Assignment # 3:	AutoML (no-code/low-code Machine Learning)
Course:	CAP 6610 Applied Machine Learning
Professor:	Dr. Oge Marques
Student Number:	Z23596812
Student Name:	Renee Raven
Term:	Fall 2022

Report: Part 1

Tool 1 : Lobe

Lobe is a friendly site that incorporates bite size info in big font and nice graphics. It targets someone new to ML. It offers image classification (label an imaged based on its content). Future development includes object detection (locate an object inside of an image) or data classification (label data in a table based on its content). A highlighted testimonial uses Lobe for improving models of nature-dependent tourism. One user states it is easy enough for his kids to use and that the kids enjoy it. Additionally, Lobe offers several ways to export the app once trained.

The "Introducing Lobe" video promises: "Build your first machine learning model in ten minutes." I followed the instructions to create a machine learning model to recognize drinking after downloading the app to my mac laptop.

1. Add the name of the task "Drink Tracker"

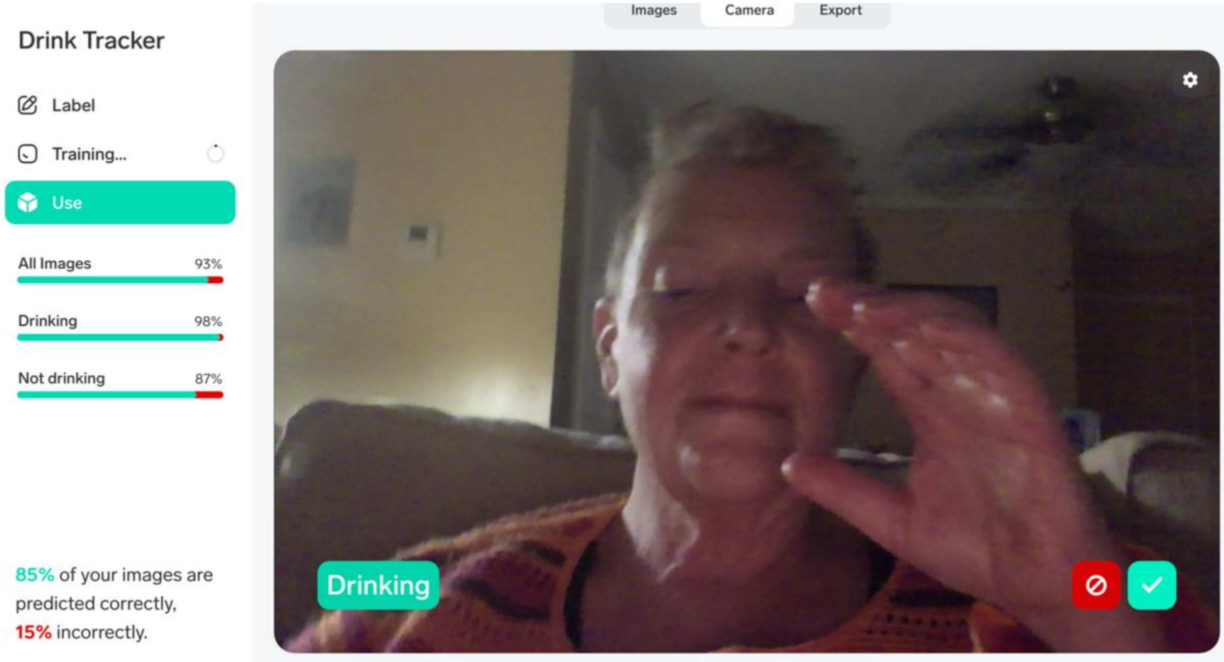
2. Go to import and use the camera function on the laptop to take pictures while drinking from a glass

3. Label those images "Drinking"

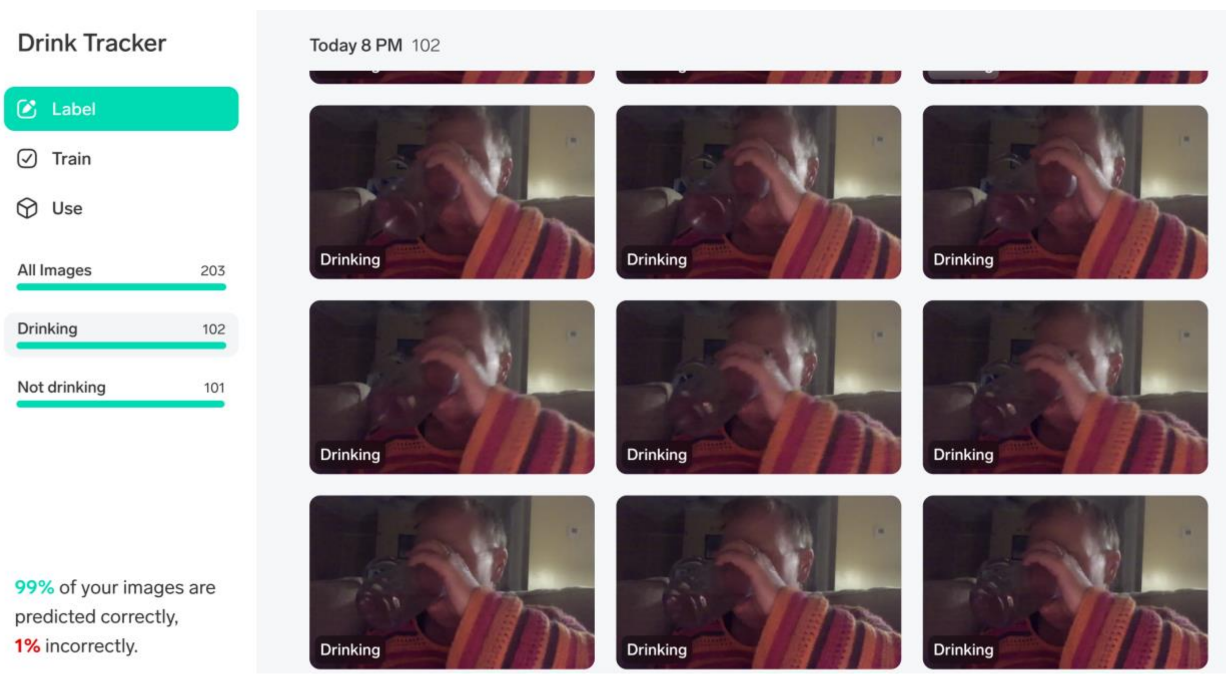
4. Take pictures while not drinking, perhaps mimicking drinking but without a cup

5. Label those images of "Not drinking"

I tested the project with poor lighting. The program started training automatically. Then I switched to the use function where a live picture from the camera is deemed "Drinking" or "Not drinking." Initially, it made several mistakes, but they were easy to correct with the red slash button or green check mark button.

