



mobilenet

In this project you will use a machine learning model to recognise objects in photos.

The screenshot shows the Scratch IDE interface with a purple title bar labeled "mobilenet". The stage area displays a night-time photograph of a city skyline with a bridge. A yellow rectangular box highlights a specific area on the bridge. Below the stage, a script is displayed in the script editor:

```
define use the machine learning model
  set result to [recognizing... please wait v]
  set result to [recognise image <backdrop image> (label) v]
  stop to yes

define get ready
  hide
  stop to no
  set result to [please wait for the ML model to be ready... v]
  wait until [is the machine learning model ready to use?]

define reset test sprite
  switch costume to [shape 6 v]
  set size to [100 %]
  show
```

The script uses the "Control" blocks to define three procedures: "use the machine learning model", "get ready", and "reset test sprite". The "use the machine learning model" procedure sets the "result" variable to "recognizing... please wait", then "recognise image backdrop image (label)", and finally stops the script. The "get ready" procedure hides the sprite and waits until the ML model is ready. The "reset test sprite" procedure changes the costume to shape 6 and sets the size to 100%.



This project worksheet is licensed under a Creative Commons Attribution Non-Commercial Share-Alike License
<http://creativecommons.org/licenses/by-nc-sa/4.0/>

1. Go to <https://machinelearningforkids.co.uk/scratch> in a web browser

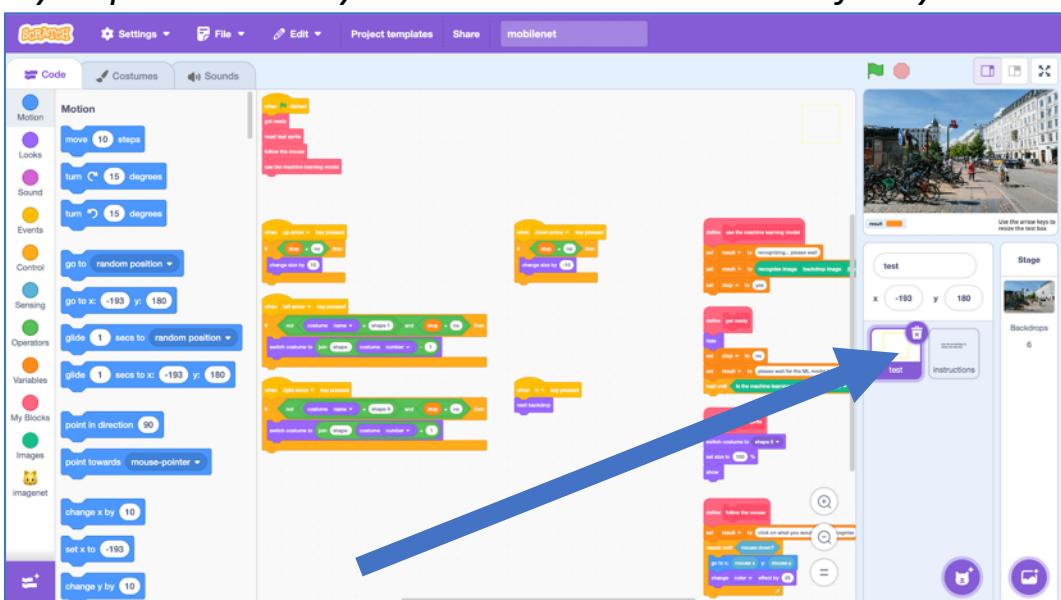
2. Click on “Project templates”

