2019 PowerAI Experts

PowerAI Vision Hands-on Lab

Deeper Dive into PowerAI Vision

Labeling video and modifying a Python program to track objects within the video

Original material:

Michael Hollinger

Ke Wei Sun

Table of Contents

PowerAI Vision Hands-on Lab 1

Introduction to the Lab Exercises 3

Lab environment 4

Prerequisites for the lab and laptop setup 4

Section 1 – Uploading the video 5

Section 2 – Labeling objects within the video 6

Section 3 – Training your first model 10

Section 4 – Using that model to perform auto-labelling 11

Section 5 – Creating a more accurate model 13

Section 6 – Quick test of your model 15

Section 7 – The time tracking Python program 15

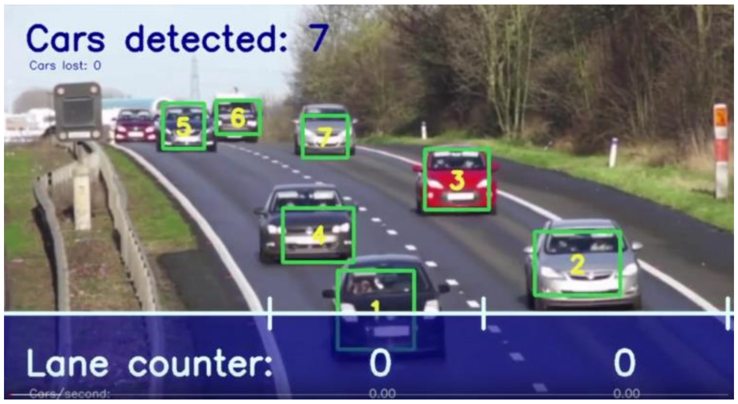
Introduction to the Lab Exercises

This hands-on lab provides you with the understanding and hands-on experience to be able to demonstrate PowerAI Vision to a client. The lab provides more than just the ability to label data and train a model for object detection or image classification. We will go through those aspects, but more importantly, this lab will give you some hands-on experience with a Python application that performs inferencing with the model you create with PowerAI Vision and lets you do some interesting things with those detected objects.

We will use stock highway video for our example but you can do this with any video. We will build a model that can detect cars on a highway but this could be modified to count products on a conveyer belt. That part is simple enough and is an interesting demonstration. But then we will take it one step further. By using a fairly simple Python application (run via a Jupyter Notebook), you will be able to take a second highlight video and break that video down into frames and call your PowerAI Vision built model to find the cars and count them as they come into view and label them. The best part, however, is that the application will also keep track of how many cars appear in the video.

This lab helps you take a video that looks like this:

And turn it into this:



We have written the Python code so that you can use it for any video of this type to detect objects in the video and count the number of objects that appear on screen during the course of the video.

Lab environment

In this lab we are using PowerAI Vision 1.1.2 to build and train a model for detecting cars object detection. We will also use Python (version 3.7) in a Jupyter Notebook to execute the sample application to build a video with the detected objects labeled inside the video. Note that we are only using the Python application to break down the video into frames, pass those frames into PowerAI for inferencing using a REST API, and then getting back a JSON document with the objects detected within the frames. There is no need for AI frameworks on your laptop in this hands-on lab because all of the deep learning (DL) work is being done on the PowerAI server side.

Prerequisites for the lab and laptop setup

On your laptop you need to install Docker. You can download docker from using the links below:

Windows: https://hub.docker.com/editions/community/docker-ce-desktop-windows

Mac OS 10.12 or higher: https://hub.docker.com/editions/community/docker-ce-desktop-mac

Section 1 – Uploading the video

As part of the hands-on lab class you should have been given access to an instance of PowerAI Vision. The first step, after logging in, is to add a data set to train with. You will download the video traincars.mp4 which is located in a box folder located here (download this file to your laptop)

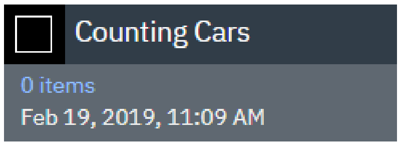
___1. Click on ***Data Sets*** in the top menu.

___2. Click on ***Create new data set*** with the big plus sign in the box.

___3. Specify the name of the data set (in the example here we will call it "Counting Cars") and click ***Create***.

___4. You will now see a square with the data set name you created (and the current date it was created) but the square will be empty

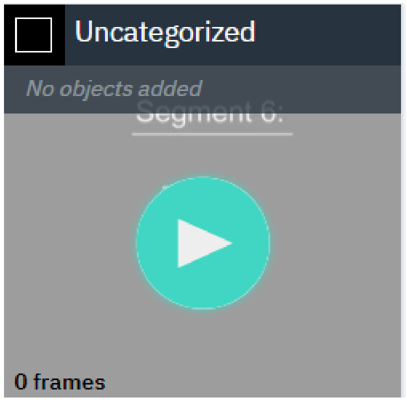
___5.



