Yandex

Yandex

CVPR 2020 hands-on tutorial

Efficient Data Annotation for Self-Driving Cars via Crowdsourcing on a Large-Scale

Alexey Drutsa, Denis Rogachevsky, Olga Megorskaya, Daria Baidakova, Ivan Semchuk

Part II:

Main components of data collection via crowdsourcing

Alexey Drutsa,
Head of Efficiency and Growth Division, Yandex

Tutorial schedule

Introduction: 30 min

Part I: 30 min
Crowdsourcing for SDC

Coffee break: 15 min

Part II: 35 min
Key components

Part III: 10 min
Intro to crowd platform

Part IV: 60 min
Data labeling demos for SDC

Lunch break: 60 min

Part V: 20 min
Brainstorming pipeline

Part VI: 90 min
Set & Run Projects cont.

Coffee break: 15 min

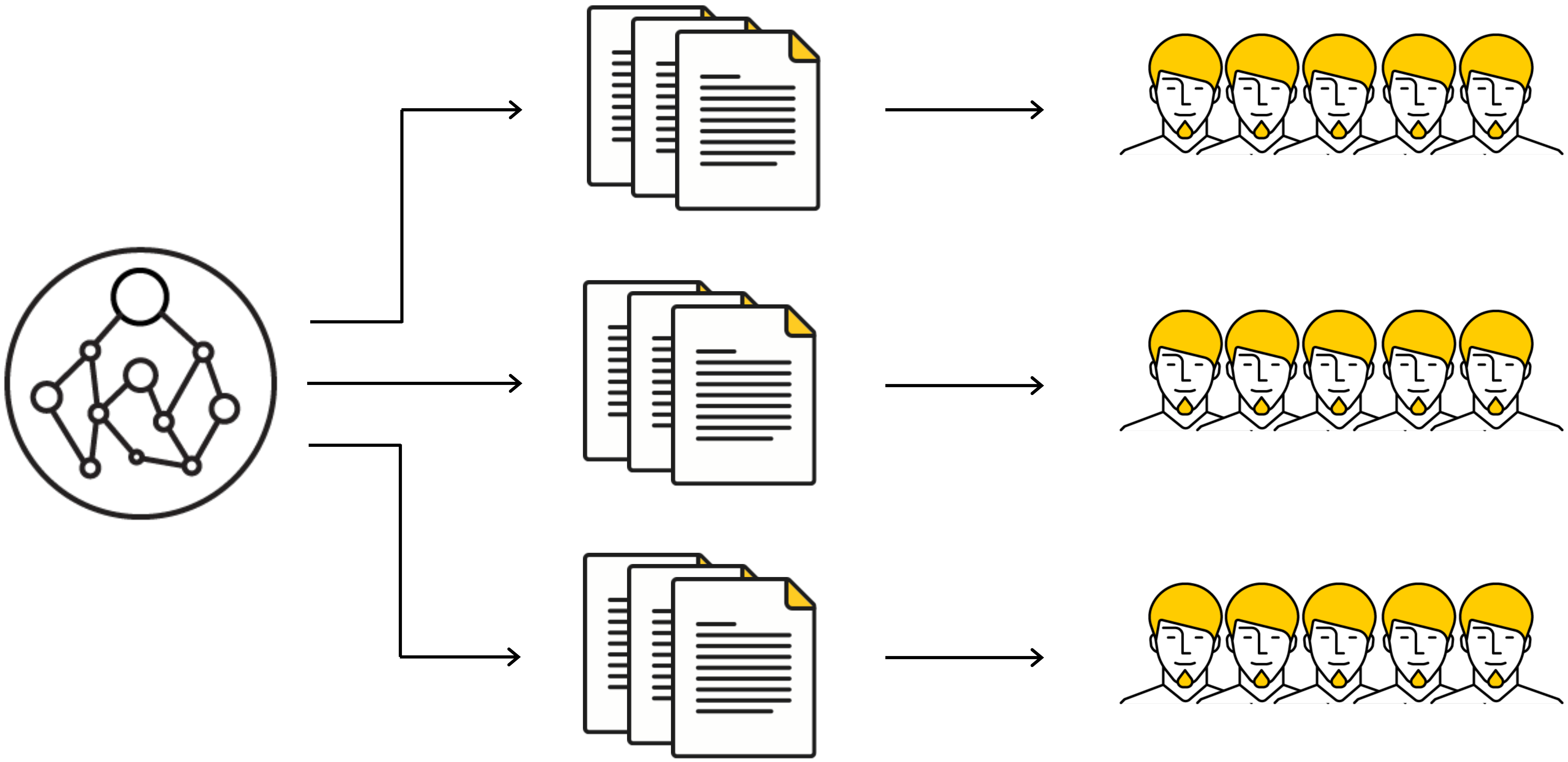
Part VII: 60 min
Theory on aggregation, IRL and pricing

Part VIII: 15 min
Results & Conclusions

Decomposition
Instruction
Task interface
Quality control
Aggregation
Incremental relabelling
&
Pricing

Decomposition

Decomposition



A big task

Projects with microtasks of different type

Cloud of performers

Decomposition: why?

Performers are usually non-specialists in your specific task

The simpler a single task is:

- the more humans can perform your task
- the easier its instruction
- the better quality of performance

A way to:

- distinguish tasks with different difficulty
- control and optimize pricing
- control quality by post verification

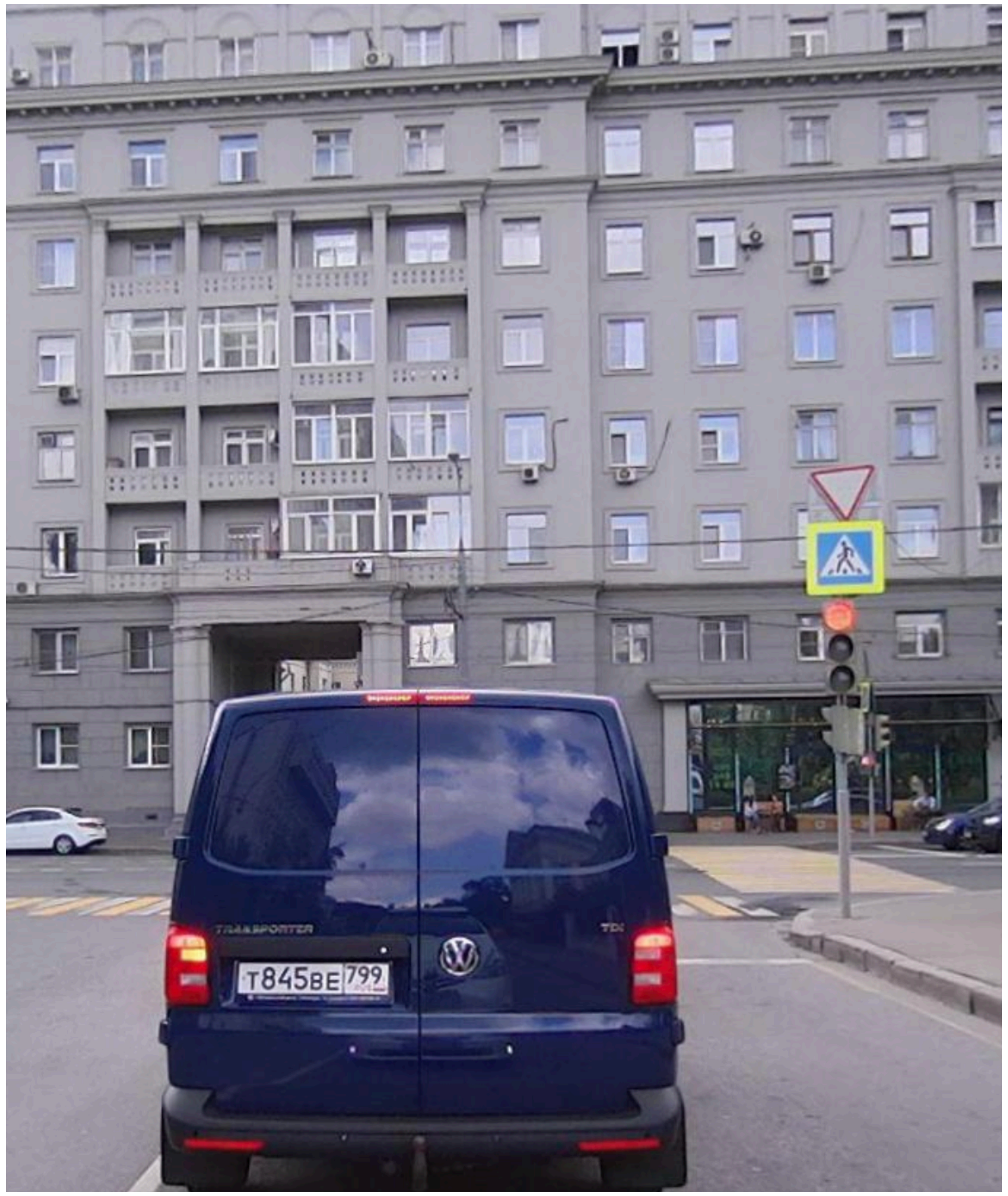
Decomposition: when?

