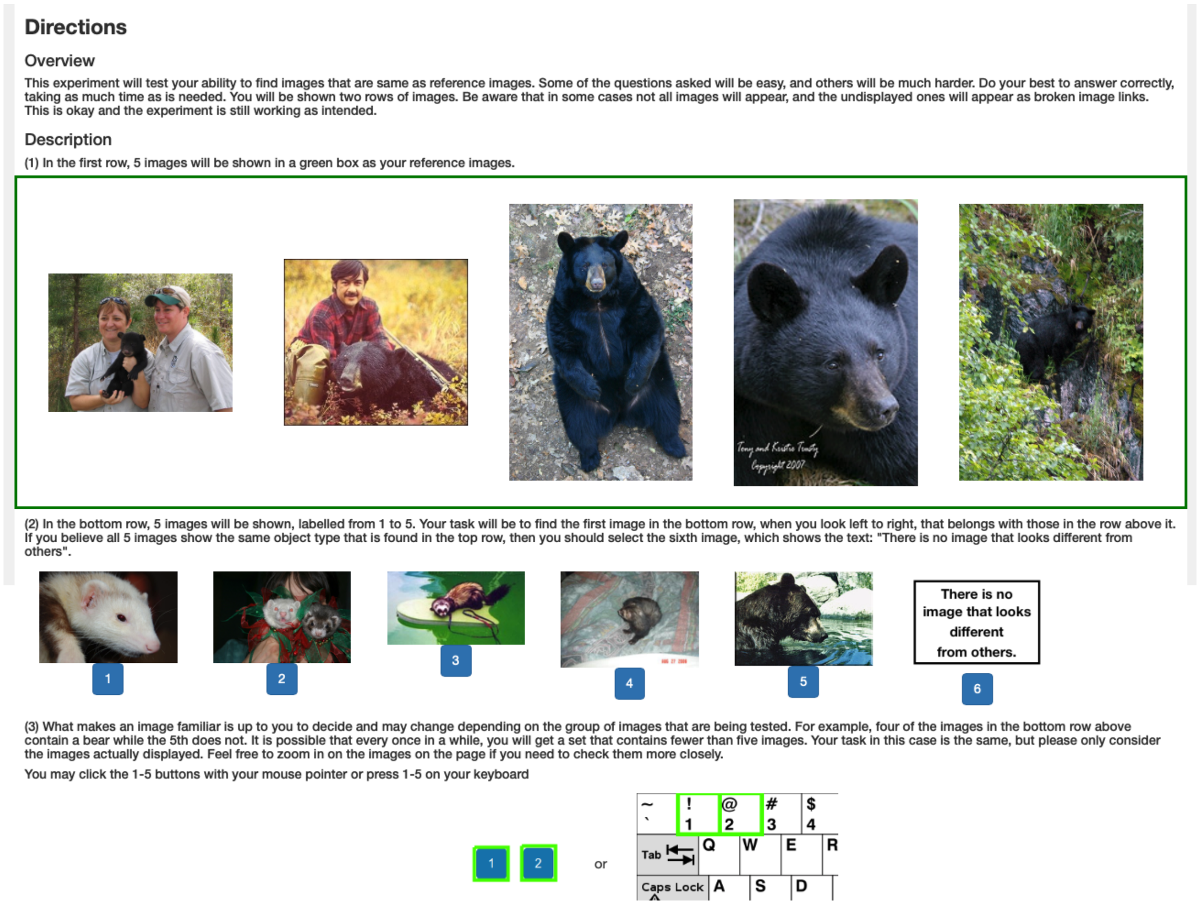
Supplemental Material: Measuring Human Perception to Improve Open Set Recognition

1 Additional Details For the Psychophysical Study

1.1 Data Collection Process on Amazon Mechanical Turk

Here we provide some additional details about our data collection process, including screen shots for key steps. The data collection application is very user-friendly. Built with Python and Flask in the back-end and distributed on Amazon Mechanical Turk, the survey can be accessed simply using a web link.



Directions

Overview

This experiment will test your ability to find images that are same as reference images. Some of the questions asked will be easy, and others will be much harder. Do your best to answer correctly, taking as much time as is needed. You will be shown two rows of images. Be aware that in some cases not all images will appear, and the undisplayed ones will appear as broken image links. This is okay and the experiment is still working as intended.

Description

(1) In the first row, 5 images will be shown in a green box as your reference images.

(2) In the bottom row, 5 images will be shown, labelled from 1 to 5. Your task will be to find the first image in the bottom row, when you look left to right, that belongs with those in the row above it. If you believe all 5 images show the same object type that is found in the top row, then you should select the sixth image, which shows the text: "There is no image that looks different from others".

(3) What makes an image familiar is up to you to decide and may change depending on the group of images that are being tested. For example, four of the images in the bottom row above contain a bear while the 5th does not. It is possible that every once in a while, you will get a set that contains fewer than five images. Your task in this case is the same, but please only consider the images actually displayed. Feel free to zoom in on the images on the page if you need to check them more closely.

You may click the 1-5 buttons with your mouse pointer or press 1-5 on your keyboard

Tony and Kristie Trusty Copyright 2007
1
2
3
4
5
There is no image that looks different from others.
6
1
2
or
~ ` ! 1 @ 2 # 3 $ 4
Tab Q W E R
Caps Lock A S D

Figure 1: Introductory page of each survey.

After a subject accesses the link, they first view an introductory page. This page includes following information:

1. Description, length and completion criteria. A brief overview about the survey, including the number of questions contained in each survey and what a subject is supposed to do to accomplish it, as well as what happens when a subject finishes answering all the questions.

2. Approval criteria and rejection criteria. In order to be approved, the subject needs to submit the survey code given at the end of a survey, as well as answer at least 3 control questions out of 5 correctly. A submission will be rejected if any of the following problems are present: (1) Failure to complete the survey. (2) Unable to provide a valid survey code. (3) Multiple submissions of an identical survey code. (4) 3 or more control questions are incorrect.

3. Troubleshooting guidelines. Information on what to do if an error shows up on the website is provided.

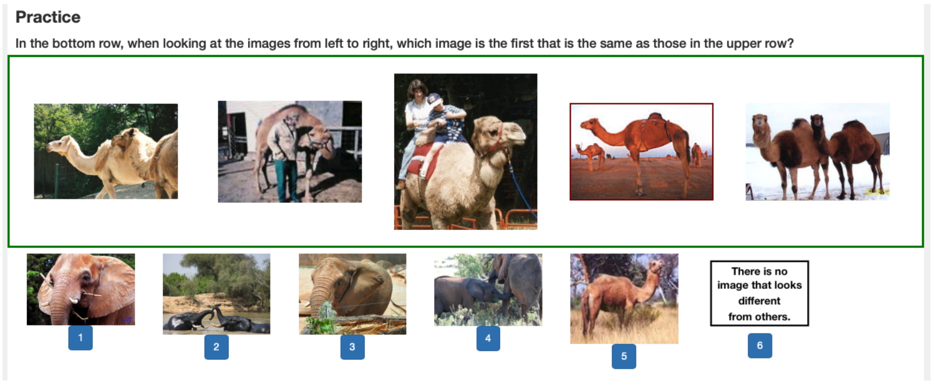

Practice
In the bottom row, when looking at the images from left to right, which image is the first that is the same as those in the upper row?
1
2
3
4
5
There is no image that looks different from others.
6

Figure 2: Practice question of a survey.

After the subject reads the introduction, the website proceeds to the consent page, where the subject is provided the information about the study, the purpose of the research, how long it takes to finish a survey, how to end a survey, how much we pay for a survey, and the privacy policy. The subject must check a box at the bottom of this page to agree to participate.

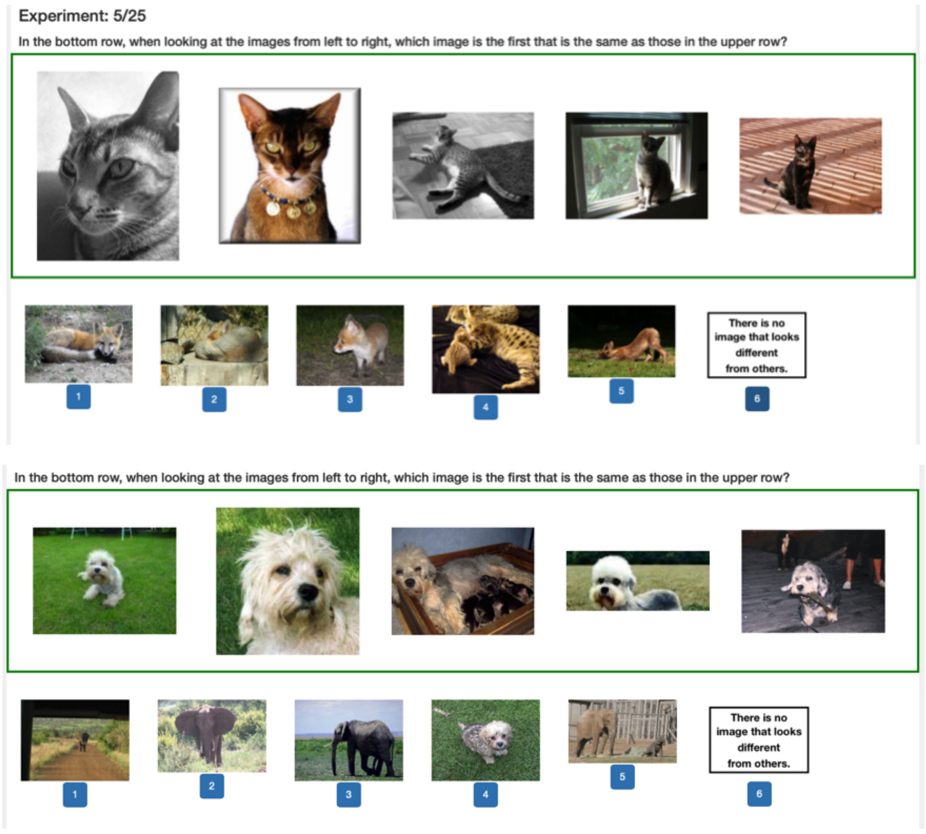
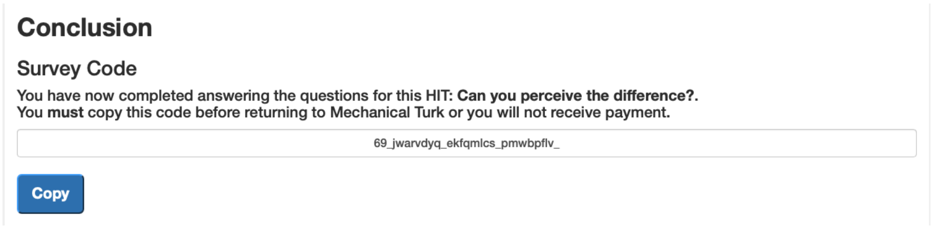

Experiment: 5/25
In the bottom row, when looking at the images from left to right, which image is the first that is the same as those in the upper row?
1
2
3
4
5
There is no image that looks different from others.
6
In the bottom row, when looking at the images from left to right, which image is the first that is the same as those in the upper row?
1
2
3
4
5
There is no image that looks different from others.
6

Figure 3: The different surveys vary in difficulty. The question at the top of this figure is more difficult because of the similarity between class cat and class fox; the question on the bottom of this figure is easier because of the large difference in visual appearance between dogs and elephants.

The survey then moves to the directions page as shown in Fig. 1. This page has an overview of the format of the questions that will appear in the survey, and it introduces the question format in detail: (1) showing the reference images in the top row in the green box. (2) showing the six options for selection in the bottom row. (3) giving direction on how to make a selection by either clicking on the buttons under an image or pressing a number key on a keyboard.

After the directions, the survey moves to a practice page that has a sample question (as shown in Fig. 2) to make sure that the subject understands how to answer the questions. The survey will only move forward to the real questions after this practice question is answered correctly.

For the actual data collection, 25 survey questions are presented to the subject, who will answer them as each question appears. Among all the questions, there are easier questions and harder questions, as shown in Fig. 3. After all of the questions are answered, a conclusion page will appear, and a survey code is provided for payment; a example of this is shown in Fig. 4



Conclusion

Survey Code

You have now completed answering the questions for this HIT: **Can you perceive the difference?.**
You **must** copy this code before returning to Mechanical Turk or you will not receive payment.

69_jwarvdyq_ekfqmlcs_pmwbpflv_

Figure 4: Conclusion page with the survey code and instruction on submitting the code for payment.

1.2 Class-Level Reaction Time Analysis

Here we provide all the box plots for the class-level RT distributions for the 40 known classes used in the behavioral experiments. To understand these plots, consider the first one. In Fig. 6, the first figure shows the case when class 4 (cat) is used as the host class (the reference class / known class) and paired with other classes, which means all RTs are collected for class 4 (cat) specifically. Although the maximum RTs when paring with different classes vary by quite a bit, we can tell from the plot that the minimums, lower quartiles, and medians are rather close among all the class pairings. This observation indicates that class level RT will not provide useful additional information for supervised machine learning training. The same is true of the other 39 classes shown below. To make the figures easier to read and understand, Table 1 shows the mapping between class index from ImageNet335 and the names of these classes.

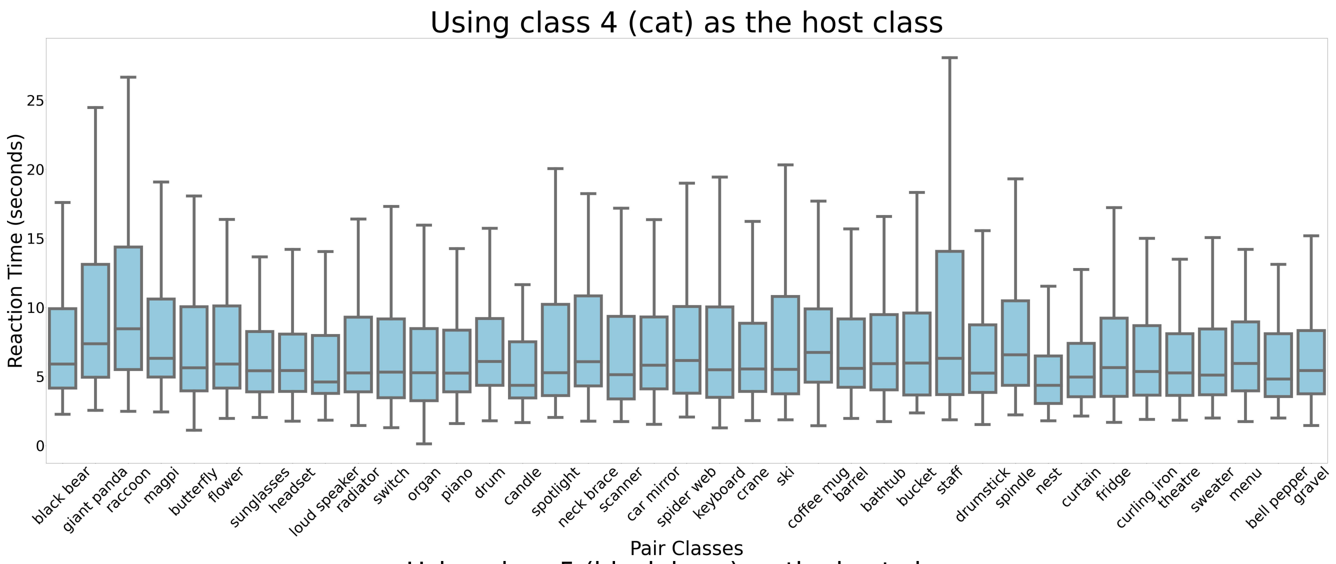
1.3 Image Level Reaction Time Analysis

Going deeper, we also look at the RT at the sample level. In our data collection process, each class is paired with 40 classes including itself, which means each image in the considered classes gets paired with multiple classes. The final RT used for each image is the average of all recorded RT measurements across all of the classes it is paired with.