3. Click on “mobilenet”

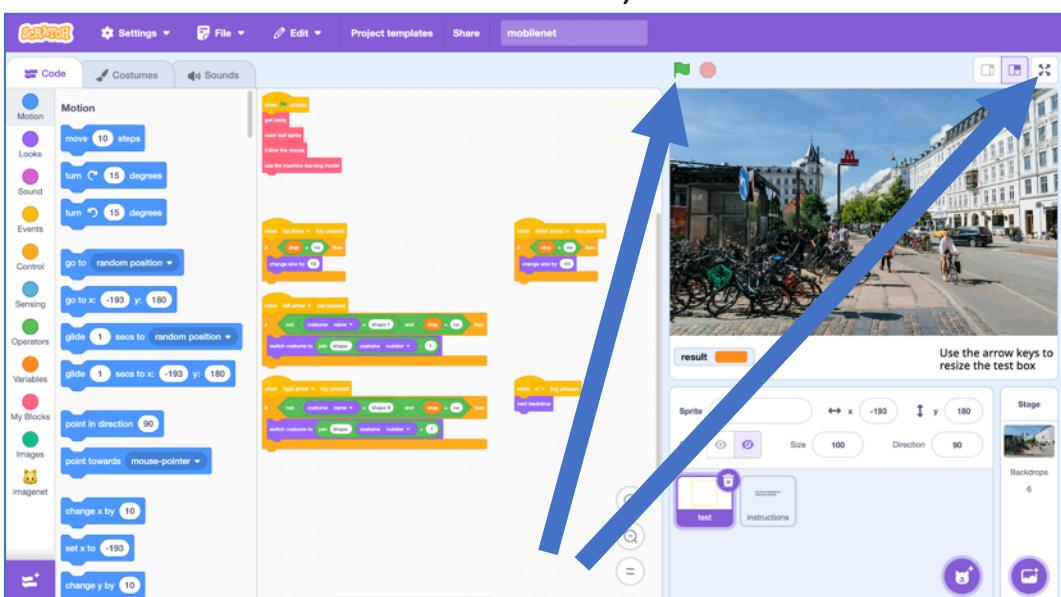
Wait for the project to download – it is large, so this may take a moment.

4. Look at the code in the “test” sprite

Try to predict what you think this code will do before you run it.



5. Click on the full-screen button, then click on the Green Flag



6. Choose an object in the scene – for example, the coffee mug

7. Move the mouse to move the flashing, coloured box.

Position the box so the object you've chosen is in the middle of the box.

Use the arrow keys to change the size and shape of the test box:

up – makes the test box larger

down – makes the test box smaller

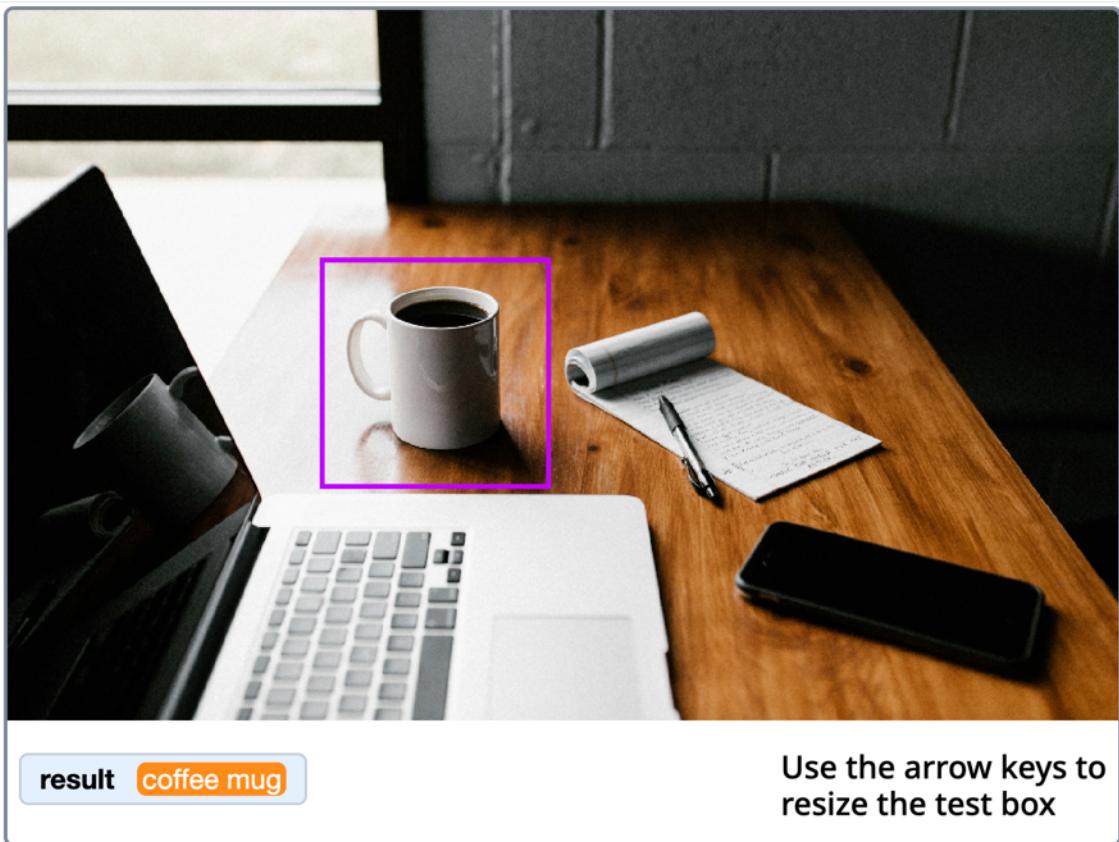
left – makes the test box taller / narrower

