Programming Assignment 2:
Interactive Deep Learning via Concept Bottleneck Modeling

In this assignment, you will develop an interactive deep learning pipeline for bird species classification. This assignment is designed based on Koh et al. (ICML 2020). You are highly encouraged to read this paper first before starting the assignment.

Dataset

You will use a bird dataset, Caltech-UCSD Birds 200 2011 (CUB-200-2011), as the training and test data. Please download the dataset via this link. This dataset is very large. Make sure you have enough disk space before downloading and unzipping this dataset.

Caltech-UCSD Birds 200 2011 (CUB-200-2011) is a challenging image dataset annotated with 200 bird species (mostly North American). It was created to enable the study of subordinate categorization, which is not possible with other popular datasets that focus on basic level categories (such as PASCAL VOC, Caltech-101, etc). The images were downloaded from the website Flickr and filtered by workers on Amazon Mechanical Turk. Each image is annotated with a bounding box, a rough bird segmentation, and a set of attribute labels.

- Number of categories: 200

- Number of images: 11,788

- Annotations per image: 15 Part Locations, 312 Binary Attributes, 1 Bounding Box

You can find more description about the content in this dataset and how this dataset was created from this paper. In this document, we will focus on explaining the folder structure and files in the dataset.

Structure

images/

This folder contains all bird images. These images are organized in subdirectories based on species. In total, there are 200 subdirectories (one for each bird species). Check the Images and Class Labels section for more info.

parts/

This folder contains the 15 part locations per image. See PART LOCATIONS section below for more info.

attributes/

322 binary attribute labels from MTurk workers. See ATTRIBUTE LABELS section below for more info.

Images and Class Labels

List of image files (images.txt)

The list of image file names is contained in the file images.txt, with each line corresponding to one image:<image_id> <image_name>

Train/test split (train_test_split.txt)

The suggested train/test split is contained in the file train_test_split.txt, with each line corresponding to one image: <image_id> <is_training_image>.

List of class names (classes.txt)

The list of class names (bird species) is contained in the file classes.txt, with each line corresponding to one class: <class_id> <class_name>

Image class labels (image_class_labels.txt)

The ground truth class labels (bird species labels) for each image are contained in the file image_class_labels.txt, with each line corresponding to one image: <image_id> <class_id>

Bounding Boxes

Each image contains a single bounding box label. Bounding box labels are contained in the file bounding_boxes.txt, with each line corresponding to one image: <image_id> < x> < y> < width> < height>. <image_id> corresponds to the ID in images.txt, and < x>, < y>, < width>, and < height> are all measured in pixels. We won't use the bounding boxes in this homework.

Part Locations

We won't use part labels in this homework. But I still suggest you go over this information since the attributes are labeled w.r.t. different parts of a bird.

List of part names (parts/parts.txt)

The list of all part names is contained in the file parts/parts.txt, with each line corresponding to one part:<part_id> <part_name>. Check Figure 2 in the dataset paper for more description of the bird parts.

Part locations (parts/part_locs.txt)

The set of all ground truth part locations is contained in the file parts/part_locs.txt, with each line corresponding to the annotation of a particular part in a particular image:<image_id> <part_id> < x> < y> < visible>.

<image_id> and <part_id> correspond to the IDs in images.txt and parts/parts.txt, respectively. < x> and < y> denote the pixel location of the center of the part. < visible> is 0 if the part is not visible in the image and 1 otherwise.

MTurk part locations (parts/part_click_locs.txt)

A set of multiple part locations for each image and part, as perceived by multiple MTurk users is contained in parts/part_click_locs.txt, with each line corresponding to the annotation of a particular part in a particular image by a different MTurk worker: <image_id> <part_id> < x> < y> < visible> < time>.

<image_id>, <part_id>, < x>, < y> are in the same format as defined in parts/part_locs.txt, and <time> is the time in seconds spent by the MTurk worker.

Attribute Labels

List of attribute names (attributes/attributes.txt)

The list of all attribute names is contained in the file attributes/attributes.txt, with each line corresponding to one attribute: <attribute_id> <attribute_name>

List of certainty names (attributes/certainties.txt)

The list of all certainty names (used by workers to specify their certainty of an attribute response of is contained in the file attributes/certainties.txt, with each line corresponding to one certainty: <certainty_id> <certainty_name>

MTurk image attribute labels (attributes/image_attribute_labels.txt)

The set of attribute labels as perceived by MTurkers for each image is contained in the file attributes/image_attribute_labels.txt, with each line corresponding to one image/attribute/worker triplet:<image_id> <attribute_id> <is_present> <certainty_id> < time>

<image_id>, <attribute_id>, <certainty_id> correspond to the IDs in images.txt, attributes/attributes.txt, and attributes/certainties.txt respectively. <is_present> is 0 or 1 (1 denotes that the attribute is present). <time> denotes the time spent by the MTurker in seconds.

Class attribute labels (attributes/class_attribute_labels_continuous.txt)

Attributes on a per-class level--in a similar format to the Animals With Attributes dataset--are contained in attributes/class_attribute_labels_continuous.txt. The file contains 200 lines and 312 space-separated columns. Each line corresponds to one class (in the same order as classes.txt) and each column contains one real-valued number corresponding to one attribute (in the same order as attributes.txt). The number is the percentage of the time (between 0 and 100) that a human thinks that the attribute is present for a given class

Step 1. Train the concept model

In the first step, you need to build a CNN model to recognize concepts (i.e., the 312 binary attributes) in bird images. You can choose to train your own CNN model from scratch. You can also use a pre-trained model, such as Inception_v3, and then finetune it on the bird dataset. No matter which way you choose, please provide a clear description about your model and how you train it in your report.

