CS-632 Course project

About: The goal of this project is to give you something fun & memorable to build that's useful down the road (this would be a good project to bring up, for example, in an interview - if you were asked "what have you worked on lately?"). As we're doing this project instead of a final, you should expect to spend considerable time on it. I recommend starting early, so I can help you with any questions that pop up, and make sure you're not blocked on the mechanics (e.g., installation issues, webcam issues, etc).

Pick your project: You may chose to complete the recommended project below, or you may propose your own.

Resources: These Keras examples (5.1, 5.2, 5.3) provide good coverage of the techniques you'll need to train an image classifier. To capture images from a webcam, I recommend OpenCV.

Project 1) The Incredible Teachable Machine

For inspiration, see https://teachablemachine.withgoogle.com/.

About: In this project, you'll write a program that can be trained to recognize images captured by a webcam. For example, your program might be trained to recognize when the user is smiling, frowning, or making a silly face. Or, it might be trained to identify the type of object they're holding, say - a book, a coffee cup, or a tennis ball. Or, to recognize whether the family cat appears in the video with them.

To make it interesting, you will not know what types of images your program has been trained to recognize in advance. And to make it habitable for the user - your program will work with video streams. To collect training data, your program will capture a few seconds of video in which the user demonstrates each scene they want to recognize. For example, to train a classifier for *smiling vs. frowning vs. silly faces* - the user will capture a few seconds of video of each pose. Your program will extract frames from the video, use them to train an image classification model, and begin using it to classify images it receives from the webcam.

Equipment required: laptop with a webcam. If your laptop doesn't have a webcam, ask the instructor for an extra USB webcam.

Design

Your program will have three phases **a) collecting training data**, **b) training and evaluating a model**, and **c) classifying images**. Here's an outline of what you should aim to do in each phase.

Collecting training data.

- When your program starts, the user will want to collect training data for each of the categories they want to recognize. You can assume there is a predefined number of categories (say, three) - and hard code your program to work with that number.

- To collect training data, the user will select each of the categories in turn. Use the webcam to capture 5 to 30 seconds of video (experiment to figure out what length works best). Either save the video to disk and extract frames later when it's time to train your model, or you can extract frames in real-time. If you like, you can explore other strategies of collecting training data (for example, perhaps you can prompt the user to change poses, rather than having them manually select each scene).

- When the user has finished collecting training data, it's time to train your model.

Train and evaluate a model.

- Here, you'll train an image classifier that can predict what category of frame it sees. If you haven't already, extract the frames from your training videos. You'll want between 100 and 2000 frames per category. Experiment to find a number that works well.

- Preprocess the images before starting training to reduce the resolution to something manageable, and so the data matches the format your model expects.

- Given these images and labels (which you know from the category), you can train a classifier similarly to Assignment #2. I recommend experimenting with Transfer Learning (see the considerations section below) to do this efficiently.

- You'll want to evaluate your model, and run an experiment to determine how many epochs you should train for. You should include a discussion of this in your writeup.

Classify images.

- Once your model is trained, your program should begin classifying images it receives from the webcam. Be sure to preprocess them in the same way as your training data.

- Give the user feedback (visual, or text) to indicate which category the current scene is being classified as.

- Bonus: take an action based on the classification result (play a sound, show a color, etc).

Considerations

- **Extract frames**: Although we are working with video, we will not consider *interactions* between frames. You may treat the output of the webcam as a collection of independent images - and train an image classifier as you normally would for Cats vs. Dogs in Assignment #2. To do so, you will need to extract the frames from the video.

- **Input preprocessing:** Most image classifiers work with relative small images (say, 32x32 pixels). What size is appropriate for your project? Your webcam probably captures images at far higher resolution, but it will not be possible to train a model that runs efficiently with data that large. Instead, you will need to **preprocess** the images in some way - e.g., by downsizing them - before training your model. Remember to preprocess "testing data" identically when you begin classifying images. Also, be sure to convert color values to be between 0 and 1 (not 0 and 255).

- **Transfer learning**: When you train your classifier, you will be working with a small amount of training data (on the order of 100-1000 examples per class). Think: will you train an image recognition model from scratch, or will you use transfer learning on top of a pretrained model? If so, does it make sense to use an off the shelf model (e.g., VGG-16)? Why or why not?

- **Underfitting vs. Overfitting**: When users train their models, it will be helpful to report how accurate you think the model is in practice. To do so, you will need to decide how long to train (e.g., determine what number of epochs is likely to give a good result). You may wish to use some of your training data for validation to figure this out. Be sure to train your final model using all the data available.

- **Performance:** after your model is trained, it isn't necessary to classify images in real-time (especially if your laptop isn't super fast). You can make predictions at a best-effort pace.

- **User Interface**: any UI you prefer is fine. You can even write a console program that's text only, so long as it matches the functionality above (and has at least one window so the user can see the output of the webcam as they capture training data, and as they classify images once the model has been trained).

Extra credit: if you complete this project and are looking for an added challenge, consider also allowing the user to train a model to recognize *sounds*. This will be difficult, and you can see this tutorial for advice.

Project 2) Propose your own

If you'd like to propose your own project, email the instructor a description, and an outline of your design (similar to above) by the evening of 10/31 (at the latest). You *must* receive approval

before starting on your own project. I'll get back to you with comments and an an *approve* / *approve with modifications* / or *reject* that week.

Note: you do not have to use Deep Learning or image classification - there are many great datasets you can explore with Decision Trees or Random Forests, check out these for examples: https://archive.ics.uci.edu/ml/datasets/

Submission instructions

Please submit a link to your github repository on blackboard by Monday December 18th (the day before the last day of class). We will have project presentations on Tuesday the 19th.

In addition to code, your GitHub repository should contain:

- A link to a YouTube video (can be unlisted) showing your project in action.

- README.md - containing a **writeup** (figure, one to two pages describing your work). The reason I am not having you produce an extensive write-up (like a research style paper) is because I believe the code will be more useful to you down the road. That said, you should at a minimum discuss the considerations above in your README - this will be part of your grade.

Grading: This project is worth 60% of your grade. Grading is identical between project 1 and project 2.