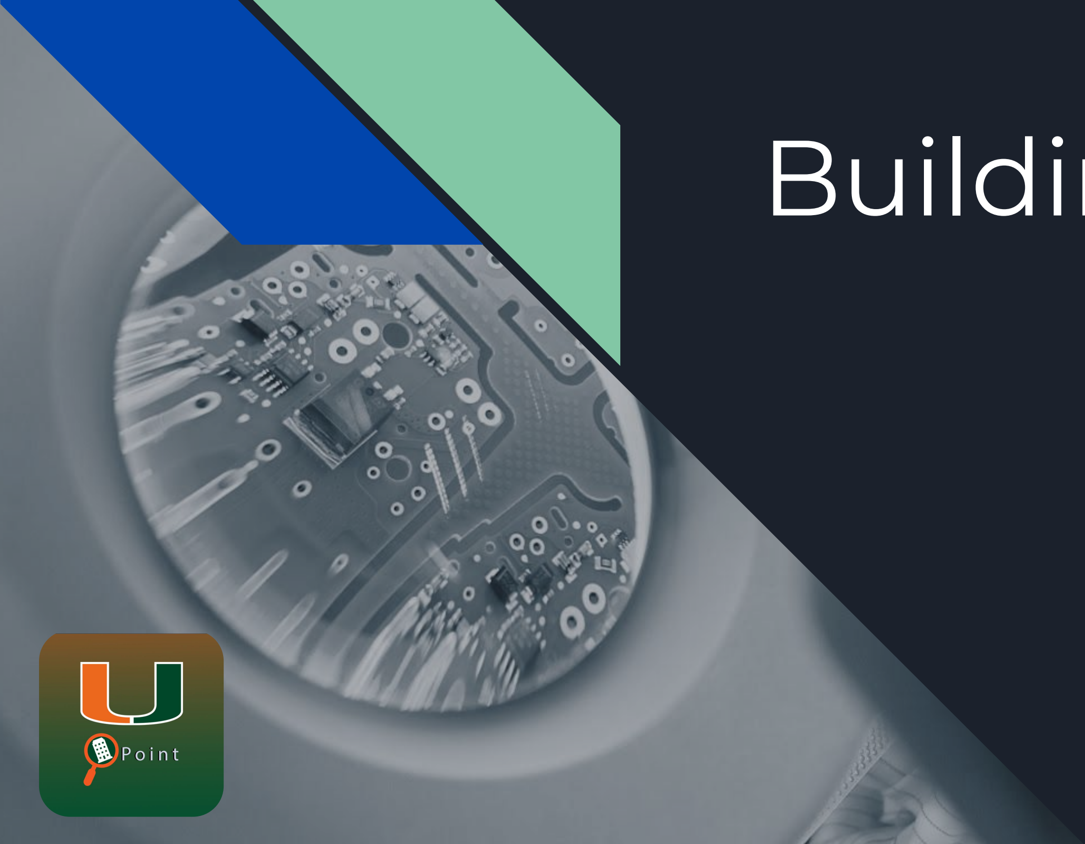


BuildingPoint

David Elimelech

Nicholas Magal

Rafael Marques Braga

The Problem

The Idea

Overview

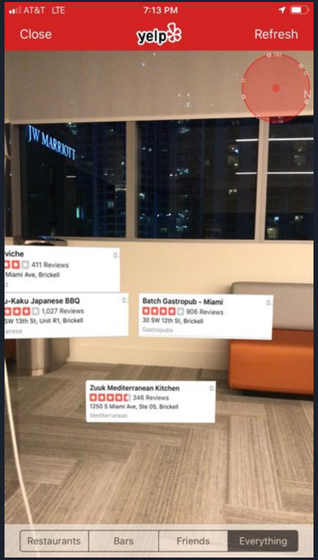
- Construct a reliable, efficient, and robust machine learning model enhanced by geolocation capable of detecting buildings

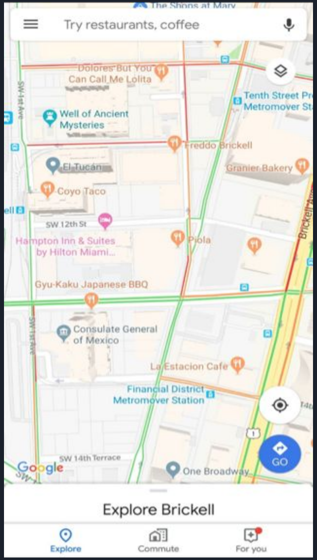
- Integrate the model into a user-friendly mobile application that will allow the user to interact with the model and benefit from its service

- Document our findings, and ensure scalability

Inspiration

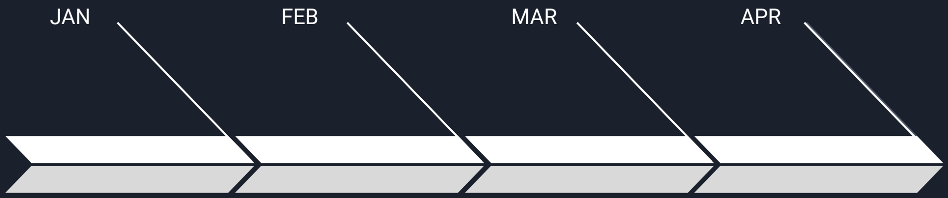
- Other apps like Yelp or Google Maps attempt to accomplish the same thing, but only work with geolocation and orientation sensors, leaving the user to have to figure out his surroundings for himself




