Appendix

of

SmartCS: Enabling the Creation of Machine Learning—Powered Computer Vision
Mobile Apps for Citizen Science Applications without Coding

A: Other Use Cases

This section briefly describes the other apps created using our platform.

*TidalNow*

There are different dimensions of animal biodiversity (species richness, phyletic richness, and functional diversity) in tide pools. Activities involving observation of wild organisms in the tide pool can provide recreational and learning opportunities (Fairchild et al. 2018). TidalNow is a citizen science mobile app developed by a high school student. The app uses a machine learning model to identify different types of saltwater marine species in tide pools. The ML model integrated into the app is trained to detect five different species: giant green anemones, ochre stars, lined chiton, sea lemons, and black turban snails. The ML model for this app was trained using about 600 images for each class. Although apps like iNaturalist or Google Lens can recognize these specimens, they require server-side processing and internet connectivity. However, many tide pools are located on beaches with limited or no internet access. As this app works without internet connectivity, it works perfectly fine in these remote locations (Figure A.1).
Figure A.1. The TidalNow App is shown here. Similar to other apps, this app (a) shows the detected object using a bounding box, and (b) the selected template has a built-in pull-up panel that was customized to present additional information about the detected objects.

**Sk.in**

Skin conditions are more prevalent than other illnesses in all countries worldwide (ALEnezi 2019). Some skin diseases can be lethal (Allugunti 2022). Although the advancement of lasers and photonics based medical technology has made it possible to diagnose skin diseases much more quickly and accurately, the cost of such diagnosis is still very expensive (ALEnezi 2019). So, there is a lot of research interest in detecting skin diseases using ML-based computer vision (Srinivasu et al. 2021; Shanthi, Sabeenian, and Anand 2020; J. Rathod et al. 2018). Even though some of these recent works demonstrate very accurate skin disease detection using CNN, there are not many works that can do this in real time on mobile platforms. Sk.in is a mobile application that utilizes ML object detection to categorize dermal conditions as bacterial, fungal, parasitic, viral infections, or allergic reactions in real time. Sk.in intends to increase the efficiency of diagnosing and treating generalized skin conditions for the public and is designed for everyday use. Developed primarily as an educational tool by a high school student, this app also facilitates data collection on skin infections across a diverse demographic. The
ML model can also be trained to detect more skin diseases, such as melanoma and other types of skin cancers (Allugunti 2022). Figure A.2 (a)-(c) displays the app's capability of detecting bacterial infection, allergy, and viral infection.

Figure A.2. (a)-(c) Demonstrates that the "Sk.in" App can detect bacterial infection, allergy, and viral infection. (d)-(e) Shows example results from the sea lions and seals detection and differentiation app, where detected seals are highlighted using green bounding boxes and sea lions are shown using magenta bounding boxes.

**Seal vs Sea Lion**

Biodiversity analysis is important for many research groups, such as those with a focus on biological science, aquaculture, marine biology, etc. Researchers may need to collect data about some endangered species; other times, they need data to analyze the biodiversity in some specific area (Willi et al. 2019; Wood et al. 2021). In this use case, we trained a model with images of sea lions and seals to demonstrate our app's usability for these types of research projects. Many sea lion species are considered endangered (Chilvers and Meyer 2017) and collecting data about them is needed for marine biology research and conservation groups (Brown et al. 2020). However, differentiating between
seals and sea lions can be challenging for non-expert participants (Wood et al. 2021). Using our ML-powered app, the participants can detect and differentiate between these two species (Figure A.2 (d-e)). With further training data and re-training of the model, this app can be modified to detect and differentiate among various sub-species (Hann et al. 2018). The same concept can be applied to create educational and data collection apps about other animal species.

![Vehicle Object Detection](image)

**Figure A.3.** The vehicle object detection app is shown here. The app is used to collect vehicle data by mounting it on the windshield of a car.