If

- your task requires an answer selected among more than 3-5 variants
- your task has a long instruction hard to read

then your task requires decomposition

Case of decomposition: a lot of questions



All questions in one task

Bad practice

What type is the vehicle?

- Car
- Bus
- Truck
- Motorcycle
- Bike
- Tractor
- None of the above

Is there a pedestrian?

- Yes
- No

Is there a traffic light?

- Yes
- No

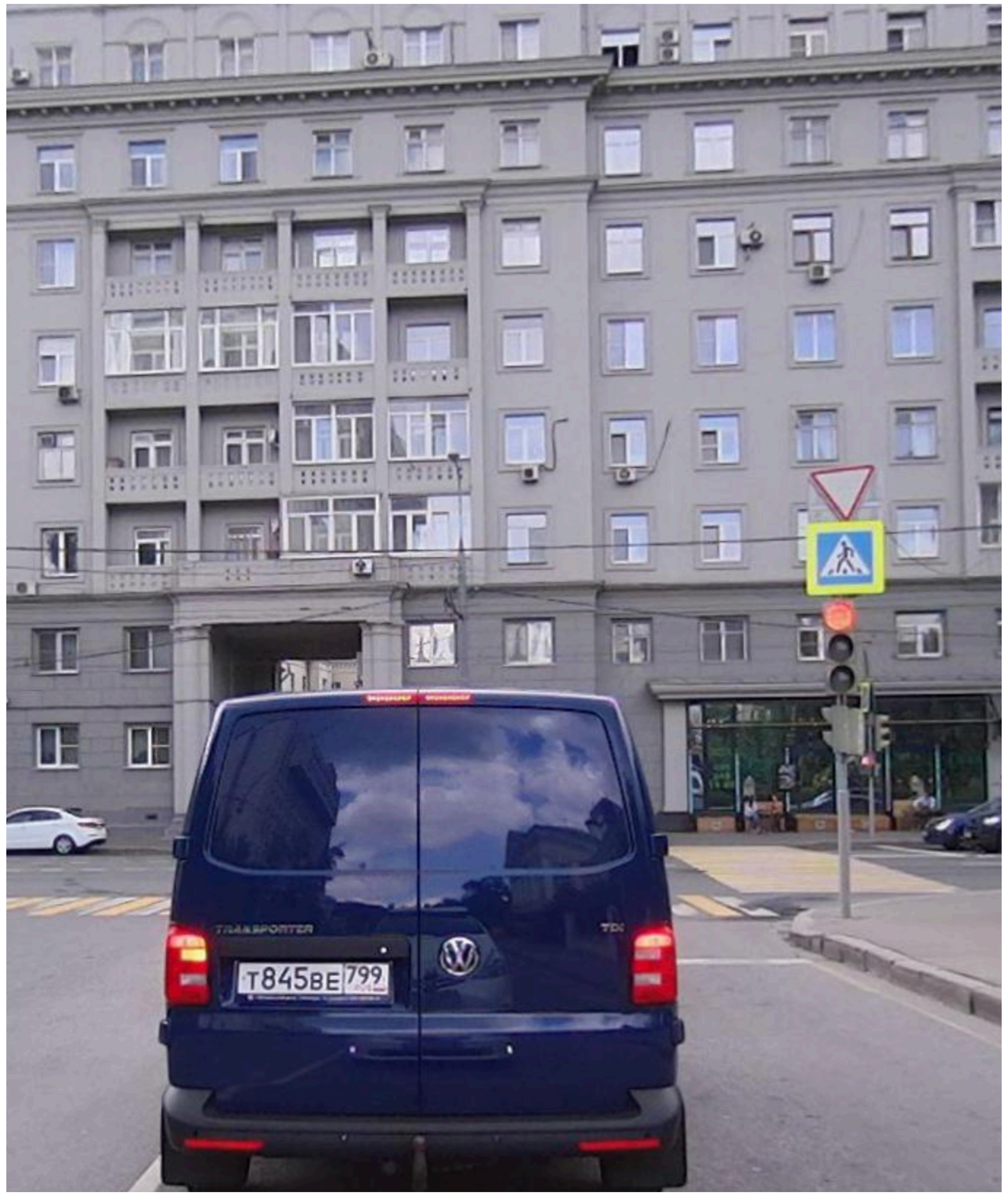
What color is the vehicle?

- White
- Black
- Brown
- Red
- Other

Where is it situated?

- On a grass
- On a sidewalk
- On a carriageway
- It is flying
- None of the above

Case of decomposition: a lot of questions



Each question in a separate task — Good practice

What type is the vehicle?
- Car
- Bus
- Truck
- Motorcycle
- Bike
- Tractor
- None of the above

Is there a pedestrian?
- Yes
- No

Is there a traffic light?
- Yes
- No

What color is the vehicle?
- White
- Black
- Brown
- Red
- Other

Where is it situated?
- On a grass
- On a sidewalk
- On a carriageway
- It is flying
- None of the above

Case of decomposition: need to verify answers

The task:
Highlight all cars on the photo

Problem: highlighting can be done in different ways.

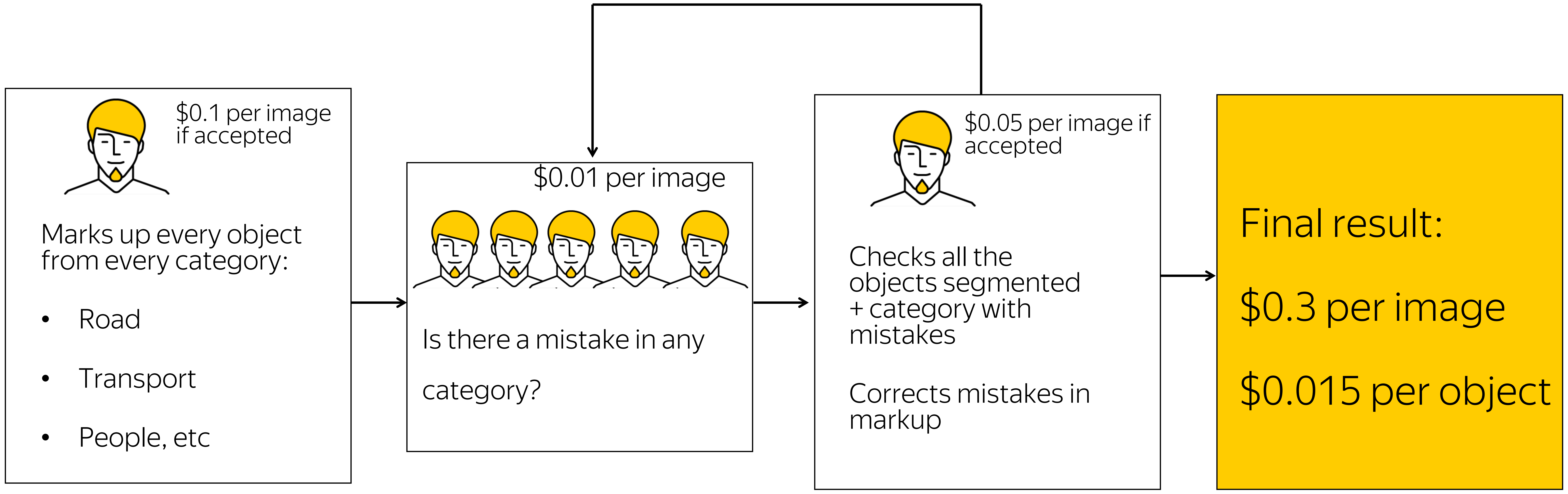
Hence, it is difficult to make:

- comparison with control answers
- aggregation of answers from different performers

A good solution

A task for another performer:
Is the highlighting of all cars made correctly?

Real example: decomposition for segmentation


$0.1 per image if accepted
Marks up every object from every category:
• Road
• Transport
• People, etc
$0.01 per image
Is there a mistake in any category?
$0.05 per image if accepted
Checks all the objects segmented + category with mistakes
Corrects mistakes in markup
Final result:
$0.3 per image
$0.015 per object

Instruction

Instruction: a typical structure

- Goal of the task to be done
- Interface description
- Algorithm of required actions
- Examples of good and bad answers
- Algorithm and examples for rare cases
- Reference materials