___6. Click on the Counting Cars data set

___7. Download traincars.mp4 which is a 94 second clip showing traffic on a highway. We will use the video to identify and count cars on the road. Download the file from this box link: https://ibm.box.com/s/a1mllvzxlho6snkm9z7elypgha7nzw79

___8. On the next screen there will be a place where you can either drag the training video onto (***Drop files here***) or you can click on ***Import files*** button to navigate to the location of the video we want to use for training. Select the traincars.mp4 file and you will see it uploaded into this data set.



Section 2 – Labeling objects within the video

If we were performing image classification or object detection (as you learned about in the PowerAI Vision lesson of the PowerAI Level 2 course http://smarter-sellers.mybluemix.net/#/systems/course/717/details), you would be able to work directly on the image to classify it or label objects within the image. However, this file is a video and so we first must select a set of frames that we want to use for labelling objects.

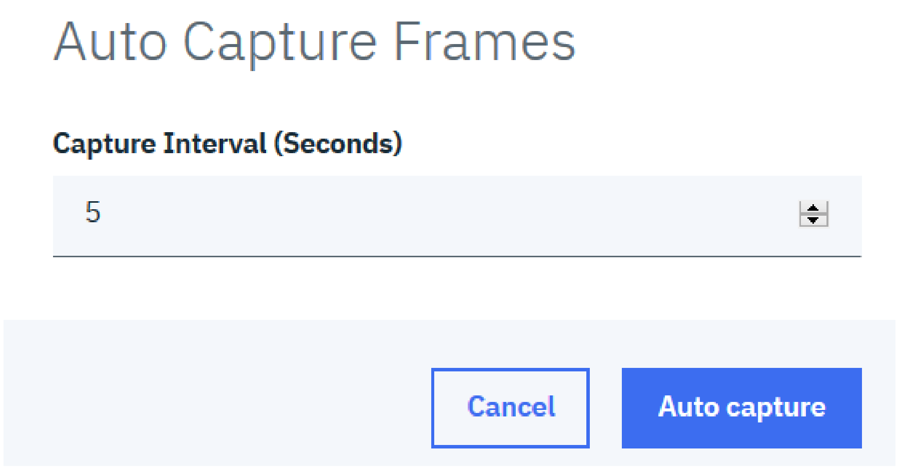
___9. First click on the image (the checkbox next to the *Uncategorized* label) and then click on the ***Label objects*** button.

On this screen, you have 3 options

- Capture frame – this will take the current frame from the video and capture that frame as an image. You can play the video and pause it on the frame you like and then click capture video to capture a frame.
- Auto capture frames – this allows you to have PowerAI Vision automatically capture a sample frame every N seconds. This is the option we will use in this step of the demo.
- Auto label – this will allow you to use a trained model to automatically label more objects found in various frames within the video. We will use this option in a future step in this hands-on lab.

___10. Click on Auto capture frames

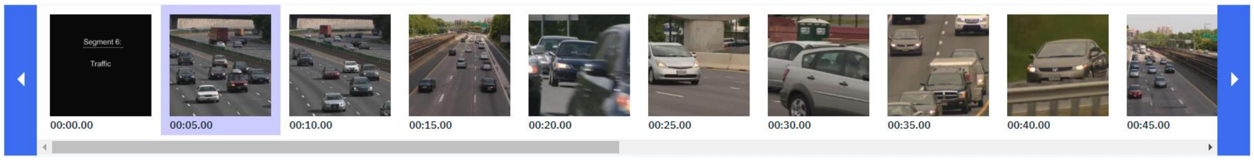
___11. In the pop-up window we want to set the Capture Interval (seconds) to 5 seconds and then click on ***Auto capture***. This will give you several frames to label.


Auto Capture Frames

Capture Interval (Seconds)

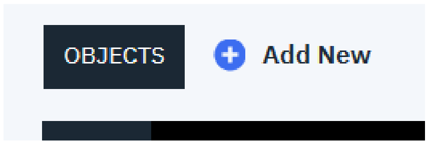
5

___12. At this point you will now see the original video on the upper left side of the screen. Along the bottom of the screen there are 20 frames of the video (one for every 5 seconds) with the first frame (at time 00:00.00 selected). On the upper right corner of the screen is the frame that is selected in the lower row of the screen.



