**Team.** Riya Parikh and Tomas Samuolis

**Elevator Pitch.**
We want to create a binary image classifier using a neural network which is robust against mislabeled training data.

**Context.**
We thought of this because of a problem that Tomas had during his internship during the summer where he was looking to classify whether some insurance claims should receive a certain service. While the positive class correctly identified all the claims which had received the service, the negative class contained all claims which had not received the service, INCLUDING claims that should have received the service, but had not been noticed, and thus did not receive it. This meant that the negative class had some number of claims which looked just like the positive class, but were labeled 'wrong.' This made modeling very difficult, particularly because there was no real idea of how many mislabeled claims there were. This project aims to begin to explore whether a solution to a problem like this is feasible by using image classification, as the models for basic image classification on its own are well established and work well. We want to see if we can make a model which can classify images even with some incorrect training data; more importantly, can it relabel/identify the errors in its own training data with itself? This makes sense to do at various percentages of misclassification.

**Methods. Explain what AI/ML methods you will try to tackle the problem. As opposed to the elevator pitch, here you should be specific about what you will try.**

a. We will use convolutional neural networks (built off of an existing model like VGG16) for image classification, testing what level of impurity in the training data can be persevered against to attain high levels of accuracy in the test set. We plan to deliberately inject label noise into a varying portion of the training dataset by randomly misclassifying a percentage of the images. This is meant to simulate real-world scenarios where some labels on data may be incorrect, whether due to human error or machine settings. One problem with using something specifically like VGG16 is that it may be so good that the mislabeled data is just not significant enough to actually affect it, so we may need to chop off more of the model or pick a weaker one to use.
b. Additionally, after training the CNN on the noisy dataset, we aim to implement self-training/ reinforcement learning methods to examine if the network can learn to correct itself over time. After an initial training, we hope that the model can reclassify its own training data (to be more accurate than it was given), and then retrain on its own predictions, as labels assigned to the noisy data points through the neural network will replace their prior labels.

c. One thing we may want to try is confidence thresholding such that the model only replaces the old data label with the new data label if its confidence in that new label exceeds a certain threshold.
d. Another idea that we have considered similar to confidence thresholding, is adding some sort of confidence score to each training example before the model is made. Some sort of unsupervised learning model like KMeans (or something more advanced) could try to first create its own labels for our categories, and if training labels do not match the unsupervised labels, the model may be less confident in the label.
e. In order to test our model's performance, we are going to keep track of assigned labels before/after a pass through the NN. We will see how many points change in their classification (with special detail to those going from correctly to incorrectly or incorrectly to correctly classified within each class). Moreover, we will also track accuracy of the models over the course of some iterations.

**Data source.**
Because the premise of our project is identifying mislabeled classes, any images which are labeled can work to explore the goal. At the moment, we are interested in examining image data that can be split into binary classes, with one class being a target, in the form of "X" and "Not X". In a real world application, a model like this would be implemented when the creators were not confident in the labels of their data, so any topic in which this would be a possibility would be more immediately applicable in the real world. However, in theory, you could do this by classifying pictures of cats and dogs with labels of "cat" and "not a cat" where some of the "not a cat" images are actually pictures of cats (if we used something more "random" like this, it would be more of an exploration and proof of concept rather than immediate application). Because we aren't as concerned with exactly what we are classifying, we would appreciate further guidance on possible datasets we could apply to our problem statement.
As a specific source, we found that Google's Open Images dataset may be relatively straightforward to implement: we could pick a target class (cats) and a non-target class (could be pictures of dogs, or could be pictures of a group of other animals that we just label as "not cats"). We aren't really sure how extensive this process will all be, so we don't know how complex to go in terms of the data.

**Code resources.**
We plan to utilize resources such as TensorFlow and Keras for building and training our CNN. Additionally, we may reference online tutorials and repositories that demonstrate techniques in self-training and confidence thresholding. As a side note, we will reference stack overflow and ChatGPT for debugging issues.

**What's new? Explain very clearly what part of the code will come from you instead of the code resources. In addition, mention any additional planned effort on your part that**

**we should consider when deciding whether the project does enough that's new - like gathering a new dataset, for example.**
The code that will come from us includes the implementation of the self-training mechanism and the confidence thresholding logic. We will also design and execute experiments to evaluate the effectiveness of our approach with various levels of misclassification. Furthermore, we will gather and preprocess our chosen dataset, ensuring that our work involves both data handling and model training. While the baseline architecture may be similar, the final goal of the project is about determining whether our model can identify the mislabeled training instances rather than just optimizing its own classification.

**Plan.**

a. Milestone 1 (October 31): Finalize the dataset selection and preprocessing steps. We will also complete the initial implementation of the CNN model and begin training it on the clean dataset to establish a baseline accuracy.
b. Milestone 2 (November 15): Implement the label noise injection, self-training method, and confidence thresholding. We will run initial tests and evaluate how the model handles the misclassified data.
c. Final Submission (December 9 or before): Complete the model evaluations, write the summary paper, and prepare for the lightning talk presentation. We will ensure that our experiments are well-documented, highlighting the impact of label noise and self-correction on model performance.

**Proposed Demonstration or Evaluation.**
To evaluate our project, we will conduct an experiment where we measure the model's performance at various percentages of misclassification (e.g., 0%, 10%, 20%, and 30%). We will compare the model's accuracy, precision, recall, and F1-score at each level of noise.

Experiments

a. Variation in Label Noise: We will systematically vary the percentage of mislabeled training data (e.g., 0%, 10%, 20%, and 30%) to observe how model accuracy changes with increasing noise levels.
b. Effect of Confidence Thresholding: We will evaluate the impact of different confidence thresholds on the relabeling process. We will test thresholds at various levels (e.g., 0.6, 0.7, 0.8) to determine how they influence the model's ability to correctly relabel and improve classification performance.

By conducting these experiments, we aim to assess the robustness of our neural network against mislabeled data and understand the effectiveness of our self-training approach.

---

Alternate data sources if Google Open Images doesn’t work:

Acute Lymphoblastic Leukemia (ALL) image dataset | Kaggle
Wildfire Prediction Dataset (Satellite Images)
Malaria Cell Images Dataset
Chest CT-Scan images Dataset