Try restaurants, coffee
Well of Ancient Mysteries
El Tucán
Coyo Taco
Hampton Inn & Suites by Hilton Miami...
Gyu-Kaku Japanese BBQ
Consulate General of Mexico
La Estacion Cafe
Financial District Metromover Station
One Broadway
GO
Explore Brickell
Explore
Commute
For you

Demo

Project Timeline



Planning

Planned out which technologies we were going to use to build our model and our app.

Machine Learning Research

Began looking into CNNs

Experimented with Tensorflow/Keras

Researched transfer learning

Gathered data

Model Implementation

Made our model using Tensorflow/Keras

Utilized a transfer learning approach

App Development

Incorporated the model into an Android application with a user friendly GUI.

What was accomplished?

1. Created a machine learning model with over 90% accuracy
2. Created a database hosting building information
3. Incorporated user geolocation into classification
4. Synthesized the model into a user-friendly mobile application

Our Stack

Tensorflow, along with its Keras API (both built on Python) was used to train our ML model

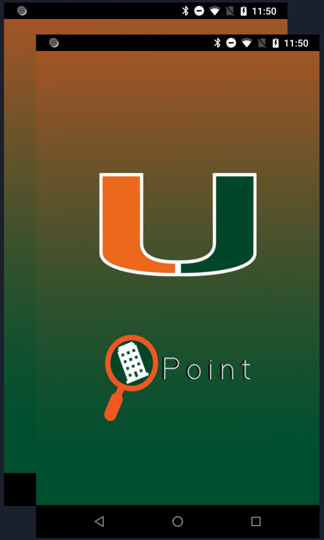
We then used Firebase's …

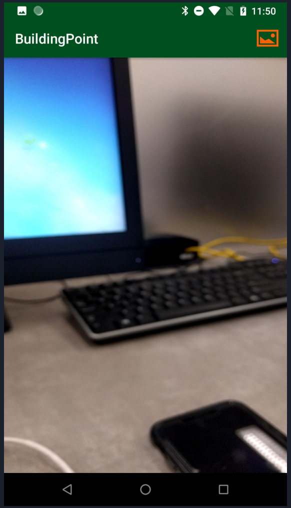
- ML Kit to host our model
- Firestore database to hold building information

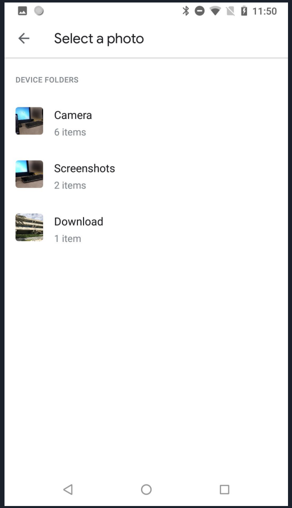
Our ML model was finally integrated into an Android application, working seamlessly together with Firebase's resources

Android App Overview

- Our app has three main activities
  - Splash
  - MainActivity
  - Result Page
- Theme
  - Orange and Green. Go Canes!






