September 2021

# ESTABLISHING AN UNKNOWN CLASS FOR CLASSIFICATION ALGORITHMS BASED ON A LOCAL OR GLOBAL CONFIDENCE LEVEL THRESHOLD

HP INC

# Establishing an unknown class for classification algorithms based on a local or global confidence level threshold

## Abstract

A machine learning model is given for classifying images according to different aspects. Even though this model had a good accuracy on the test set, when it was deployed into production, we (the model developers) started to face many cases with failures due to input images that were not covered in our training or test sets. We've identified that these failures commonly have smaller confidence levels, and hence applying a threshold value to exclude low confidence predictions could work as a solution to avoid failures. In order to establish this threshold value, we explored different threshold values with the confidence scores returned by the model on different subsets of images. When below the defined threshold, the output is then considered *unknown* (i.e., not belonging to any of the established classes) and the SDK client will be able to define a specific behavior for these cases. Also, the threshold can be configured to be adaptive regarding each of the classes that the model was trained to output.

## Problems Solved

When a machine learning classification model fails, the user experience can be highly affected depending on the use case. For example, in an auto-orientation use case, where the output from the model is used to rotate the image, if the input image is already in the right orientation and the model fails to detect it, the output image can be shown in the wrong orientation, affecting the user experience and forcing the user to edit the image that was already in the right position.

Using the confidence level threshold returned by the classification model reduces the aforementioned problem by setting a boundary (i.e., a threshold value) where the model can perform well, enabling the user to select the action when the classification is *unknown* (i.e., below an established threshold value). Moreover, this value can be adjusted according to the available data and the user behavior to select the best threshold adaptively.

## Description

As aforementioned, adding an *unknown* class among the other ones allows us to avoid bad behaviors when the machine learning model is not confident enough over the predicted class. However, by adding this extra class, we can also lose some correct predictions below the defined threshold. To evaluate the influence of this addition, we split the outputs of the model into 4 possibilities:

- **Correct (corr):** predictions that are above the threshold value and matches the ground truth label;
- **Fail (fail):** predictions that are above the threshold value and do not match the ground truth label;
- **Unknown correct (unk_corr):** predictions that are below the threshold value and matches the ground truth label;
- **Unknown fail (unk_fail):** predictions that are below the threshold value and do not match the ground truth label.

Hence, we intend to define a threshold value that maximizes the *correct* and *unknown fail* predictions, while minimizing the *fail* and *unknown correct* ones. To quantify it, we defined herein the following metrics:

- **Precision**: is the ratio between the correct predictions over the sum of correct and failed predictions. It measures the accuracy of the model regarding the predictions that were not classified as *unknown*.
- **Recall**: is the ratio between the correct predictions and all the predictions (i.e., disregarding the threshold value). It measures the accuracy of the model regarding all predictions.
- **Responsivity:** is the mean between the **precision** and the **recall** value. This measures how well our model is responding to different inputs. High responsivity means that the model is predicting most of the inputs correctly with a score above the threshold; while low scores can mean both that our model is returning too many scores below the threshold value, or our model is wrongly classifying many inputs with scores above the threshold.

With these definitions, we proceeded to determine the unknown threshold value, $\tau_{unknow}$, in the following manner:

1. Evaluate the model in three distinct datasets: the testset, a QA dataset, and in a dataset containing not-supported image classes called *unsupported*;
2. Iterate over threshold values ranging from [0.5,1.0] with a 0.01 increase step;

3. Generate plots for the 4 possible prediction scenarios (i.e., correct, fail, unknown correct, and unknown fail) for all supported image categories in our datasets regarding one output (i.e., if a model predicts both the image class and orientation, for each image class, we plot the curves for the orientation outputs). Example of this plot is illustrated in Figure 1.
4. Based on our experiments, we get the best benchmark threshold value by analysing the intersection (crossing) between the different plotted curves using the following algorithm:

**PSEUDOCODE:**

```
$\tau_{unknow}$ → output
$\tau_{min}$ ← input
$\tau_{if}$ = where($u_f(\tau) = fail(\tau)$)
$\tau_{ic}$ = where($u_c(\tau) = fail(\tau)$)

if $\frac{\partial u_f(\tau\ )}{\partial \tau}|_{\tau=\tau_{if}} > \frac{\partial u_c(\tau\ )}{\partial \tau}|_{\tau=\tau_{ic}}$ then
>>$\tau_{unknow} = \tau_{if}$
else then
>>$\tau_{unknow} = \tau_{ic}$
end if

if $\tau_{unknow} < \tau_{min}$ then
>> $\tau_{unknow} = \tau_{min}$
end if
```

**PYTHON 3:**

```
th_unknown → output
thresholds ← input
unk_corr ← input
unk_fail ← input
fail ← input
th_min ←  input

import numpy as np

d_uc = np.diff(unk_corr)
d_uf = np.diff(unk_fail)

cross_uc = np.argwhere(np.diff(np.sign(\
np.subtract(unk_corr, fail)))).flatten()
cross_uf = np.argwhere(np.diff(np.sign(\
np.subtract(unk_fail, fail)))).flatten()

if d_uc[cross_uc[0]-1]>d_uf[cross_uf[0]-1]:
   th_unknown = thresholds[cross_uc[0]]
else:
   th_unknown = thresholds[cross_uf[0]]

if th_unknown<th_min:
   th_unknown = th_min
```

where $\tau$ are the evaluated thresholds, $u_f$ is the *unk_fail* curve, $u_c$ is the *unk_corr* curve, $\tau_{if}$ is the threshold value in which $u_f(\tau) = fail(\tau)$, $\tau_{ic}$ is the threshold value in which $u_c(\tau) = fail(\tau)$, and $\tau_{min}$ is a minimum threshold value. This minimum threshold value can vary depending on the application, and it is defined by the developers. For instance, it can be equal to 1 over the number of classes (e.g., if 10 classes, the minimum threshold value would be equal to 1/10 = 0,1).