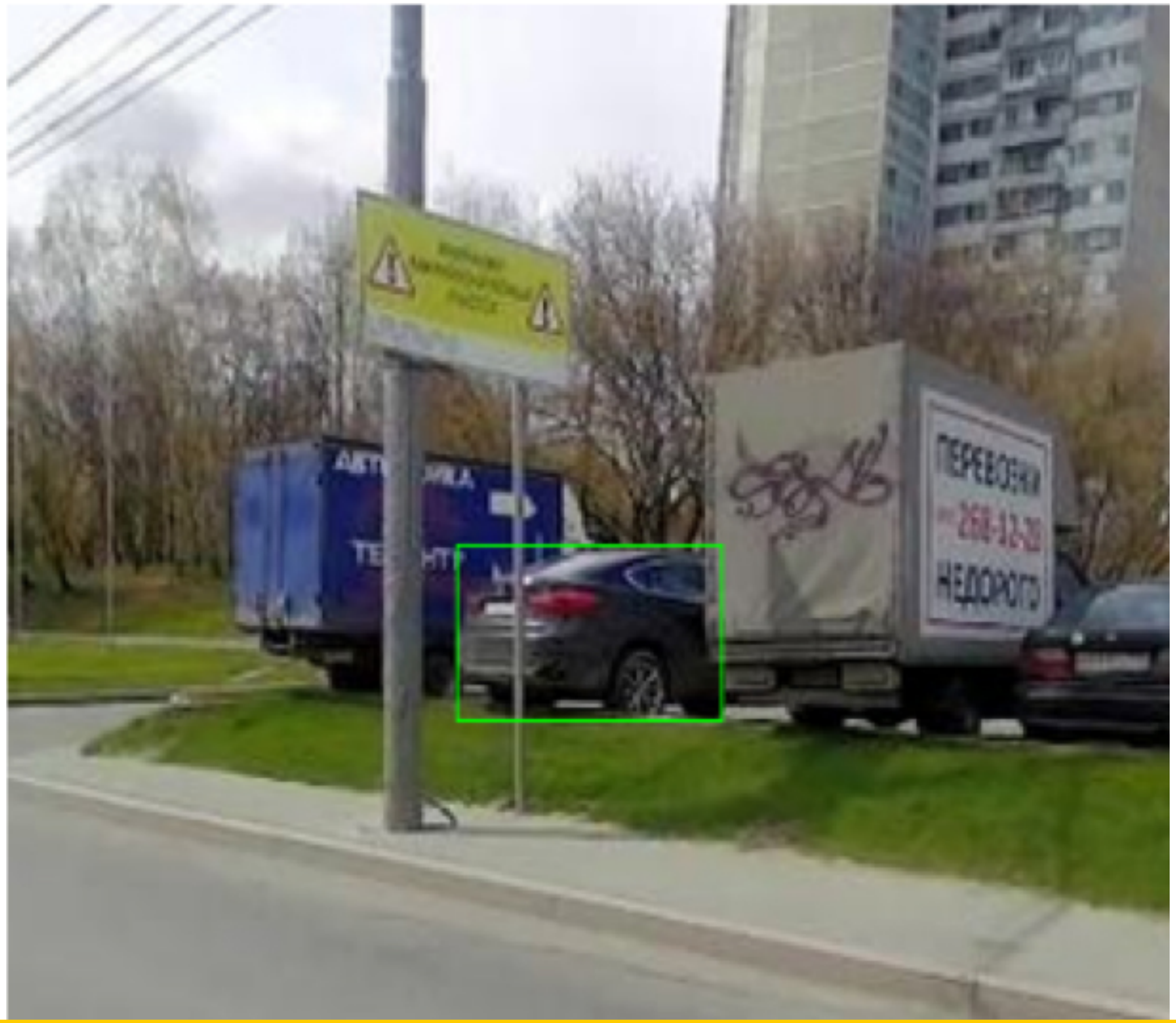
Most pitfalls are there

Instruction ambiguity for a rare case: example

Is the outlined object a car?

Yes

No



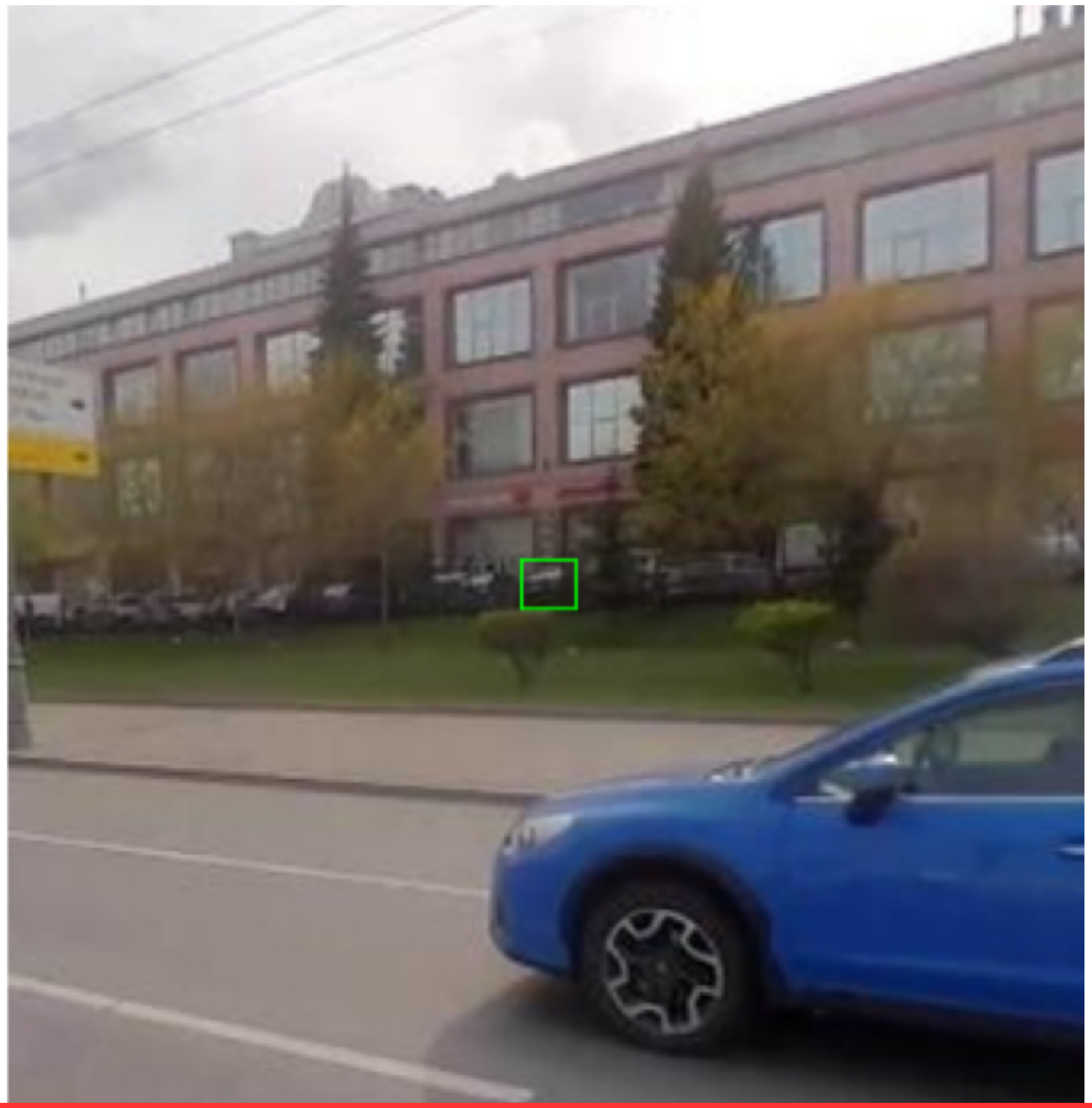
OK: the answer and the task seem clear

Instruction ambiguity for a rare case: example

Is the outlined object a car?

Yes

No



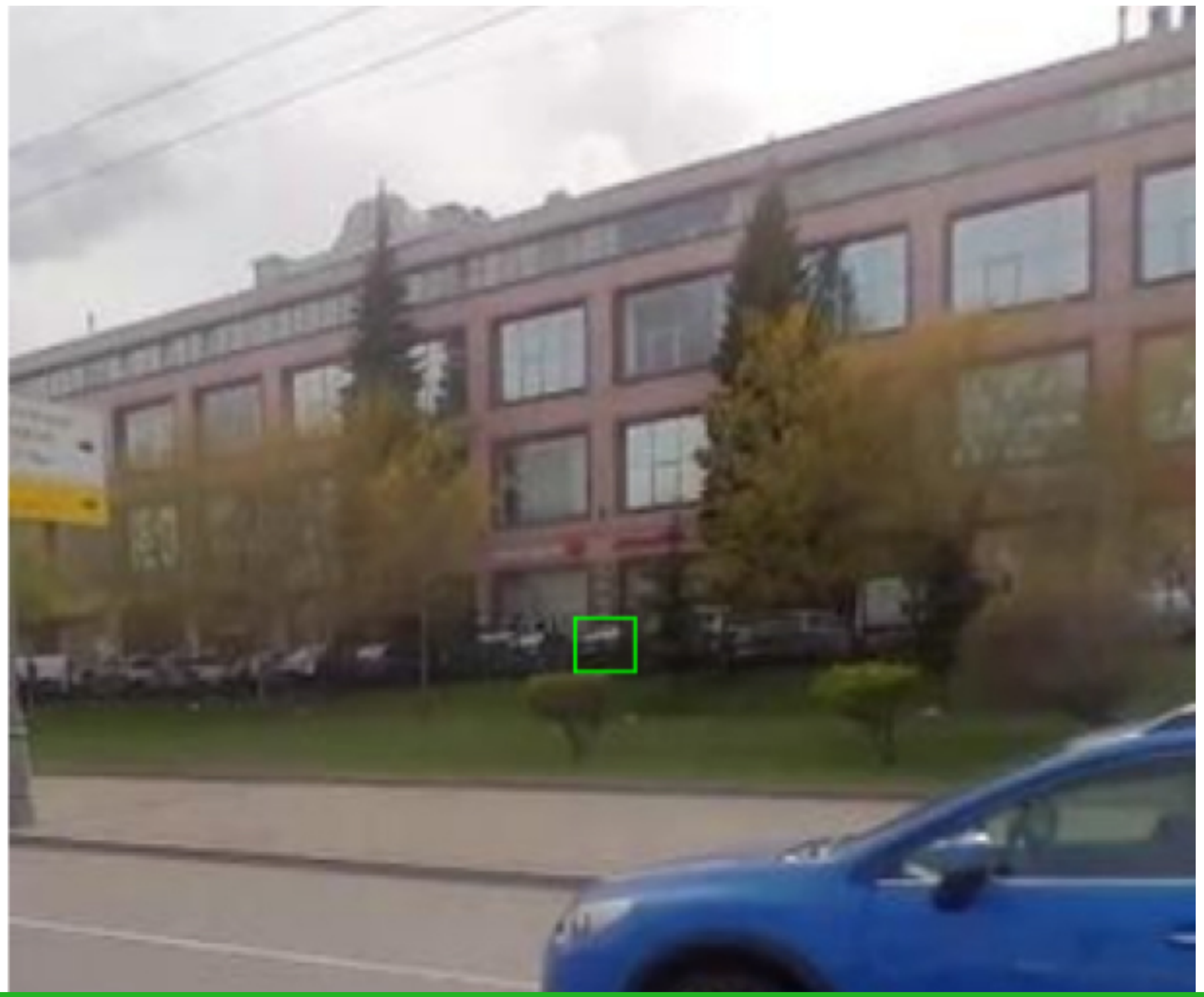
What is the correct answer?