11:50

Select a photo

DEVICE FOLDERS

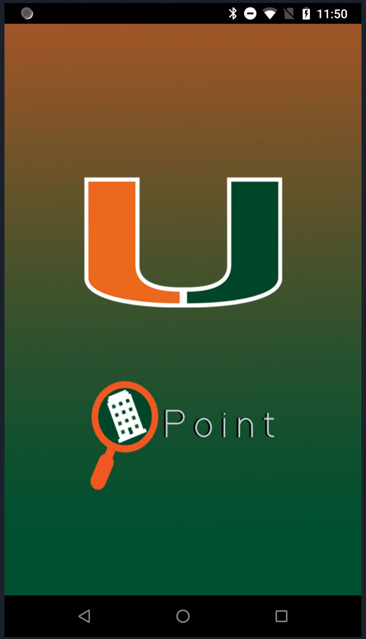
Camera
6 items

Screenshots
2 items

Download
1 item

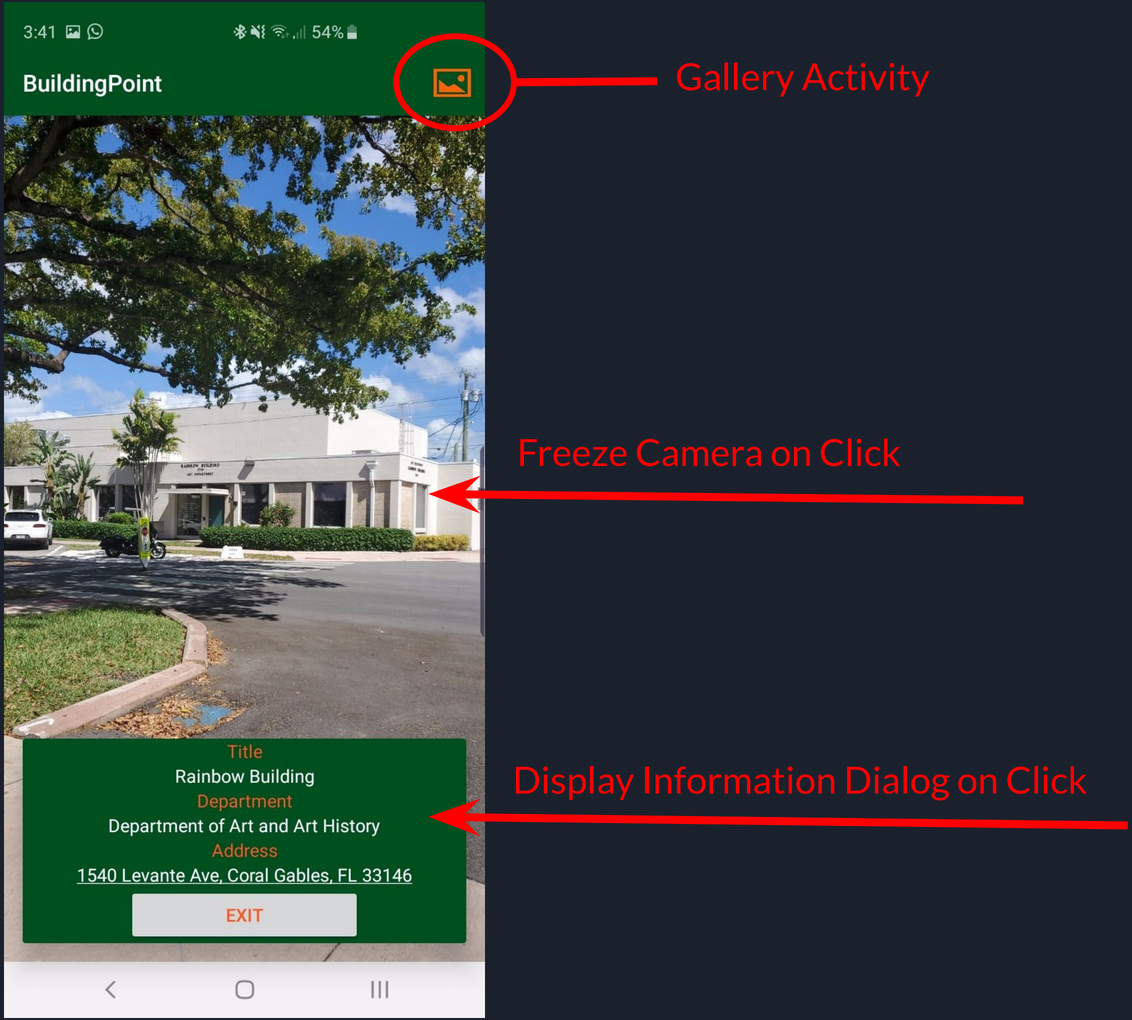
Splash

- Shown on app launch
- BuildingPoint logo and UM logo
- Displayed for 3 seconds, automatically takes user to camera page



MainActivity

- The homepage of app
- Camera preview
- Information dialog



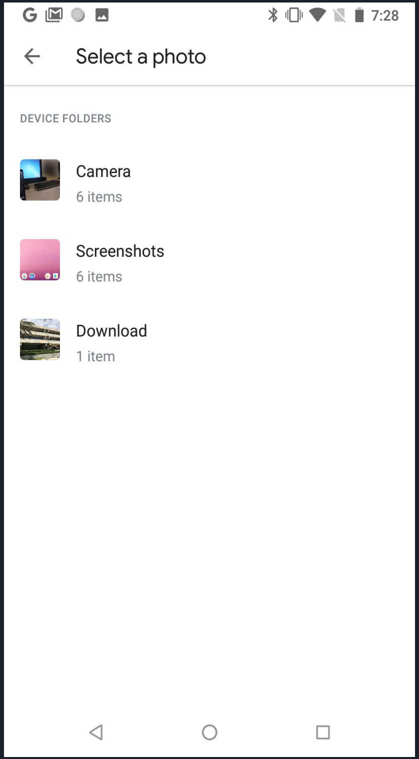
3:41 54%

BuildingPoint

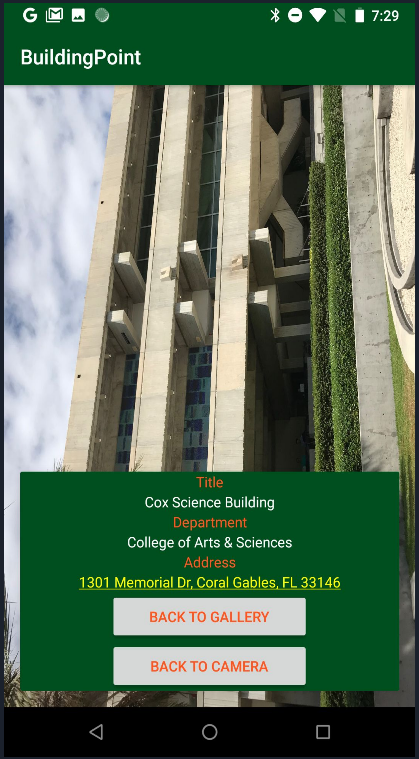
Title
Rainbow Building
Department
Department of Art and Art History
Address
1540 Levante Ave, Coral Gables, FL 33146