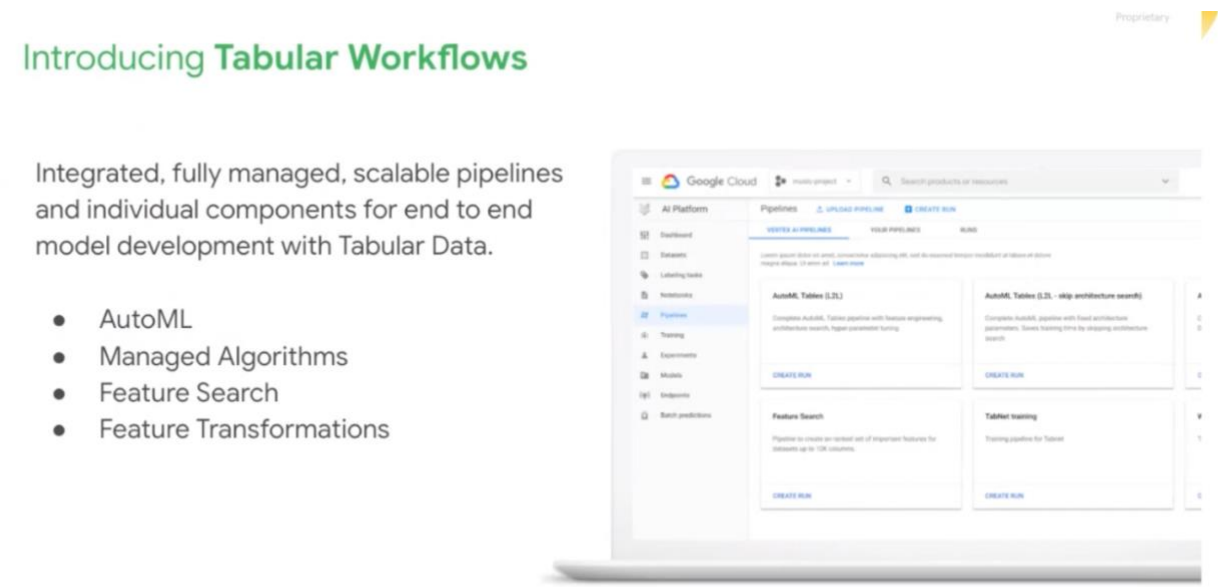


With each correction, the model retrained and the performance greatly improved. Eventually, it correctly predicted 99% of images.