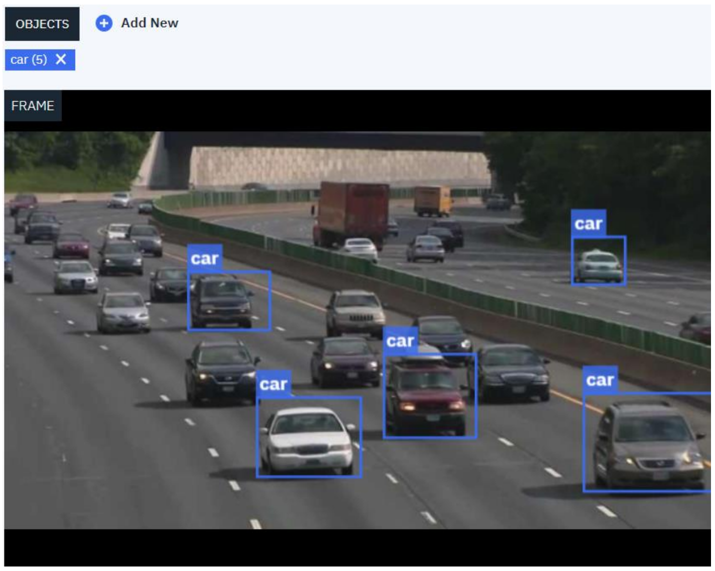
___13. On the upper right side of the screen you can see the currently selected frame and above the image is the ability to add new objects. In our example we will only label cars.

Click on the ***Add New*** button and type in “car” and then click on *OK*.



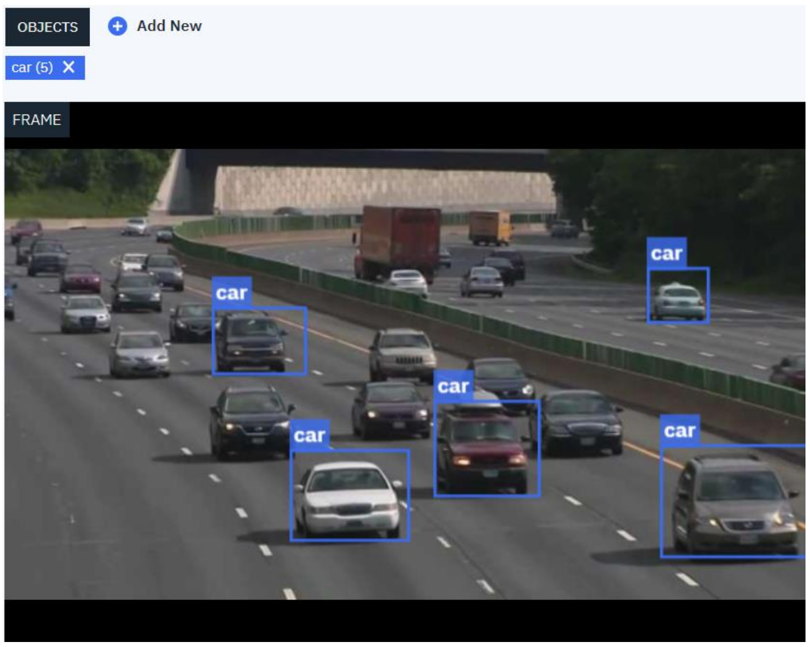
___14. You can now label the frame by selecting each car drawing a bounding box around each car. You do not have to label every car in the image.

___15.



Note that if you accidentally create a bounding box and want to delete it, you can click on the box (not the label but click on the edges of the box itself) and hit the delete key to remove it. Or if you want to change the size of the box, simply click on an edge of the box and you will see the 4 corners now have white boxes on them that will allow you to adjust the size of the bounding box.

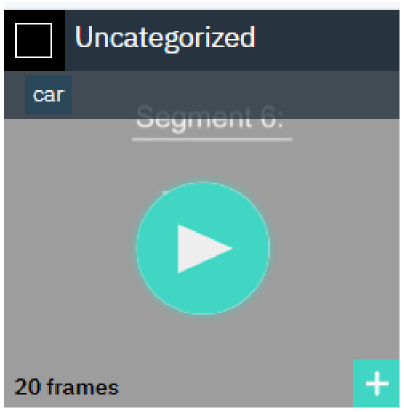
___16. Now select the frame at the bottom of the screen from time index 00:05.00 and label the cars on that screen as well. Notice that in this frame there are different cars as traffic keeps moving.