Considering there are thousands of individual images that have an associated RT value, we are not able to show the RT variance for all classes for every single image. Here we randomly pick 4 images to show the image-level RT variance for each class it is paired with in Figure 19 and Figure 20. From these plots we can see that the RT varies a lot when a single image is paired with different classes. Thus we consider the difficulty for recognizing whether an image belongs to the reference class (the class that is shown in the first row in the green box in each question) or not depends on not only the intrinsic features

Label	Class Name	Label	Class Name	Label	Class Name	Label	Class Name	Label	Class Name
4	cat	5	black bear	10	giant panda	11	raccoon	55	magpie
73	butterfly	91	flower	108	sunglasses	114	headset	115	loud speaker
133	radiator	135	switch	141	organ	142	piano	144	drum
154	candle	155	spotlight	156	neck brace	162	scanner	163	car mirror
164	spider web	171	keyboard	172	crane	173	ski	202	coffee mug
205	barrel	206	bathtub	207	bucket	236	staff	237	drumstick
238	spindle	273	nest	274	curtain	314	fridge	315	curling iron
343	theatre	386	sweater	402	menu	403	bell pepper	410	gravel

Table 1: Label mapping for the class names of the 40 known classes.



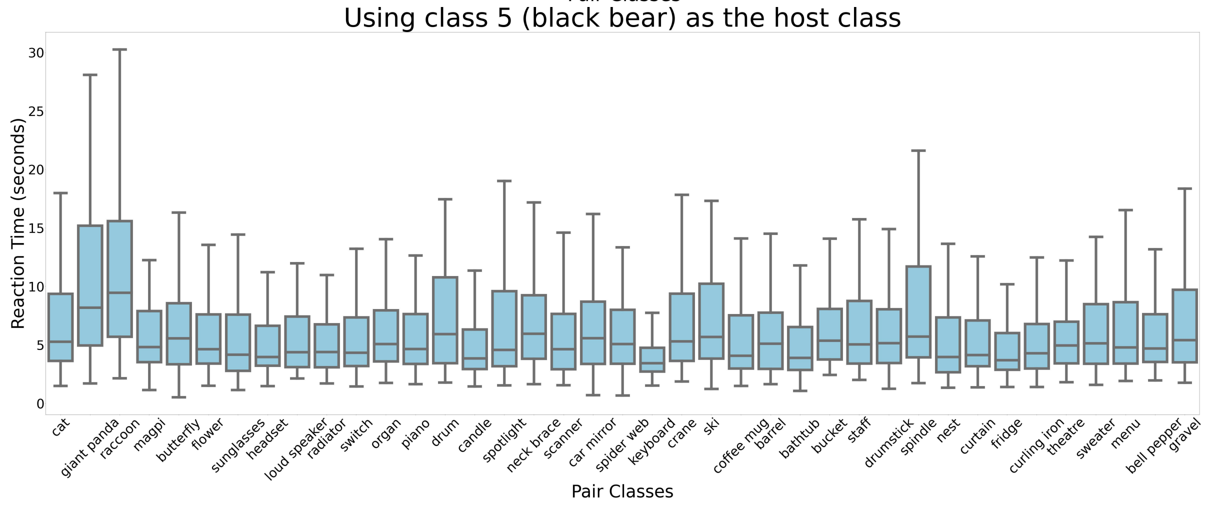


Figure 5: Using class 4 and class 5 as the host class

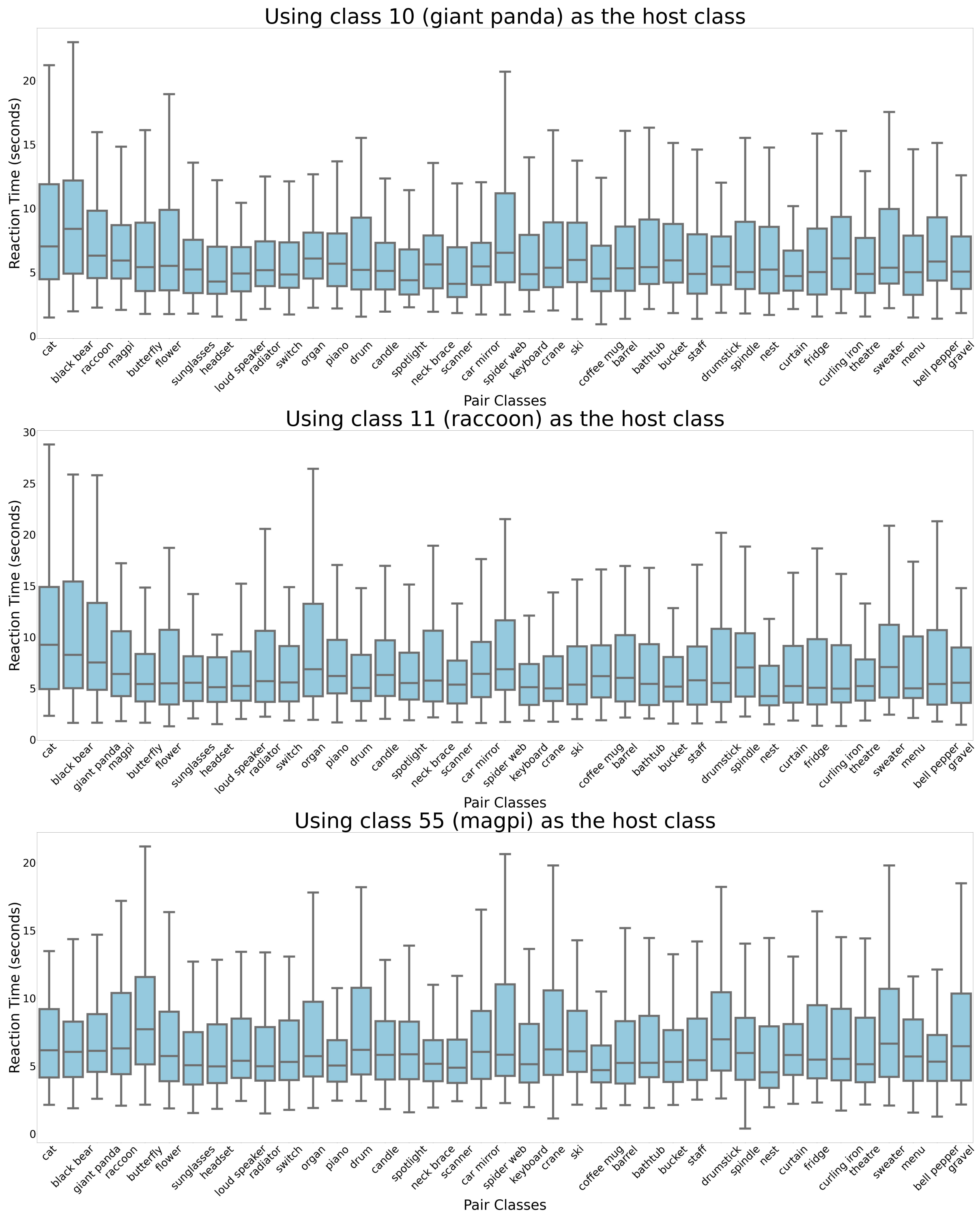


Figure 6: Using class 10, 11 and 55 as the host class

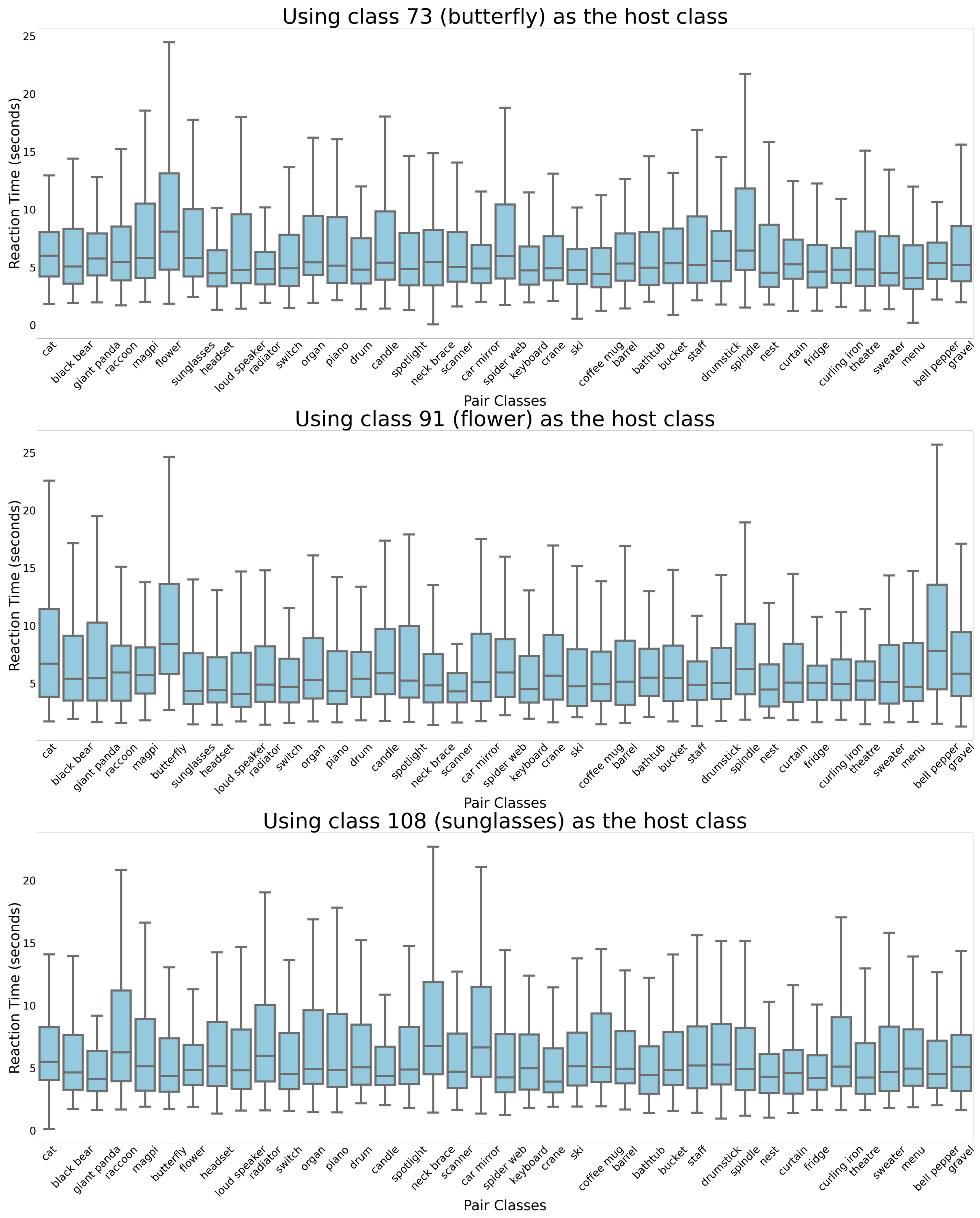


Figure 7: Using class 73, 91 and 108 as the host class

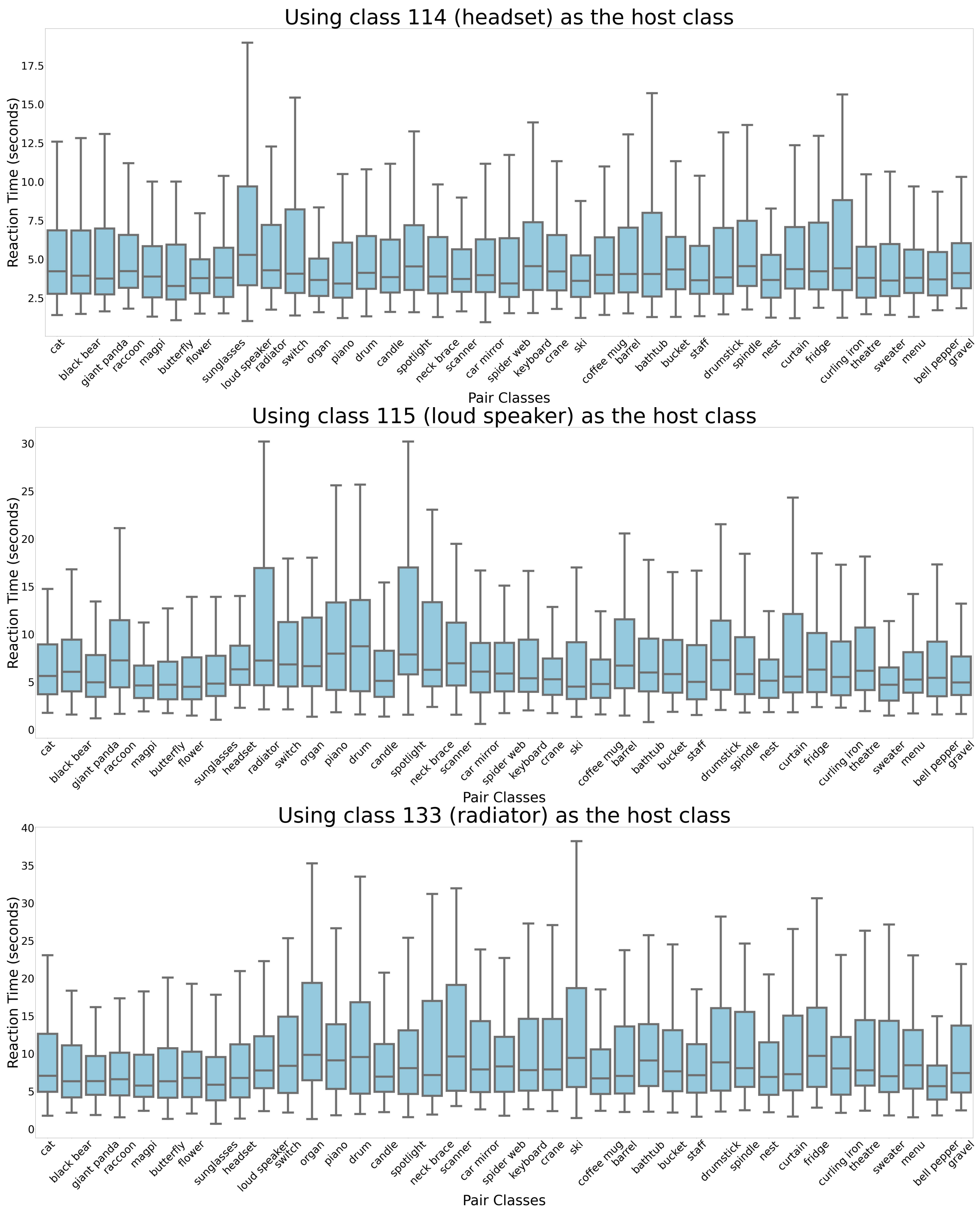


Figure 8: Using class 114, 115 and 133 as the host class

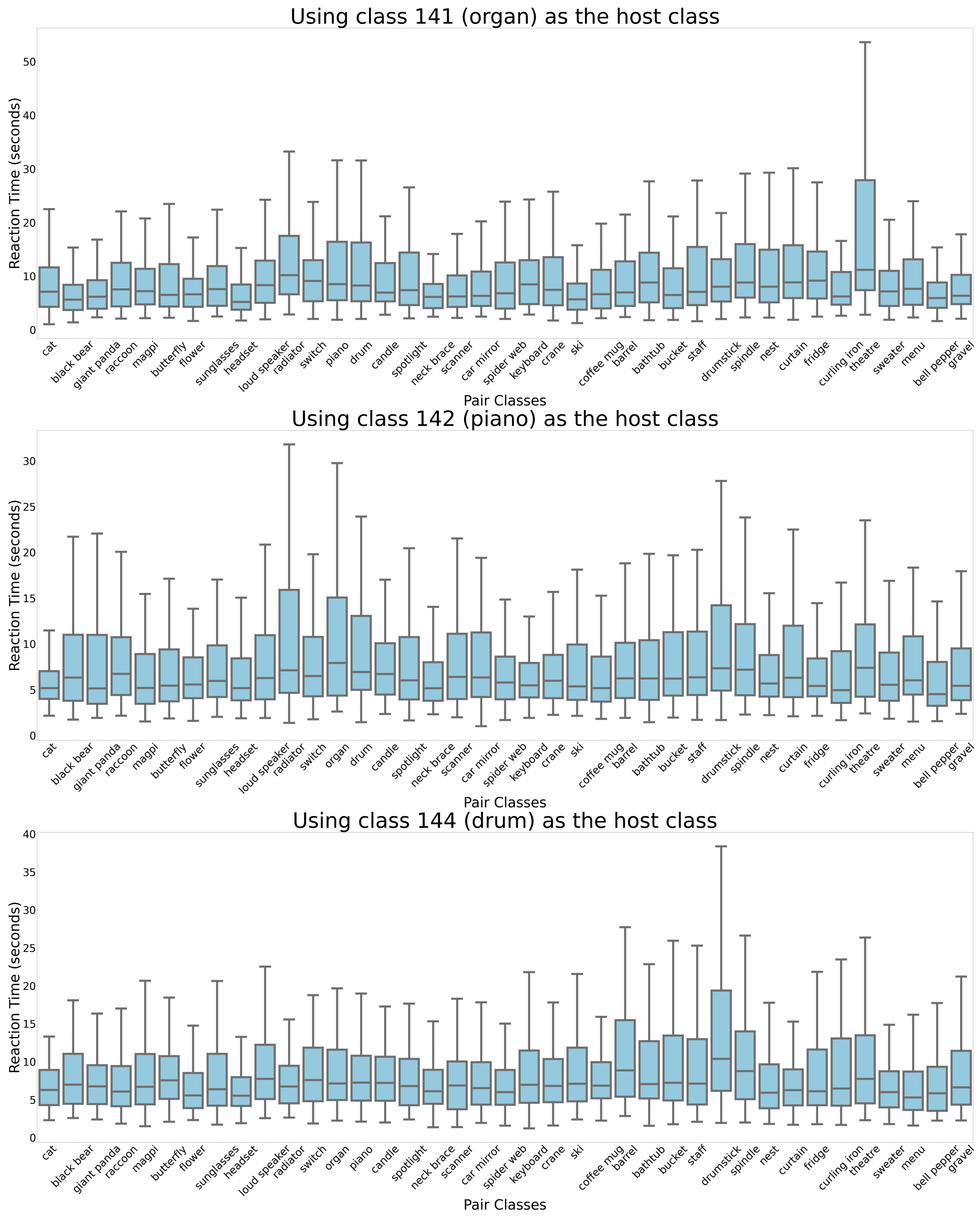
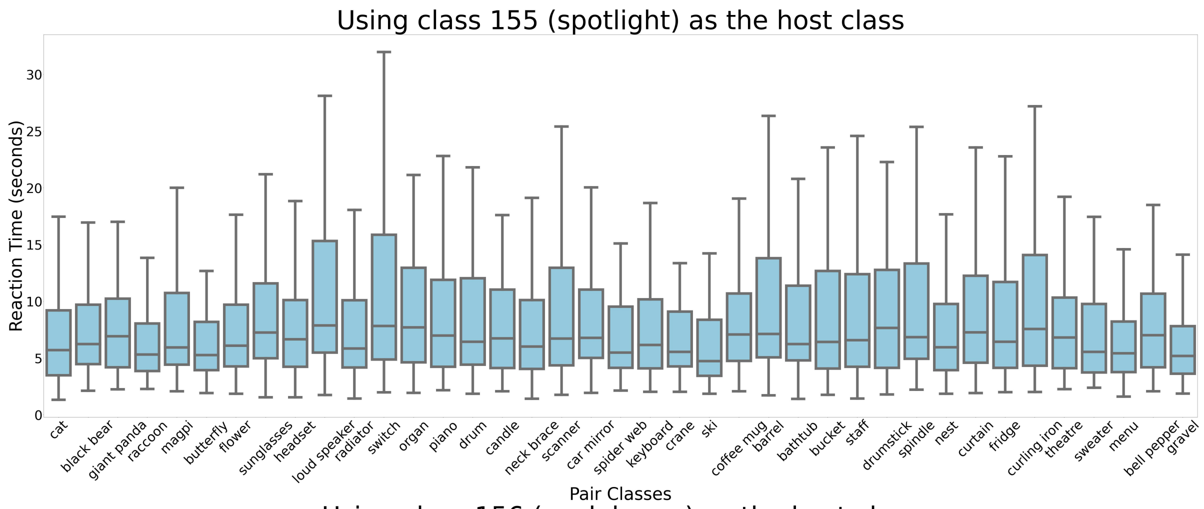
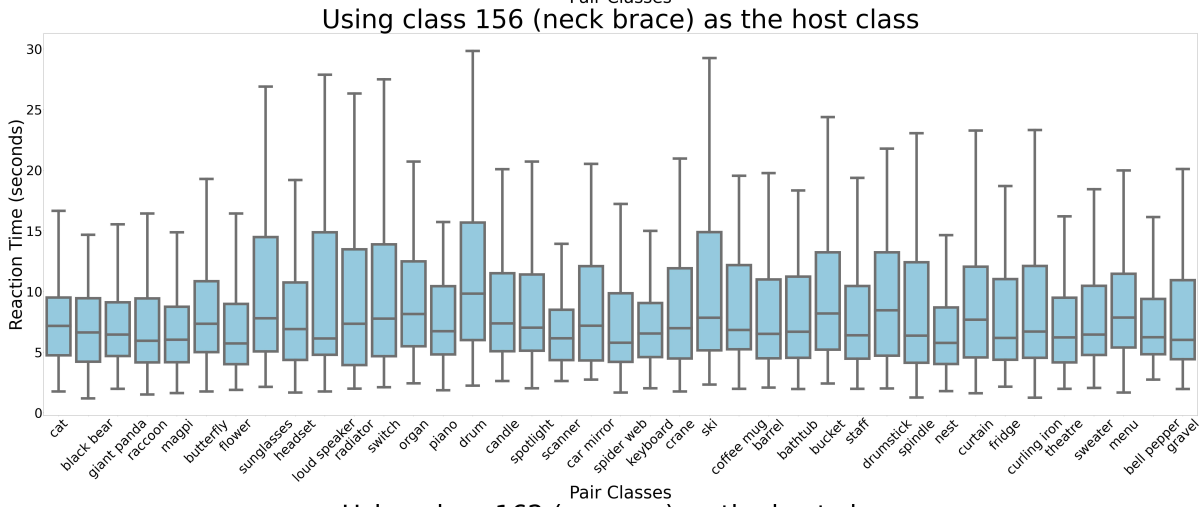


Figure 9: Using class 141, 142 and 144 as the host class





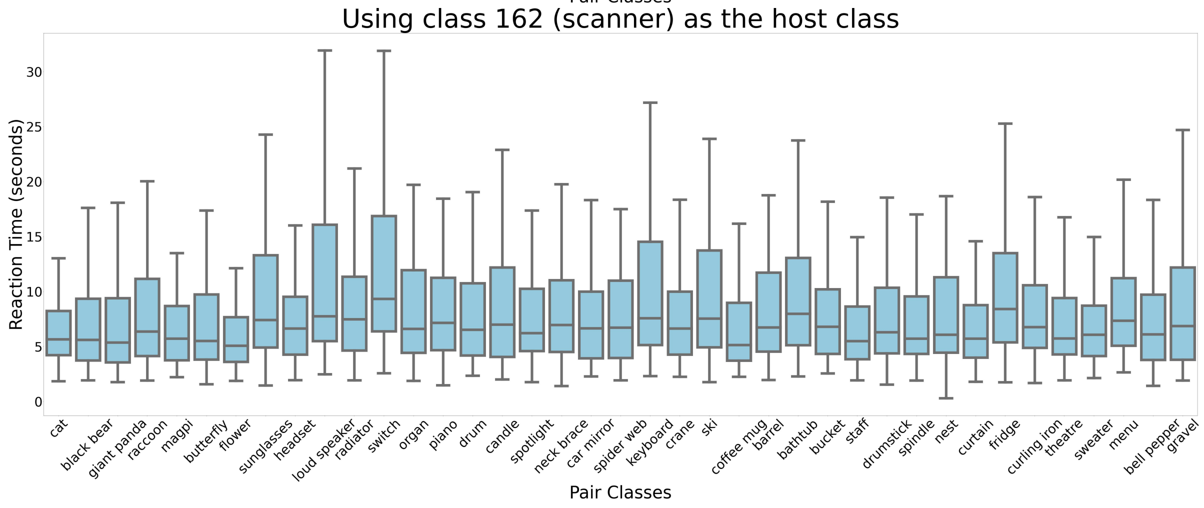
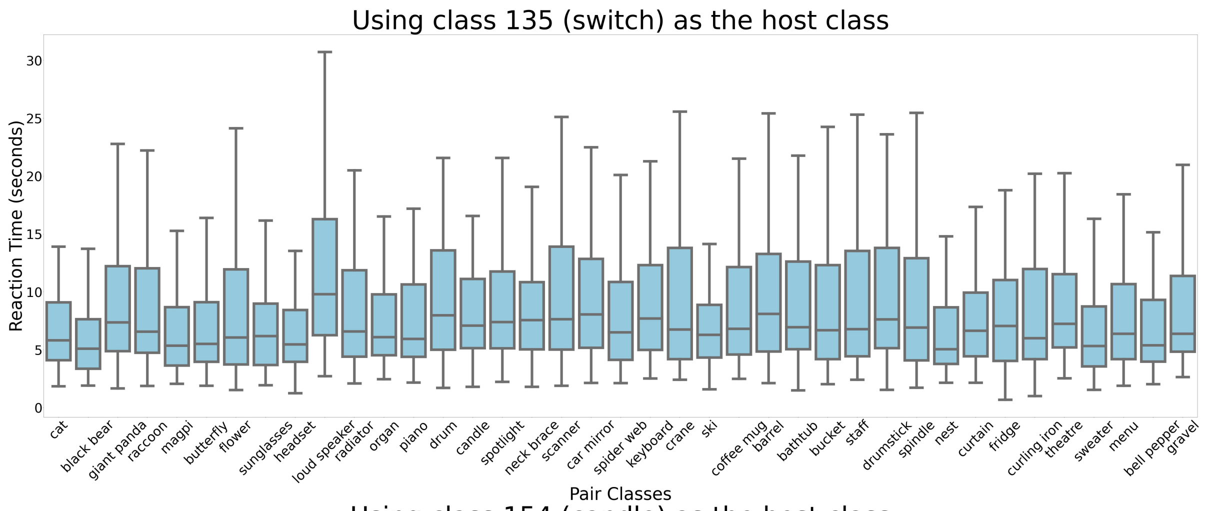
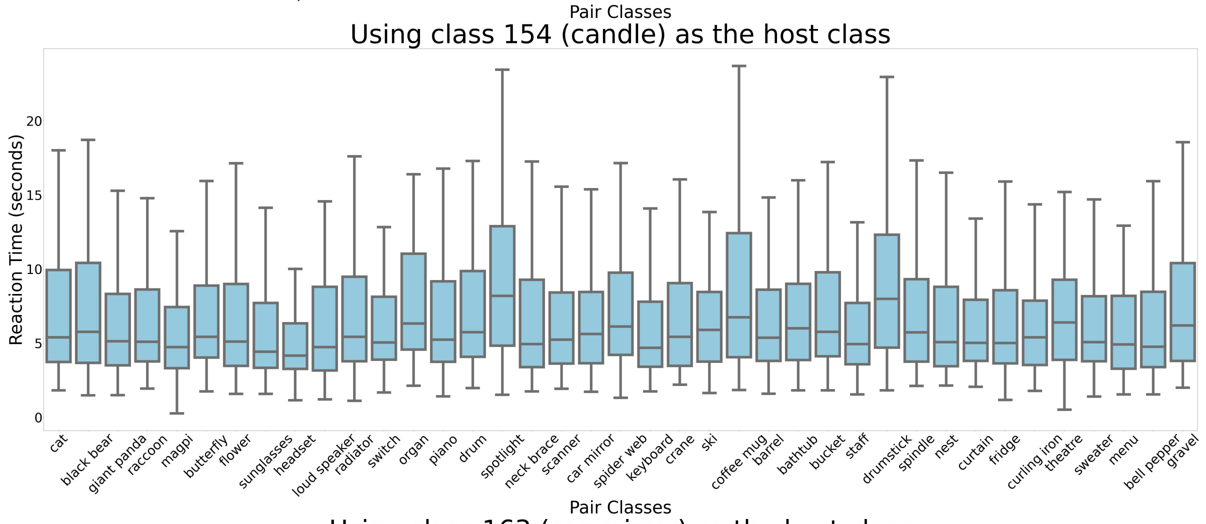