ResultPage

- Secondary Page
- Static Preview
- Information dialog



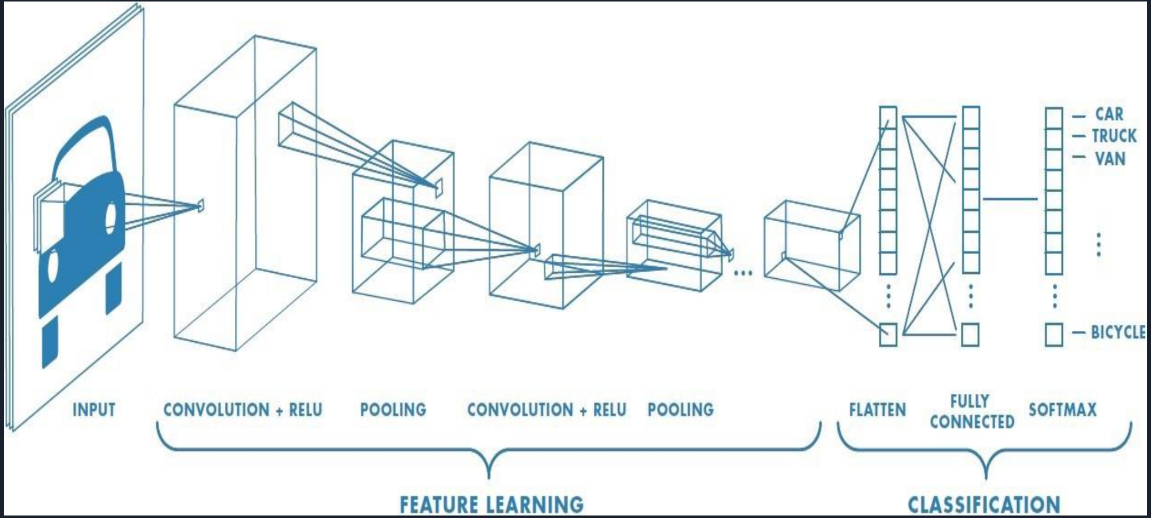
After Choosing a photo from Gallery Classify Photo



Title
Cox Science Building
Department
College of Arts & Sciences
Address
1301 Memorial Dr, Coral Gables, FL 33146

Machine Learning

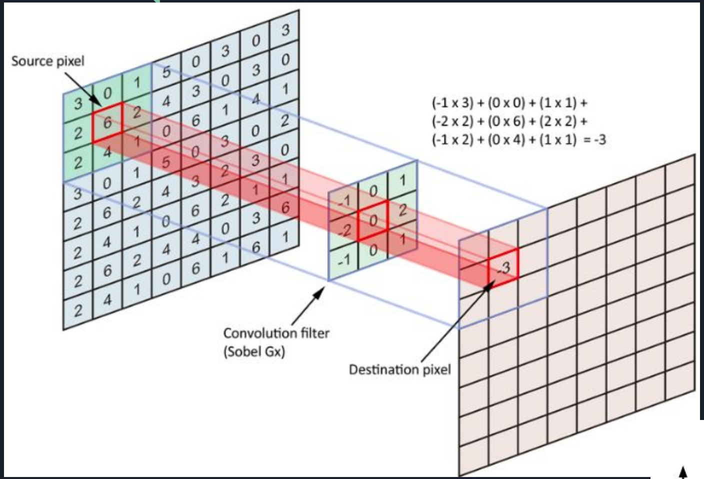
What is a CNN?



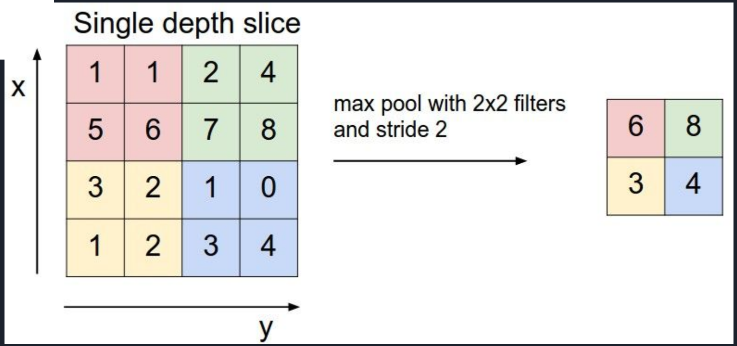
A convolutional neural net, in most cases, is the most ideal paradigm to classify images.

