How to validate your deep learning model with the Diffgram SDK—Tutorial

Anthony Sarkis Follow
Feb 1 · 6 min read

Do you have an existing deep learning model in production? Need to validate it's performance? An easy way to do this is through the diffgram python SDK.

We will cover

- Installing and configuring
- How to track your version
- How to export changes

The end result is an export of your data, with changes flagged.

Installing and configuring

1. Install the library

```
pip install diffgram
```

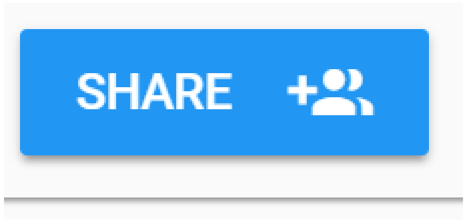
2. Download samples

Clone or download the github samples

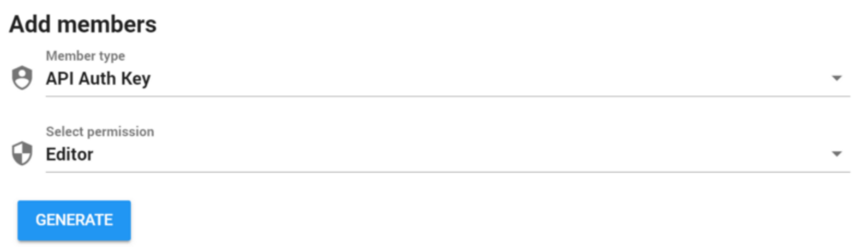
3. Provision authentication credentials

If you haven't already, create a new account, enabled the builder API, and create a new project.

- Click on the Share button on the top-right corner of the window.



- In member type, select 'API Auth Key'

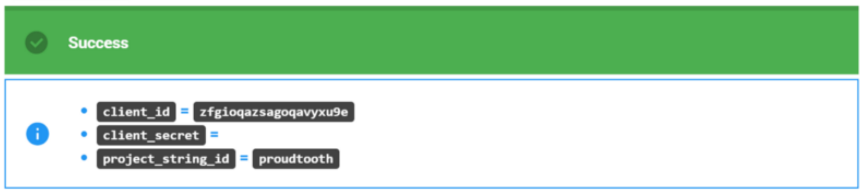


Add members

Member type: API Auth Key

Select permission: Editor

- Choose desired permissions level. 'Editor' is required to upload media.
- Save the client secure in a secure place



Success

- client_id = zfgioqazsagoqavyxu9e
- client_secret =
- project_string_id = proudtooth

- Update the credentials

```
CLIENT_ID = "replace"
CLIENT_SECRET = "replace"
PROJECT_STRING_ID = "replace"
```

4. Create a new label

Let's open `sample_new_label.py` and review the code:

First we create the diffgram client `diffgram = Diffgram()`

Then we authenticate it with the credentials we changed in settings: `diffgram.auth(client_id = settings.CLIENT_ID …)`

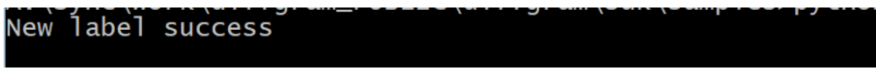
This auth process will also set our default directory and download existing labels.

We see there is an example of a label `apple = {'name' : 'apple'}` Let's change it to cat `cat = {'name': 'cat'}`

We only need one label for this: `diffgram.label_new(label)` So we can delete the rest of the code in the sample. If you have a large project with many labels you could import them with a loop here.