___17. Repeat the labeling in step 14 for the all 20 frames in the video. Note that as a data domain expert, it's up to you to decide what should be considered "a car on the screen on this frame". If only half a car shows should you label it? If the object is a truck or a bus, do you want to count it? That is up to you to decide.

___18. After you have completed all the labeling, click on ***Done editing*** in the top left.

You will now be back on the Data Sets page and you will see the icon for the video file now indicates the 1 car label and that it contains 20 frames. If you click on the green plus sign you will see all 20 frames and what objects are labeled within each frame



Section 3 – Training your first model

Now that we have 20 frames and multiple objects labeled, we will use these images to train a model. As you might expect, the model may not be very accurate because we have not labeled enough images. We will take care of that in Section 4.

___19. On the data sets page click on the ***Train model*** button

___20. This will bring up the model training options. First is the model name. The default for model name is <<data set name>>_model so in our case, **Counting Cars_model**

___21. The type of training that we are doing here is **Object detection** because we want to detect which company logs we can find within the video. Click the ***Object detection*** radio button.

___22. When you do this you will be given two models to choose from. The first is optimized for accuracy and the second is optimized for speed. For this exercise, since we only have 18 frames to train, select ***Optimized for accuracy***.

___23. Under Advanced options we suggest you reduce the Max iteration to 1000 as you won't need 4000 iterations for this small data set.

___24. Click on the ***Train*** button.

You will now be brought to the training page which shows an estimated amount of time to train the model and the Loss vs. Iteration graph.

It will likely take several minutes to complete training. If you are demoing PowerAI Vision to a client this is a situation where you will likely want to have the model pretrained and then (just like on a cooking show where the meal is already cooked in the oven and they just take it out on the show), you can show a client how easy it is to label and how easy it is to train and then just jump over the training step to the already trained model. When the training is complete you will have the option to see the *Model details* or you can *Deploy* the model. If you click on the model details at this point you will likely see very low accuracy and low precision due to the fact that we had very little data to train on.

___25. We are going to use this model to automatically label more data. Click on ***Deploy***

model. This will pop up a window that will allow you to change the model name and then click ***Deploy***. You can set the name to anything you like. In my example, I called it “Counting Cars_model_poor” because I know I have not yet labeled enough data to get a good model.

Section 4 – Using that model to perform auto-labelling

We now want to go back and use the auto-labelling feature of PowerAI Vision and leverage the model that has low accuracy to label some data for us and so that we don’t have to do it all manually.

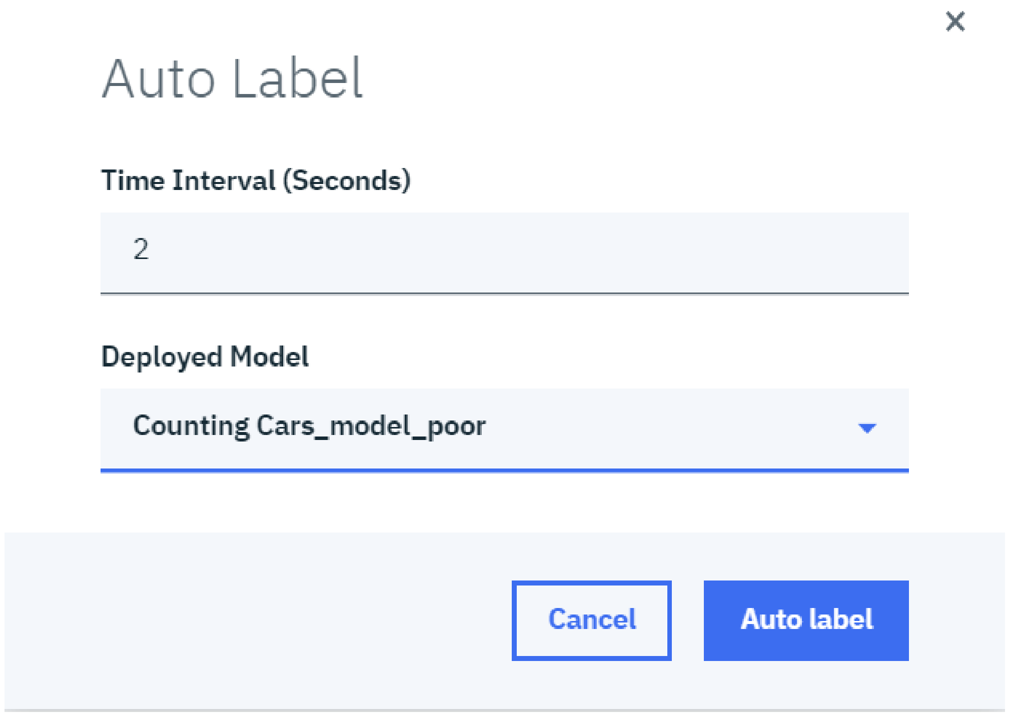
___26. Click on the ***Data Sets*** tab at the top of the page.