Figure 10: Using class 155, 156 and 162 as the host class





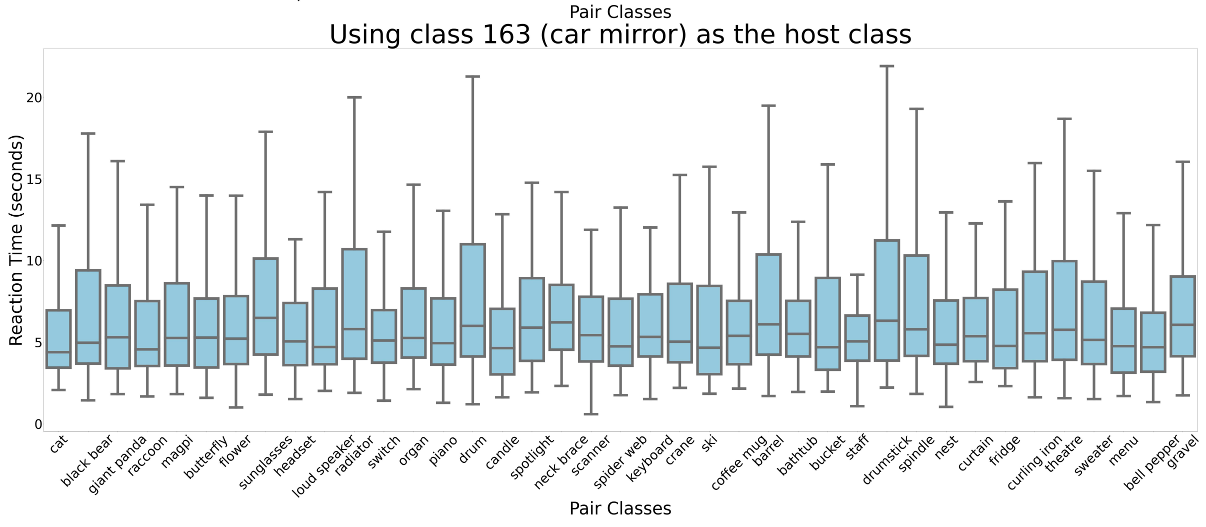


Figure 11: Using class 135, 154 and 163 as the host class

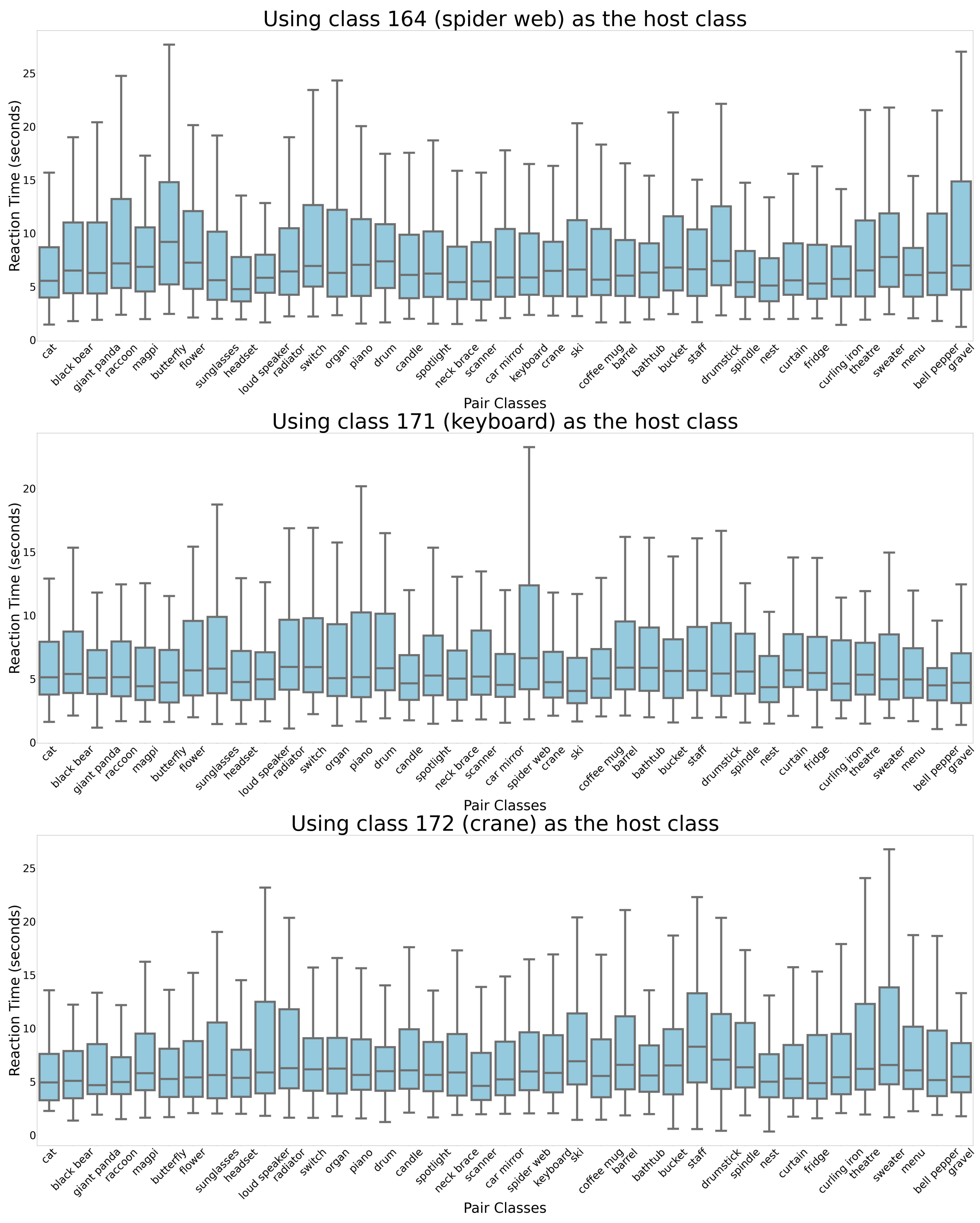


Figure 12: Using class 164, 171 and 172 as the host class

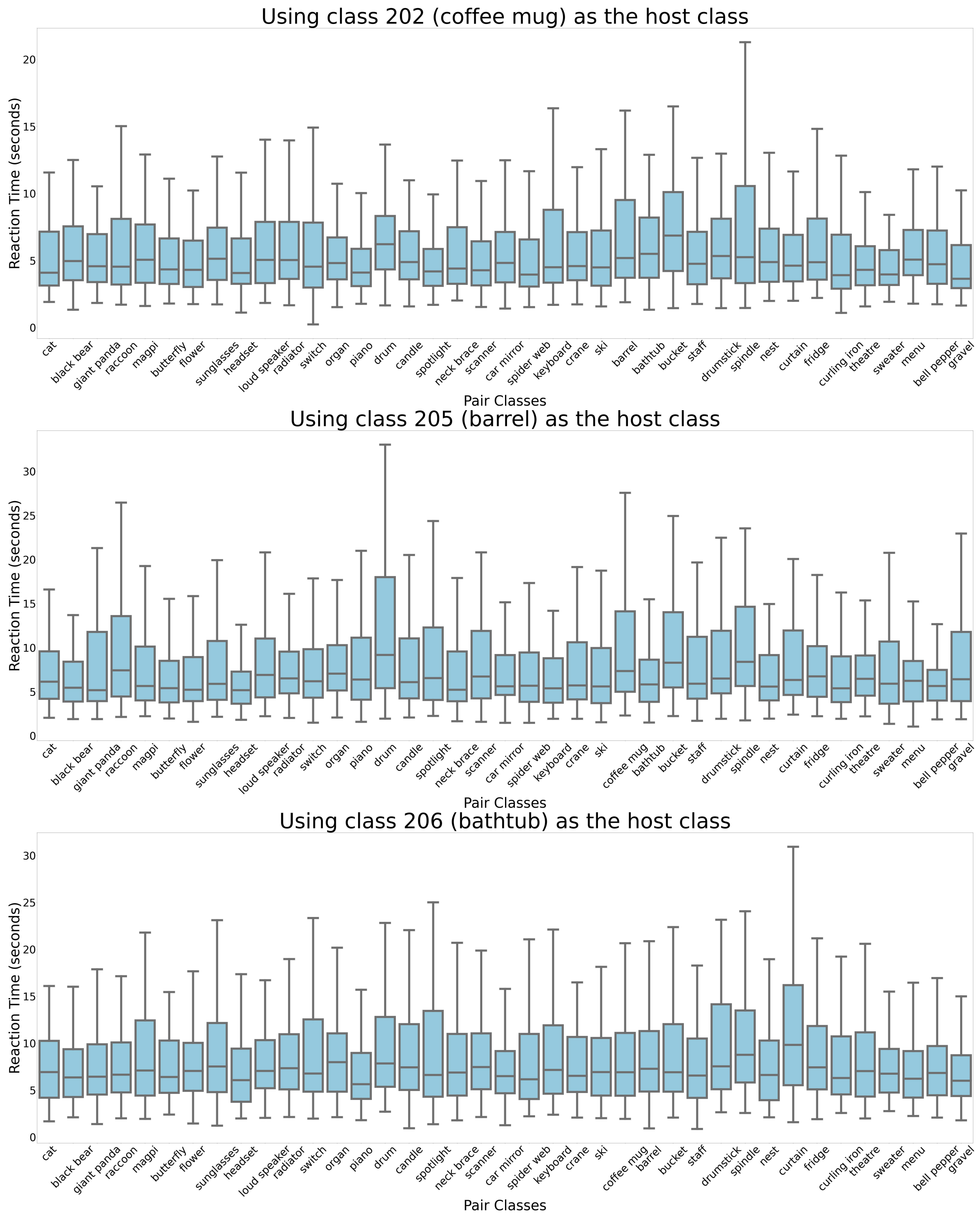


Figure 13: Using class 202, 205 and 206 as the host class

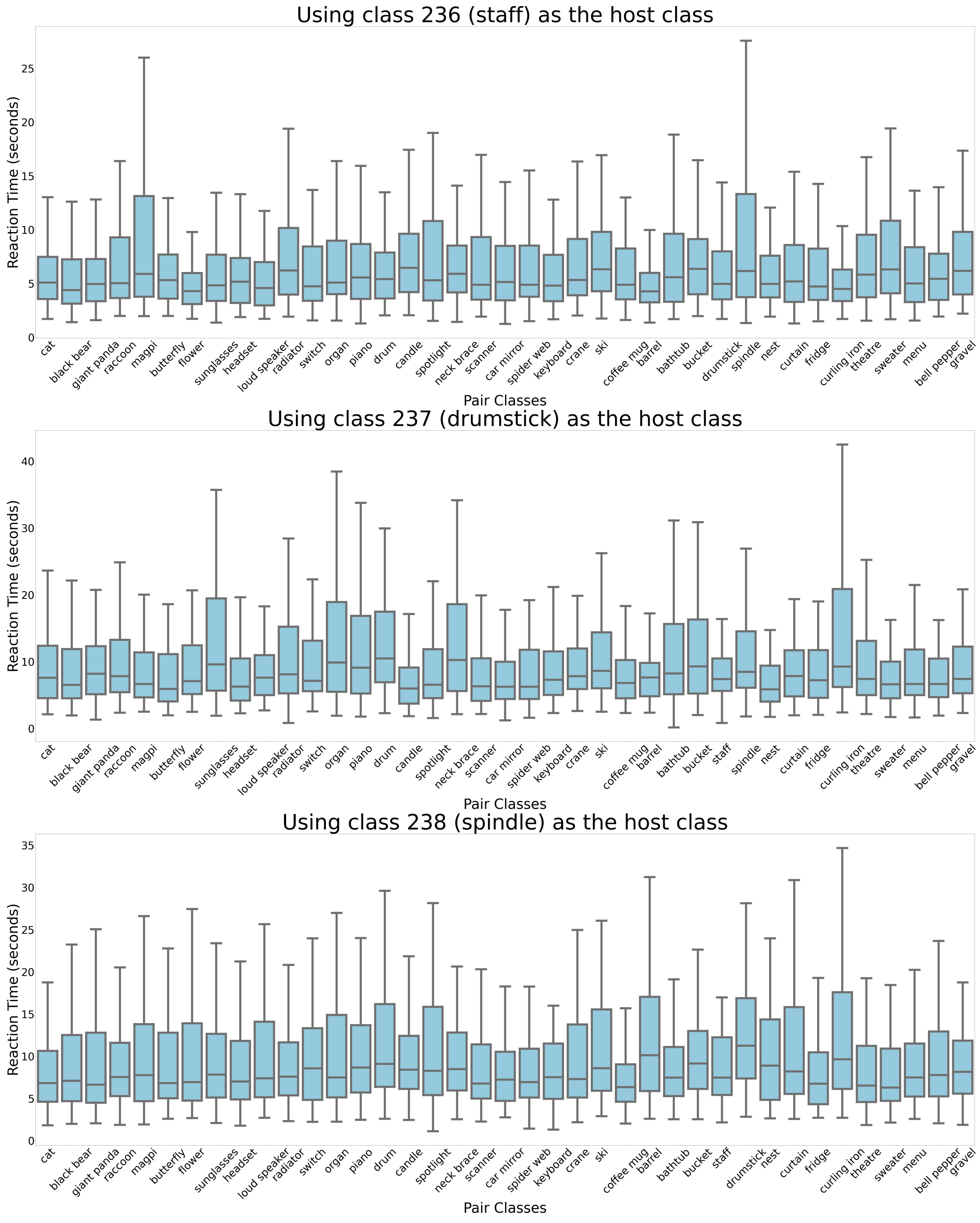


Figure 14: Using class 236, 237 and 238 as the host class

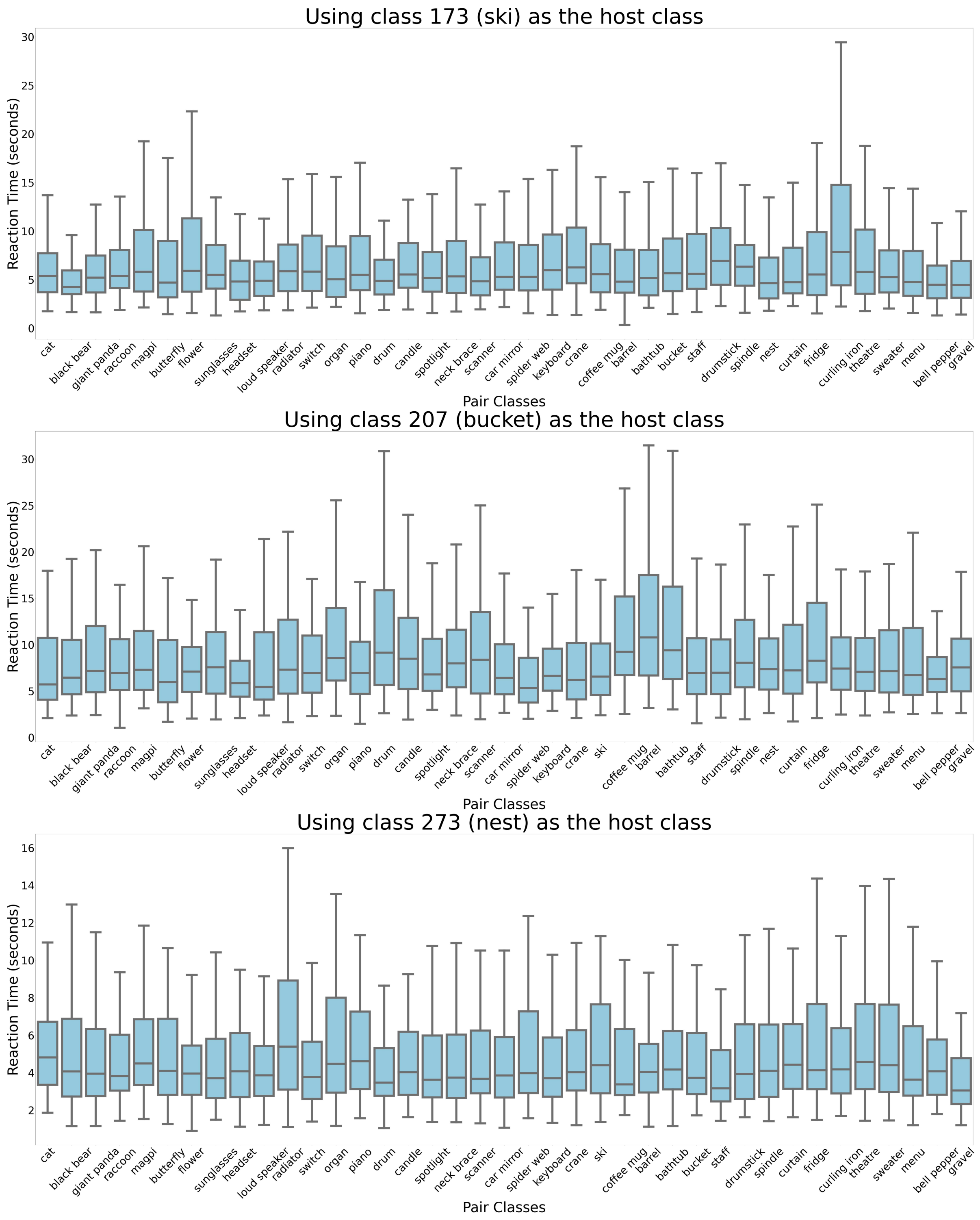


Figure 15: Using class 173, 207 and 273 as the host class

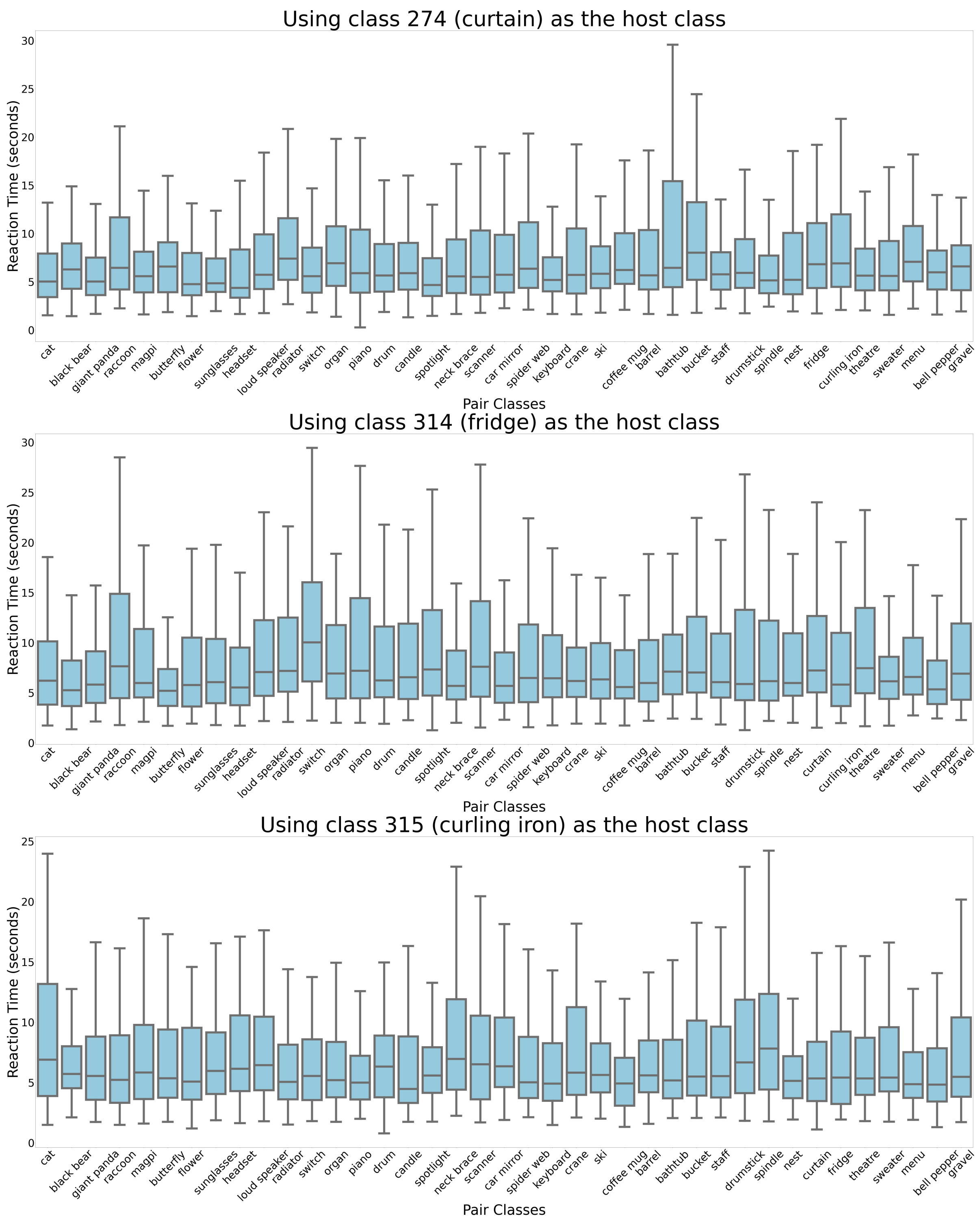


Figure 16: Using class 274, 314 and 315 as the host class

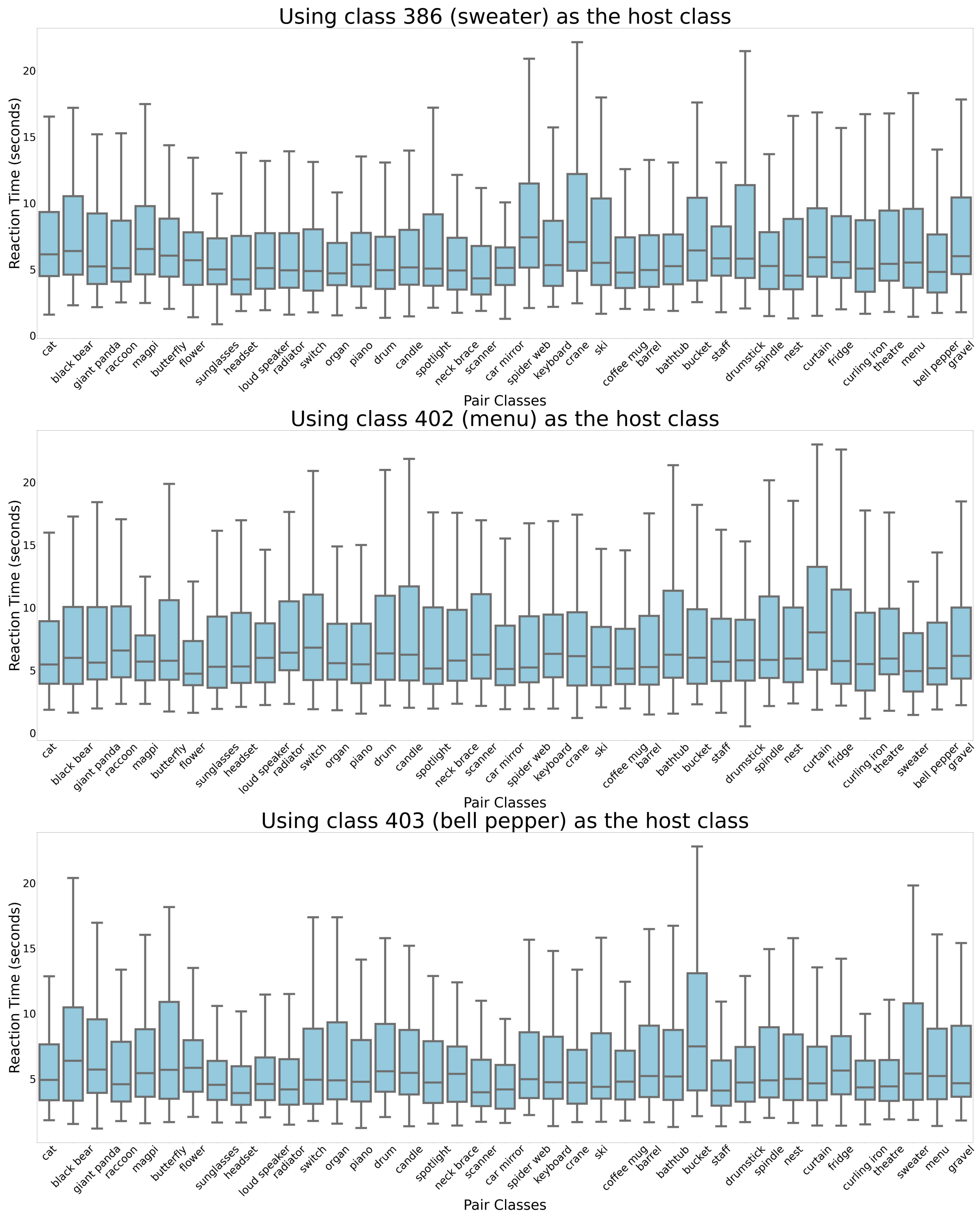
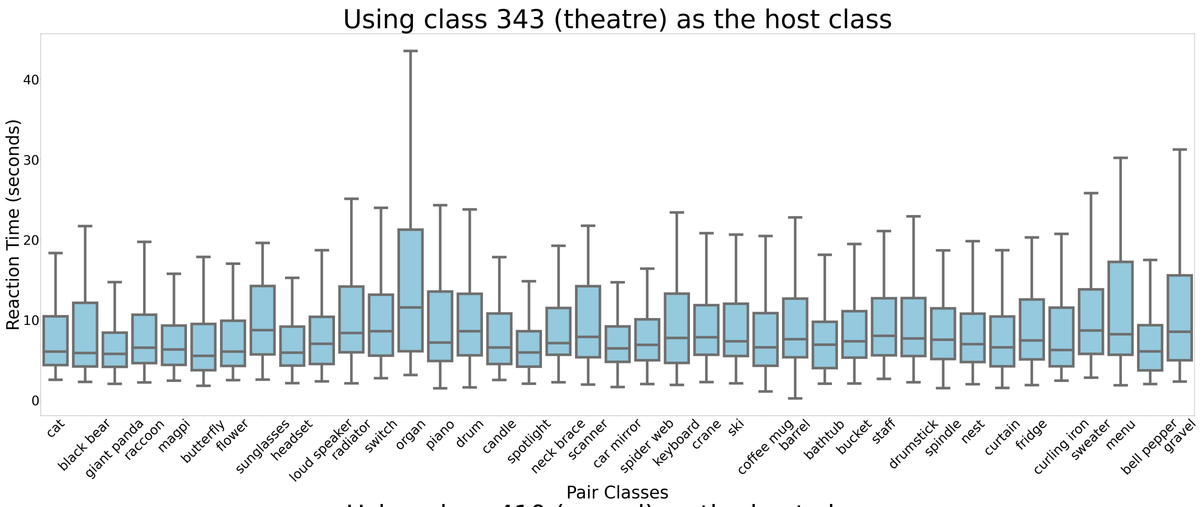


Figure 17: Using class 286, 402 and 403 as the host class