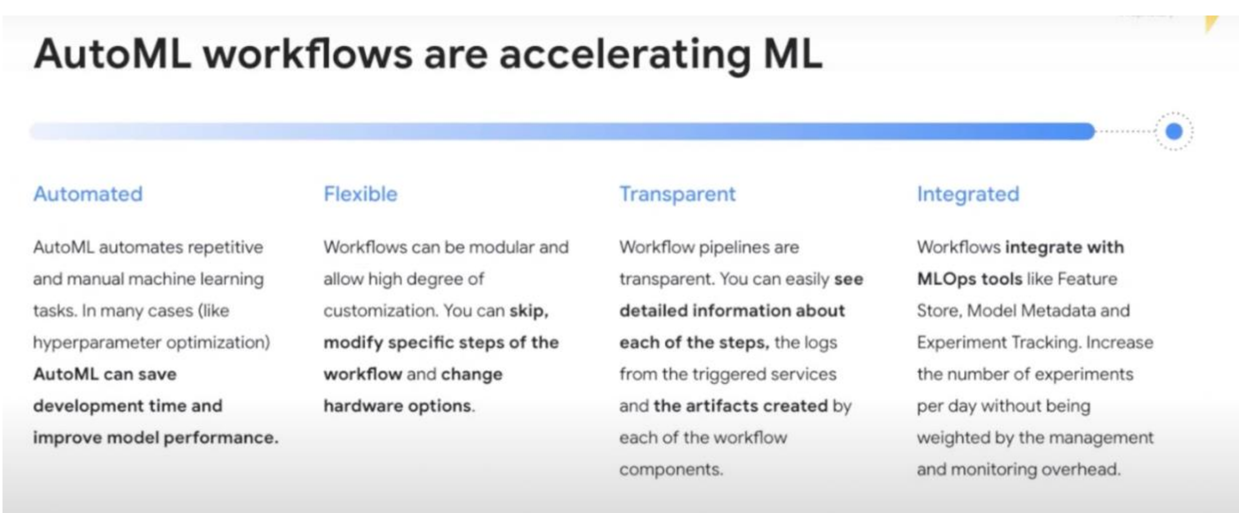


Tool 2 : Google AutoML

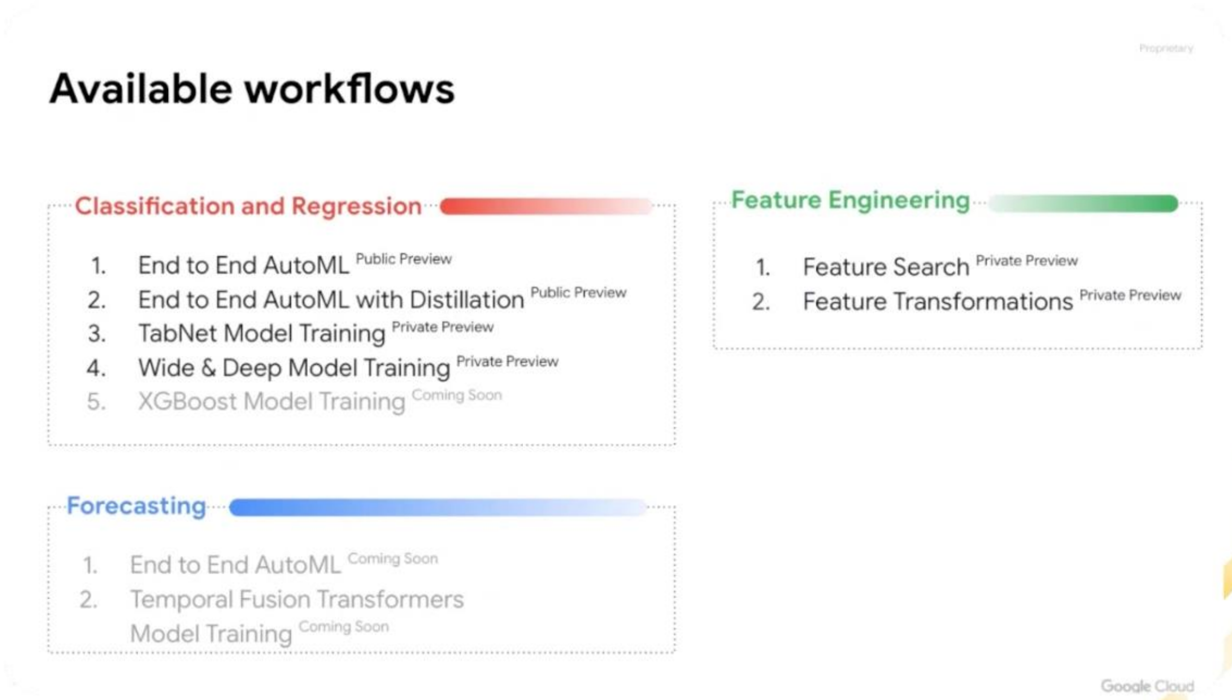
Google AutoML targets software developers with some understanding of machine learning, but not enough to utilize it for business purposes. AutoML helps users create their own custom, high-quality, models to answer their business concerns. The site offers a short video which introduces tabular workflows which are scalable pipelines that minimizes complexity and high maintenance costs. Endorsements include Lowe's (they use it for inventory tracking) and General Mills (usability, scalability, and reliability).



The video also talks about the 2 most common problems with standardized ML pipelines: lack of transparency and lack of flexibility. They suggest AutoML workflows provide flexibility (modifiable) and transparency (detailed information in logs and artifacts).

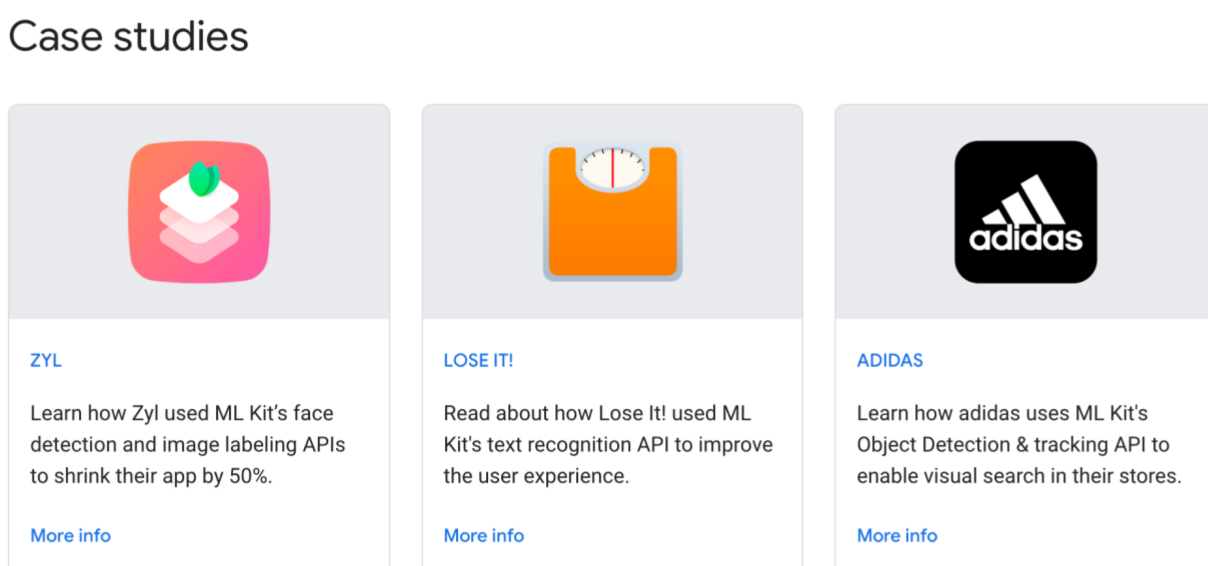


Google AutoML offers a menu of classification, regression, and feature engineering tools. Forecasting, including end to end AutoML and temporal fusion transformers model training, are in development.



Tool 3 : Google ML Kit +

Google ML Kit is aimed at the mobile app developer. It brings Google's "on-device machine learning expertise to Android and IOS apps." It offers face detection, text detection and object detection integration, as evidenced by the highlighted case studies:



ML Kit offers sample apps that are either quickstart (simple examples to launch quickly) or showcase apps (combines multiple ML Kit APIs with Material Design components to generate polished apps), all optimized for mobile.

Two of the more interesting sample apps are pose detection (real time identification of the body postion) and face mesh detection with 468 3D points and edges. Use cases of face mesh include real-time AR filters, selfie capture, and video chat.