right – makes the test shorter / wider

8. Click the mouse

A machine learning model will attempt to recognize the image contained in the coloured box.

The result will be displayed at the bottom.

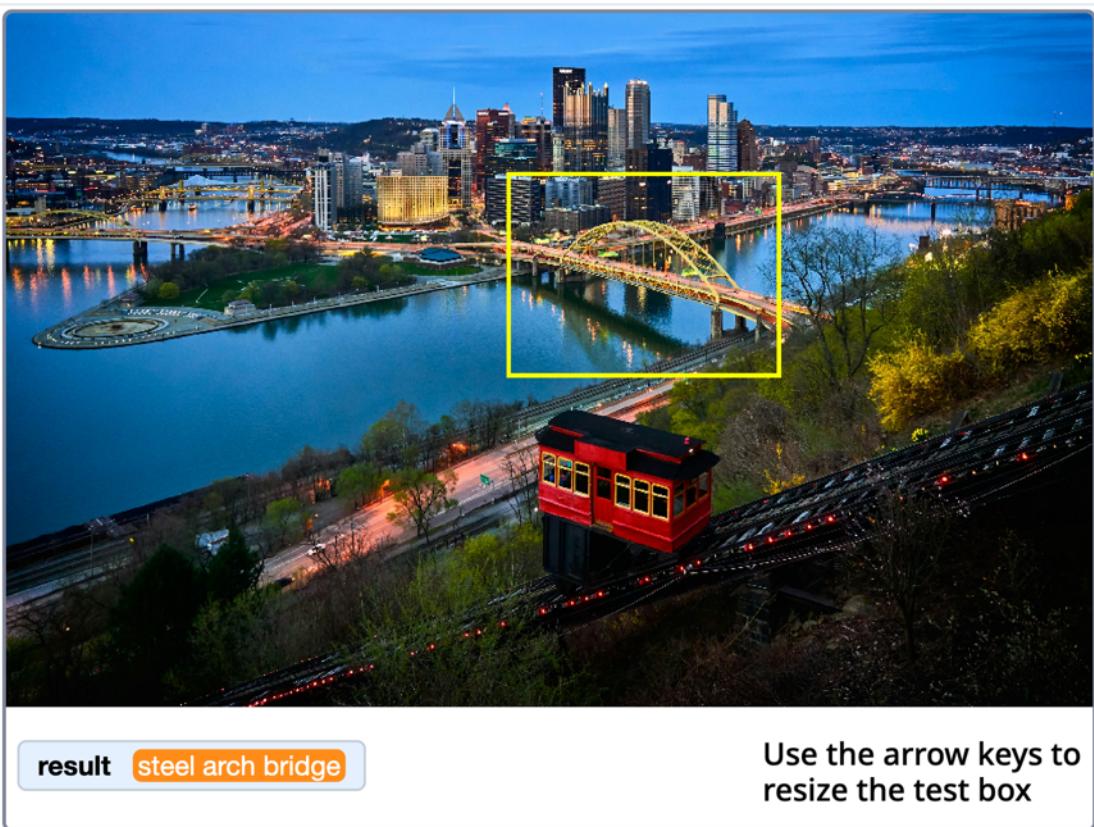


9. Press the **N** button on your keyboard to move to the next scene

10. Click the **Green Flag** and try recognising something else.