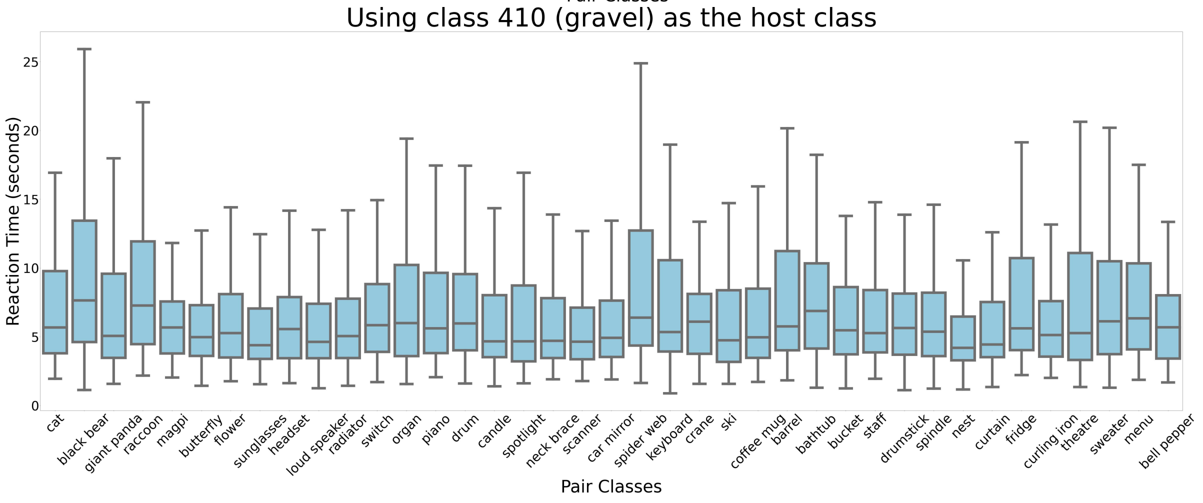
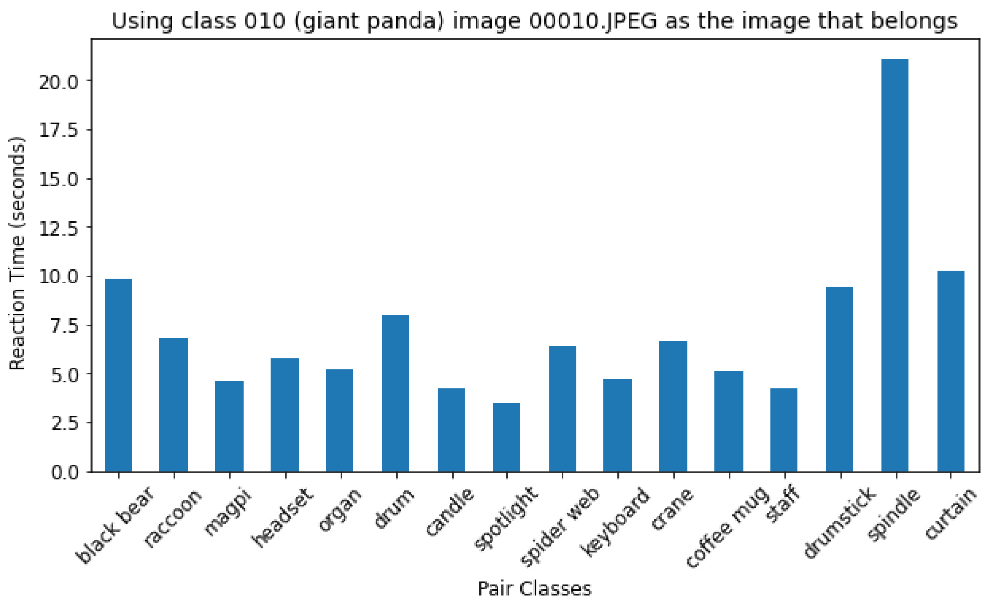


Figure 18: Using class 343 and 410 as the host class

of an image, but also what class it is paired with. This is significant for visual recognition in computer vision, because the training set captures those relative comparisons, and that is likely why these measurements are useful for supervised training.



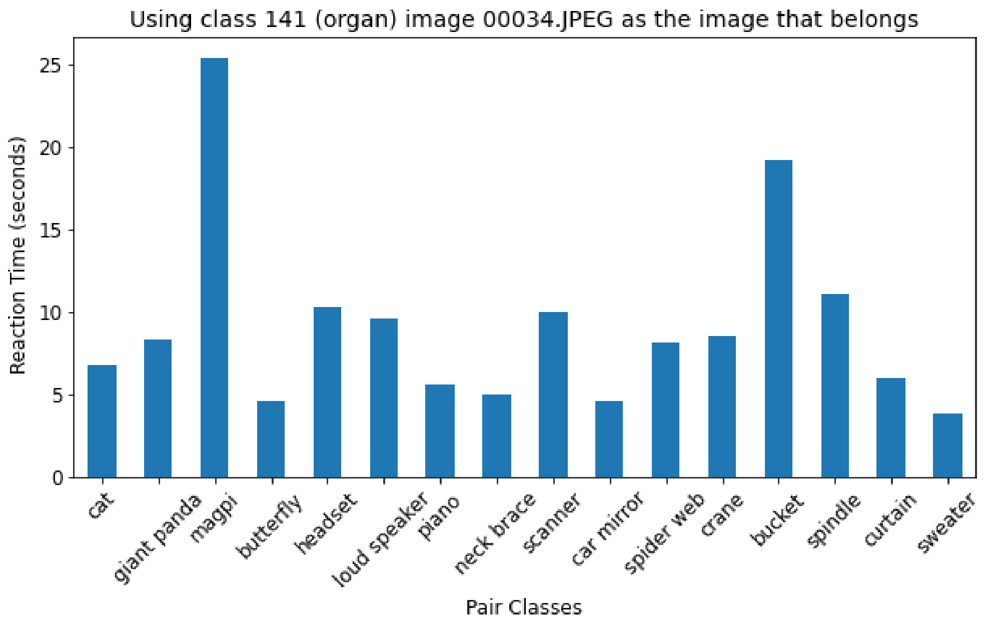
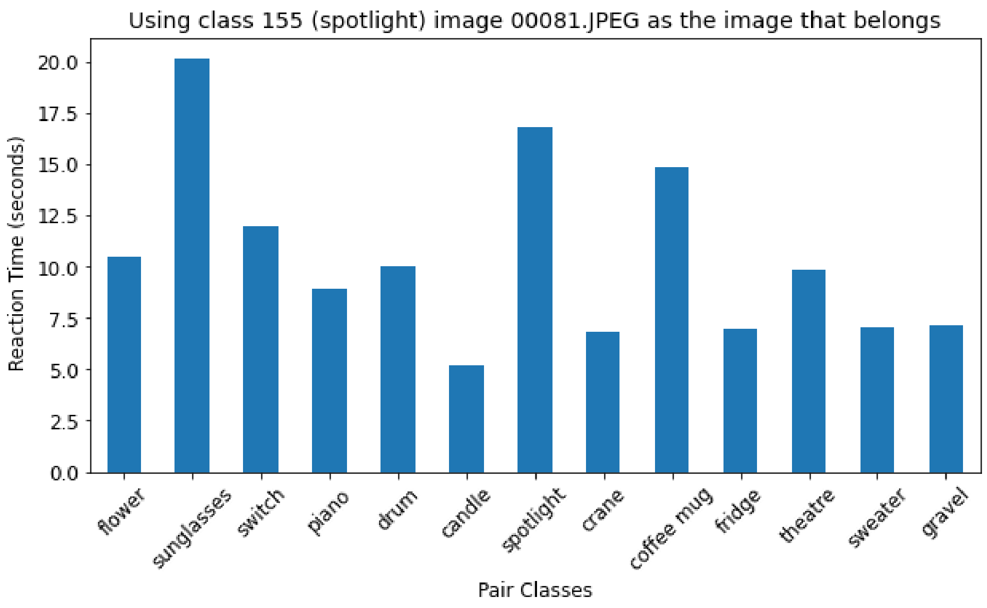


Figure 19: Average Reaction Times (RT) by class when collecting data on class 10 (giant panda) image 10 and class 141 (organ) image 34.



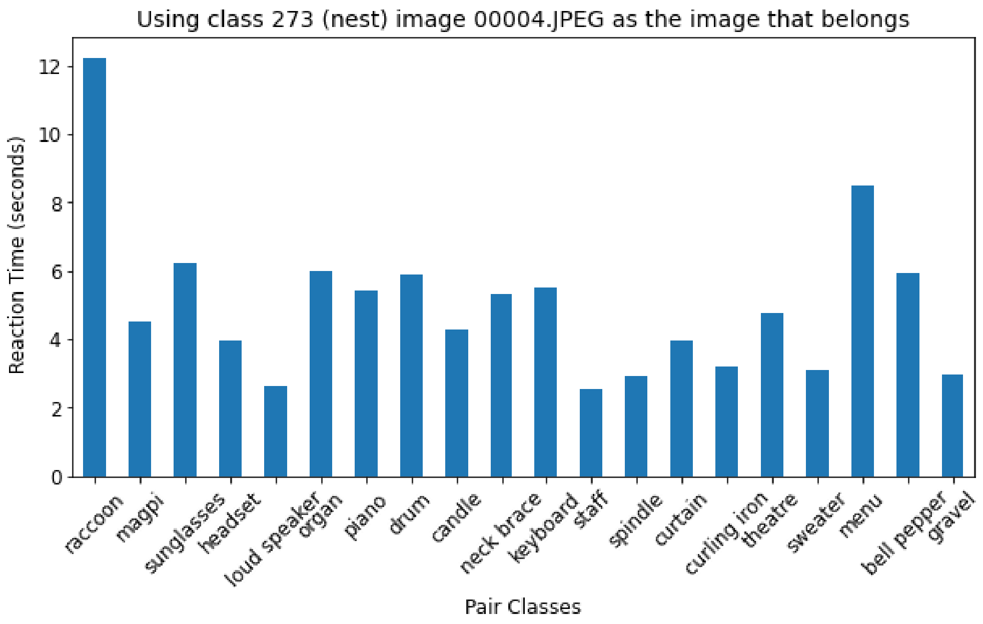


Figure 20: Average Reaction Times (RT) by class when collecting data on class 155 (spotlight) image 081 and class 273 (nest) image 004.