Face mesh detection **NEW**

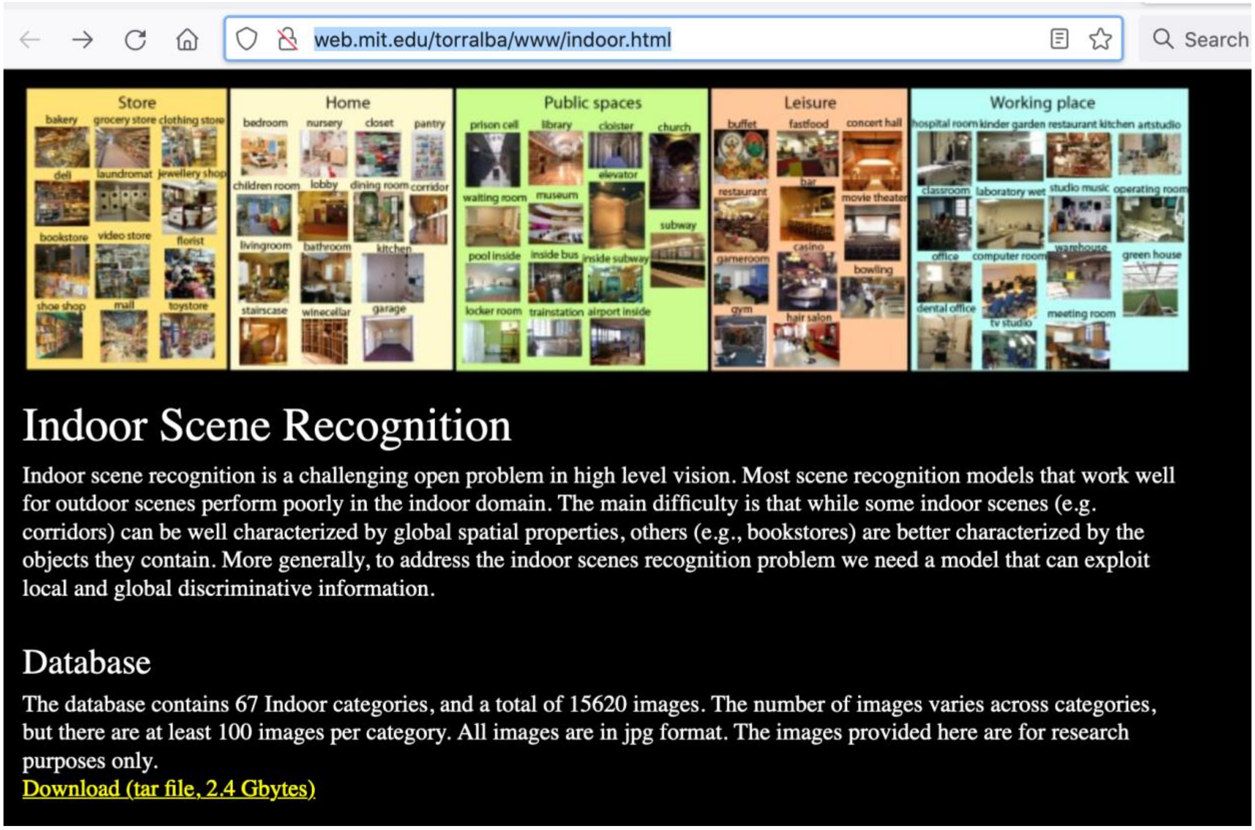
Detect face mesh info on close-range images.
Get started

But Google ML Kit offers a wide range of capabilities including: face recognition, text recognition, translation, image labelling, object identification and tracking, barcode scanning, and smart reply where replies are automatically generated based on the flow of text conversations.

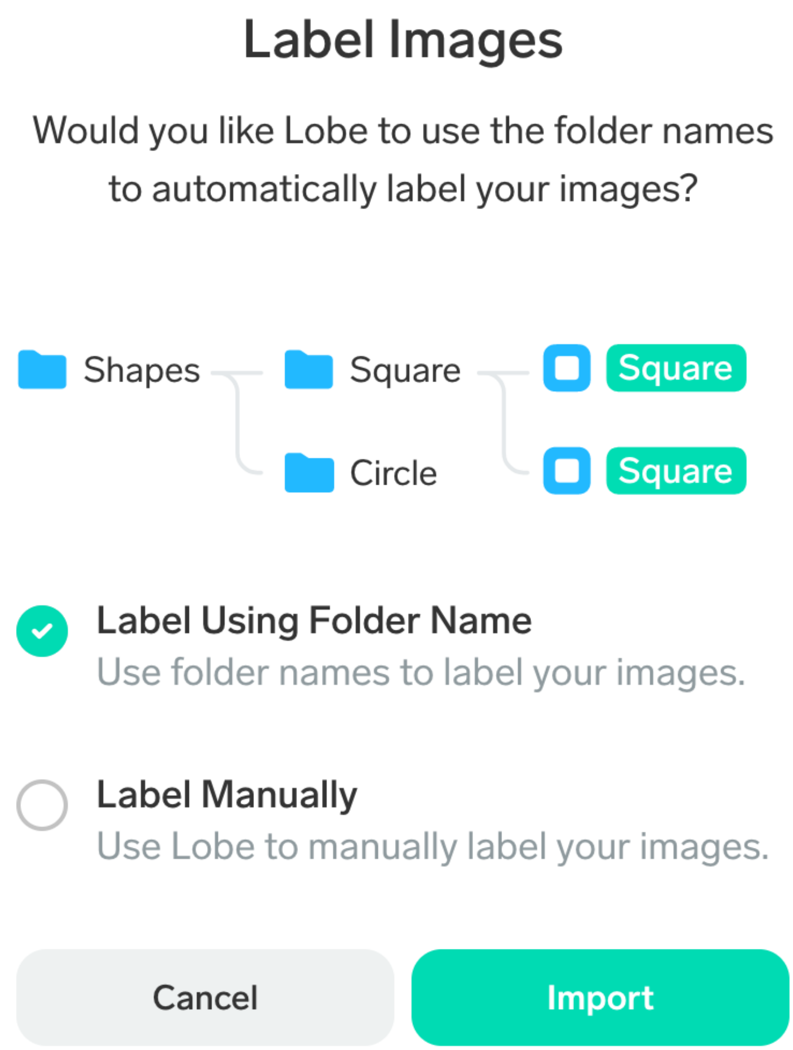
Report: Part 2

I was pretty impressed with the simplicity of Lobe.ai, so I looked forward to challenging it with images from 2 categories that are fairly close. Indoor scene recognition from MIT offers a collection of 67 image categories where each category contains at least 100 unique images.