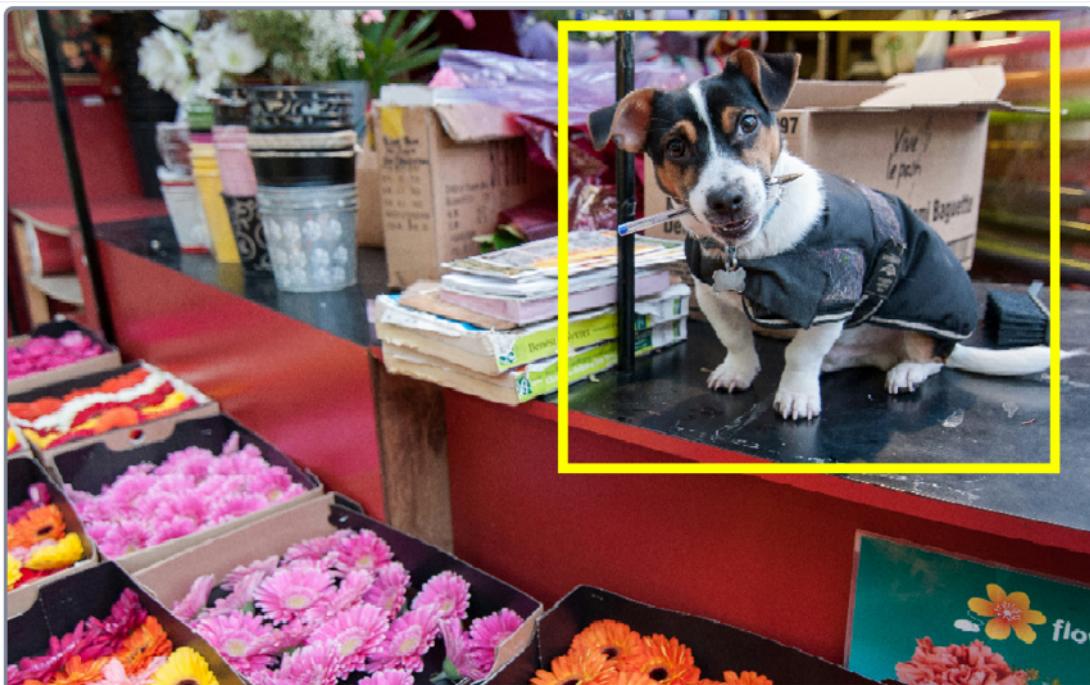


11. Go through a few scenes. Try a few objects in each scene.
What does the model recognize correctly? What does it get wrong?



12. How specific is the model in your testing?

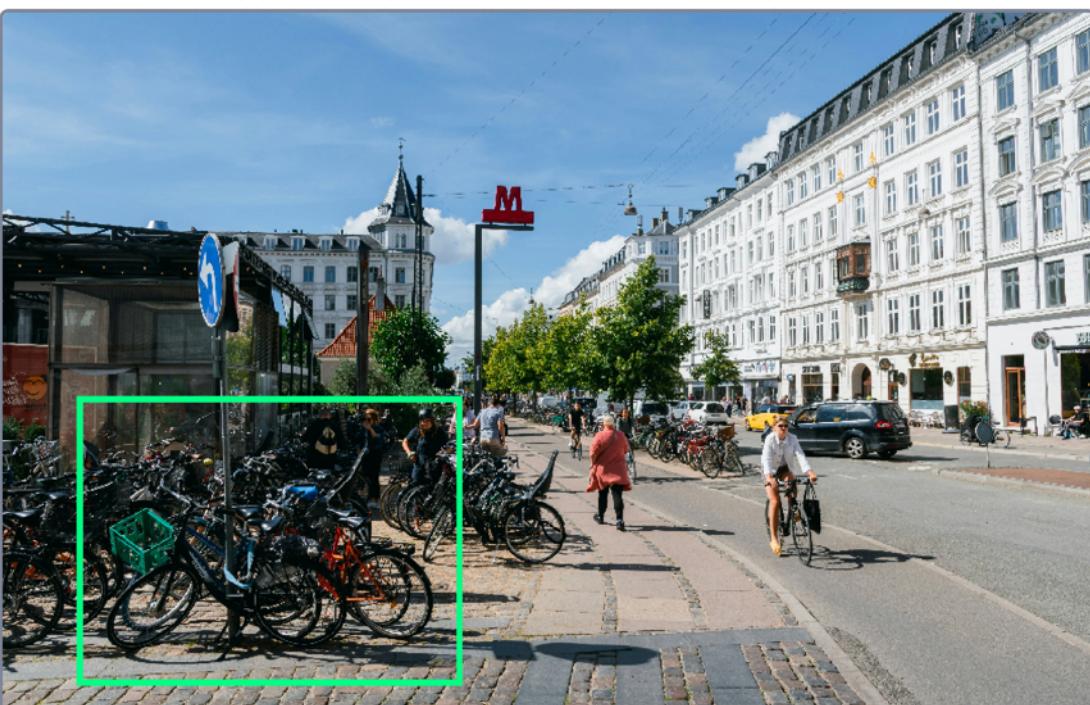
For example, instead of “dog”, it attempted to recognise a specific breed.
Was it correct?