**Vehicle Object Detection**

The vehicle detector is a citizen science mobile app for collecting video data about road objects relevant to autonomous vehicle research. It was created as part of an autonomous vehicle research project in collaboration between a university research group and a company from two different countries. The app can be used for road object detection and data collection by mounting the phone next to the windshield of a car. Various standardized datasets exist for autonomous vehicle research, such as Kitti, Waymo, NuScene, etc. (Kang, Yin, and Berger 2019). However, these are collected using arrays of advanced sensors mounted on specialized vehicles (Janai et al. 2020). This app
facilitates small-scale experiments and data collection worldwide, employing a simple and inexpensive mobile setup. Even using the app with multiple mobile devices to collect multiview datasets would be much cheaper than using the traditional autonomous vehicle data collection setup. Therefore, it can be used to collect data about various exotic vehicles, such as three-wheelers, rickshaws, etc., using citizen science from different developing countries. These vehicles are not commonly seen in developed countries and, thus, not present in the standard datasets (Kato et al. 2015).

The other apps developed with the platform include those for detecting plant leaf diseases, identifying beach debris, recognizing types of building architecture, assessing fingernail conditions, counting blood cells, and classifying ultrasound image types.

B: Summary of Used ML models

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Inference Speed (ms)</th>
<th>mAP for COCO objects</th>
<th>Mobile model size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSD MobileNet v1</td>
<td>48</td>
<td>29.1</td>
<td>5</td>
</tr>
<tr>
<td>SSD MobileNet v2</td>
<td>39</td>
<td>28.2</td>
<td>5</td>
</tr>
<tr>
<td>EfficientDet D0</td>
<td>39</td>
<td>33.6</td>
<td>6</td>
</tr>
<tr>
<td>EfficientDet D1</td>
<td>54</td>
<td>38.4</td>
<td>8</td>
</tr>
<tr>
<td>EfficientDet D2</td>
<td>67</td>
<td>41.8</td>
<td>11</td>
</tr>
<tr>
<td>YOLOv8m</td>
<td>32</td>
<td>50.2</td>
<td>49</td>
</tr>
</tbody>
</table>
Table B.1: This table presents a summary of the ML models tested and supported on our platform. The information provided here helps the app creators to select a model for training. The faster the inference speed (second column), the better real-time performance the app gains. The higher mean Average Precision (mAP) in the third column represents better precision for detecting objects. Comparing the second and third columns shows that higher precision requires more inference time, leading to slower than real-time performance. The app creator needs to decide about the trade-off between these two. The fourth column shows the approximate final size of the converted trained model, which may impact performance on older devices with lower computational resources.

The table in this appendix lists the ML models tested on our platform, enabling app creators to select the most appropriate model for training (Abadi et al. 2015). In our platform's current selection of compatible models for mobile devices released up to 2023, YOLOv8m is the largest recommended model that runs smoothly on a typical consumer mobile device, with a saved weight size of 49 MB. However, even though YOLOv8m shows better performance metrics on benchmarks, we found that EfficientDet D2 offers more stable performance overall during our testing with a few current-generation smartphones (Apple's iPhones and Google's Pixels). Over time, with the introduction of more powerful mobile devices and larger compatible models, these can be included in our platform without significant modifications.
The number of images needed to train an object detection model can vary widely depending on several factors, such as the complexity of the task, data quality, and model architectures. Decent results can still be achieved even with hundreds of images per class. Shahinfar, Meek, and Falzon (2020) suggest an inflection point of around 150-500 images per class, beyond which the earlier sharp performance gains start to level off. However, Bochkovskiy, Wang, and Liao (2020) argue that for optimum accuracy, having at least 2000 different images for each class is desirable. As a rough guideline, for a simple object detection task with a few object classes and relatively consistent object appearances and backgrounds, a few hundred to a few thousand images might suffice, especially if transfer learning is utilized. For more complex tasks or a larger number of object classes, tens or even hundreds of thousands of images might be needed.

The number of classes supported for training depends on the model type and architecture. For instance, according to official documentation, EfficientDet can support up to 999 classes (Tan, Pang, and Le 2020), whereas YOLOv8 (Ultralytics 2023) does not have a defined hard limit for the number of classes. However, the choice of model determines the number of classes; it is not a limitation of our platform, SmartCS, since we can incorporate newer versions of models that support more classes.

Although transfer learning cannot be performed within the app due to technological limitations and the resource constraints of mobile devices, which render model training infeasible, models can be updated periodically with newly collected data by retraining them using transfer learning on more powerful machines or cloud servers. Additionally,
we provide pre-trained models derived from well-known public image datasets, such as MS-COCO, to serve as a starting point for training on new datasets via transfer learning.

References