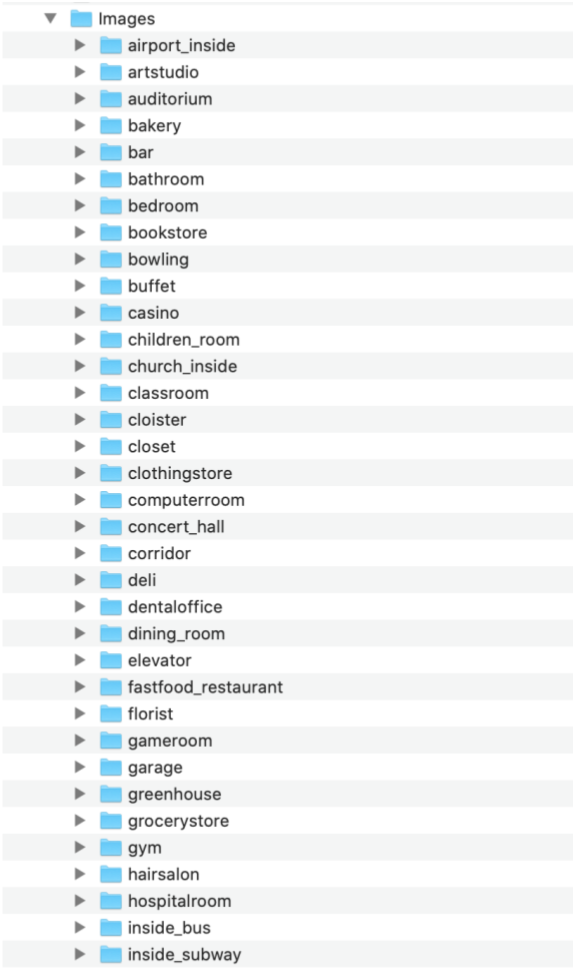
http://web.mit.edu/torralba/www/indoor.html

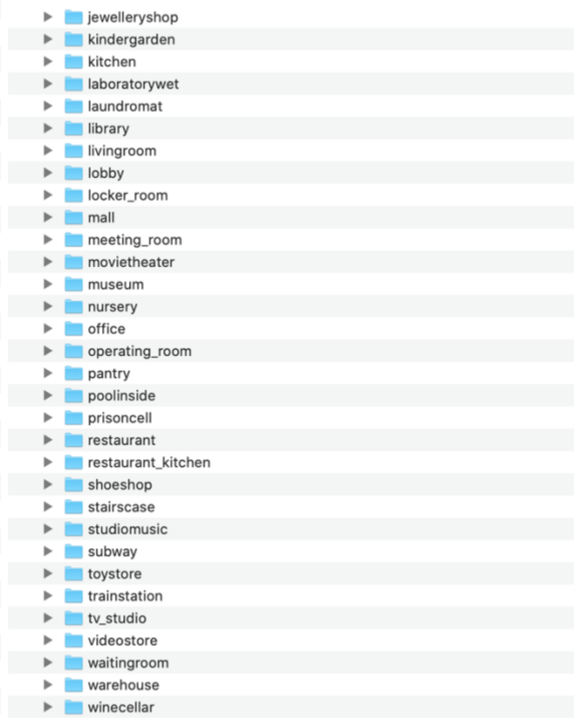


This particular dataset uses labelled folders to collect the images, making it ideal for working with Lobe.

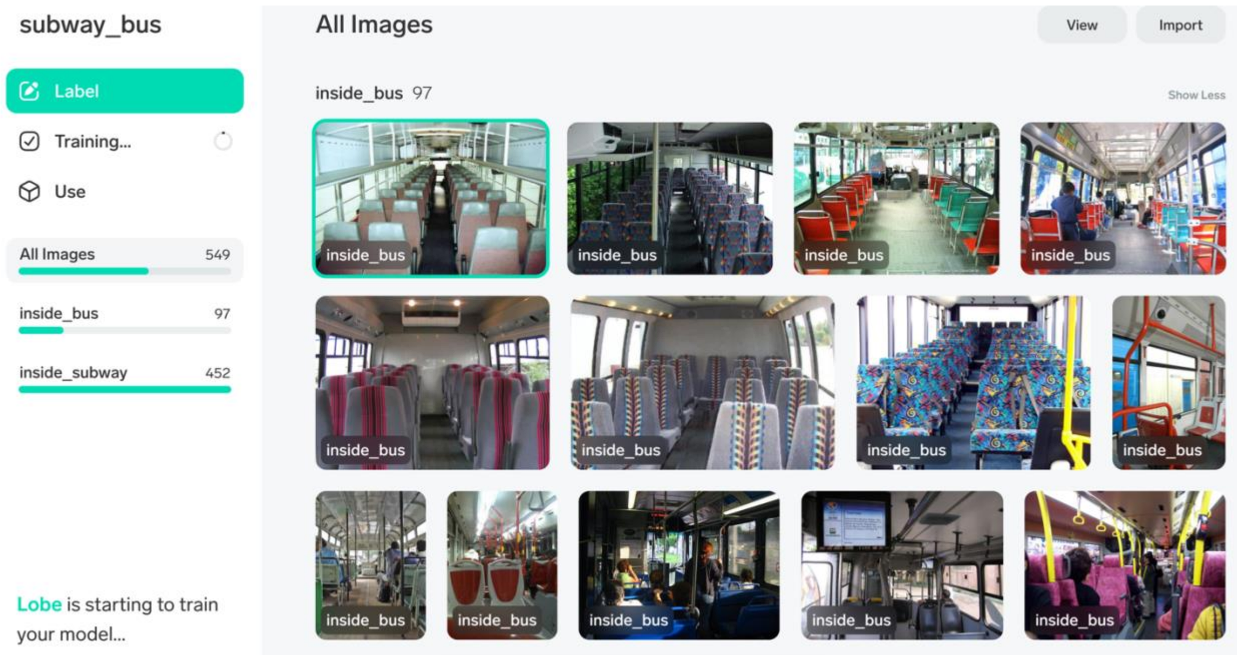


From the 67 potential categories, I selected indoor_bus and indoor_subway, since I suspected they would have similar images.