- CNN is composed of two main parts
  - Feature Learning
    - Extract features from the image
    - Deeper you go, the more abstract the features
    - Breaks down the image, reducing image parameters
  - Classification
    - Uses features for classification

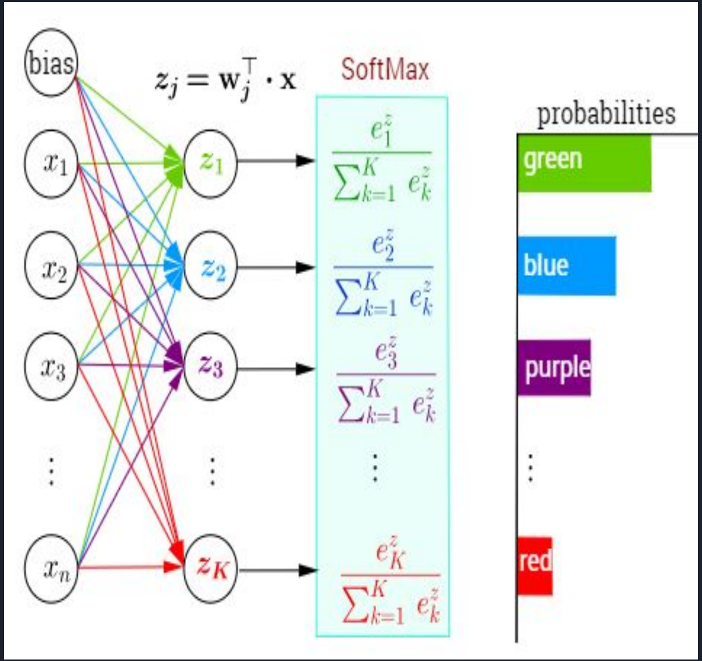
Feature Learning


Source pixel
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3
Convolution filter
(Sobel Gx)
Destination pixel

- Stack of convolutional and pooling layers
- Convolutional layers are responsible for extracting features
- Pooling layers are responsible for downsampling the feature map (generalization)



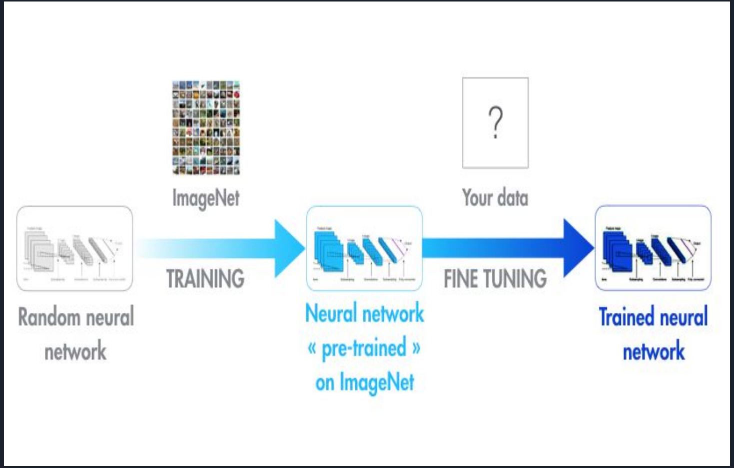
Classification



bias
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
SoftMax
probabilities
x_1
x_2
x_3
x_n
z_1
z_2
z_3
z_K
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
green
blue
purple
red

- Densely connected neural network
- Responsible for training the model to classify images by updating the weights
- Output is a series of probabilities of the input image belonging to each class

Transfer Learning



- Originally, we wanted to train our own model from scratch, but ...
  - Time consuming
  - Computationally intensive
- Transfer Learning
  - How does it work?
    - Don't need to teach the model what it already knows
  - What do we gain?
    - We only need to fine-tune the model, saving us a lot of time

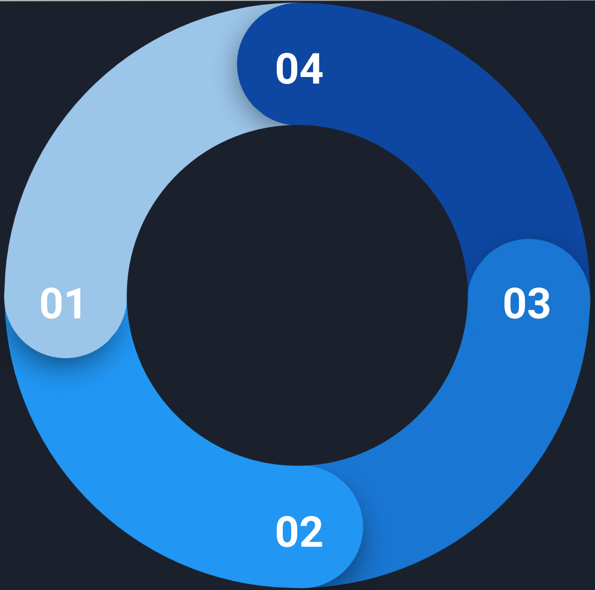
Implementation Cycle

Preprocess Data

Formatting, Normalization, One-Hot Encoding, Split data into Train & Test, Augment data

