Homework 7: Mobile ML

Due: November 24, 2025, 11:59 PM
Submit: via the class TEAMS site
Grading: 30 points total (+ up to 5 pts Extra Credit)

Goal

Build a full image-classification pipeline:

1. Use **Chrome + Fatkun Batch Download** to collect images from the web and sort them into separate subfolders (one per class).
2. Use **Roboflow** to clean and prepare the dataset (train/valid/test split, resizing, normalization).
3. Use **Create ML (Image Classifier)** to train a mobile ML model.
4. Embed the model into a simple **iOS app** that runs on an iPhone and classifies images from the Photos library.

Behavior Details

Your iOS app should:

- Let the user **pick an image** from the Photos library.
- Show the **selected image** on screen.
- Run **on-device classification** with your Create ML model.
- Display:
    - The **predicted class label** (e.g., “Prediction: mug”), and
    - The **top confidence** (e.g., “Confidence: 0.87”).
- Handle failure gracefully (e.g., “Could not classify image.”).

Learning Objectives

- Practice **dataset curation**: search strategy, filtering, and organizing images into class folders.
- Use **Fatkun Batch Download** in Chrome to quickly download large sets of images.
- Use **Roboflow** to:
    - Import folder-based class labels,
    - Split into **Train / Validation / Test**,
    - Apply basic preprocessing (resize, normalization).
- Train an **image classifier** using **Create ML**, interpret metrics, and export model.
- Embed a Core ML image classifier into a **SwiftUI iOS app** and run it on iPhone.

Tasks (30 pts)

Part A — Dataset Collection with Fatkun (5 pts)

- Install Fatkun Batch Image Downloader (Chrome extension).
- Focus on the following object classes: **mug, book**, **keyboard**, **backpack**.
- For said domain, search for images on Google Images.
- For each of your 4 classes:
    - Use Fatkun to select and download images relevant to that class.
    - Aim for at least **120 raw images per class**.
- After downloading, organize into a root folder, e.g. Dataset/, with structure:
    - Dataset/mug/
    - Dataset/book/
    - Dataset/keyboard/
    - Dataset/backpack/
- Remove obviously problematic images (wrong class, text-heavy memes, tiny thumbnails, duplicates).

Part B — Roboflow Dataset Prep (7 pts)

- Create a new project on **Roboflow** as an **Image Classification** project.
- Upload your Dataset/ folder; ensure Roboflow uses **folder names as labels**.
- Perform a **dataset split**:
    - Train: **70%**
    - Validation: **20%**
    - Test: **10%**
- In the **Preprocessing** / **Augmentation** steps:
    - Resize images to a fixed size (e.g., 224×224 pixels).
    - Apply any basic augmentation you like (small rotations, etc.), but keep it reasonable.
- Export the dataset in a **Create ML-compatible image classification format**.
- Download the exported dataset and verify it has the expected **train/valid/test** splits.

Part C — Train the Image Classifier in Create ML (6 pts)

- Open **Create ML** and create a new **Image Classifier** project.
- Import the Roboflow-exported dataset folder.
- Use default settings for the first run (you may experiment later if you want).
- Train the model and inspect:
    - Training accuracy,
    - Validation accuracy,
    - Per-class performance (if available).

- Target: at least **70% validation accuracy**. If your accuracy is low, briefly note likely reasons in your README (e.g., noisy images, small dataset).
- Export the trained model as a .mlmodel file with a clear name, e.g., ObjectClassifier.mlmodel.

Part D — iOS App Integration (7 pts)

- Create a new **iOS app** in Xcode (SwiftUI).
- Add your .mlmodel to the Xcode project (drag into the project, ensure "Add to targets" is checked).
- In your SwiftUI app:
  - Provide a **button** to pick an image from the Photos library.
  - Show the selected image in the UI.
  - Run the Core ML model on the image.
  - Display:
    - Predicted label as a prominent text (e.g., large font),
    - Confidence (rounded to 2 decimal places).
- If you have **a real iPhone** run your app on that. If not, run it on the simulator. If the simulator cannot cope with the task, have a graceful exit.
- Layout: simple, readable, and responsive; avoid cramped or overlapping elements.

Part E — Code Quality & Writeup (5 pts)

- Organize your project with clear structure (e.g., Views, ML helpers).
- Use descriptive names for variables, functions, and files.
- Remove unused imports and dead code.
- Include a short **README.md (≤250 words)** that explains:
  - How you collected and cleaned the images,
  - Roboflow preprocessing choices and train/valid/test split,
  - Create ML results (validation accuracy, any observations).

Extra Credit (+5 pts max)

You may implement ONE or more of the following (max +5 total):

1. **Live Camera Mode (+3 pts)**
   - Add a mode that uses the camera to classify a live preview or a freshly captured photo.
2. **Top-3 Predictions (+2 pts)**
   - Show the **top 3 predictions** with labels and confidences sorted by probability.
3. **Prediction History (+2 pts)**
   - Keep a simple in-app log of past classifications (image thumbnail + label).

Clearly document any extra credit features in your README

Deliverables

- A **zipped Xcode project** that builds and runs on an iPhone.
- Your **Create ML .mlmodel file** included inside the Xcode project.
- The **Roboflow project link** (or screenshots of your dataset and split) in the README.
- README.md (≤250 words) with the information described above.

Grading Rubric

- A. Dataset Collection with Fatkun — 5
- B. Roboflow Dataset Prep — 7
- C. Create ML Training — 6
- D. iOS App Integration — 7
- E. Code Quality & Writeup — 5

  Total: 30 points