2 Descriptions of Baseline Approaches.

2.1 Standalone Classifiers

For the experiments involving standalone classifiers, we used the features that are generated by MSD-Net [1] when trained with the original Cross-Entropy loss on the ImageNet335 dataset. The SVM-based methods described below, as well as the EVM algorithm, were not originally designed to apply to feature sets of high dimensionality. When we use the original extracted features directly with these classifiers, the algorithms fail to converge due to the large number of dimensions. To address this issue, we utilized the Principal Component Analysis (PCA) method to reduce the size of features. To be specific, we first standardized features by removing the mean and scaling to unit variance using a standard scaler from scikit-learn (`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html`). We then applied PCA to the standardized features such that the reduced features represented 99% of the original features. For the PCA algorithm, we used the implementation in scikit-learn (`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.IncrementalPCA.html?highlight=incremental+pca`) with a batch size of 512. Hyperparameters are set to implementation defaults unless otherwise mentioned.

To train the SVM, W-SVM, P_I-SVM and EVM, we fit the reduced features to each algorithm and obtained trained models. We then sent known samples with labels and unknown samples into the models, and obtained prediction scores. Last, we applied thresholding to each score to decide whether a sample is known or unknown. The threshold was obtained by calculating the median of predictions scores when passing the validation set through a trained model.

SVM [2] maps training samples into points in space so as to maximize the margin between different classes. It has been used in many applications in computer vision including, but not limited to, image classification, handwritten character recognition, and face recognition. Here we use the SVM implemented in scikit-learn and we apply the SGDClassifier (`https://scikit-learn.org/stable/modules/sgd.html?highlight=sgd`) to our features for simplicity. To be specific, we use the linear kernel with a C parameter of 1.

Weibull-Calibrated SVM (W-SVM) [3] introduces a Compact Abating Probability (CAP) model to tackle the OSR problem. In the proposed CAP model, the probability of being associated with a class decreases in value when a point moves from known to unknown space. Based on the CAP model, the authors propose a novel Weibull-calibrated SVM (W-SVM) algorithm. The W-SVM utilizes the statistical Extreme Value Theory for score calibration with one-class and binary SVMs. The code that is used for the W-SVM is from `https://github.com/ljain2/libsvm-openset`.

P_I**-SVM** [4] utilizes the intuition that an algorithm should be able to reject a large portion of unknown samples during the testing phase if known samples are accurately modeled during the training phase without overfitting. Based on this intuition, the P_I-SVM models positive training data with the statistical Extreme Value Theory to determine decision boundaries, and it also uses a threshold-based strategy to generate classification results. The code that is used for the P_I-SVM is from `https://github.com/ljain2/libsvm-openset`.

EVM [5] is a theoretically sound open set classifier. It is inspired by the concept of support vectors in SVM, and is modeled by measuring the distribution of sample half-distances relative to a reference point. During the training phase, the EVM trains a one-vs-rest model for each known class; during the testing phase, it generates prediction scores for each testing sample. The code used for the EVM is from `https://github.com/prijatelj/vast/tree/massive_memory_use`. The EVM parameters used are as follows: tail size=1000, cover threshold=0.5, distance multiplier=1.0, distance metric=cosine and chunk size=200.

2.2 Deep Learning-Based Methods

Researchers have been exploring deep learning-bases methods for OSR problems for several years and have come up with different approaches. We compare our MSD-Net trained with the proposed psychophysical loss with the following methods:

OpenMax [6] is the first open set classifier that can be trained with a deep neural network. It utilizes the features produced by a deep network to compute a Mean Activation Vector (MAV) and the distances for MAVs. Then it trains a Weibull model using the MAVs and their corresponding distances. During the testing phase, the model produces a probability vector of length $n + 1$, where n is the number of known samples, and the last value in the vector indicates the probability for being an unknown sample.

To reproduce OpenMax using our dataset, we chose the model that had the best performance (evaluated by top-1 validation accuracy) while training MSD-Net with Cross-Entropy loss on the ImageNet335 dataset, and passed the training set and testing set (both known and unknown) to the model to obtain features for each sample. Then we followed the pipeline implemented in `https://github.com/abhijitbendale/OSDN` to calculate MAVs, distances of MAVs, and to train the Weibull model. Lastly we used the trained Weibull model to test all of our samples. The class that has the largest probability is considered to be the classification result.

OSRCI [7] generates samples that are close to training samples yet do not belong to any training class, using Generative Adversarial Networks. It trains a network that can classify these generated images into an extra class; ideally, when tested with unknown images, these unknown samples should be classified into the extra class.

We used the code publicly provided by the authors (`https://github.com/lwneal/counterfactual-open-set`) and followed the instructions step-by-step to apply this method to our ImageNet335 data. Parameter-wise, we randomly generated 50 batches of fake images, where each batch contained 64 images, which made 3,200 fake images in total. The network was trained for 200 epochs on a single GPU. After training, we evaluated the model with both known and unknown samples in the testing set, deciding what class a sample belonged to by looking at the class that had the largest prediction score.

CROSR [8] uses latent representations for reconstruction to enhance a model's ability to learn known features and separate them from unknown samples. The method consists of a closed set classifier and an unknown detector, where the closed set classifier directly uses the target labels, and the unknown detector uses a reconstructive latent representation together with target labels.

We utilized the code from `https://github.com/saketd403/CROSR` to reproduce this method with our ImageNet335 data. We train CROSR for 200 epochs on a single GPU with a batch size of 16 and image size of 32, with an initial learning rate of 0.05, a momentum of 0.9, and a weight decay of 0.0005. After training, we found the best model by looking at the top-1 validation accuracy, and used that model for testing. Similar to the original paper, we used thresholding on the probability scores we obtained from testing samples to separate known and unknown samples; but instead of using a naive threshold of 0.5, we used the median of the probability scores obtained by using the validation data.

CAC-OSR [9] is a distance-based loss that trains known classes to form clusters around anchored class centers in the feature space, which makes it easier to distinguish known samples and unknown samples. It consists of a closed set classifier (which can be any existing network), a non-trainable parameter C that represents a set of class center points, and a new layer designed for the new loss.

Following the code provided at `https://github.com/dimitymiller/cac-openset`, we reproduce this method with our ImageNet335 data. We trained the model for 200 epochs, with a batch size of 64, and image size of 64. After training, we found the best model by looking at the top-1 validation accuracy, and used that model for testing. We calculated the median of the probability scores gained by the validation data, and used it as a threshold on the probability scores we obtained from testing samples to separate known and unknown classes.

Psychophysical Performance Loss was introduced by Grieggs et al. [10] as a psychophysical loss formulation for training artificial neural networks. In that work, behavioral experiments were conducted to collect human reaction time data associated with the ability of reading in order to improve handwritten character recognition in historical documents. We call this loss the "performance loss" in our experiments, and it is defined as:

$$\mathcal{L}_P(x) = \frac{R_{\text{max}} - R_{\text{x}}}{R_{\text{max}}} \tag{1}$$

where R_{max} is the maximum human RT across all of the human training data and R_x is the mean human RT of the current image x. $\mathcal{L}_P$ is then the image's reaction time score normalized by R_{max} to be within the range $[0, 1]$. Based on the dataset described in Section III in the main paper, R_{max} was found to be 28 seconds.

This loss is the psychophysical loss can be combined with cross-entropy loss in a weighted summation as follows:

$$\mathcal{L} = \mathcal{L}_C + \lambda \mathcal{L}_P \tag{2}$$

where $\mathcal{L}_C$ is the cross entropy loss and λ is the weight parameter for performance loss.

The performance loss leverages the information of the expected human reaction time versus the maximum RT to inform the model of the sample's difficulty for human classification. This relies on the assumptions that human RT is longer for images that are more difficult to classify by the humans and that this difficulty should be shared between humans and the model.

3 Additional Detail on Experimental Setups and Supplemental Results

3.1 Additional Detail on Reaction Time and Exit Strategy

Table 2 shows the cut-off thresholds that map bins of RT measurements to the MSD-Net exit indices, mentioned in Section IV of the main paper. The maximum is obtained by finding the largest value of RT after removing the outliers.

Threshold Percent Below	QU1 20%	QU2 40%	QU3 60%	QU4 80%	Max 100%
Reaction Time	3.5720	4.9740	7.0156	11.601	27.572
Exit Index	0	1	2	3	4

Table 2: The cut-off thresholds for human RT for known samples. This table maps RT bins to the MSD-Net exit indices. The minimum is set to 0, and the other 5 thresholds are the quintiles (QU). All the values for reaction time are in seconds.

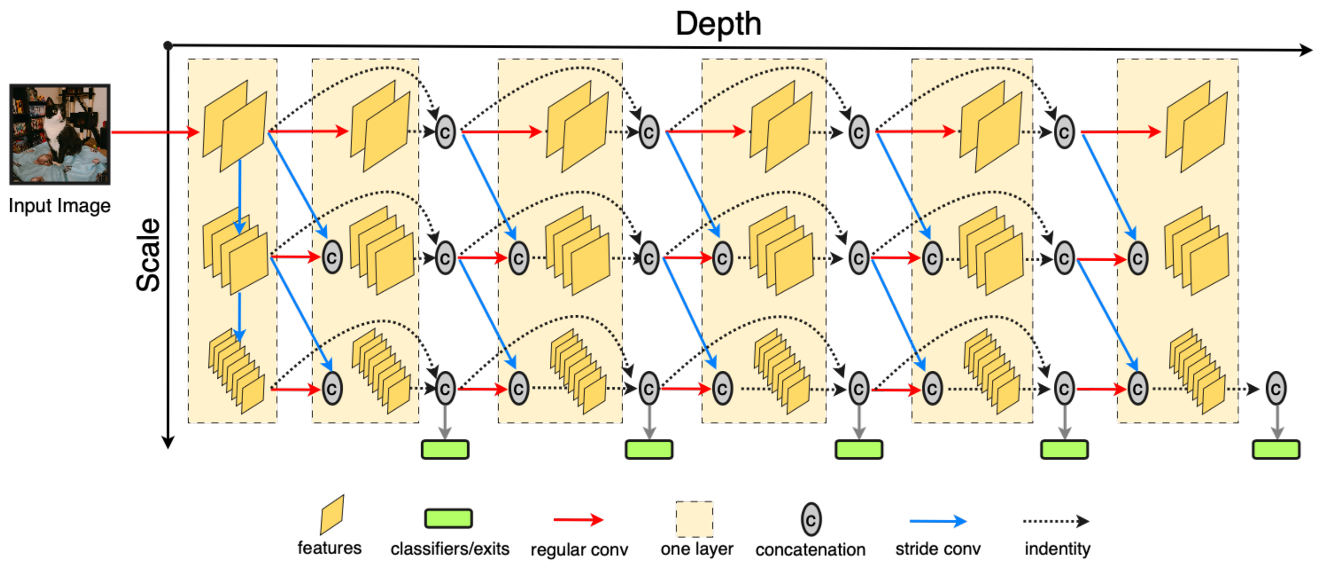


Figure 21: Network architecture of MSD-Net [1] with 5 classifiers/exits.

Figure 21 shows the MSD-Net that has 5 classifiers/exits, and as mentioned in the main paper, we map the cut-off thresholds to the 5 exits. (Figure adapted from Figure 2 in 21.)

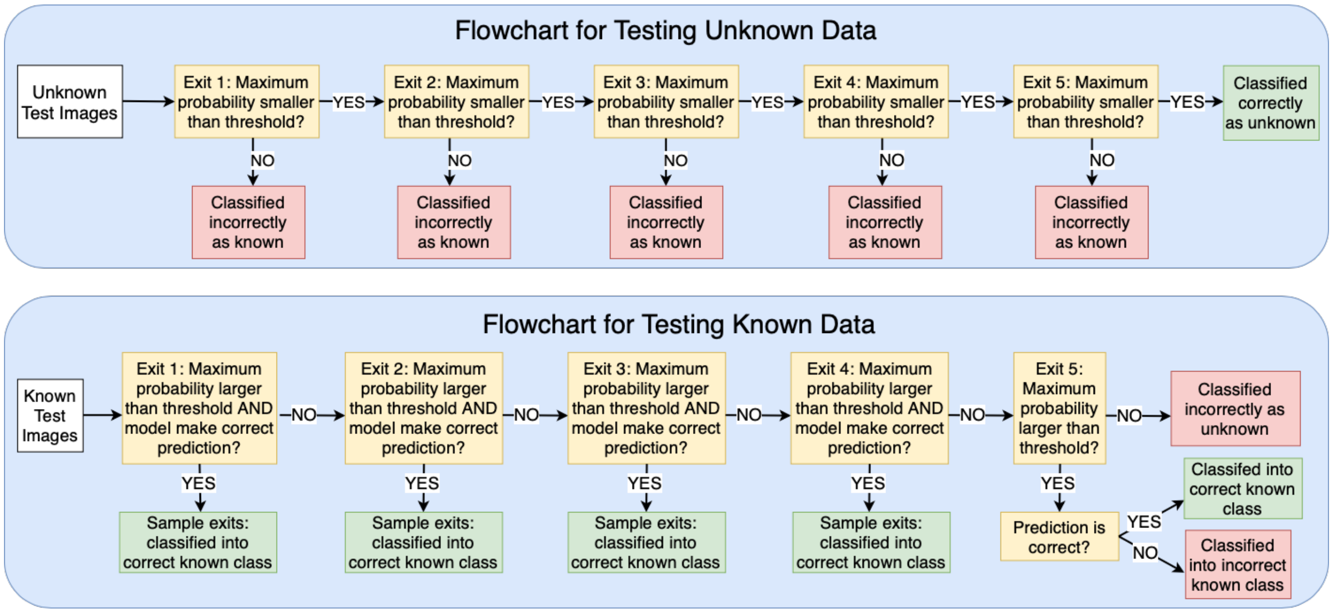

Flowchart for Testing Unknown Data
Unknown Test Images
Exit 1: Maximum probability smaller than threshold?
Exit 2: Maximum probability smaller than threshold?
Exit 3: Maximum probability smaller than threshold?
Exit 4: Maximum probability smaller than threshold?
Exit 5: Maximum probability smaller than threshold?
YES
NO
Classified correctly as unknown
Classified incorrectly as known
Flowchart for Testing Known Data
Known Test Images
Exit 1: Maximum probability larger than threshold AND model make correct prediction?
Exit 2: Maximum probability larger than threshold AND model make correct prediction?
Exit 3: Maximum probability larger than threshold AND model make correct prediction?
Exit 4: Maximum probability larger than threshold AND model make correct prediction?
Exit 5: Maximum probability larger than threshold?
Sample exits: classified into correct known class
Classified incorrectly as unknown
Prediction is correct?
Classified into correct known class
Classified into incorrect known class

Figure 22: Flowchart illustrating the detailed exit strategy. The top panel shows the exit strategy for unknown samples, and bottom one shows the strategy for known samples. In both flowcharts, yellow blocks stand for exits where the conditions are checked based on probability scores; green blocks mean the network makes a correct prediction, and red blocks mean the network makes an incorrect prediction.

With the above exit strategy, we are able to obtain the number of samples that exit from each classifier.