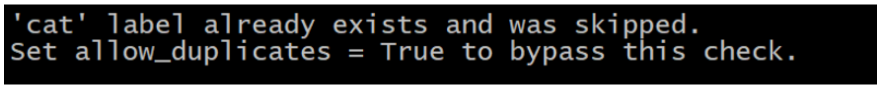
Now lets run the program `python sample_new_label.py`

Returns:



```
New label success
```

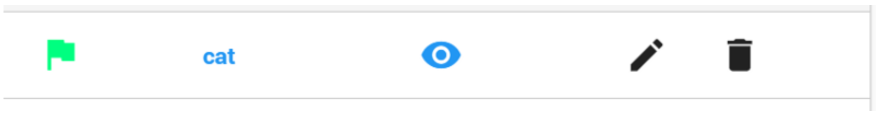
If we run the program a second time we get:



```
'cat' label already exists and was skipped.
Set allow_duplicates = True to bypass this check.
```

A label file in Diffgram reflects the meaning attached to an instance. These can be fairly complex, for example a label may have different attributes such as occlusion, categories, etc.

We can also go to Project -> Annotate in Diffgram to see the label:



5. Setup your instances

Let's open `existing_instances/sample_image_with_existing_instances.py`

We see the first part of the code is the same, creating a new diffgram client and authenticating it.

Now we define a single instance, which is a dictionary:

```
instance_alpha = {

  'type': 'box',

  'name': 'cat',

  'x_max': 128,

  'x_min': 48,

  'y_min': 97,

  'y_max': 128

}
```

In this case, a bounding box, represented by two key points (x_max, y_max) and (x_min, y_min).

What's with the 'name'? By default, the diffgram client will convert the string name (ie 'cat'), in each instance, to the matching diffgram label file (like the one we just created above.). If you would like to directly assigned the diffgram label file id you can control this with the `convert_names_to_label_files` flag.

Now we have a packet of data:

```
image_packet = {

  'instance_list' : [instance_alpha, instance_bravo],

  'media' : {

     'url' : "https://www.readersdigest.ca/wp-
content/uploads/sites/14/2011/01/4-ways-cheer-up-depressed-
cat.jpg",
        'type' : 'image'

    }

  }
```

Each packet will create one File in Diffgram. A diffgram file is raw data + encoded meaning.

The instance list, is a list of the above shown instances.

The media dictionary contains the url to the media to be downloaded and the type.

Both ‘image’ and ‘video’ types are supported. Video types require more complex handling of frame packets, not covered in this tutorial.

For managing work at scale we can attach the packet to a job. See Introduction to Jobs for a primer on this.

6. Generate signed URLs [optional]

For this tutorial, you can use a public url, such as the one included in the sample.

For production use you will need to generate signed urls from your cloud storage provider.

- Google The generate_signed_url function is an easy way to do this
- Amazon

For this tutorial, you can leave the instances and image data as is.

7. View the file in Diffgram

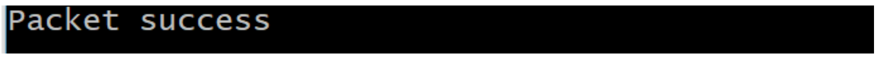
Now lets run the program `python sample_image_with_existing_instances.py`

Github link

The key line here is:

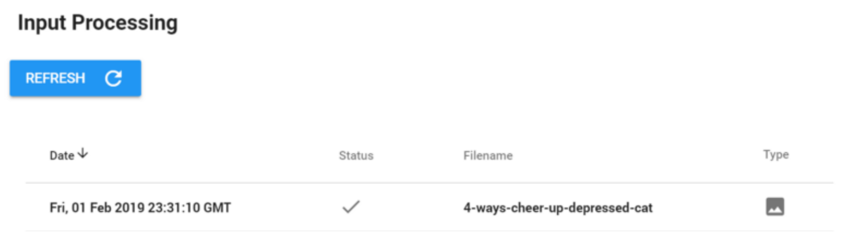
```
result = project.file.from_packet(image_packet)
```

And we get:



```
Packet success
```

Then in Diffgram we can go to project import to see the file:

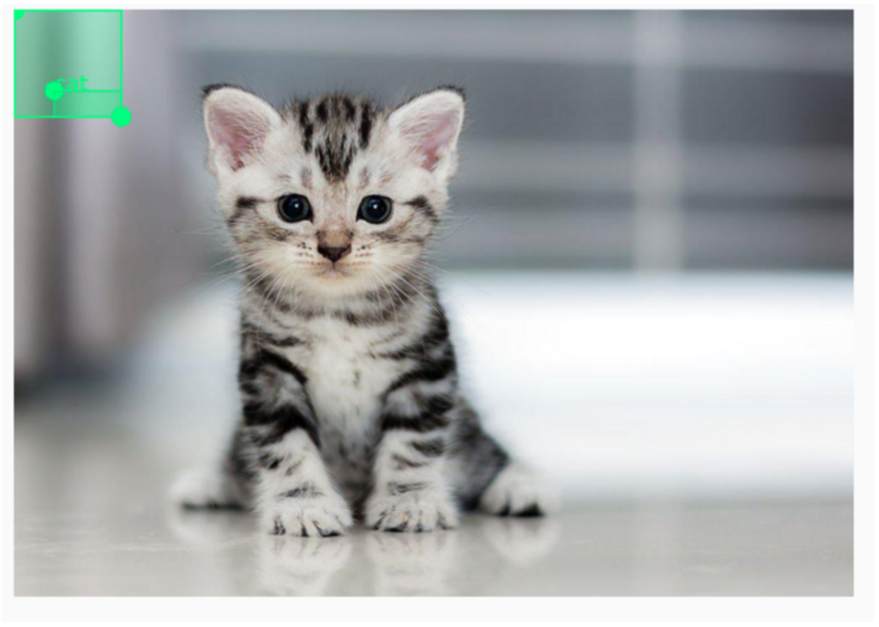


Input Processing

REFRESH

Date	Status	Filename	Type
Fri, 01 Feb 2019 23:31:10 GMT	✓	4-ways-cheer-up-depressed-cat	🖼

And in project -> Annotation we can see



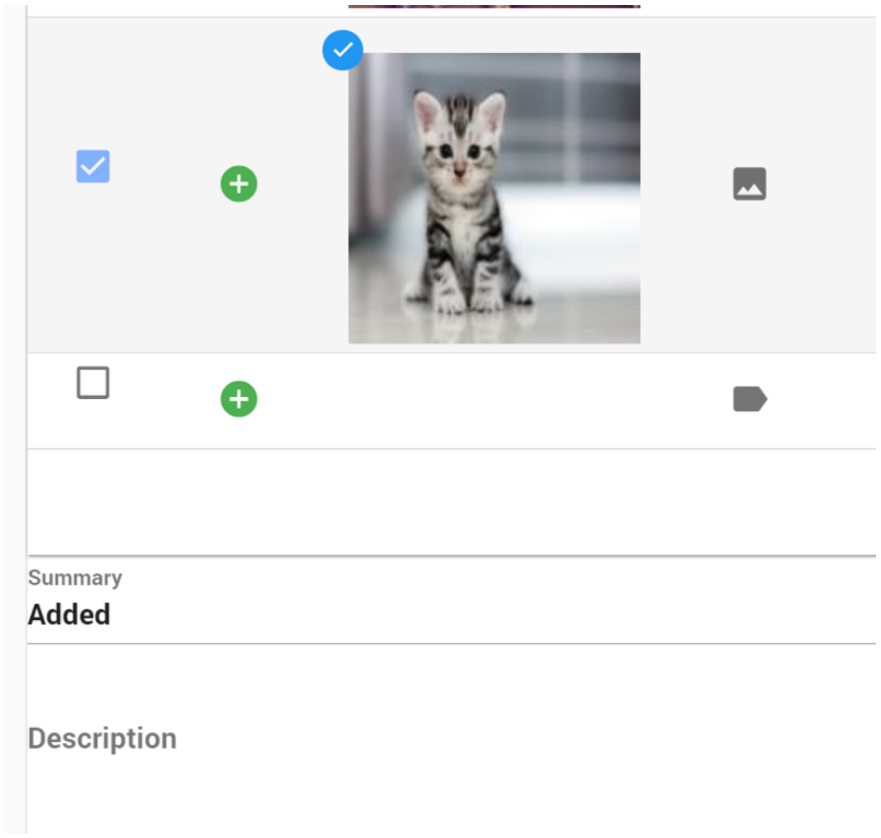
Try sharing the project with teammates to review work together.