result basset

Use the arrow keys to
resize the test box

13. What happens if you select multiple objects?



result mountain bike

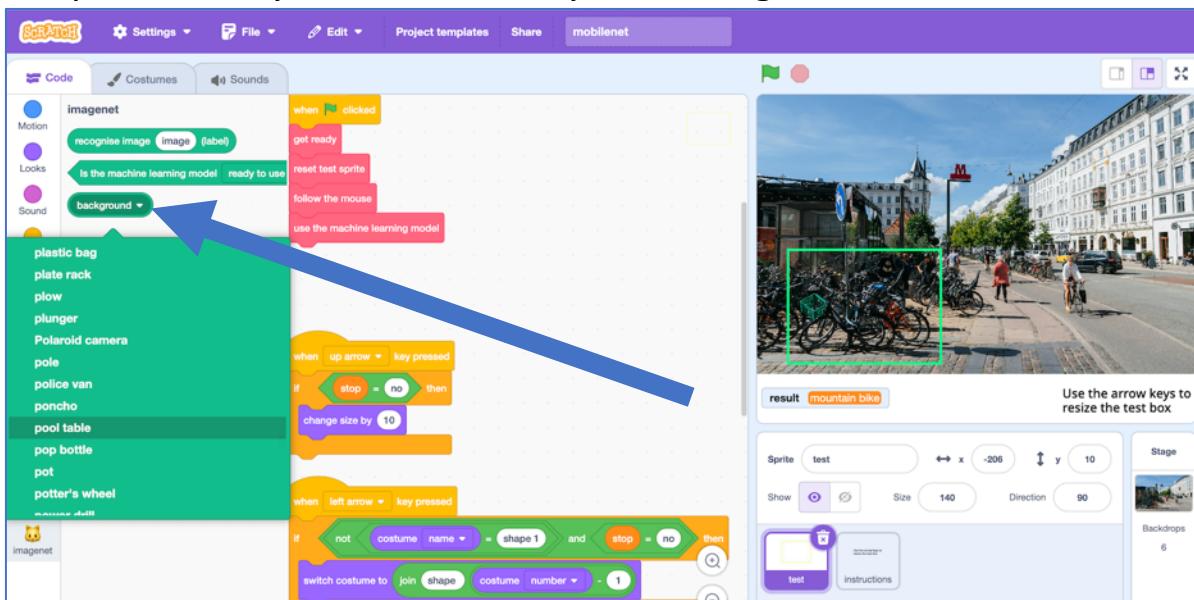
Use the arrow keys to
resize the test box

14. What objects has this model been trained to recognize?

Exit full-screen mode and scroll through the list of labels available in the reporter block in the imangenet section.

You should see a very long list of classes. These are the objects that the machine learning model has been trained to recognise.

The predictions you have seen in your testing will all be in this list.



15. What do you think of this model?

Like all machine learning models, it makes mistakes. Some objects you wanted it to recognize will have been incorrectly recognized.

How many did it get correct?

What was it good at recognizing? What was it bad at recognizing?

16. How many pictures you think were collected to train this model?

Where do you think the images were collected from?

Try to guess before you move on to the next step.

Write down your guesses to help you remember.

17. In a new web browser window, go to <https://ibm.biz/mobilenet>

When we use a machine learning system, it is helpful if the creators of the system publish the “Model card” – where they tell you how it was created.

The model you have been using is **Mobilenet v3**.

This page has the model card for this machine learning model.

18. Have a look at the model card

Model cards are complicated, so don't worry that you won't understand all of it. But there are some interesting things to find here.