Instruction ambiguity for a rare case: example

Is the outlined object
a car?

Yes

No



How to fix:

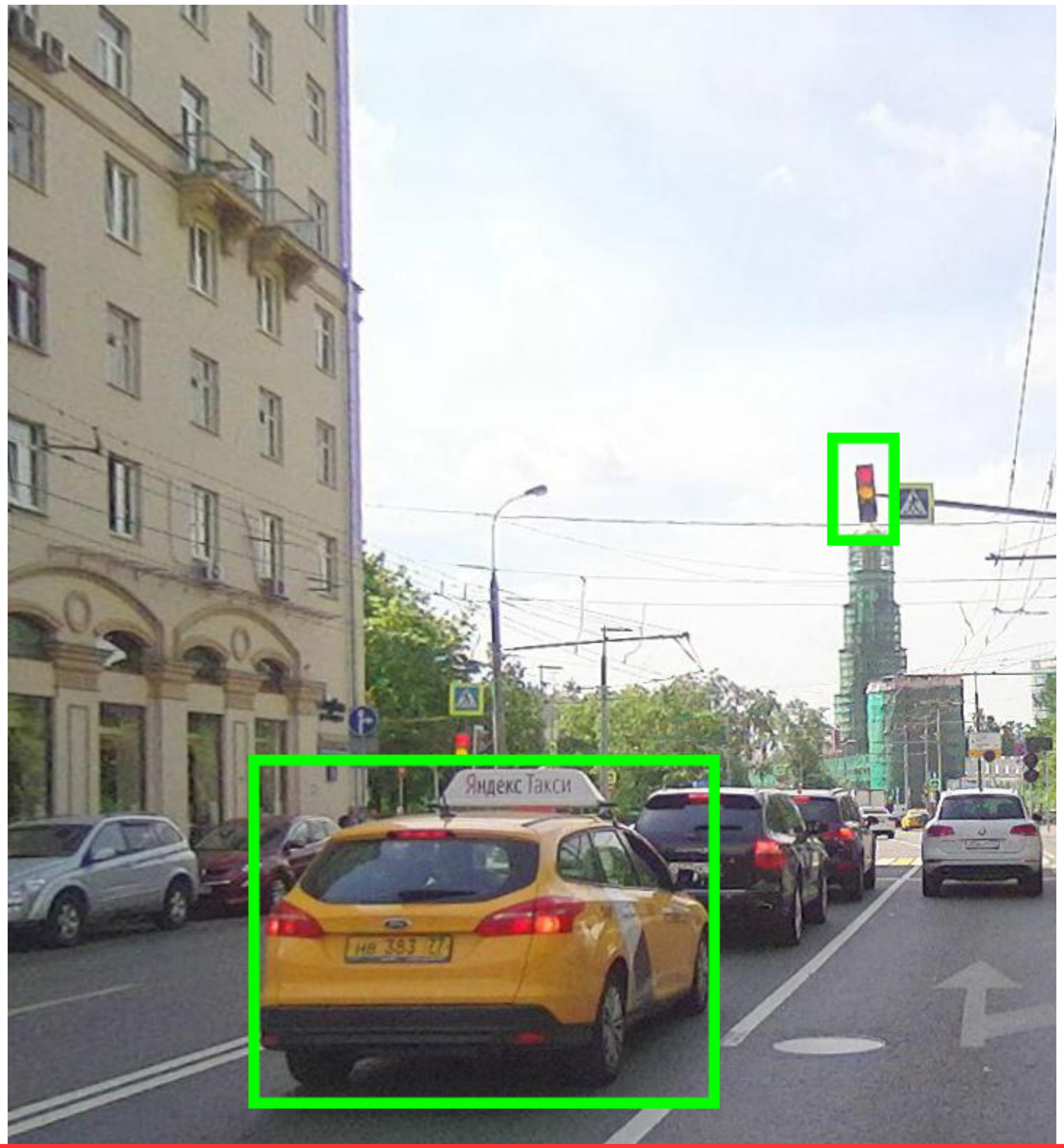
- In the instruction: clarify what you mean under “a car”

Instruction ambiguity for a rare case: example

Is the outlined object
a car?

Yes

No



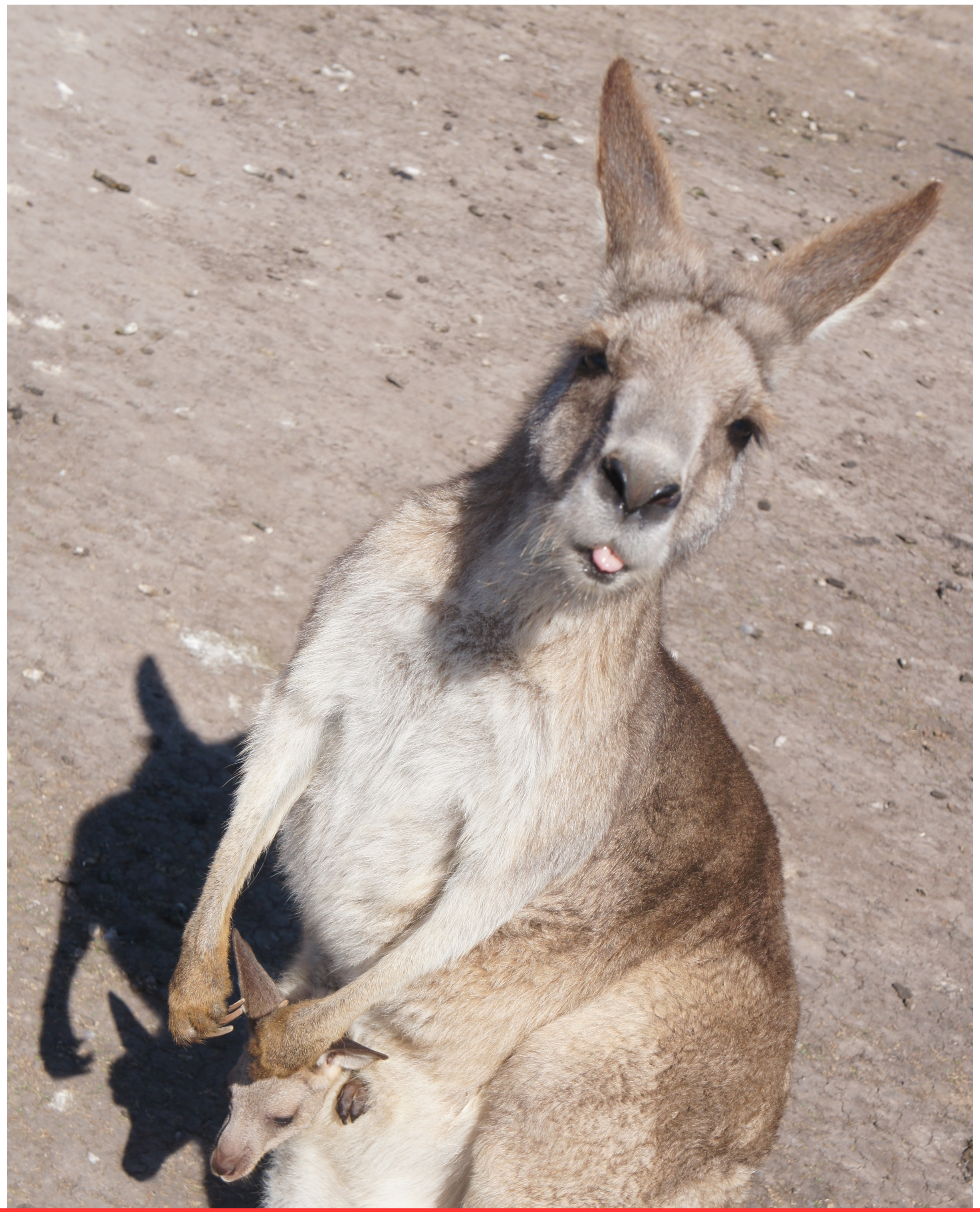
Rare case: many selections

Instruction ambiguity for a rare case: example

Is the outlined object a car?