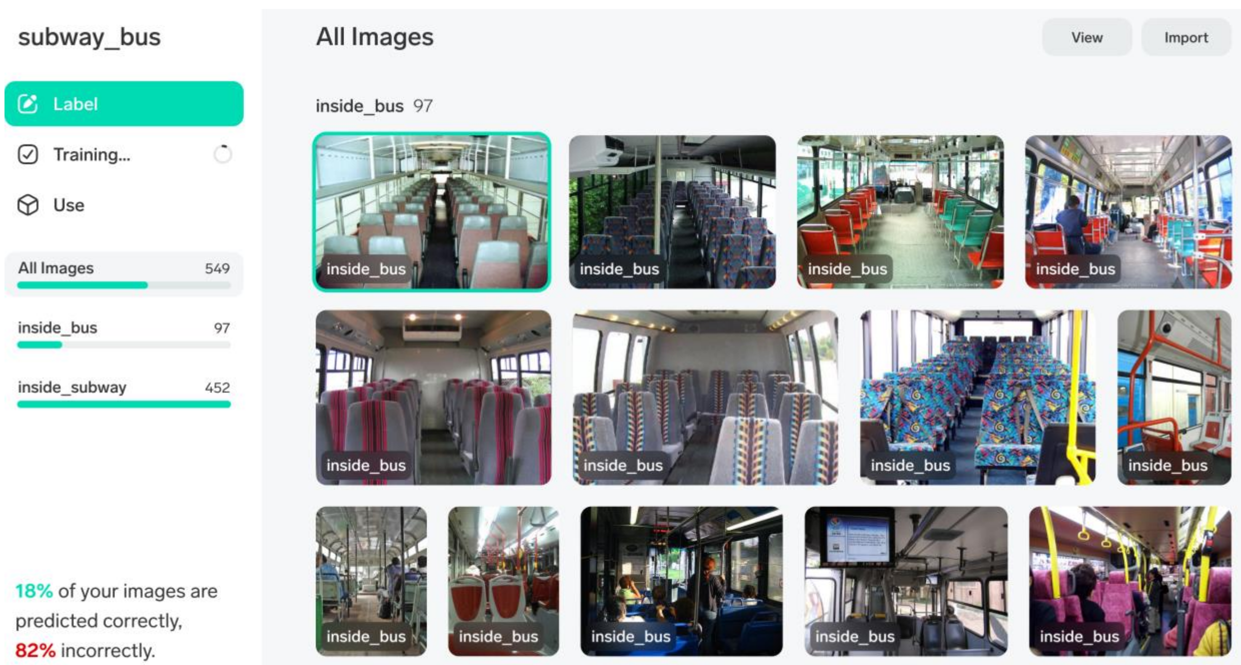


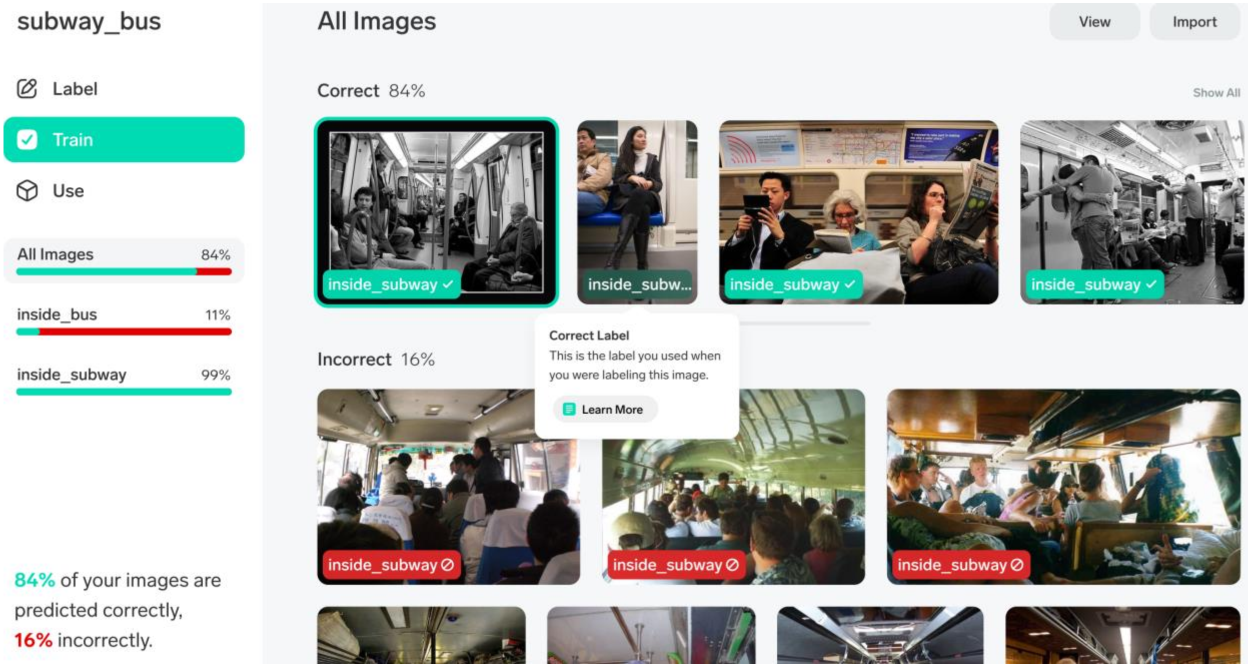


I then extracted the first 5 from each folder to perform a manual test after the model was trained. Lobe rejected 2 of the images. In total, 97 of the 98 available images from inside_bus folder were uploaded and 452 of the 453 available images in inside_subway folder were uploaded.



Training took less than 10 minutes and progress was updated periodically.