Algorithm	Seed	Exit 1	Exit 2	Exit 3	Exit 4	Exit 5 Correct Known	Exit 5 Incorrect Known	Exit 5 Unknown
$Loss_C$	0	26518	1493	1418	1403	22910	87904	108662
	1	63627	16207	8933	4654	8246	36719	95413
	2	82784	17230	11179	5347	3090	20763	90814
	3	43106	6922	4234	2780	16053	69579	101002
	4	62669	16397	8409	3937	9067	43000	93024
$Loss_C + Loss_P$	0	44446	7188	3853	2817	16794	73548	100526
	1	91774	20020	7073	5953	3335	23832	83008
	2	68176	15473	7374	4173	8036	37891	93077
	3	42712	6893	4716	2759	14970	64052	102931
	4	69760	15569	6939	3873	7850	37319	95727
$Loss_C + Loss_E$	0	78477	19404	11870	7060	3918	12590	94961
	1	84836	19423	9646	6269	3578	13629	92981
	2	78640	19970	11210	7588	3560	10549	93830
	3	84700	19036	9922	5787	3103	12117	93740
	4	83144	19443	11006	6519	3629	11869	91177
$Loss_C + Loss_P + Loss_E$	0	83407	20274	10916	6401	3772	11115	90457
	1	84046	18913	10464	6781	3642	11354	91923
	2	80207	19961	10691	6502	3778	12420	92731
	3	81406	18996	10960	6888	3381	10927	92195
	4	80554	19484	11258	7029	3900	11484	92492

Table 3: Breakdown for exit status for all 4 loss formulations. Columns 3, 4, 5, 6 and 7 stand for the number of samples that exit from each exit and are classified correctly into the known class they belong to, respectively. Column 8 stands for the number of samples that exit from exit 5, but are classified into a wrong known class. Column 9 stands for the numbers of samples that exit from exit 5 and are incorrectly classified as unknowns.

Algorithm	Exit 1	Exit 2	Exit 3	Exit 4	Exit 5 Correct Known	Exit 5 Incorrect Known	Exit 5 Unknown
$Loss_C$	55741	11650	6835	3624	11873	51593	97783
$Loss_C + Loss_P$	63374	13029	5991	3915	10197	47328	95054
$Loss_C + Loss_E$	81959	19455	10731	6645	3558	12151	93338
$Loss_C + Loss_P + Loss_E$	81924	19526	10858	6720	3695	11460	91960

Table 4: Average exit status for all 4 loss formulations.

Table 3 shows the breakdown for each loss formulation and the number of samples that exit from each exit respectively, by considering only maximum probability score and top-1 prediction, while Table 4 shows the average number for each exit from all 5 seeds, rounded to the closest integer.

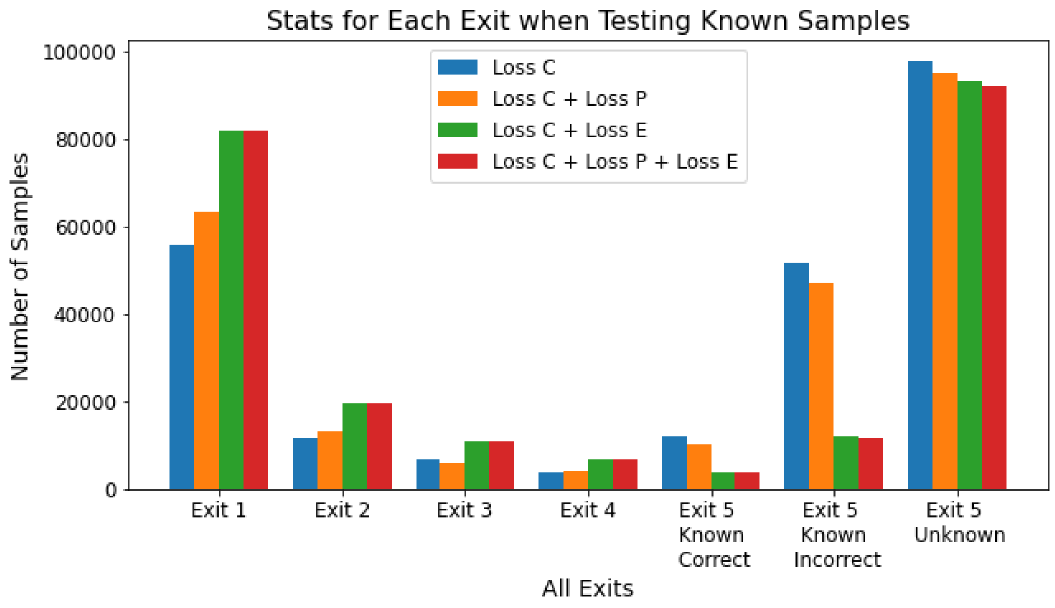


Figure 23: Statistics for each exit when testing the 4 different loss configurations with known samples.

Figure 23 illustrates the numbers reported in Table 4. We can tell from the figure that models trained with our exit loss (green and red bars) have more samples that are classified into a known class and that exit from Exit 1, 2, 3 and 4, compared to the two baseline models (blue and orange bars). Further, for models trained with exit loss, the number of samples that are classified correctly (the first five x-axis points) form a distribution that is similar with the shape of the distribution of our reaction time as shown in Figure 3 of the main paper. Models trained without exit loss do not form the same distribution, because the number of samples that exit from the 5th exit is larger than the number of samples that exit from the 4th exit. This indicates that our loss formulation forces a consistent behavior between human reaction time and machine learning models. Furthermore, models trained with exit loss have much fewer samples that are classified into an incorrect known class, and fewer samples classified as unknown at the end, which shows our loss formulation provides better decision boundaries for OSR.

Table 5 shows the breakdown for the number of samples that exit from each exit respectively, by considering only maximum probability score, while Table 6 shows the average number for each exit from all 5 seeds, rounded to the closest integer. At each exit, models trained with our exit loss have fewer samples that are incorrectly classified as known, and as a result, at the end of the model, many more examples are classified correctly as unknown with our models.

Algorithm	**Seed**	**Exit 1**	**Exit 2**	**Exit 3**	**Exit 4**	**Exit 5 Incorrect Known**	**Exit 5 Correct Unknown**
$Loss_C$	0	16289	8569	6274	4598	2718	9616
	1	15515	9827	6447	4149	2382	9744
	2	14228	7813	7225	4970	2789	11039
	3	16485	8891	5672	4582	2876	9558
	4	11983	9151	7636	5013	3635	10646
$Loss_C + Loss_P$	0	15586	8712	7060	4872	2943	8891
	1	15492	6674	3808	3313	1746	17034
	2	15437	8328	6142	4780	2772	10605
	3	16787	9721	5215	4099	2444	9798
	4	14464	9171	5995	4370	2827	11237
$Loss_C + Loss_E$	0	10900	4619	3063	2338	1268	25879
	1	11515	5000	2999	2450	1350	24753
	2	10209	4923	2976	2241	1057	26661
	3	10351	4642	3118	2287	1127	26542
	4	10472	4294	2788	1964	1254	27295
$Loss_C + Loss_P + Loss_E$	0	11063	4699	2891	2112	990	26312
	1	10271	4704	3014	2124	1140	26814
	2	11186	4614	2919	2335	1147	25866
	3	11381	4723	2816	2142	1056	25949
	4	11170	4904	3258	2290	1030	25415

Table 5: Breakdown of results for exit status for all 4 loss formulations. Columns 3, 4, 5, 6 and 7 stand for the numbers of samples that exit from each exit and are classified incorrectly as known, respectively. Column 7 stands for the numbers of samples that are correctly classified as unknown.

Algorithm	**Exit 1**	**Exit 2**	**Exit 3**	**Exit 4**	**Exit 5 Incorrect Known**	**Exit 5 Correct Unknown**
$Loss_C$	14900	8850	6651	4662	2880	10121
$Loss_C + Loss_P$	15553	8521	5644	4287	2546	11513
$Loss_C + Loss_E$	10689	4696	2989	2256	1211	26226
$Loss_C + Loss_P + Loss_E$	11014	4729	2980	2201	1073	26071

Table 6: Statistics for each exit when testing the 4 different loss configurations with unknown samples.

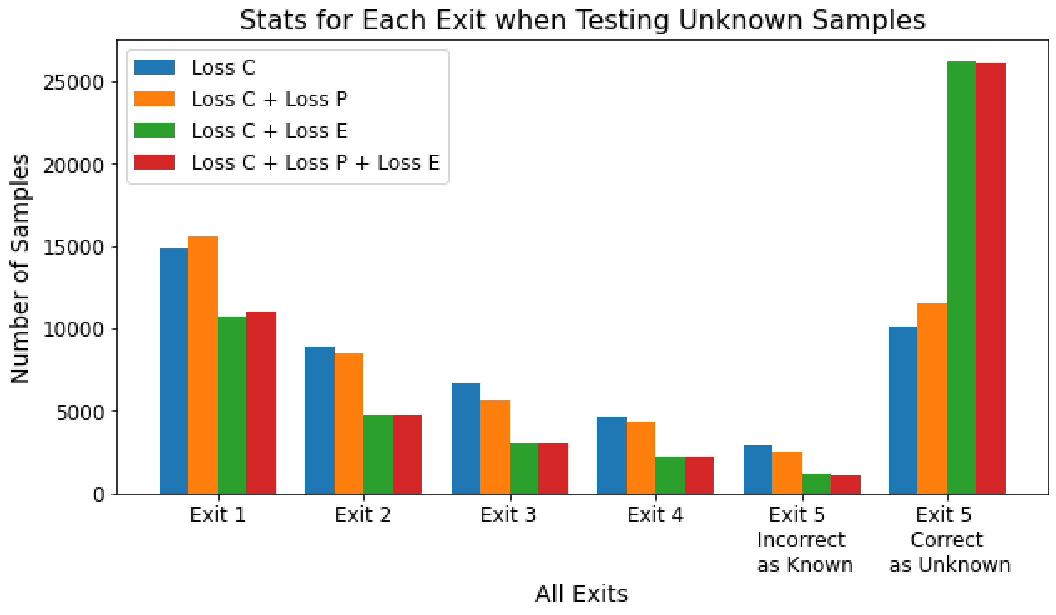


Figure 24: Statistics for each exit when testing the 4 different loss configurations with unknown samples.

3.2 Analysis of the Generalizability of Proposed Method

Our psychophysical loss function was originally designed for the MSD-Net architecture, but can be generalized to any deep network that has multiple classifiers/exits in its architecture. To explore the generalizability of our loss, we modified a simple deep network, ResNet18 [11], adding 5 classifiers as a proof-of-concept. Figure 25 shows the modified ResNet-18 architecture: we take the feature map after each convolutional block and feed those features into an average pooling layer and a fully-connected layer. A SoftMax layer is then used to obtain probability scores at each exit point.

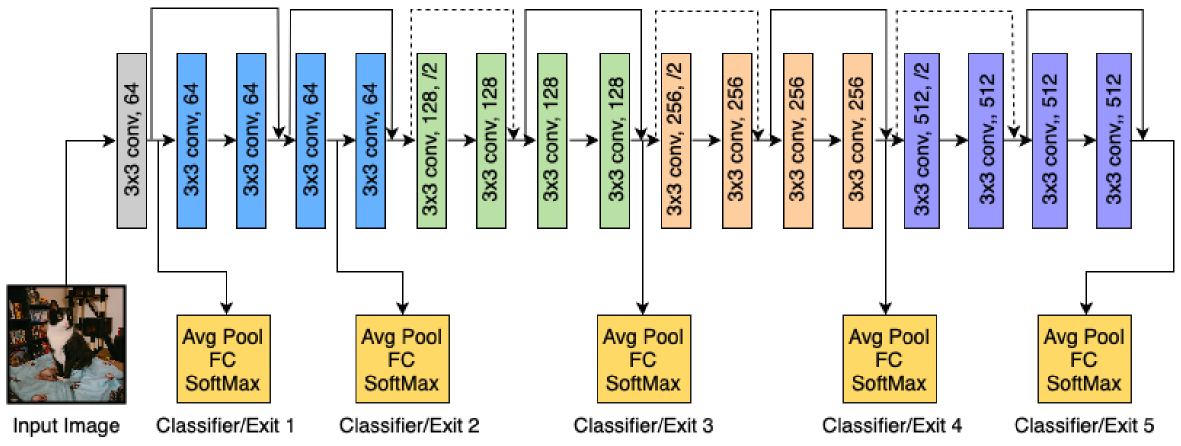


Figure 25: Network architecture of ResNet18 with the addition of 5 classifiers/exits.

In this way, we are able to train ResNet18 on ImageNet335 with exactly the same data partition and to use the same training strategy used to train MSD-Net. Table 7 shows the breakdown results for 5 seeds for each model, while Table 8 summarizes the average results with 5 seeds.

Algorithm	**Seed**	**Train Acc. Top-1**	**Train Acc. Top-3**	**Train Acc. Top-5**	**Valid. Acc. Top-1**	**Valid. Acc. Top-3**	**Valid. Acc. Top-5**
$Loss_C$	0	74.60	85.93	89.81	60.79	72.12	76.97
	1	74.88	86.56	90.23	60.97	72.38	77.58
	2	75.14	86.51	90.12	61.39	73.11	77.97
	3	**76.03**	**87.00**	**90.51**	61.27	72.70	77.72
	4	75.14	86.70	90.27	**61.52**	**72.85**	**78.06**
$Loss_C + Loss_P$	0	74.37	86.04	89.95	60.94	72.46	77.37
	1	74.63	86.24	89.92	60.84	72.37	77.40
	2	75.11	86.59	90.16	60.80	72.57	77.71
	3	75.15	86.61	90.36	60.73	72.62	77.67
	4	**75.49**	**86.90**	**90.35**	**60.99**	**73.00**	**77.74**
$Loss_C + Loss_P + Loss_E$	0	75.46	86.80	90.22	61.42	73.13	78.01
	1	76.02	**87.40**	**90.94**	61.90	73.36	78.04
	2	75.19	86.62	90.27	**62.25**	**73.42**	**78.24**
	3	75.15	86.41	90.28	61.24	73.07	77.80
	4	**76.23**	87.22	90.53	60.68	72.59	77.77
$Loss_C + Loss_E$	0	74.62	86.08	89.84	60.36	72.19	77.15
	1	73.79	85.69	89.38	59.61	71.40	76.69
	2	74.44	85.98	89.72	61.16	72.35	77.57
	3	74.71	86.41	89.95	**61.30**	**72.69**	**78.09**
	4	**75.28**	**86.59**	**90.23**	61.06	72.42	77.57

Table 7: Breakdown of training and validation accuracy in percentage for ResNet-18 for 5 seeds. Bold numbers indicate the model that performs the best on a metric among 5 seeds.

Algorithm	Train Acc. Top-1	Train Acc. Top-3	Train Acc. Top-5	Valid. Acc. Top-1	Valid. Acc. Top-3	Valid. Acc. Top-5
$Loss_C$	75.16	86.54	90.19	61.19	72.63	77.66
$Loss_C + Loss_P$	74.95	86.48	90.15	60.86	72.60	77.58
$Loss_C + Loss_P + Loss_E$	**75.61**	**86.89**	**90.45**	**61.50**	**73.11**	**77.97**
$Loss_C + Loss_E$	74.57	86.15	89.82	60.61	72.21	77.42

Table 8: Average training and validation accuracy in percentage for ResNet-18 for 5 seeds. Bold numbers indicate the model that performs the best on a metric.

Then we applied the same strategy to test these models: Table 9 shows the breakdown results for 5 seeds respectively, while Table 10 shows the averaged results.

Algorithm	Seed	TP	TN	FP	FN	F-1	MCC	Test Known Acc. Top-1	Test Unknown Acc
$Loss_C$	0	136874	12597	35470	199574	0.5380	-0.2202	22.0085	26.2071
	1	**140429**	12425	35642	**196019**	**0.5480**	-0.2151	22.5096	25.8493
	2	136966	**13739**	**34328**	199482	0.5395	**-0.2043**	22.8769	28.5830
	3	136230	11810	36257	200218	0.5354	-0.2323	**22.8847**	24.5698
	4	136495	12625	35442	199953	0.5370	-0.2206	22.7254	**26.2654**
$Loss_C + Loss_P$	0	2278	24034	24033	334170	0.0126	-0.6461	0.1697	50.0010
	1	136476	12794	35273	199972	0.5371	-0.2183	22.4962	26.6170
	2	136494	**12983**	**35084**	199954	0.5375	-0.2157	22.4656	**27.0102**
	3	**142010**	12362	35705	**194438**	**0.5524**	**-0.2128**	22.7075	25.7182
	4	139835	12140	35927	196613	0.5460	-0.2203	**22.9542**	25.2564
$Loss_C + Loss_P$ $+Loss_E$	0	136595	12474	35593	199853	0.5371	-0.2225	22.5973	25.9512
	1	**141098**	11899	36168	**195350**	0.5493	-0.2210	**23.6702**	25.7550
	2	137737	11818	36249	198711	0.5397	-0.2291	23.3700	25.5865
	3	140340	**12988**	**35079**	196108	0.5483	**-0.2076**	22.7126	**27.0206**
	4	139261	12405	35662	197187	0.5447	-0.2178	22.7994	25.8077
$Loss_C + Loss_E$	0	136336	12892	35175	200112	0.5368	-0.2173	22.2260	26.8208
	1	138123	**13377**	**34690**	198325	0.5424	**-0.2069**	22.0548	**27.8299**
	2	137736	12419	35648	198712	0.5403	-0.2208	22.1565	25.8368
	3	**139092**	12828	35239	**197356**	**0.5446**	-0.2124	**22.3470**	26.6877
	4	135248	12936	35131	201200	0.5337	-0.2190	21.8587	26.9124

Table 9: Breakdown testing results for losses based on ResNet-18. Results for seed 0 for the second loss configuration are marked in red because the model failed to perform consistently on this seed. Bold numbers indicate the model that performs the best on a metric among 5 seeds.