___27. Click on the Counting Cars data set.

___28. Select the video file we previously labeled and then select ***Label objects***.

___29. You will be brought back into the screen that was used in the labelling exercise. This time click on the ***Auto label*** button below the original video file.

___30. A pop-up screen will now ask you for the time interval you want to use for selecting frames from the video. This time let’s select a frame every 2 seconds and we will use the Counting Cars_model_poor that we trained in section 3.


Auto Label

Time Interval (Seconds)

2

Deployed Model

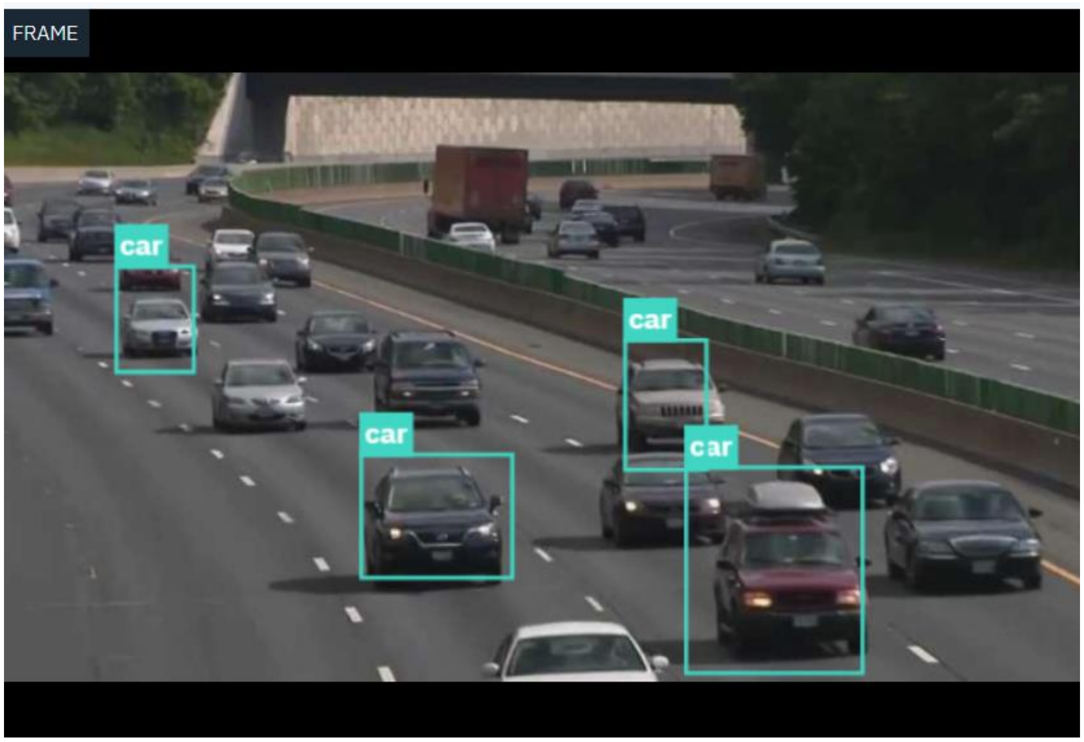
Counting Cars_model_poor

Then click ***Auto label***.

This will select out a frame from the video every 2 seconds and perform inference on that frame to try to detect new objects within the frames.

___31. You will notice now in the timeline on the bottom of the screen that there are frames for both every 2 seconds and frames for every 5 seconds (that we had previously labeled). You will also notice that the frames that are taken every 2 seconds have a light green box around the auto-labeled objects and the labels we previously created are the blue boxes. One final thing to note that for frames that are both at 2 seconds intervals **and** 5 second intervals, you will find two frames (i.e. two frames at 0 seconds, two frames at 10 seconds, two frames at 20 seconds and so on). You don't need two have both frames so just select the new auto-labeled frame and click on the ***Delete Image*** (the trash can) for the frames you don't want.

___32. If you look at the frame from the two seconds mark (00:06.00) you will likely see that it did an ok job of detecting cars but it is still not perfect.