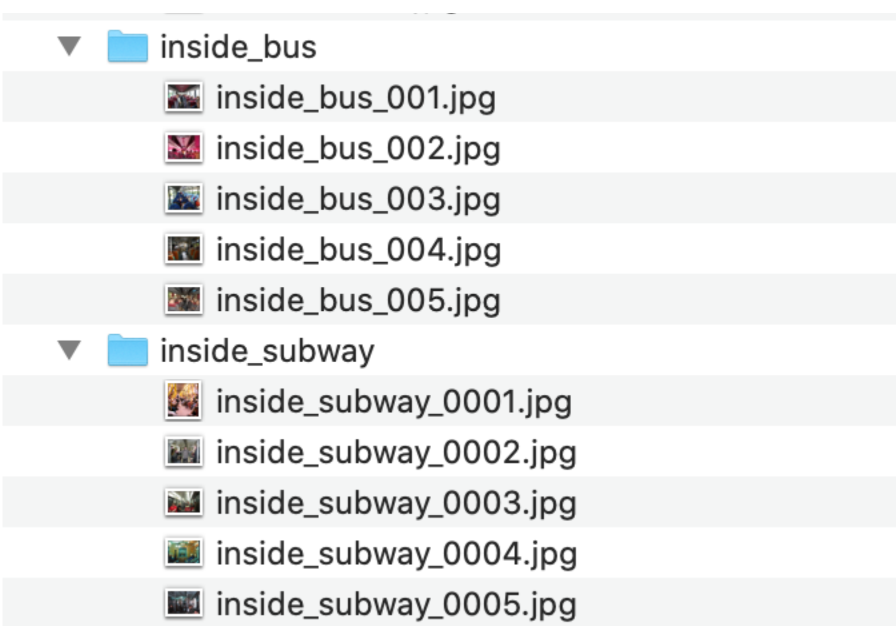




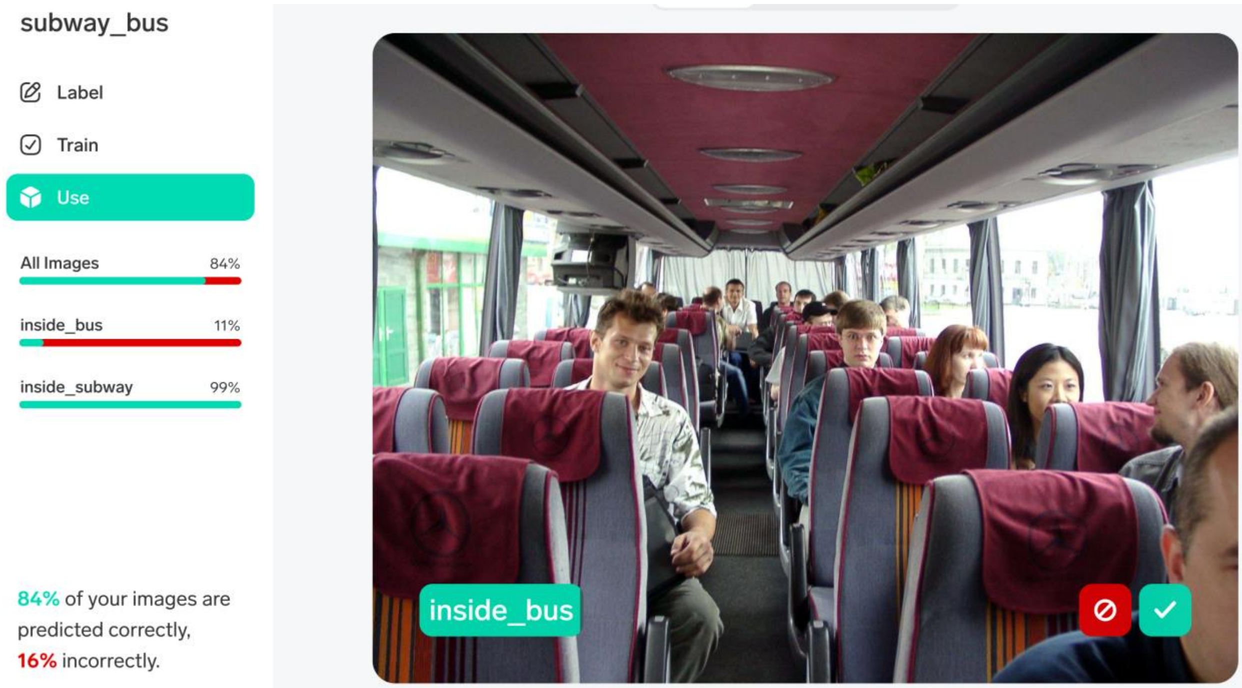
84% might sound like a great number, but Lobe didn't actually perform so well on identifying the inside_bus images (only 11%). This imbalance is likely due to the imbalance in the size of the datasets (97 vs 452 images).

This is a real weakness of the model. Unfortunately, someone who is naïve to ML problems might not catch why this is so imbalanced and think they have created a pretty model.

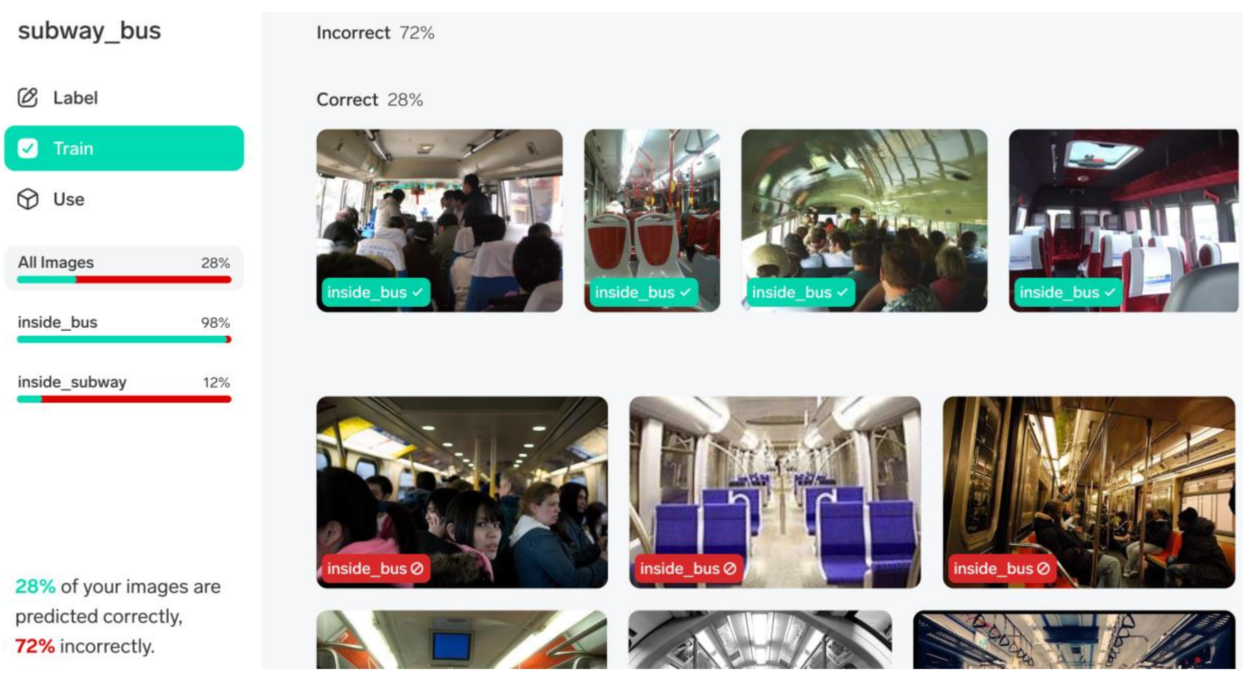
Then I tested the model with the 10 sample images.