Looking at the training, validation and testing results, we observe improvement in all three phases by applying our method to the training process. Although the improvement gained on ResNet-18 is smaller comparing to that gained using the MSD-Net architecture, it shows our method can also be successfully applied to the ResNet architecture, and potentially can be generalized to other deep networks.

We argue that a psychophysical loss can improve the performance of OSR on networks that have multiple classifiers/exits because human behavior provides richer information for deep network training, and it enforces the network to be more consistent with human on image recognition. However, these results show that the improvement gained depends on the architecture of the network; our method works better on a network like MSD-Net that has deep structure and that was intentionally designed with multiple classifiers.

Algorithm	TP	TN	FP	FN	F-1	MCC	Test Known Acc. Top-1	Test Unknown Acc.
$Loss_C$	137399	12639	35428	199049	0.5398	-0.2185	22.6010	26.2950
$Loss_C + Loss_P$	138704	1270	35497	197744	0.5432	-0.2168	22.6559	26.1505
$Loss_C + Loss_P$ $+Loss_E$	**139006**	12317	35750	**197442**	**0.5438**	-0.2196	**23.0299**	26.0242
$Loss_C + Loss_E$	137307	**12890**	**35177**	199141	0.5396	**-0.2153**	22.1286	**26.8176**

Table 10: Average testing results for multiple seeds for losses applied to ResNet-18. Results for the second loss configuration excluded results from seed 0 because the model failed to perform consistently on this seed, thus the average only calculates a result from seed 1, 2, 3 and 4.

3.3 Breakdown Results for All Methods

Table 11 shows the full breakdown of results (*i.e.*, the individual results for each of the 5 seeds) for the standalone classifier baselines, and Table 12 shows the full breakdown of results for the deep learning-based methods. The results in bold represent the best result for a specific metric among the 5 seeds for that method.

In Table 12, some values are "nan" for the Matthews correlation coefficient (MCC) score, because this metric is defined by:

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{3}$$

When $TN + FN$ is zero, the numerator in the equation is divided by zero, and thus leads to "not a number"as the result.

Algorithm	Seed	TP	TN	FP	FN	F-1	MCC	Test Unknown	Test Known Top-1
SVM	0	**164378**	25599	22468	**172070**	**0.6282**	0.0139	0.5326	0.0062
	1	163091	**27448**	**20619**	173357	0.6271	**0.0369**	**0.5710**	0.0041
	2	154708	26632	21435	181740	0.6036	0.0092	0.5541	0.0046
	3	151932	25628	22439	184516	0.5949	-0.0101	0.5332	0.0067
	4	152100	26984	21083	184348	0.5969	0.00894	0.5614	**0.0082**
W-SVM	0	350	0	48087	336078	0.0018	-0.9959	0.0923	0.0009
	1	381	0	47931	336203	**0.0020**	-0.9955	0.0934	0.0010
	2	362	0	48115	336038	0.0019	-0.9957	0.0928	0.0009
	3	**410**	0	**47101**	337004	0.0021	**-0.9951**	**0.1101**	**0.0011**
	4	392	0	48112	**336011**	**0.0020**	-0.9954	0.0981	0.0010
P_I-SVM	0	**891**	201	48066	**335357**	**0.0046**	**-0.9872**	0.0832	**0.0028**
	1	807	208	48121	335379	0.0042	-0.9881	0.0810	0.0026
	2	851	210	48010	335444	0.0044	-0.9875	0.0827	**0.0028**
	3	790	240	**48001**	335484	0.0041	-0.9879	**0.0851**	0.0027
	4	830	**247**	48057	335381	0.0043	-0.9873	0.0832	**0.0028**
OpenMax	0	187520	19882	28182	132576	0.7000	-0.0004	0.4137	0.0036
	1	226722	13591	34473	94014	0.7792	-0.0077	0.2828	0.0037
	2	**249973**	9967	38097	69771	**0.8225**	-0.0089	0.2074	**0.0047**
	3	189123	**20942**	**27122**	132893	0.7027	**0.0157**	**0.4357**	0.0034
	4	228974	13520	34544	**91346**	0.7844	-0.0029	0.2813	0.0040
EVM	0	1977	47796	271	334471	0.0117	0.0010	0.9944	0.0047
	1	1293	47870	197	335155	0.0077	-0.0014	0.9959	0.0036
	2	1142	**47918**	**149**	335306	0.0068	0.0017	**0.9969**	0.0035
	3	1952	47812	255	334496	0.0115	**0.0022**	0.9947	0.0050
	4	**3602**	47494	573	**332846**	**0.0211**	-0.0039	0.9881	**0.0061**

Table 11: Testing results for standalone classifier baselines.

We also conducted a series of experiments for the four methods that utilized features that were reduced in dimensionality by PCA in order to explore algorithm sensitivity to different PCA settings. Recall from above that we applied PCA such that the reduced features represented 99% of the original features. In these experiments, we picked three larger fixed feature sizes that are reasonable for balancing information content and available computational resources: 250, 500 and 1000 dimensions. The corresponding results are shown in Table 13, Table 14 and Table 15. As can be seen, there is barely any change in the results, which indicates that our chosen dimensionality reduction scheme is reasonable, since increasing the dimensionality does not provide additional information. It is worth mentioning that after increasing the dimensionality, the EVM cannot detect known samples anymore and classifies every test sample as unknown. We use dashed lines to indicate this in the following tables.

References

[1] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," in *ICLR*, 2018.

[2] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, 1995.

[3] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE T-PAMI*, vol. 36, no. 11, 2014.

[4] L. P. Jain, , W. J. Scheirer, and T. E. Boult, "Multi-class open set recognition using probability of inclusion," in *ECCV*, 2014.

[5] E. M. Rudd, L. P. Jain, W. J. Scheirer, and T. E. Boult, "The extreme value machine," *IEEE T-PAMI*, vol. 40, no. 3, 2018.

[6] A. Bendale and T. E. Boult, "Towards Open Set Deep Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1563–1572. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Bendale_Towards_Open_Set_CVPR_2016_paper.html

[7] L. Neal, M. Olson, X. Fern, W.-K. Wong, and F. Li, "Open set learning with counterfactual images," in *ECCV*, 2018.

[8] R. Yoshihashi, W. Shao, R. Kawakami, S. You, M. Iida, and T. Naemura, "Classification-reconstruction learning for open-set recognition," in *IEEE/CVF CVPR*, 2019.

[9] D. Miller, N. Sunderhauf, M. Milford, and F. Dayoub, "Class anchor clustering: A loss for distance-based open set recognition," in *IEEE/CVF WACV*, 2021.

[10] S. Grieggs, B. Shen, G. Rauch, P. Li, J. Ma, D. Chiang, B. Price, and W. J. Scheirer, "Measuring human perception to improve handwritten document transcription," *IEEE T-PAMI*, vol. 44, no. 10, 2022.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

Algorithm	Seed	TP	TN	FP	FN	F-1	MCC	Test Unknown	Test Known Top-1
OSRCI	0	336179	39	48025	205	0.9330	0.0027	0.0008	**0.0040**
	1	**336384**	0	48064	**0**	0.9333	nan	0	0.0038
	2	321613	**2052**	**46012**	14771	0.9137	-0.0020	**0.0427**	0.0030
	3	**336384**	0	48064	**0**	**0.9333**	nan	0	0.0037
	4	332311	581	47483	4073	0.9280	-0.0001	0.0121	0.0036
CROSR	0	54089	41069	6995	282295	0.2722	0.0138	0.8545	0.0205
	1	50193	41568	6496	286191	0.2554	0.0131	0.8648	0.0174
	2	**56809**	40773	7291	**279575**	**0.2837**	0.0153	0.8483	**0.0209**
	3	54324	41114	6950	282108	0.2732	0.0152	0.8554	0.0201
	4	51308	**41625**	**6439**	285076	0.2604	**0.0172**	**0.8660**	0.0174
CAC-OSR	0	21776	45255	2809	314608	0.1207	0.0085	0.9416	0.0005
	1	20999	**45569**	**2495**	315385	0.1167	0.0145	**0.9481**	0.0004
	2	20825	45546	2518	315559	0.1158	0.0132	0.9476	0.0005
	3	**22820**	45366	2698	**313564**	**0.1261**	**0.0156**	0.9439	**0.0006**
	4	20008	45498	2566	316376	0.1114	0.0086	0.9466	0.0005
$Loss_C$	0	150031	9616	38448	186401	0.5716	-0.2342	20.00	15.97
	1	164704	9744	38320	171731	0.6106	-0.2039	20.27	30.22
	2	**170872**	**11039**	**37025**	**165576**	**0.6278**	**-0.1742**	**22.97**	**35.56**
	3	157431	9558	38506	179002	0.5914	-0.2204	19.89	21.73
	4	169667	10646	37418	166768	0.6243	-0.1819	22.15	29.87
$Loss_C + Loss_P$	0	163659	8891	39173	172774	0.6070	-0.2177	18.50	22.32
	1	**190692**	**17034**	**31033**	**145756**	**0.6833**	**-0.0528**	**35.43**	**38.09**
	2	166445	10605	37459	169990	0.6161	-0.1886	22.06	30.68
	3	150340	9798	38266	186093	0.5727	-0.2311	20.38	21.42
	4	165341	11237	36827	171094	0.6140	-0.1820	23.38	30.91
$Loss_C + Loss_P$ + Loss$_E$	0	**154791**	26312	21755	**181657**	**0.6035**	0.0050	54.74	**37.08**
	1	153070	**26814**	**21253**	183378	0.5994	**0.0085**	**55.78**	36.81
	2	152738	25866	22201	183710	0.5973	-0.0052	53.81	36.01
	3	151808	25949	22118	184640	0.5949	-0.0059	53.98	36.15
	4	153610	25415	22652	182838	0.5992	-0.0097	52.87	36.33
$Loss_C + Loss_E$	0	152906	25879	22188	183542	0.5978	-0.0047	53.84	35.88
	1	**15865**2	24753	23314	**177796**	**0.6121**	-0.0089	51.50	**36.78**
	2	148619	26661	21406	187829	0.5869	-0.0024	55.47	35.95
	3	153397	26542	21525	183051	0.5999	0.0054	55.22	36.42
	4	153493	**27295**	**20772**	182955	0.6011	**0.0160**	**56.79**	**36.78**

Table 12: Testing results for deep learning-based baselines.

Algorithm	**Seed**	**TP**	**TN**	**FP**	**FN**	**F-1**	**MCC**	**Test Unknown**	**Test Known Top-1**
SVM	0	156428	1519	1485	180020	0.6328	-0.0055	0.5057	0.0060
	1	**159912**	1608	1396	**176536**	**0.6425**	**0.0020**	0.5352	0.0041
	2	152790	1578	1426	183658	0.6228	-0.004	0.5252	0.0042
	3	146332	1527	1477	190116	0.6044	-0.0107	0.5083	0.0069
	4	143729	**1645**	**1359**	192719	0.5969	-0.0048	**0.5476**	**0.0079**
W-SVM	0	320	0	51021	333165	0.0017	-0.9964	0.0911	0.0008
	1	331	0	51321	**332855**	0.0017	-0.9963	0.0914	**0.0009**
	2	315	0	50206	333985	**0.0016**	-0.9964	0.0910	0.0008
	3	**341**	0	49985	334180	0.0018	**-0.9961**	0.0921	**0.0009**
	4	334	0	**49912**	334260	0.0017	-0.9962	**0.0929**	**0.0009**
P_I-SVM	0	721	189	**48210**	335386	0.0037	-0.9892	0.0821	0.0024
	1	744	**211**	48328	335223	0.0039	-0.9891	0.0839	0.0025
	2	721	205	48290	335290	0.0037	**-0.9889**	0.0829	0.0024
	3	749	188	48381	**335188**	0.0039	-0.9892	**0.0864**	0.0024
	4	**812**	201	48922	334571	**0.0042**	-0.9891	0.0821	**0.0026**
EVM	0	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-

Table 13: Testing results for the four methods that use features after the application of PCA to reduce the representation to 250 dimensions.

Algorithm	**Seed**	**TP**	**TN**	**FP**	**FN**	**F-1**	**MCC**	**Test Unknown**	**Test Known Top-1**
SVM	0	146845	1602	1402	189603	0.6059	-0.0057	0.5332	0.0062
	1	**151455**	1666	1338	**184993**	**0.6191**	**0.0008**	0.5545	0.0040
	2	144019	1663	1341	192429	0.5978	-0.0034	0.5535	0.0048
	3	146845	1602	1402	189603	0.6059	-0.0057	0.5332	0.0062
	4	133415	**1703**	**1301**	203033	0.5663	-0.0069	**0.5669**	**0.0081**
W-SVM	0	298	0	52012	**332196**	0.0015	-0.9967	**0.0899**	0.0008
	1	310	0	51882	332314	**0.0016**	-0.9966	0.0891	0.0008
	2	305	0	51841	332860	**0.0016**	-0.9966	0.0887	0.0008
	3	310	0	**51202**	332994	**0.0016**	**-0.9965**	0.0881	0.0008
	4	**330**	0	51303	332873	**0.0016**	**-0.9965**	0.0881	0.0008
P_I-SVM	0	751	**228**	50092	333435	0.0039	-0.9891	0.0812	0.0025
	1	758	221	50101	333426	0.0039	**-0.9890**	0.0814	0.0025
	2	**763**	211	**50085**	333447	**0.0040**	-0.9892	0.0819	0.0024
	3	755	226	51012	332513	0.0039	-0.9891	**0.0825**	**0.0026**
	4	758	222	51018	**332508**	0.0039	-0.9892	0.0821	0.0025
EVM	0	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-

Table 14: Testing results for the four methods that use features after the application of PCA to reduce the representation to 500 dimensions.

Algorithm	**Seed**	**TP**	**TN**	**FP**	**FN**	**F-1**	**MCC**	**Test Unknown**	**Test Known Top-1**
SVM	0	135028	1737	**1267**	201420	0.5712	-0.0039	0.5782	0.0067
	1	**139877**	1745	1259	196571	**0.5857**	**-0.0006**	0.5808	0.0039
	2	132666	1748	1256	203782	0.5640	-0.0045	0.5818	0.0045
	3	135028	1737	**1267**	201420	0.5712	-0.0039	0.5782	0.0067
	4	121726	**1815**	1189	**214722**	0.5299	-0.0066	**0.6041**	**0.0084**
W-SVM	0	280	0	53292	330934	**0.0015**	-0.9970	0.0812	0.0007
	1	**288**	0	53288	**330930**	**0.0015**	**-0.9969**	0.0822	0.0007
	2	276	0	**53197**	331033	0.0014	-0.9970	0.0821	0.0007
	3	275	0	53229	331002	0.0014	-0.9979	**0.0827**	0.0007
	4	281	0	53212	331013	**0.0015**	**-0.9969**	0.0821	0.0007
P_I-SVM	0	731	235	**51241**	332299	0.0038	-0.9892	0.0808	0.0025
	1	735	240	51255	332276	0.0038	-0.9891	0.0812	0.0025
	2	744	**252**	51265	**332245**	**0.0039**	**-0.9889**	0.0818	0.0024
	3	**750**	243	51254	332259	**0.0039**	**-0.9889**	0.0815	**0.0026**
	4	753	247	51259	332247	**0.0039**	**-0.9889**	**0.0816**	**0.0026**
EVM	0	-	-	-	-	-	-	-	-
	1	-	-	-	-	-	-	-	-
	2	-	-	-	-	-	-	-	-
	3	-	-	-	-	-	-	-	-
	4	-	-	-	-	-	-	-	-

Table 15: Testing results for the four methods that use features after the application of PCA to reduce the representation to 1000 dimensions.