The screenshot shows the 'mobileNet_v3' model card on Kaggle. The left sidebar has 'Models' selected. The main content area shows 'Model Details' with an 'Overview' section. It mentions that MobileNet V3 is a family of neural network architectures for efficient on-device image classification and related tasks, originally published by Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig Adam in 2019. The 'Downloads' section shows 1.22M total downloads with 193K in the last 30 days. The 'Tags' section includes 'Image Classification' and 'Image Feature Vector'. The 'Usability' section shows a score of 9.33. A blue arrow points to the 'Authors' section, which lists the names of the creators.

For example, here are the names of the people who first created it.

There are two versions: "large" (22.1 MB) and "small" (10.34 MB). The "large" model has a higher accuracy (it makes more correct predictions).

The screenshot shows the 'mobileNet_v3' model card on Kaggle. The left sidebar has 'Models' selected. The main content area shows 'Model Details' with an 'Overview' section. It mentions that MobileNet V3 uses a multiplier for the depth (number of features) in the convolutional layers to tune the accuracy vs. latency tradeoff. The 'Downloads' section shows 1.22M total downloads with 193K in the last 30 days. The 'Tags' section includes 'Image'. The 'Usability' section shows a score of 9.33. A blue arrow points to the 'Comparison Table' section, which provides a quick comparison between the 'large' and 'small' variants. The table shows the following data:

Size	Depth multiplier	Top1 accuracy (%)	Pixel 1 (ms)	Pixel 2 (ms)	Pixel 3 (ms)
Large	1.0	75.2	51.2	61	44
Large	0.75	73.3	39.8	48	32
Small	1.0	67.5	15.8	19.4	14.4
Small	0.75	65.4	12.8	15.9	11.6

This TF Hub model uses the TF-Slim implementation of `mobilenet_v3` as a large network with a depth multiplier of 0.75.

You used the "small" model in this project. Why do you think the project used the small version if the large model gives better answers?

*The model card explains that the training data used to create this model was the “**Imagenet**” set of photos.*

This is such a famous training set that they don’t explain what it is, so we need to look elsewhere for more detail about this.

A screenshot of a Kaggle model card for "mobilenet_v3". The left sidebar shows navigation options like Home, Competitions, Datasets, Models (which is selected), Code, Discussions, Learn, and More. The main content area is titled "mobilenet_v3" and includes sections for Model Card, Code (11), Discussion (1), and Competitions (0). The "Training" section contains text about the checkpoint exported from the model and its weights being obtained by training on the ILSVRC-2012-CLS dataset. The "Fine-tuning" section provides instructions for consumers of the model.

You can find information about imangenet at <https://image-net.org>

A screenshot of the ImageNet website. The header features the "IMAGENET" logo and navigation links for Home, Download, Challenges, and About. Below the header, a text block describes ImageNet as an image database organized according to the WordNet hierarchy, containing over 14 million images and 21841 synsets indexed. At the bottom of the page, there is a note about not being logged in and links for Login and Signup.

*This tells you that imangenet contains **over 14 million images**. Compare that with your guess. Were you close?*

It describes how images were found from searching across the Internet, especially using searches in Google, MSN, Yahoo, and Flickr.

*All of these 14 million images needed to be labelled by a human to train the computer how to do this. This was done by **crowd-sourcing** (paying lots and lots of humans to each label some images).*

The screenshot shows a web browser displaying the 'Dataset' section of the ImageNet page on Wikipedia. The left sidebar contains a navigation tree with links to 'Top', 'History', 'Significance for deep learning', 'Dataset' (which is expanded), 'Categories', 'Image format', 'Labels and annotations', 'ImageNet-21K', 'ImageNet-1K', 'Later developments', and 'History of the ImageNet challenge'. The main content area under 'Dataset' includes a paragraph about the annotation process, a note about being the world's largest academic user of Mechanical Turk, a summary of the original plan for 50M images, and a detailed breakdown of the dataset statistics from April 30, 2010.

Dataset [edit]

(Top)

History

Significance for deep learning

Dataset

Categories

Image format

Labels and annotations

ImageNet-21K

ImageNet-1K

Later developments

History of the ImageNet challenge

Dataset

ImageNet [crowdsources](#) its annotation process. Image-level annotations indicate the presence or absence of an object class in an image, such as "there are tigers in this image" or "there are no tigers in this image". Object-level annotations provide a bounding box around the (visible part of the) indicated object. ImageNet uses a variant of the broad [WordNet](#) schema to categorize objects, augmented with 120 categories of [dog breeds](#) to showcase fine-grained classification.^[6]

In 2012, ImageNet was the world's largest academic user of [Mechanical Turk](#). The average worker identified 50 images per minute.^[2]

The original plan of the full ImageNet would have roughly 50M clean, diverse and full resolution images spread over approximately 50K synsets.^[18] This was not achieved.