Yes

No



Rare case: no selection

Instruction ambiguity for a rare case: example

Is the outlined object a car?

Yes

No

404: Cannot download the image

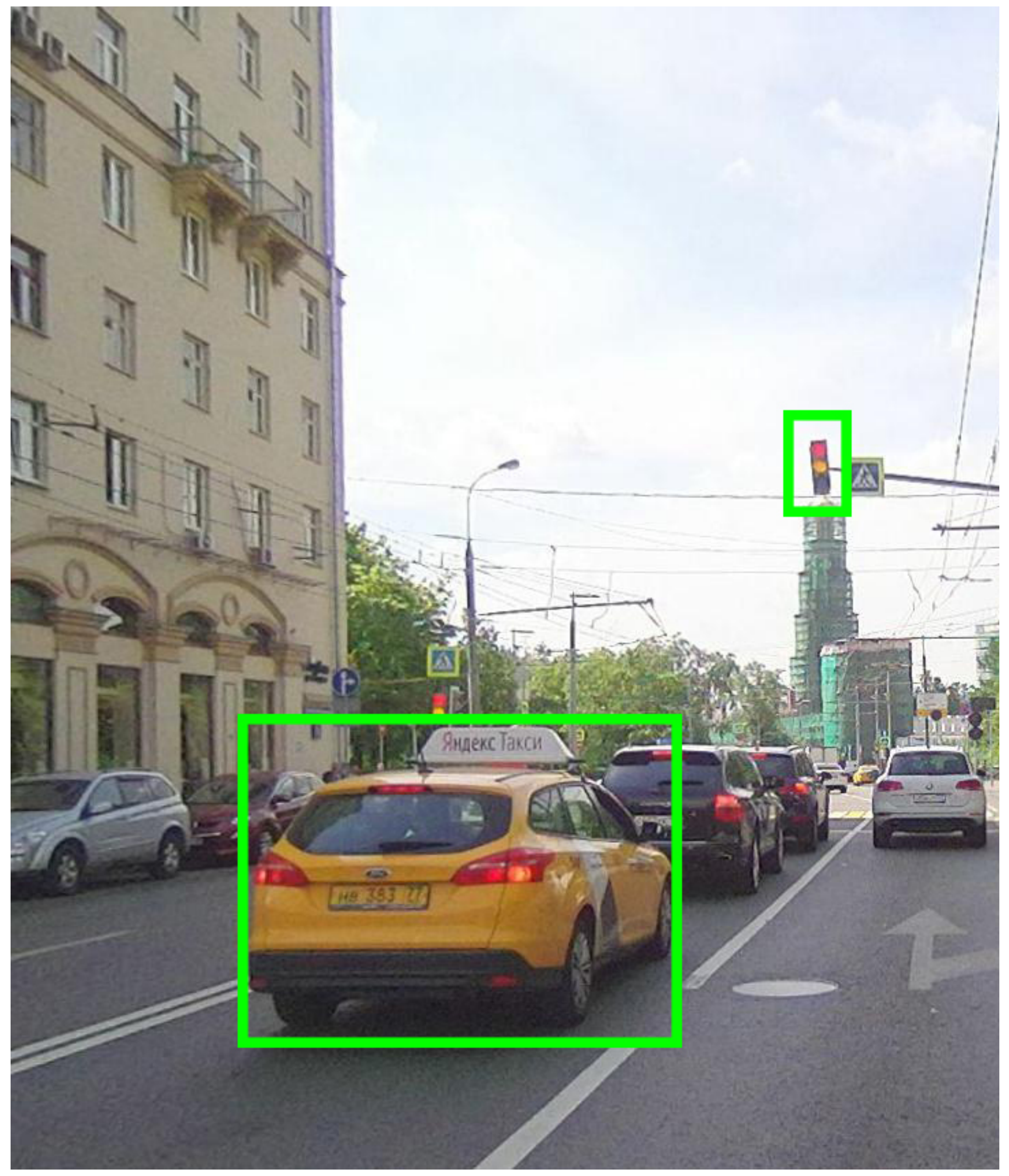
Rare case: image has not been shown

Instruction ambiguity for a rare case: example

Is the outlined object a car?

Yes

No



It is difficult to predict situations of any kind, but you can:

- In the instruction: clarify what should be done in a non-standard situation
- In the interface: add a text field to allow a performer to report the case

Task interface

Task interface: summary on best practices

For faster performance:

- Hot key combinations for checkboxes / radio buttons / buttons
- Reduce navigation to third-party sites
- Effective composition of a task template
- Optimal position of tasks on a page

For better quality and less errors:

- Dynamic interface (show/hide input controls depending on user actions)
- Adaptive interface (good view for any device and screen resolution)
- Always test your interface (template testing)
- Dynamic validation of input data (e.g. a text is less than 3 words)

Quality control

Quality control

“Before” task performance

- Selection of performers
- Well-designed instruction

“Within” task performance

- Golden set (aka honey pots)
- Well-designed interface
- Motivation (e.g. performance-based pricing)
- Tricks to remove bots and cheaters (e.g. quick answers)

“After” task performance

- Post verification (acceptance)
- Consensus between performers and result aggregation

Selection of performers

Filter by static properties (e.g. education, languages, citizenship, etc.)

Filter by computed properties (e.g. browser, region by phone/IP, etc.)

Filter by skills:
- to select proper specialization
- to control quality level on your tasks
- to get performers with best quality on past projects

Educate to perform your tasks:
- Use training tasks to show how to perform tasks
- Use exam tasks to evaluate education level

Golden set (aka honey pots)

Tasks with known correct answer
shown to performers to evaluate their quality

Best practices

- Distribution of answers in golden set = distribution in whole set of tasks
- But should contain rare answer variants with higher frequency
- Refresh your set of honey pots regularly
  to avoid bots and cheating
- Automatic golden set generation via performers:
  tasks with answers of high confidence
  (e.g. aggregation of answers from a large number of performers)

Motivation

- Bonuses for a good quality within a period
- Gamification (e.g. achievements, leader boards, etc)
- Price depending on quality

Will be discussed in Part VIII

Tricks to remove bots and cheaters

- Control fast responses
- Check whether a link has been visited
- Check whether a video has been played
- etc.

Post verification (acceptance)

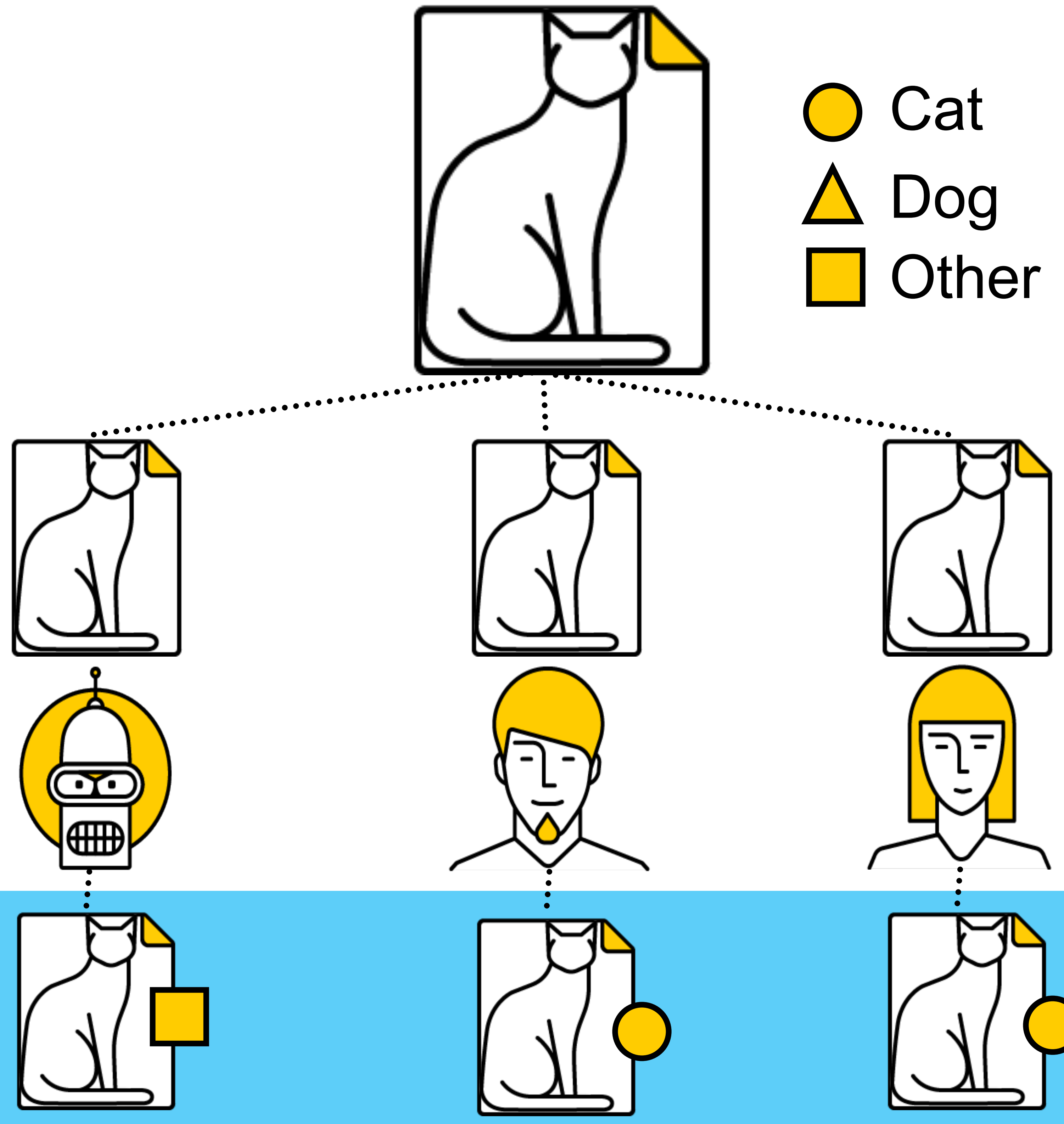
A performer gets money only if his answer is accepted

