Re-Factoring Labels in Roboflow

* Regarding the labels, it seems that it would have to be manually done to refactor all the labels as they stand right now. We can refactor the labels when we generate datasets, but in the dataset itself as we view it, we would have to go through by hand to rename all the labels such as grapefruit.

Vegetables

* Cassava Leaves: 227 Unlabeled
* Plantains: 194 Cooked/151 Raw Unlabeled
* Yams: 178 Unlabeled
* Spinach: 184 Unlabeled
* Okra: 214 Unlabeled
* Onion: 183 Unlabeled
* The images are all crawled on my end; I have not labeled them yet, as I have been focusing mostly on the fruits. I should be able to have them labeled by the end of the week.

Cooked Foods

* A couple of concerns from my end regarding the cooked foods. I think the images are well-taken, but the performance of models on cooked foods may vary from food to food. For the fish, I think we will see much better performance since the fish all take similar shapes and are presented in similar ways. However, for the stews and meats, I think that there will be quite a lot of noise and it will be hard to train a good model for such foods.
* I also think that cooked foods are not region-specific but rather family specific, as every family may prepare foods differently? Not sure how accurate this is, but just a guess on my end.

YOLO v8 vs. YOLO v7 vs. YOLO v5/6

* While the versions have minor differences, I don’t believe they are significant enough to impact our project. I chose YOLOv7, as even though it’s not as recent as YOLOv8, there is more material out there for projects done with YOLOv7 than YOLOv8, which is too recent to have much work performed with it. I believe there are also minor performance advantages to using YOLOv7 from the papers that I have read on the topic.

Edge Computing/Thinking of the way a ML Model Works

* From my understanding, we have a low computational cost. However, I think this is easily avoidable just by the idea that we don’t have to store the model; we just have to keep the weights stored. We generate predictions using these weights, so if we are to store the weights on the front-end, we can generate predictions both quickly and at low costs.
* As we get more images, we can re-train our models and tune our weights. This can happen on local machines/local servers where we have a bit more overhead to work with. On the edge side, we can simply store the weights for quick and accurate predictions; we just need to update these weights when we retrain our models. I don’t think we should be training on-device, as a few images will not make any major difference in terms of model performance. We should simply keep the weights for predictions, and update these weights as we train our models.
* Idea for dataflow:

1. Collect train/test images in one place. How would we label these images?
2. Train the model on the dataset and track the best weights. We push these weights to a central API/Server (some format for storage and easy access) to be used for quick predictions.
3. Every 250/500/1000 images, we retrain our model to better tune our weights and push these new weights, generating better predictions as we get more images.

* The big idea here is that we **avoid training on-device**. Rather, we only **infer on-device**, and do everything else on the back-end.
* Additionally, do we need inference in real time? Since we are analyzing nutritional values and the nutrition present, would it be easier to upload the images, perform inferences on the back-end, and to gather the information on the back-end? Or are we using inference in real time to provide feedback on what the people in the DRC should focus on incorporating into their diet?