The summary statistics given on April 30, 2010:^[20]

- Total number of non-empty synsets: 21841
- Total number of images: 14,197,122
- Number of images with bounding box annotations: 1,034,908

This was even more work than asking people to label 14 million images.

People make mistakes, so every picture had to be labelled by a few different people.

If the people all agreed, then their answers could be trusted.

If they didn't agree, more people needed to label the image.

Every training image was labelled by between 2 – 5 people.

Did you guess what a huge project it was to train this model?

19. Compare what you've learned about the mobilenet model with what you thought of how good the model is at recognizing things.

Remember, you've been using the "small" variant of the model.

The "large" variant is a little better than what you've seen.

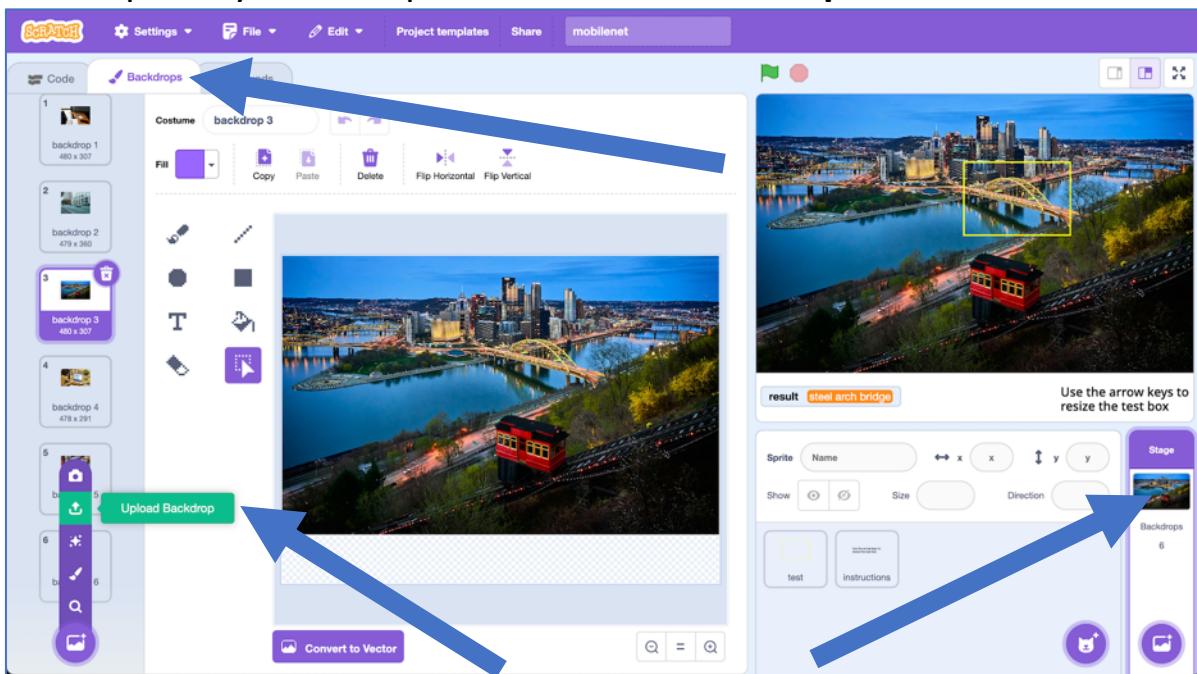
20. Find a new picture to test the model with

Search the web for scenes, landscapes, photos from towns, office photos, photos of shops...

The best images for this have several objects you can use to test the model.

You can take your own digital photos if you want.

21. Upload your new photo as a new backdrop



22. Try testing mobilenet with your new image. *How does it do?*

What have you done?

You've been using one of the most well-known models in machine learning.

You've seen that machine learning models are not perfect. They are impressive. They can do a lot. But they make mistakes.

You've learned what a “model card” is. Not all artificial intelligence systems publish a model card, but it is a good practice that encourages transparency.

You have seen the amount of work that goes into collecting the training sets needed to create real-world machine learning models.