5. *(Optional)* Add a margin of 5-10% to the determined threshold value;
6. If the number of classes is greater than 1:
   a. *(If global threshold)* Take the max value among the determined thresholds for the different possible classes;
   b. *(If local threshold)* Determine the unknown threshold value for each of the possible classes;

7. Verify with the aforementioned metrics (precision, recall and responsivity) how well the model behaves with the final(s) threshold(s) value(s) on the other two datasets.

From Figure 1, following the above steps, the best benchmark threshold value would be approximately 0,86. Figure 2 illustrates the derivative behavior of the *unknown correct* and *unknown fail* curves.

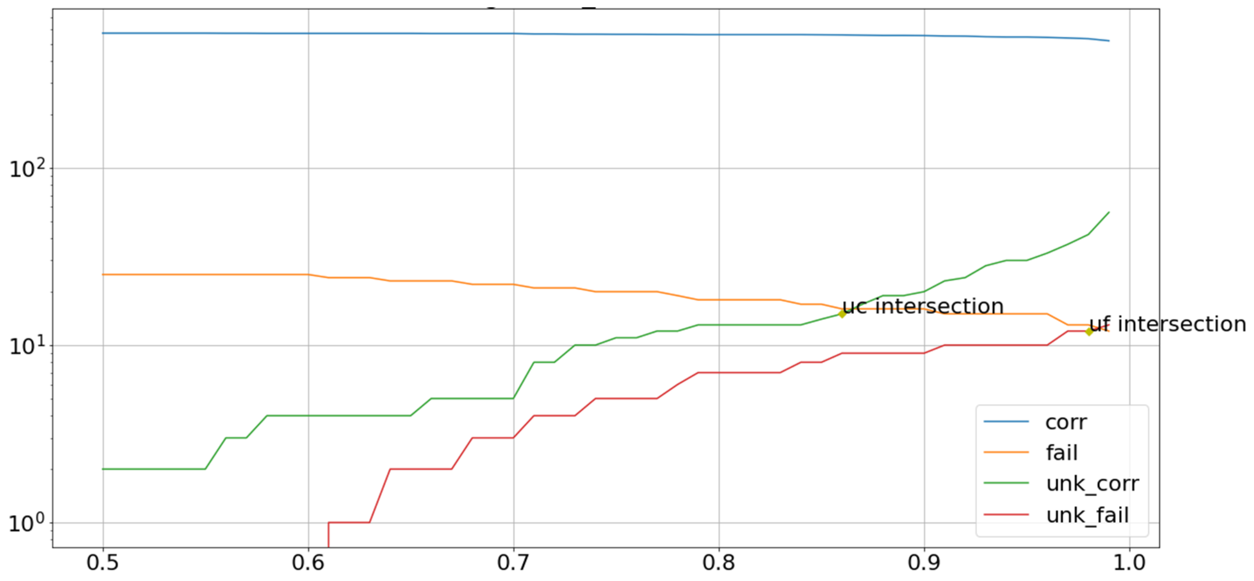

**Figure 1.** Example of plot for visualizing the impact of different threshold values on the 4 possible output scenarios. The *uc intersection* label is the point where the *unk_corr* and the *fail* curves intersect, while *uf intersection* is the point where *unk_fail* and the *fail* curves intersect.

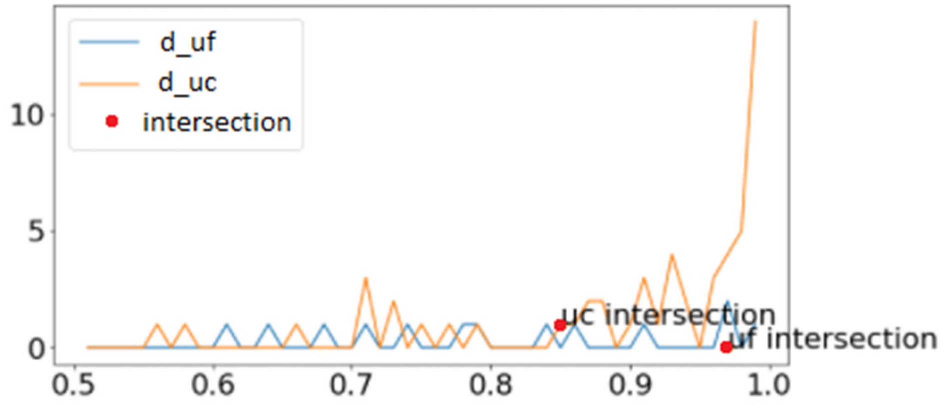

**Figure 2.** Derivatives from Figure 1 for *unk_fail* (d_uf) and *unk_corr* (d_uc) curves. The *uc intersection* label is the point where the *unk_corr* and the *fail* curves intersect, while *uf intersection* is the point where *unk_fail* and the *fail* curves intersect.

***Disclosed by Lucas Kirsten, Ricardo Ribani, João Melo, Jean Bainha, HP Inc.***