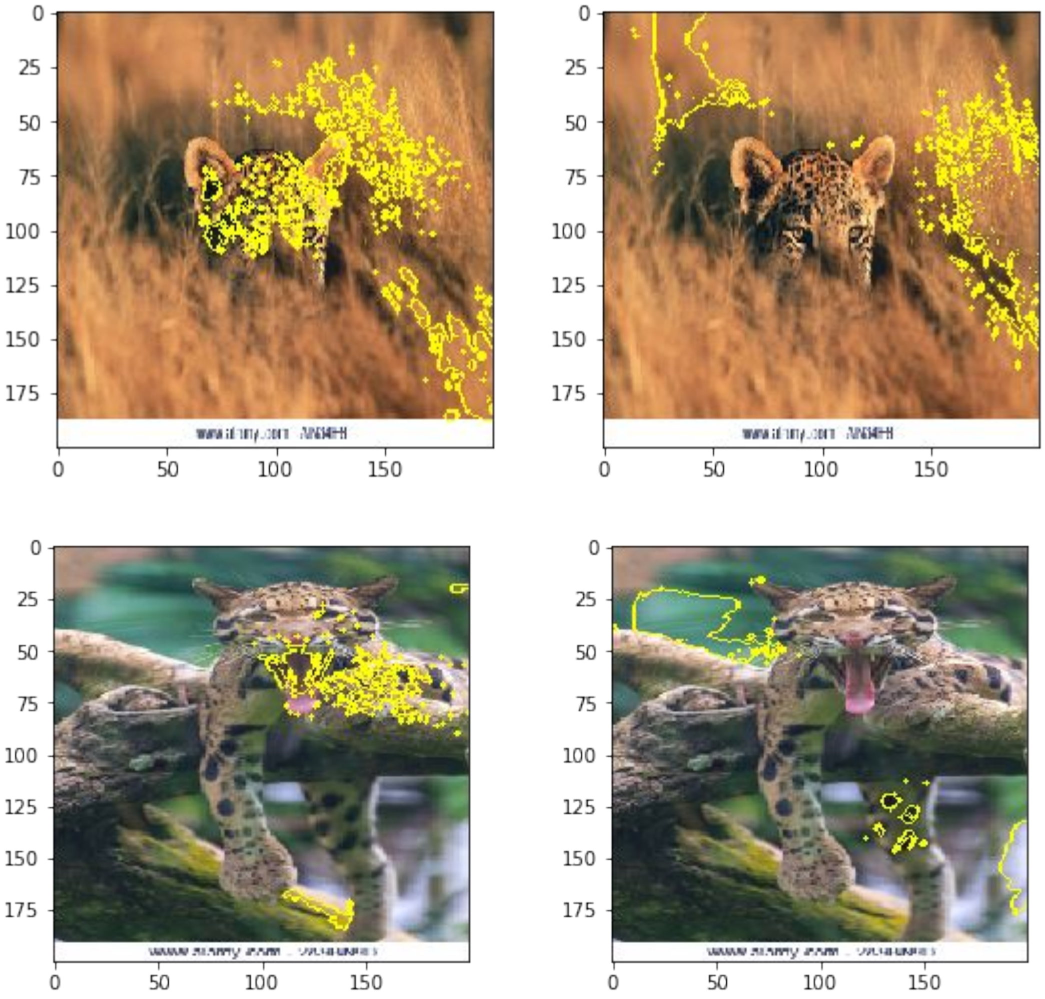


LIME Demo (cont.)

The images on the right were two of the images in which the model predicted incorrectly.

The left images represent what the model predicted, while the right represent the actual category was.

It seems that for these images, they did not show enough of the creatures' body, leading to its mispredictions.



Limitations

- Since this focused on image classification, I found it unnecessary to provide statistical analyses or hypothetical testing
- Building deep learning models in general requires a lot of dedication in addition to time, especially if one were to build a “perfect” model
- There were not many options of using GPU to train the model instead of CPU, and Keras is known to be slower than PyTorch, limiting my ability to experiment

What happens next?

The model I have experimented already achieved a decent performance, with 97% training accuracy and 88% validation accuracy. However, it is far from perfect, so some possible next steps:

- Find a way to detect the creatures' body, especially if they are far in the background in an image
- Use a larger dataset or use a larger target size for the model
  - More images dominantly depicting the animal's body
  - Or, more diverse image set (assuming **first point** is achieved)
- Use a different CNN architecture besides ResNet50 (ResNet101, VGG16, InceptionV3, etc.)