- Is used when a task is sophisticated
  (neither golden set nor consensus models work)

- Can be performed on your own, but

You can use other crowd performers via a task of different type
Thus, you deal with hierarchy of projects (you apply decomposition)

Consensus between performers



Upload multiple copies of each object to label

Performers assign noisy labels to objects

Derive quality from consensus

Works well only if most performers have good quality

Will be discussed in Part VI

Quality control: skills

Skill is a variable assigned to a performer

Can be used to automatically calculate

- answer correctness rates (via control tasks, agreement, post-verification)
- behavioral features (e.g., fast response rate)
- binary information on execution of particular projects
- any their combinations and other features

Can be used for automatic decision making:

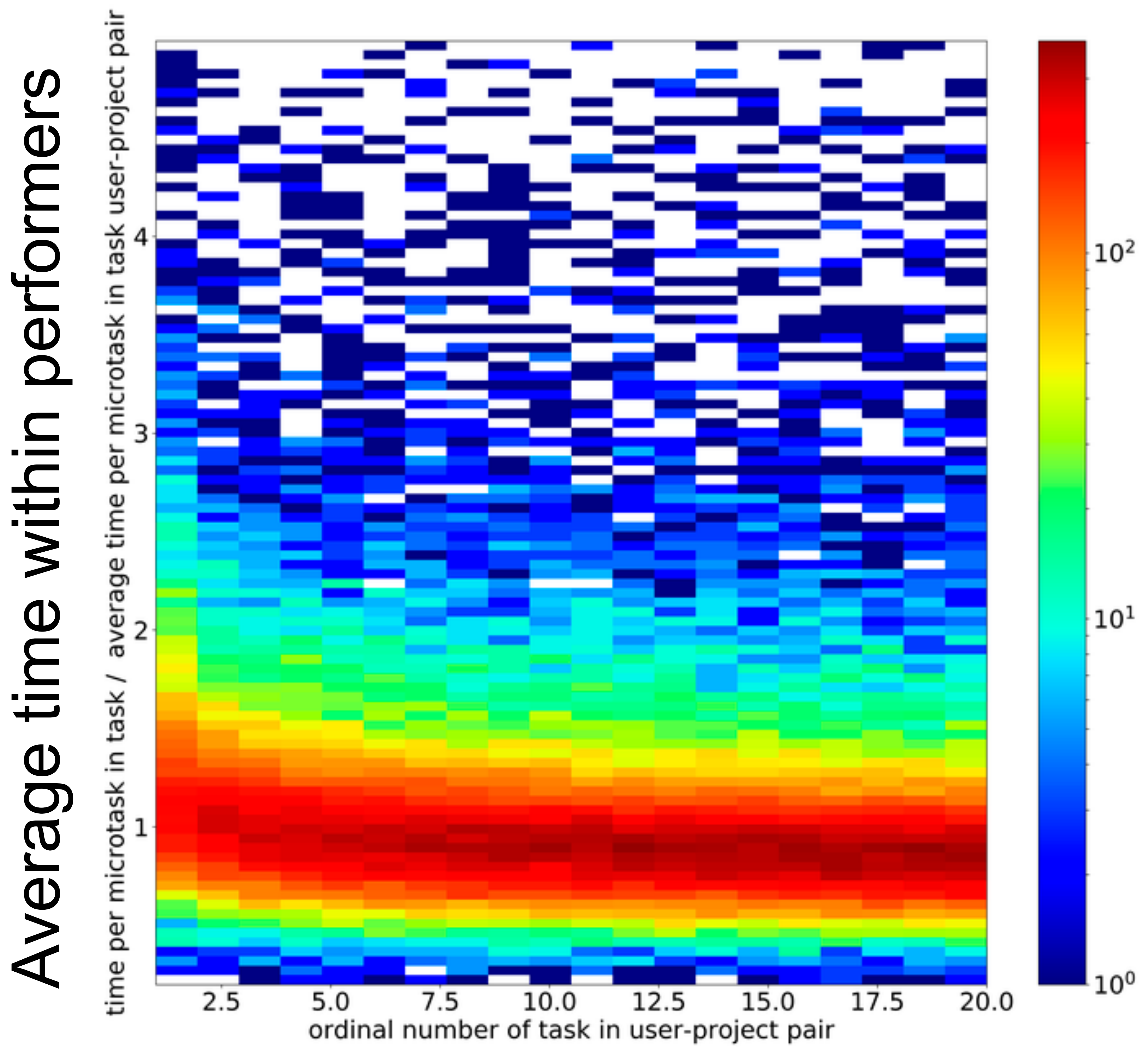
- access control to certain projects and tasks
- e.g., revoke access to your tasks if a skill becomes too low

Thinking (cogitation) vs reflexes

› Skills based on a single signal are easy to game

It is difficult to force a performer to think (cogitate) instead of to use/train reflexes

A representative crowd project



# tasks made by a performer

Best practice for a good skill

Combine different signals to get a skill robust to gaming

- Combine agreement signal with control tasks or post-verification
- Add behavioral information: execution time, CAPTCHA, etc.