Step 2. Train the concept-based bird species classifier

In the second step, you will train another neural network to predict the bird species given the concepts recognized by the concept model. You can use a simple MLP or any other model architecture you prefer. We will build an independent bottleneck pipeline in this homework. Therefore, you only need to train this classifier with ground-truth concept (attribute) labels as input, rather than the concepts recognized by the real concept model.

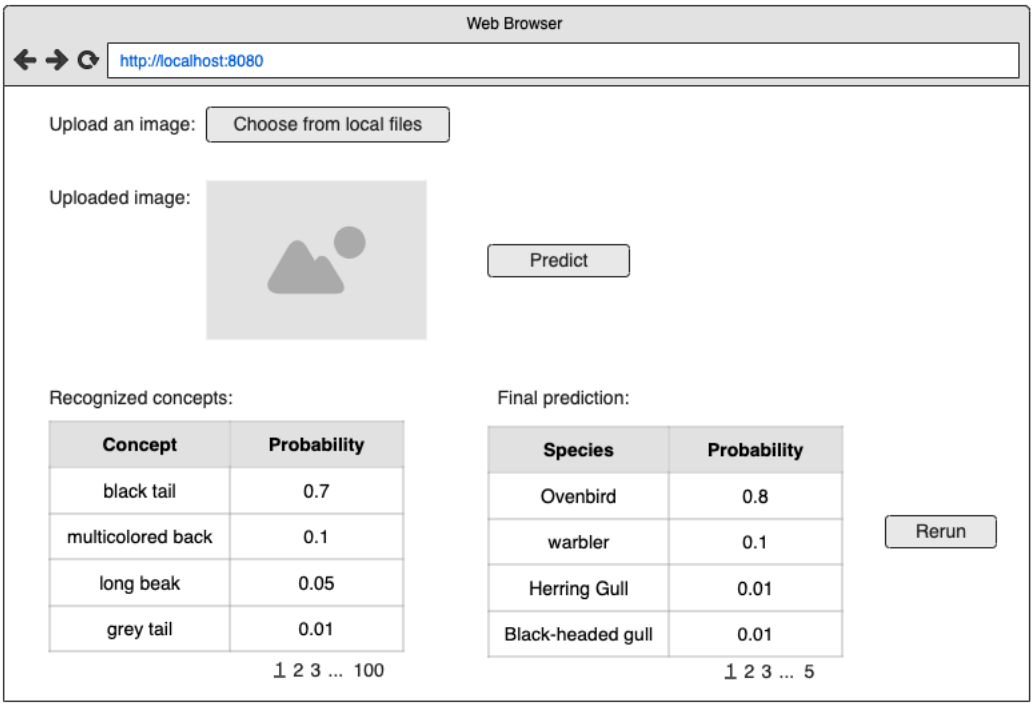
Check Section 3 in Koh et al. (ICML 2020) for the definition of independent bottleneck.

Step 3. Write an inference script and test

Now you have trained the two models. It's time to combine them to make an inference on a given image. Your inference script should take a bird image as input and then runs the concept model to detect the binary attributes in the image. The recognized concepts will then be fed into the classifier to make the final prediction. Finally, write a test script to measure the accuracy of your pipeline on the test set of the bird dataset.

Step 4. Develop the user interface for human intervention

As the final step, you need to build a simple web interface for your concept bottleneck pipeline. Please find a UI sketch below.



This interface should allow users to upload a bird image from local files and then render the uploaded image on the website. After the user clicks on the Predict button, the inference script should be triggered in the backend to recognize concepts in the image and make the final classification. Then, the recognized concepts and the final prediction should be rendered in two separate tables. The concept table should list the probability of recognized concepts in a descending order. Similarly, the final prediction table should list the probability of possible bird species in a descending order. Since there are 312 binary attributes and 200 bird species, which are too many to render in a flat table, your tables should support pagination (e.g., 10 items per page). The probability column of the concept table should be editable, so that users can intervene and update the probability. After the concept table is updated, the user can click the Rerun button to run the classifier to make the prediction again based on the updated concept table.

Bonus Points

The original concept bottleneck paper by Koh et al. (ICML 2020) also introduced two other types of bottleneck pipelines---sequential bottleneck and joint bottleneck. The only difference from independent bottleneck is the training method. You will earn 10 points if you train a sequential bottleneck and another 15 points if you train a joint bottleneck.

Submission Instruction

First, please create a PDF document to describe the model architecture and hyperparameters of your concept model and your classifier. Then describe how you train these two models. Please specify the library dependencies and the commands to run your training scripts, test scripts, inference scripts, and the web application. Please also include the accuracy of your concept model, classifier, and the entire pipeline on the test data. There is no page requirement on this PDF document.

Second, please make a demo video to demonstrate (1) how to use your UI to make a prediction on a bird image and (2) how to customize the concept recognition results and redo the prediction.

Finally, please put your source code, the PDF document, and the demo video into a zip file and upload it to BrightSpace. Don’t include your training and test data, since they are too large.