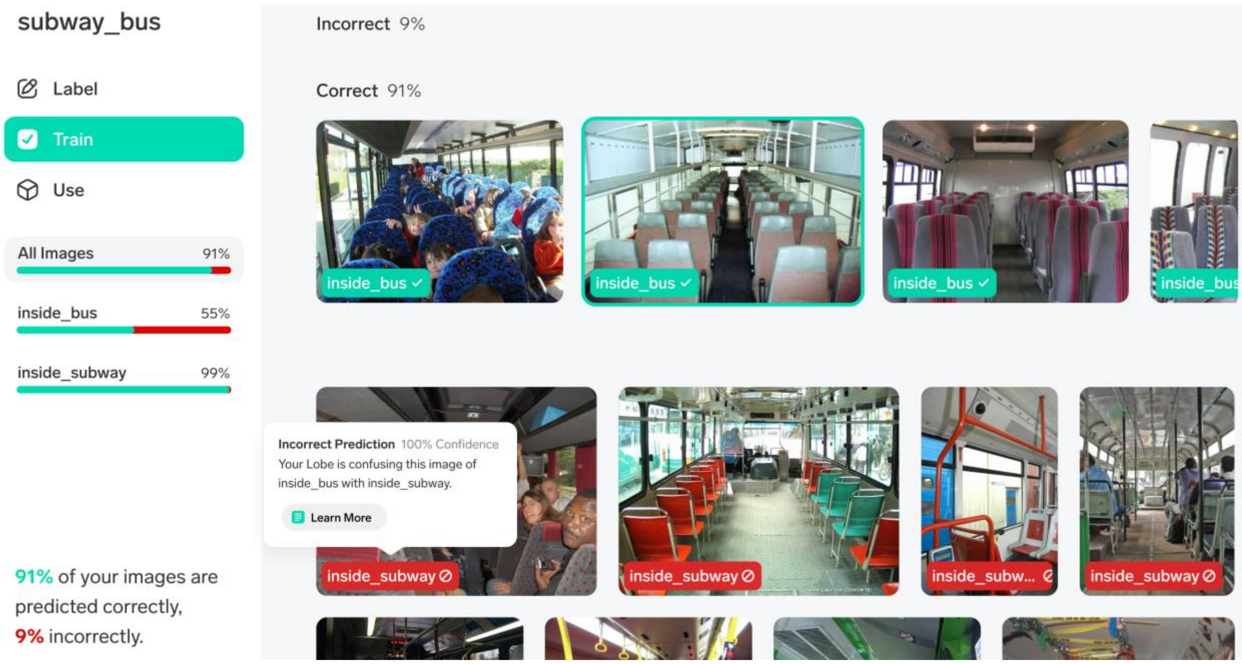


The Use function was used to check the test images. The first image from the inside_bus test files was loaded. Every time a new image was loaded, the user was prompted to indicate if it was right (red slash circle) or right (green check mark).

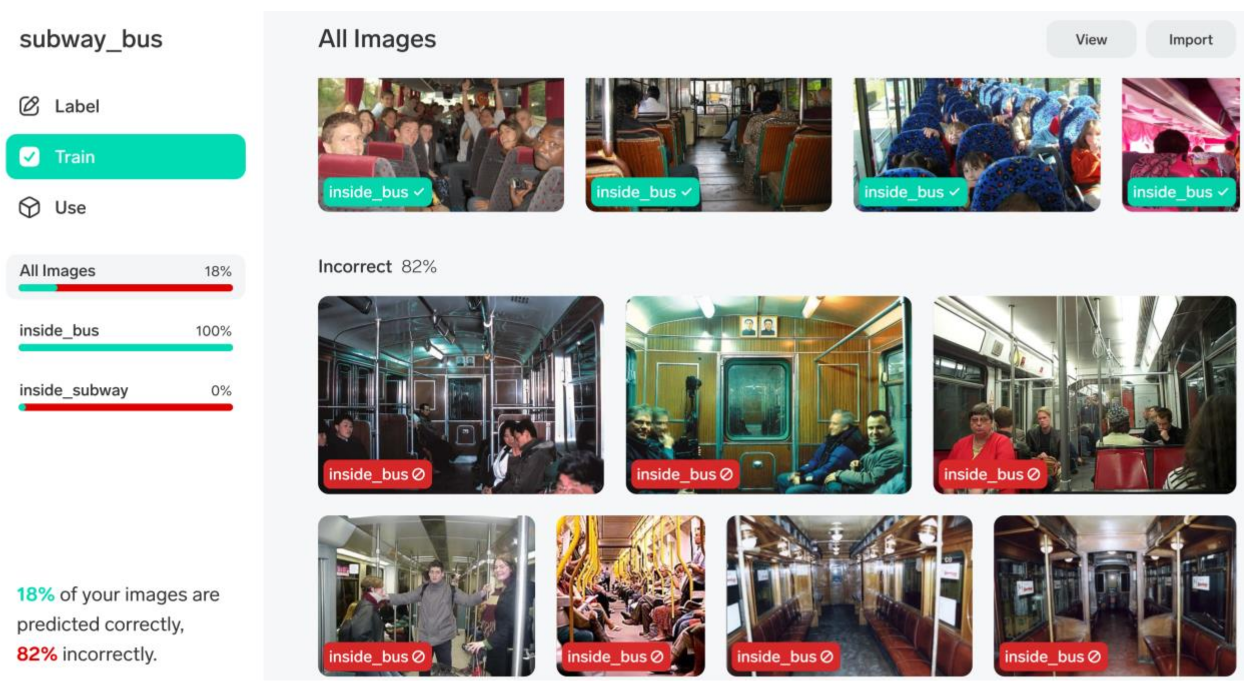


2/5 of the inside_bus images were incorrectly identified. 3/5 of the inside_subway images were incorrectly identified. During testing, the percentage of correct/incorrect flip-flopped. It seemed to depend on whether the model was over predicting inside_bus or inside_subway.





And the final percentage, after all 10 test images were loaded, was pretty dismal.



With a total of 5/10 test images correctly identified during the testing phase, it is clear this model performs at chance level and has not developed an algorithm to make useful predictions. This may be because of the imbalanced datasets. It would be interesting to see if it did better with 2 categories with an equal number of images and/or 2 categories that are more dissimilar during future experiments.