Use this skill in quality-based pricing

Quality control: performer life cycle

Training task

Train performers to execute your tasks

- All tasks are control ones
- There are hints that explain incorrect answers

Exam task

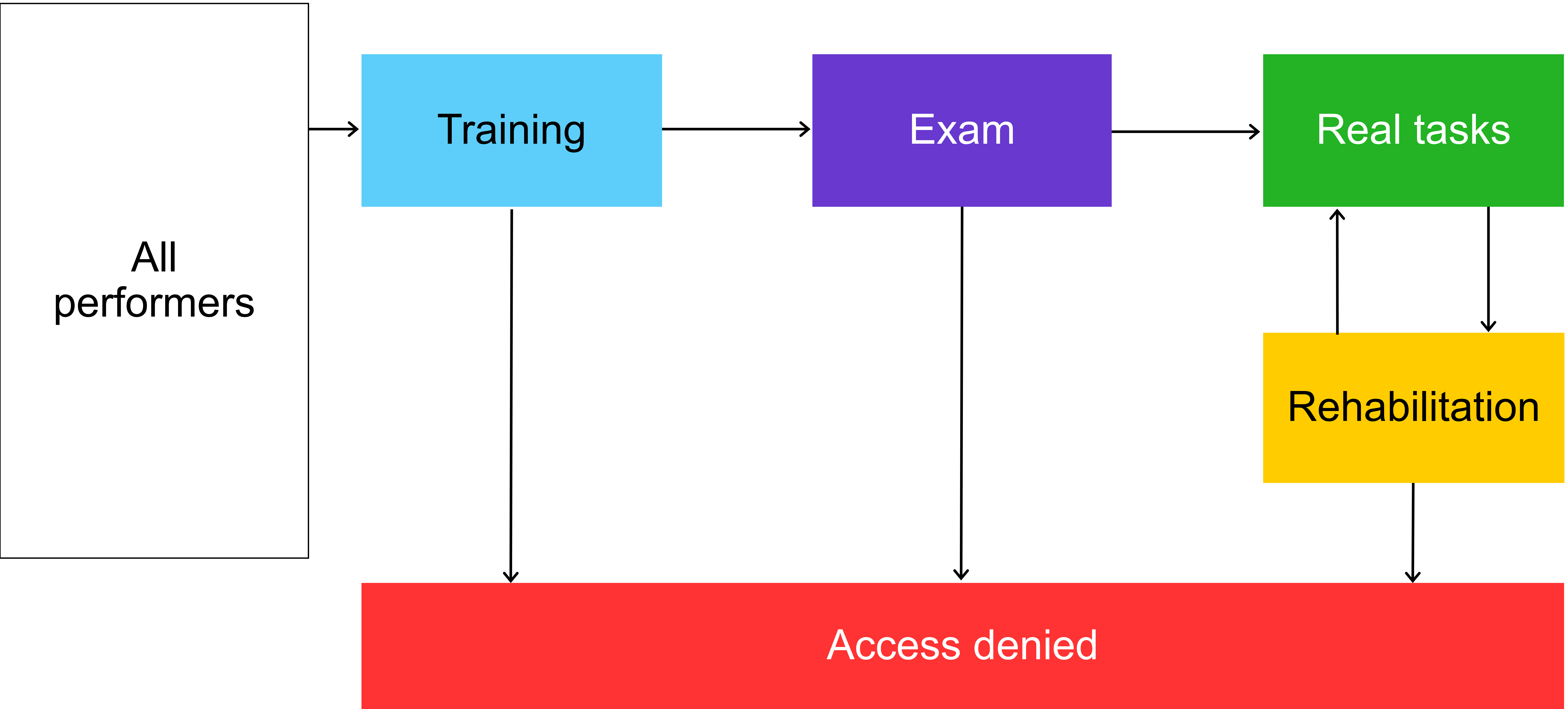
Control the results of training

- All tasks are control ones
- No hints and explanations

- A good exam should be:

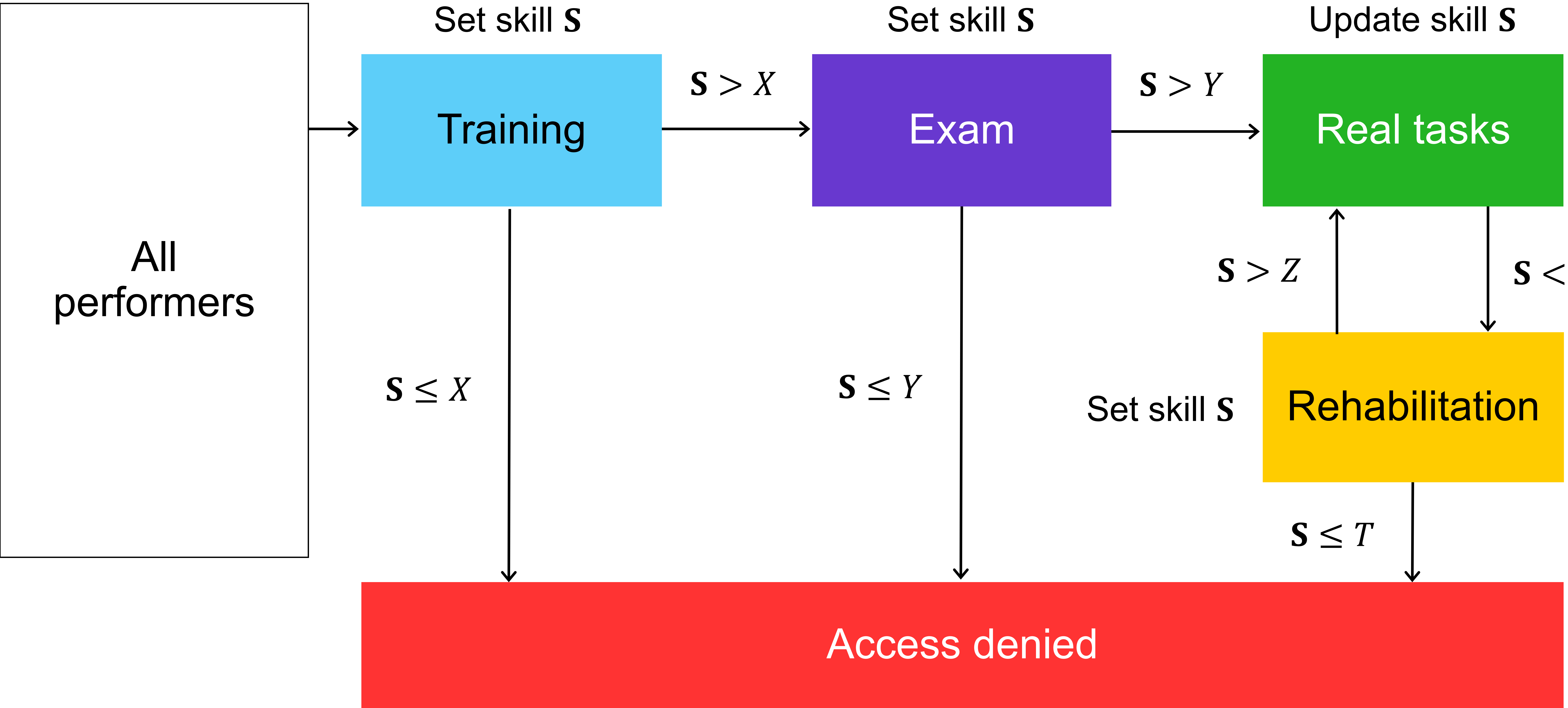
1. passable
2. regularly updated
3. small

Recommended life cycle of performers



Recommended life cycle of performers

Let quality be controlled by means of a skill $\mathbf{S}$



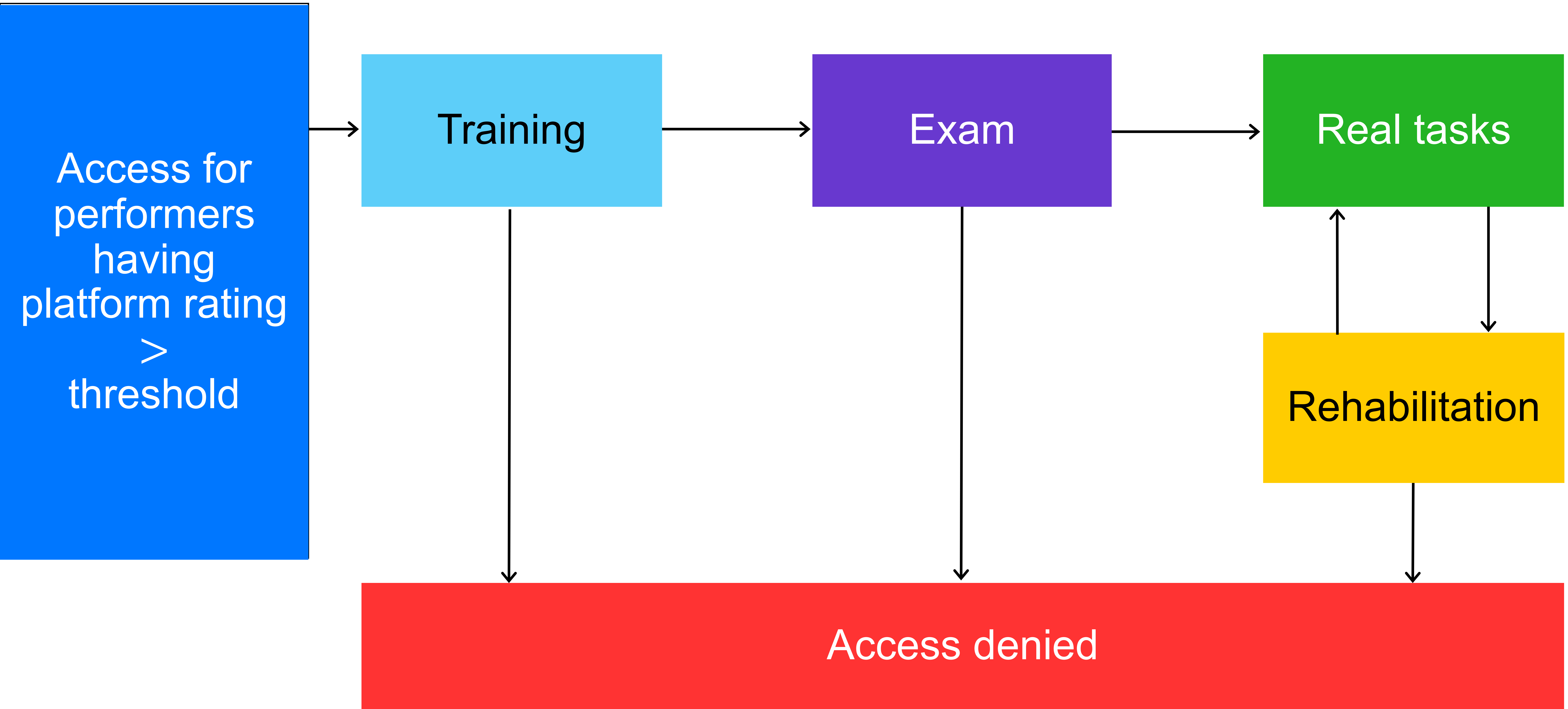
Rehabilitation task

Give a change to those who failed the skill threshold accidentally

- Rehabilitation is similar to an exam task, but with another access criterion
- Remind that there is a chance to observe low quality of a good performer

$$\mathbb{P}(\text{correct}) \approx \frac{1}{n}\sum_{i=1}^{n} y_i \pm \frac{1}{2\sqrt{n}}$$