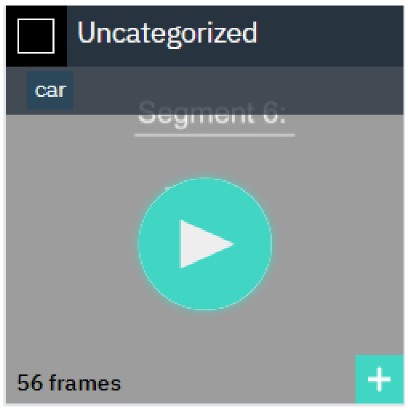
___33. Go through each of the frames and adjust the boxes as needed. You may notice that some frames were not labeled When you see these frames with unlabeled cars, just draw a bounding box around the car. If a car is mislabeled, you should click in the label and delete the bounding box..

___34. Continue to go through all the frames to make sure all the cars are labelled accurately.
NOTE: Generated labels are all in lower case characters whereas labels you define can be upper or lower-case characters.

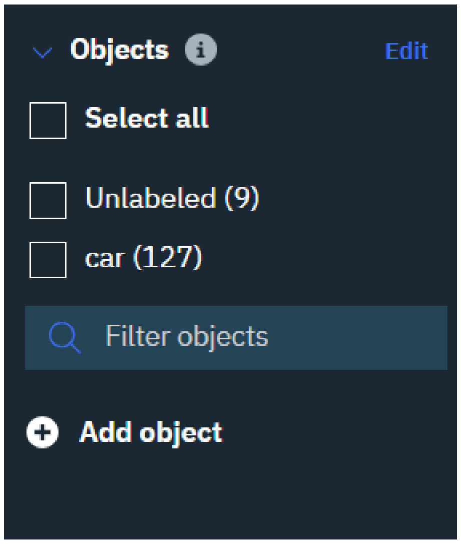
___35. As you can see, the auto labeling approach is much easier than manually labelling each frame. When complete click ***Done editing***.

Section 5 – Creating a more accurate model

We will now use the auto-labelled data to build a more accurate model. In the Data Sets page you will now see that there are somewhere around 56 frames depending on how many you deleted that had no objects or were duplicates.



Click on the left side navigation on the Objects drop down twisty you can see how many of each object has been labelled. Note that if your browser is not very wide, the left-hand navigation will be collapsed behind the hamburger menu (3 horizontal bars). Click the hamburger to see the Objects drop down list. In my case there are 9 frames with no objects in them and there are 127 images of cars (your results may vary).



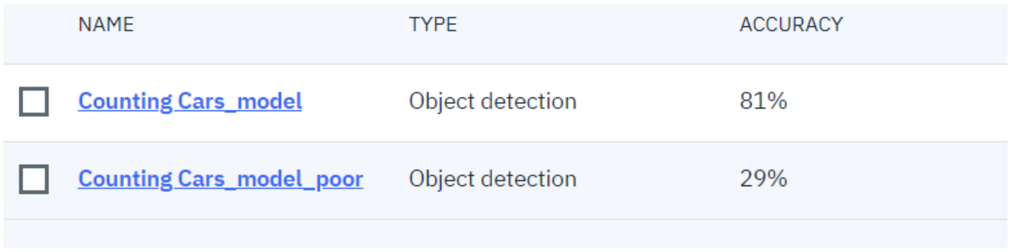
___36. As in the previous example, click on ***Train model*** then click on the ***Object detection*** radio button and train the model ***Optimized for accuracy***. Under advanced options you can select 1000 iterations for purposes of this hands on lab and then click ***Train***.

NOTE: You may not have an available GPU resources to conduct this second model Training. In this hands on lab class, each group has been given an instance with just 1 GPU. If you have a model deployed it will consume that GPU and therefore you must delete the deployed model in order to train a new model. Click on the ***Deployed Models*** tab and delete your current *Counting Cars_model* to make a GPU available in order to train a new model.