Create Model

Import MobileNet model, Freeze early convolutional layers



Implement into App

Integrate model into Android app

Train/Test Model

Train our model using training set

Evaluate our model using testing set

Performance and Remarks

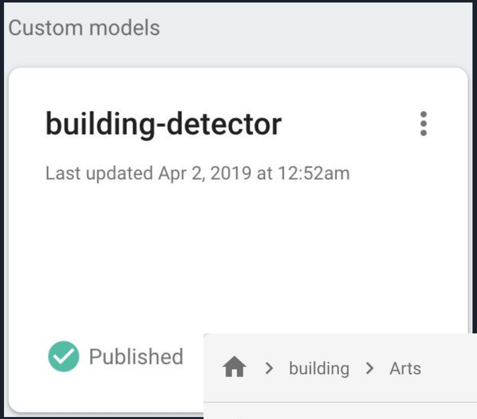
- Over 90% accuracy on testing and training sets in 10 epochs
  - Training completed in under 20 minutes

- Small size of model
  - 13MB, despite millions of parameters
  - Optimal for mobile application

- Result: a lightweight, scalable, high-performing model

Firebase

Setting up Firebase

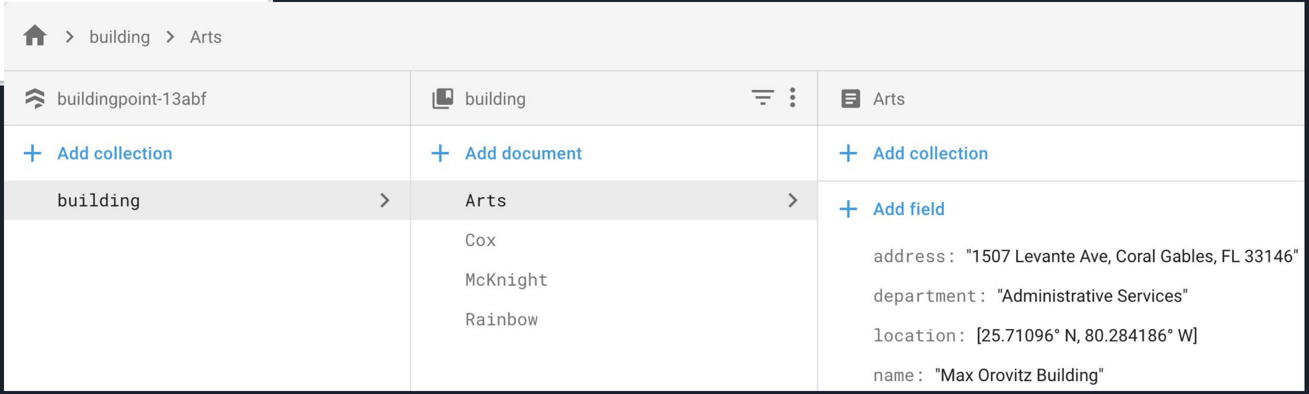

Custom models

building-detector

Last updated Apr 2, 2019 at 12:52am

Published

- Used Firestore database to create a collection of documents referring to each building


building > Arts

buildingpoint-13abf | building | Arts

Add collection | Add document | Add collection

building | Arts | Add field

Cox

McKnight

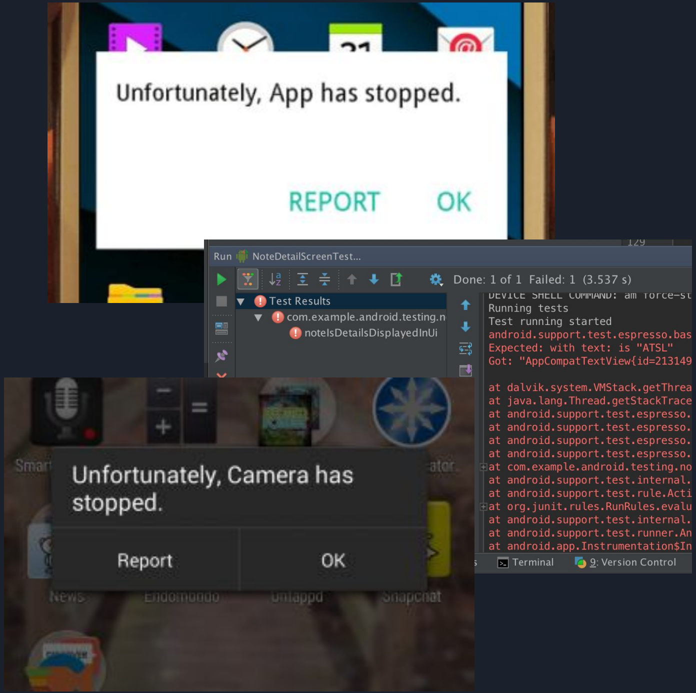
Rainbow

```
address: "1507 Levante Ave, Coral Gables, FL 33146"
department: "Administrative Services"
location: [25.71096° N, 80.284186° W]
name: "Max Orovitz Building"
```

Custom ML Kit

- Saved our ML model as a Tensorflow Lite model, and imported it into Firebase's ML Kit

- Saved Locally


Unfortunately, App has stopped.

