Smartphone Detection and Cost Estimation

Jake Garrison

UNIVERSITY of WASHINGTON

UNIVERSITY OF WASHINGTON
ELECTRICAL ENGINEERING

Motivation

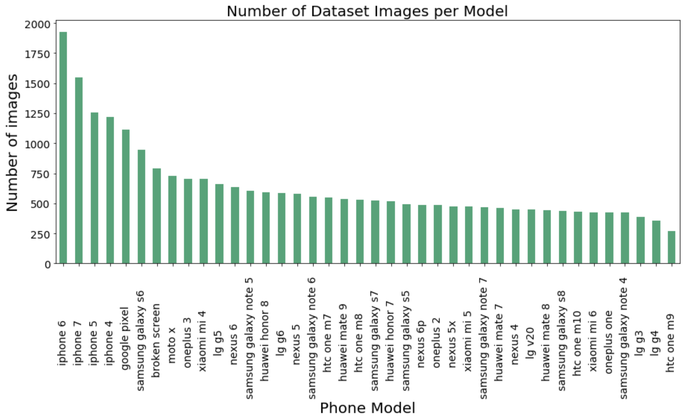
Using a convolutional neural net (CNN) purely for object detection is well understood. There is less research emphasizing the learning of context, for example object cost, for a given detected object.

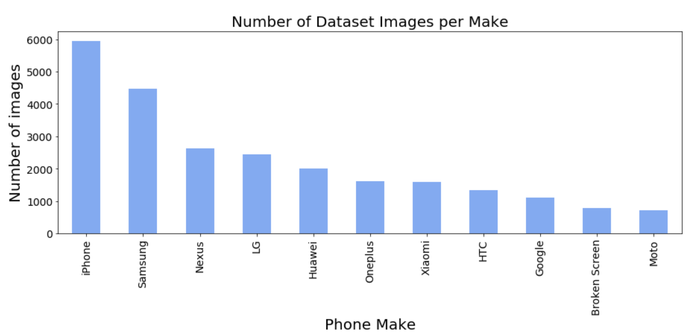
Smartphones were chosen for this project because:

- **There are various models with unique features**
- **Most people have a smartphone, or at least recognize them**
- **There are several images on the internet that can be used in a dataset.**

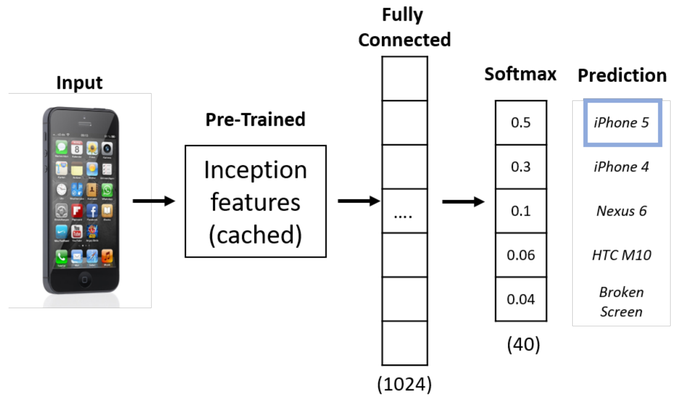
Dataset Construction

1. Preliminary **research** on the most popular phone make and models in the last four years.
2. Developed a **script to find and download images** based on keywords from popular search engines like Google, Bing, Yahoo and Baidu and remove corrupt and duplicate results
3. Identified **50 phone models** and searched for images using keywords like *new, used, refurbished, for sale*. Ideally the images are not stock photos.
4. Added a ***broken screen* class** to detect broken phones
5. Gathered around **30K photos** with around 600 per phone model.
6. Utilized an existing CNN (darknet) trained on COCO to **remove about 5k images** that weren't labeled as a *mobile phone,* greatly reducing false positives.
7. Finally **combined similar phone models** such as the iPhone 6, 6s and 6 plus into one class, resulting in **11 phone makes** and **40 unique phone models**. See the histograms below.





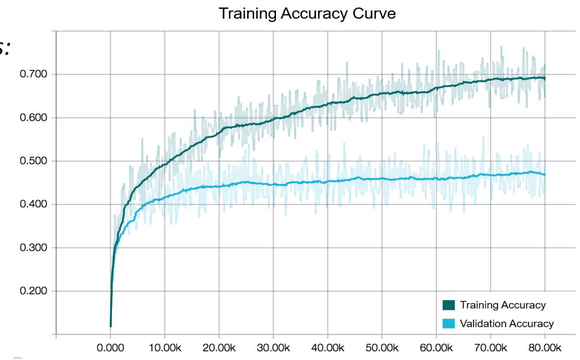
Training the Convolutional Neural Net



Transfer learning is used to retrain the last layer of inception V3

Hyperparameters:

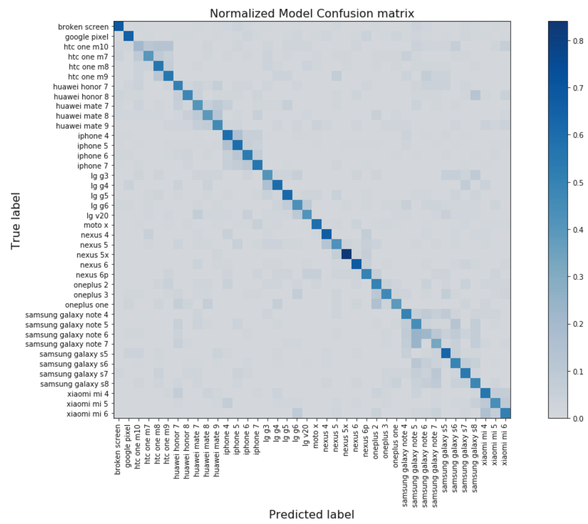
- **Epochs:** 80,000
- **Learning Rate:** 0.01
- **Batch size:** 100
- **Train/Val/Test Split:** 80/10/10
- Random brightness
- Random flipping



The network is trained to minimize the **cross-entropy** loss. The training and validation curves above depict the network learning and improving over time. After 25k epochs, the validation flattens indicating learning is slowing down.

Accuracy Evaluation

Training	Validation	Test (Model)	Test (Make)
75%	51%	54%	73%



Building the iOS App

An iOS app was designed to use the camera and an input the to network for fast **predictions on both the phone model and estimated cost.**

Model Prediction

When the app is open, camera frames are **instantly evaluated by the network** and phone models with a confidence above 5% are. In order to smooth the output displayed, a running sum parameterized by an **update and decay value** is used. If no phone is seen (the confidences are all below 5%), a message is displayed indicating no phone is seen.

Cost Estimation

For each network prediction, a **cost is also estimated and displayed**. Cost is calculated as a **weighted sum** of the market price of each phone, weighted by the confidence outputted by the network. Next, a **discount is applied** based on the confidence level of the 'broken' label. This way phones with wear and tear, or broken screens are devalued.

For example if the model outputs 0.5, 0.3, 0.10 and 0.06 for four models that cost $120, $180, $130, and $210 respectively, and the broken label has a confidence of 0.04 or discount of 4%. The following equation shows how the output cost is calculated.

$$estimated\ cost = discount \times \sum_i confidence_i \times cost_i$$

$$(1 - 0.04) \left(\begin{bmatrix} 0.50 \\ 0.30 \\ 0.10 \\ 0.06 \end{bmatrix} \cdot \begin{bmatrix} \$120 \\ \$180 \\ \$130 \\ \$210 \end{bmatrix} \right) = \$134$$

App Screenshots