___37. When the model has been trained you can click on ***Model details*** and you should see a much higher accuracy and precision. At this point, if you were creating a demo for a client you can perform this step again by training every 1 second to gain more accuracy but for the purposes of this demonstration we will simply use this new model for testing

___38. Click on Deploy Model

___39. At this point you will now see two models deployed and you can delete the older, less accurate model.



	NAME	TYPE	ACCURACY
☐	Counting Cars_model	Object detection	81%
☐	Counting Cars_model_poor	Object detection	29%

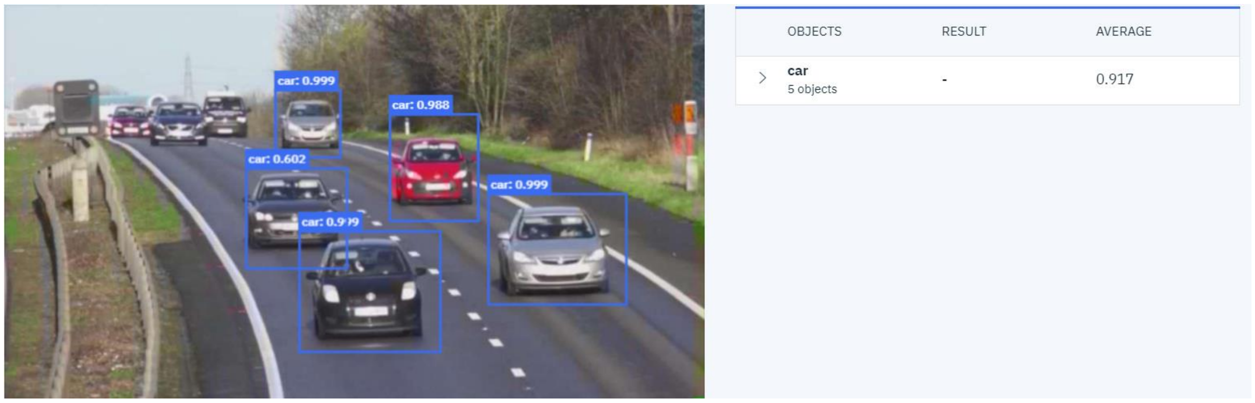
Section 6 – Quick test of your model

You can test the model that has been deployed. Simply click on the name of the model which brings you to a page that shows you the API endpoint (you will use this in the Python application in Section 7). You have the option to launch a tool that will take a test video and perform inference on frames within that test video; to quickly test the deployed model, we will just test a single image.

___40. Download the sample image from this Box location https://ibm.box.com/s/s5dazrfij8ql4aw55cq1lz2izlpuefxn

___41. Drag and drop that sample image into the *Test Model* section of the page in the ***Drop files here*** box

___42. PowerAI will perform inference on the image and detect the advertisers it can find in the image. In my example it was able to find all 4 advertisers with a very high confidence value.



OBJECTS	RESULT	AVERAGE
car 5 objects	-	0.917

Section 7 – The car counting Python program

It's quite interesting to see that PowerAI Vision is able to detect the objects in a previously unseen frame/image. However, what a client really wants is the ability to detect objects within a video (as that's the source we were using to build the model in the first place). We won't have clients using the PowerAI Vision "Test model" page to run their business …they will use the inferencing API within their own applications to detect objects within a video stream.

To that end, you will now use an application we have created to break down a previously unseen video into frames, call PowerAI Vision to perform the inferencing using the REST API, and then perform a few actions on each frame (labeling the object with a colored dot and keeping track of the total amount of time each advertisement was visible in the video).

To perform this, we will use a test video (which is from the same hockey game but was not part of our training data). This test video is called testcars.mp4 and has already been preloaded to your Docker container in the /countingcars directory. To run this test, you will need to start your Docker container and run a Jupyter notebook.

___43. You will need to pull down the Docker Container from the Docker Hub. To do this, you will need to open up a command window and type in the following command:

docker pull nrudnick/countingcars:base1

___44. To start the lab which is located in your Docker container, you should execute the following command.

docker run -it --rm -p 8888:8888 nrudnick/countingcars:base1 bash

___45. To run the Jupyter notebook, you should execute the following command.

jupyter notebook --ip=172.17.0.2 --allow-root --no-browser --notebook-dir=/countingcars

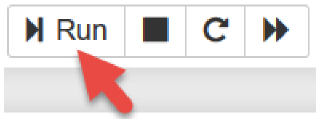
___46. When you started your Jupyter notebook, you will be given a URL that you can use to open up your notebook. You will need to change the ip address of that URL to localhost (or 127.0.0.1) as the example below shows. The token you will use is unique to your session and must be used. You should now navigate to the browser of your choice (I recommend Google Chrome but Mozilla and Microsoft Edge will also work) and open the Jupyter notebook you just started using the directions above. An example is shown belowl:

http://localhost:8888/?token=<token>

___47. The notebook we are going to be using for this lab is called **Counting_cars..ipynb**. Click on this notebook and a new window will open.

This will open up the Python application in the notebook. If you are not familiar with a notebook, there are a series of cells that either contain comments or code. The first cell gives you an overview of the application we are about to go through.