REPORT
OK
Run
NoteDetailScreenTest...
Done: 1 of 1 Failed: 1 (3.537 s)
Test Results
com.example.android.testing.n
noteIsDetailsDisplayedInUi
Terminal
9: Version Control

```
DEVICE SHELL COMMAND: am force-st
Running tests
Test running started
android.support.test.espresso.bas
Expected: with text: is "ATSL"
Got: "AppCompatTextView{id=213149

at dalvik.system.VMStack.getThrea
at java.lang.Thread.getStackTrace
at android.support.test.espresso.
at android.support.test.espresso.
at android.support.test.espresso.
at android.support.test.espresso.
at com.example.android.testing.no
at android.support.test.internal.
at android.support.test.rule.Acti
at org.junit.rules.RunRules.evalu
at android.support.test.internal.
at android.support.test.runner.An
at android.app.Instrumentation$In
```

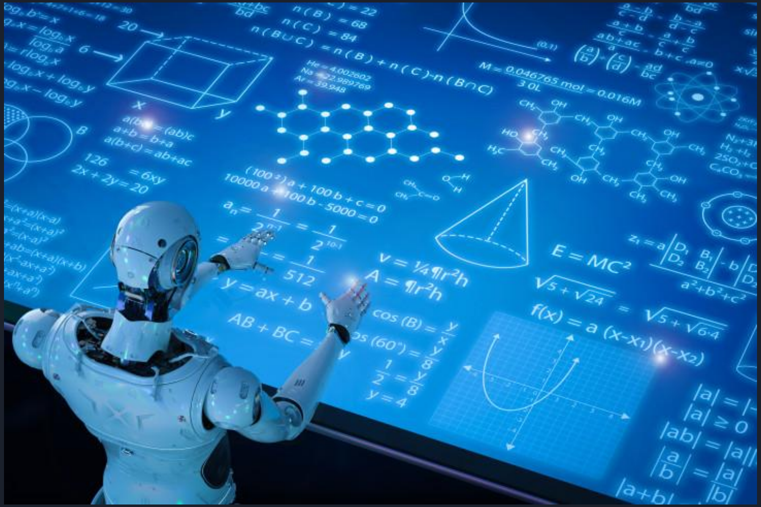
Unfortunately, Camera has stopped.

Technical Challenges

Which technology?

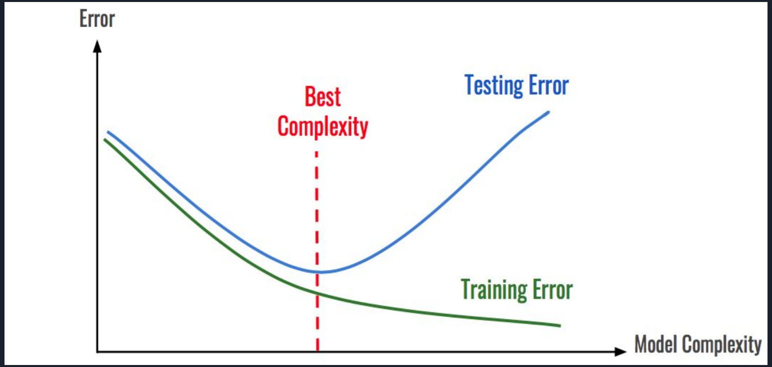
- Originally, planned on using Google's Flutter
  - Built on Dart
  - Can create apps native to both iOS and Android
- Lack of Documentation
  - Custom ML model
- Android

How to build our CNN?



- Initially wanted to build a CNN from scratch using TensorFlow
    - This was technically challenging and required a great amount of resources
    - Time Constraint
- Went with transfer learning using Keras, a time and resource efficient option.
- What architecture to use?
- Thought about integrating the geolocation into the classification part of the CNN itself as extra inputs

ML Model Overfitting



- While training our model, we were experiencing severe overfitting

- To combat overfitting
    - Added images
    - Cut epochs
    - Augmented data
    - Reduced complexity of our model (removing preprocessing function)
    - Increased data shuffling

Gathering and Cleaning Data

- Large volumes of the buildings we wanted to train our model did not exist.
- We took at least 130 pictures from different angles of each building.
- In order to get more data, we applied data augmentation.
- Deleted photos with similar angles.

Sluggish Application

- Slow and unresponsive app initially

- Solutions
  - Download local model of machine Learning
    - Was able to do this because the model was lightweight!

  - Reduce database queries
    - Query snapshot instead of individual buildings

What to do with geolocation?

- Originally, we used the geolocation to increase the probability of the nearest building coming out of the ML model
    - Problem
        - Geolocation does not really tell you what building you are looking at, but rather which building you cannot be looking at
- Instead used geolocation as a validation step
    - If user location is within 100 meters

Confidence/Background Noise

- How do we gauge how confident we are in our model?

- Establish a confidence threshold
  - If probability is < 70%, alert as background noise
  - Rather have a more strict app that tells the user it couldn't classify the image then classify it incorrectly

- Definitely a lot of work that can be done here to fine-tune this process, especially as more buildings are added to database.