Grant initial access to top performers



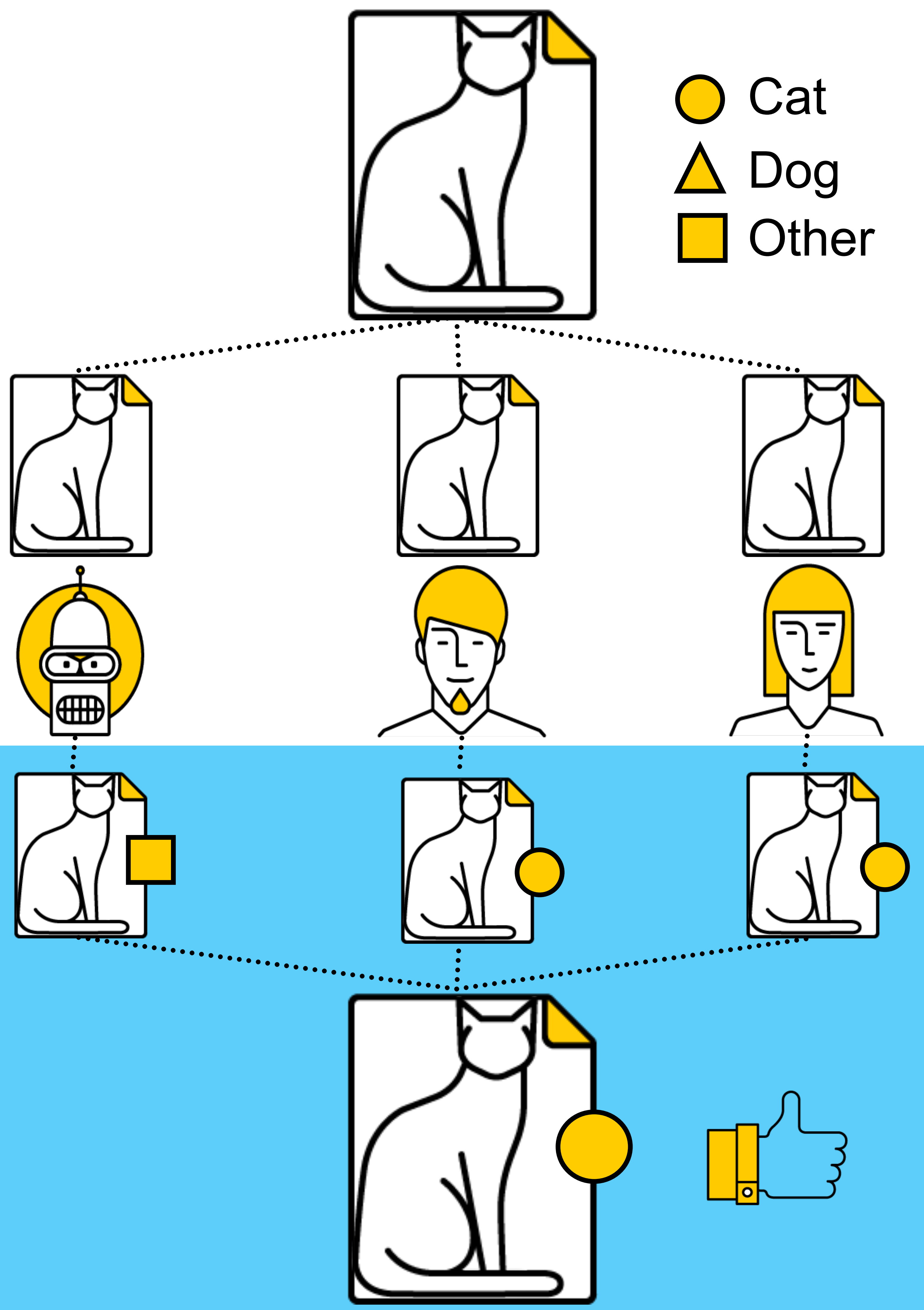
Platform rating *

is calculated based on performer behavior on all existed tasks within the platform

* is available on Yandex.Toloka

Aggregation

Aggregation



Upload multiple copies of each object to label

Performers assign noisy labels to objects

Aggregate multiple labels into a more reliable one

The simplest way:
assign the most popular answer (Majority Vote)

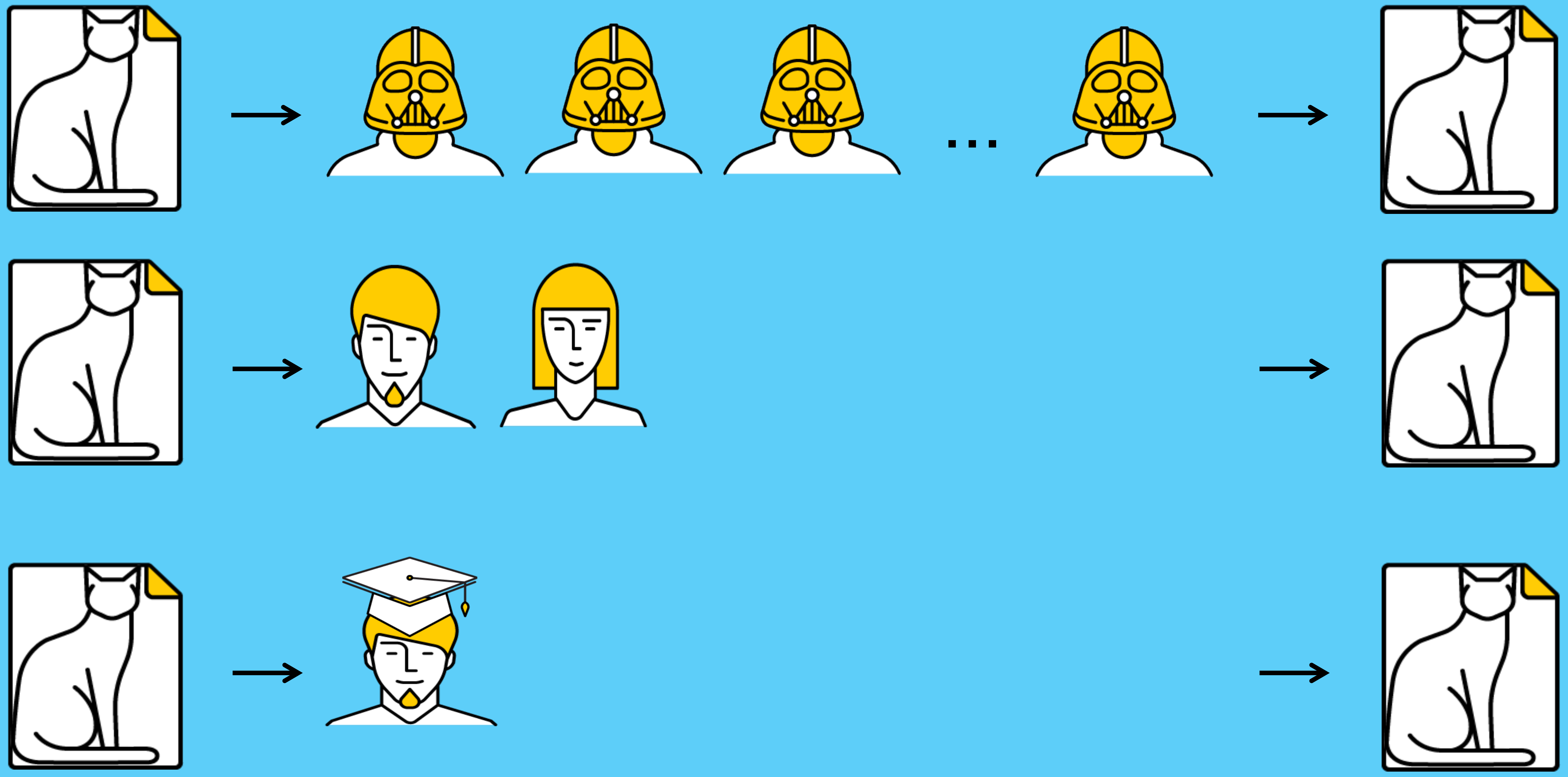
There are more sophisticated methods

Will be discussed in Part VI

Incremental relabelling & Pricing

Incremental relabelling

Obtain aggregated labels of a desired quality level using a fewer number of noisy labels



Several unknown performers

A few performers with known good quality

One expert with high quality

Will be discussed in Part VIII

Pricing depends on

Task design:

- Payment is made per a batch of microtasks (aka a task suite)
- Time required to perform a task: control hourly wage

Market economy aspects:

- The lower supply of performers is (e.g. due to specific skills), the higher price
- How quickly do you need accomplished tasks (latency)?

Result quality:

- Incentivize better performance by a quality-dependent price

Will be discussed in Part VIII

IF

Good decomposition

THEN

Simple instruction

Easy to use task interface

Performers do tasks with better quality

Easy to control quality

Standard aggregation models work well

Easy to control and optimize pricing

Yandex

Thank you!
Questions?

Alexey Drutsa

Head of Efficiency and Growth Division

adrutsa@yandex-team.ru

https://research.yandex.com/tutorials/crowd/cvpr-2020