8. Commit the file(s)

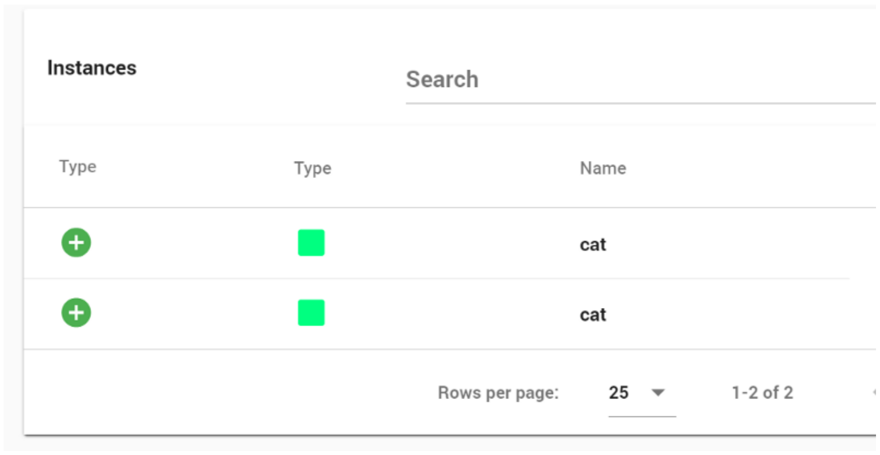
By committing the file we tell Diffgram we want to track the file, like git. After each file is committed, any new changes will show up in source control.

Diffgram doesn't track files until you tell it too—this allows you to use it like a working directory and only track the files you want.

- Go to source control
- Select the file(s)
- Write a summary as needed



Then in history we can see:



Instances

Search

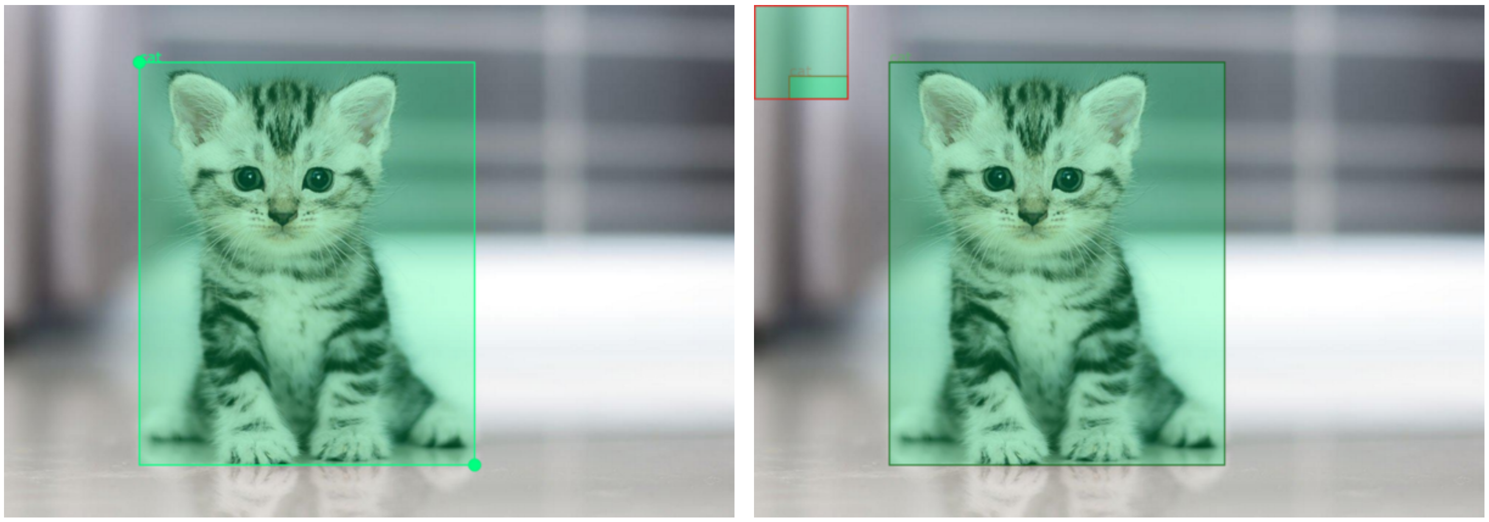
Type	Type	Name
⊕	🟩	cat
⊕	🟩	cat

Rows per page: 25 1-2 of 2

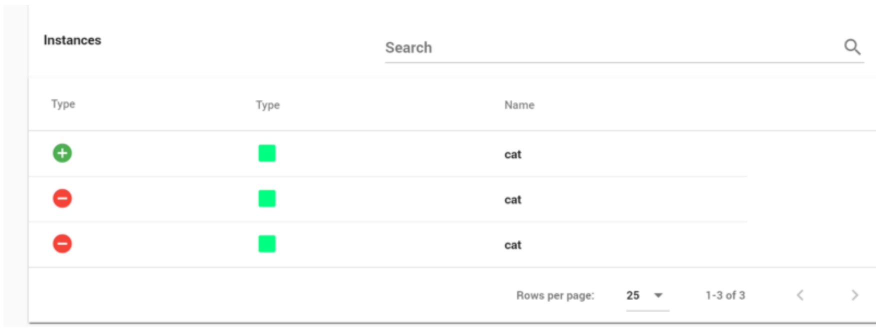
9. Validate the file(s)

Now let’s go back to annotate and fix this clearly awful inference!

- Go to edit mode
- Select the two incorrect instances and delete them
- Create a new instance and save the file



Now if we go to source control we can see:



The red instances have been removed, and the green ones added. (Unchanged ones will show up in white)

This is useful for a visual confirmation, but if we are working with a large volume of files we likely want to export the results in a machine readable format.

10) Export it

- Go to project -> Export
- File comparison: `vs_original` and then Generate
- Download in your preferred format

We can see the file has 3 instances attached, with 2 deleted and 1 added:

```
a6cd75ee5103888bf15fe0b6b2c28637f3ae480538d9bfde7207136749a3
4860:
  file:
    id: 82478
  image:
    height: 700
    original_filename: 4-ways-cheer-up-depressed-cat
    video_id: null
    width: 1000
  instance_list:
  - change_type: deleted
    hash:
9c1ffb1ba6633a1a4b4dddb5521eb213317dce28d37f9addd90298f0eebc
6a36
    height: 31
    label:
      id: 16
      label_name: cat
    type: box
    width: 80
    x_max: 128
    x_min: 48
    y_max: 128
    y_min: 97
  - change_type: deleted
    hash:
b1ebfcc2560dd921e32ffee5c500d257ab8f35576a3fb32aaba67575b69d
9565
    height: 127
    label:
      id: 16
      label_name: cat
    type: box
    width: 127
    x_max: 128
    x_min: 1
    y_max: 128
    y_min: 1
  - change_type: added
    hash:
c69ceb32d8c0f63189896e5f27caf281e2d2cafea3cc865f1f40348a3c8d
6b29
    height: 550
    label:
      id: 16
      label_name: cat
    type: box
    width: 459
    x_max: 644
    x_min: 185
    y_max: 628
    y_min: 78
```

We can now:

- Compute validation statistics
- Retrain our model using the corrected data

Thanks for reading!

PS Working on a large scale project? Check out FAN for reducing cost.

1: Storing as an environment variable is more secure