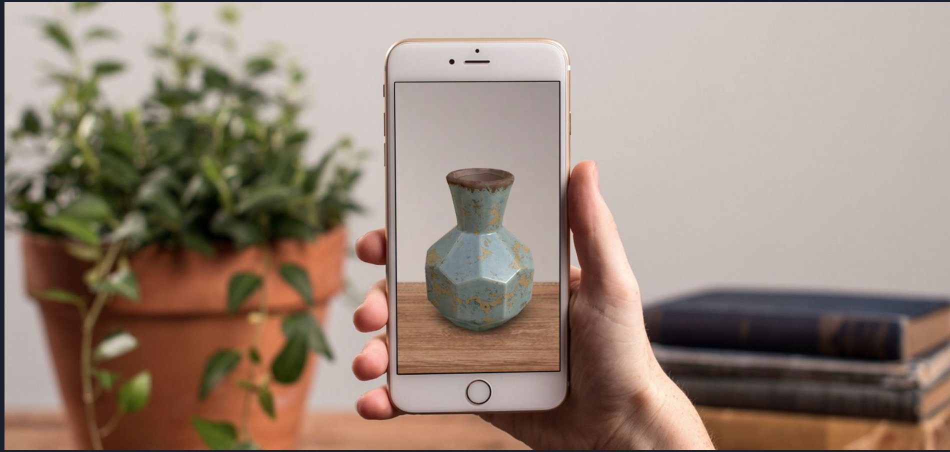
Bugs

- As expected, we faced numerous bugs.
  - Examples Include:
    - Conflicting packages in Python
    - Camera crashing
    - Switching activities crashed app due to ML model still classifying

Looking Forward

AR

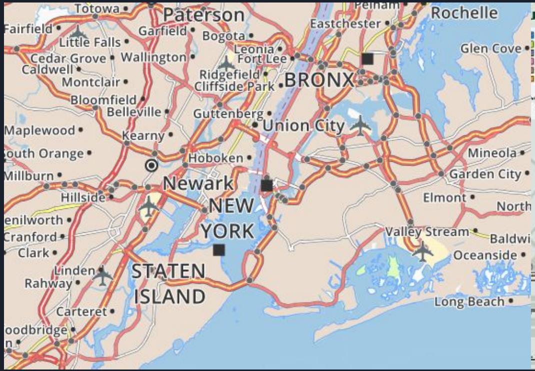
- Keep camera actively previewing
- Superimpose AR billboard
- Track building



ARCore

Add more Buildings- Campus/Off Campus

- Completely incorporate campus into app
- Approach UM app developers
- Add buildings using a Bing images or Google images API so it can be implementable in a city
- Have a program where users can take photos of buildings for model



Better Confidence algorithm

Other algorithms to improve the confidence levels could be used including

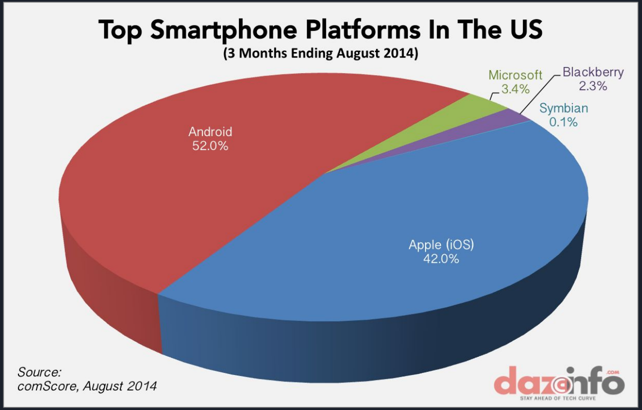
- Increasing the probability of the closest buildings by a certain increment
- Checking the 3 most probable buildings from the building recognition algorithm and comparing their distance to the user’s current location
- Find a better algorithm that takes into account the user’s direction, weather and time of the day.

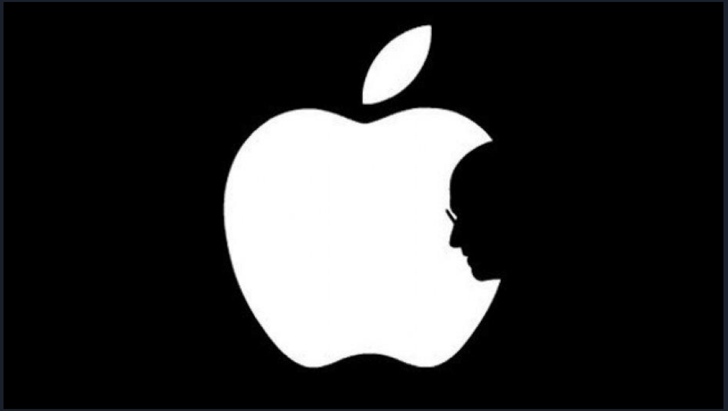
Machine learning model by Geography

- On a large scale machine learning model, the complexity of the model increases considerably
- To reduce complexity, domains by geography should be separated
- Create separate machine learning models by geography with limited domains

Create iOS Version

- Apple contains massive market share
- Lots of potential in iOS market
- Flutter or Swift?





Conclusions

&

Thank You!