The first code cell is labeled "In [1]". You can click on the run button at the top of the screen to execute each cell one at a time.



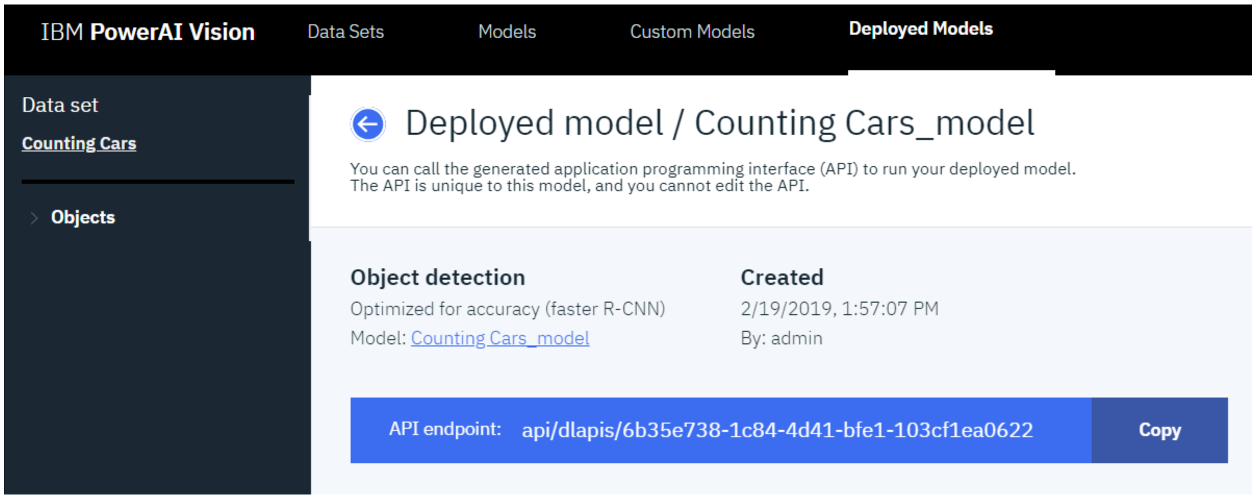
Because the first cell is just a description (markup) cell, it does nothing.

The next cell imports a set of modules that we use within the python code.

___48. Click ***Run*** on cell 1

The next cell describes the contents of Code Cell 2. Code Cell 2 has the PowerAI Vision variable that you will need to change in order to execute this demonstration.

***** The thing you need to change is the *pai_api* variable. This needs to point to the URL of the test system you are running on and the API key that you get from the deploy model screen for the model you created in the previous step. You just need to click on the copy button next to the API endpoint and replace the value in cell 2 with the copied value.**



Deployed model / Counting Cars_model

You can call the generated application programming interface (API) to run your deployed model.
The API is unique to this model, and you cannot edit the API.

Object detection
Optimized for accuracy (faster R-CNN)
Model: Counting Cars_model

Created
2/19/2019, 1:57:07 PM
By: admin

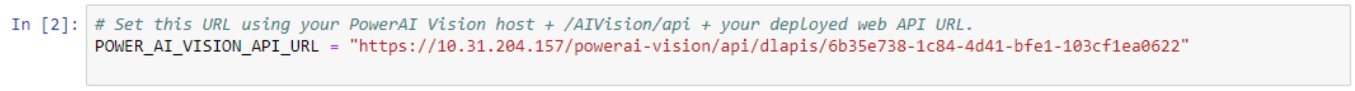
API endpoint: api/dlapis/6b35e738-1c84-4d41-bfe1-103cf1ea0622

Cell 2: After making your changes to cell 2 click on Run for that cell. Note there is no output for cell 2.

The rest of the notebook requires no modifications. To execute the rest of the lab, press the 'RUN" icon for each cell. The last step in the lab produces a video where you can see how accurate your model is detecting and counting cars.

Most steps execute very quickly but some of them take a minute or so to complete. Please wait until each step completes before executing the next step. You will know when the step completes when the cell gets an execution step number. Before the step executes, the step will have no number. While the step executes, an '*' appears and when the step completes, the step number appears.

As you can see from the example below, this step is complete as it has a '2' for a step number.



```
In [2]: # Set this URL using your PowerAI Vision host + /AIVision/api + your deployed web API URL.
        POWER_AI_VISION_API_URL = "https://10.31.204.157/powerai-vision/api/dlapis/6b35e738-1c84-4d41-bfe1-103cf1ea0622"
```

END OF LAB