Deployment Manual

Computer Vision for Object Detection in Medicine

Team 6 - GOSH Drive

Page Index

Training a Model	**3**
Web API	**3**
Client Application	**6**
Appendix	**8**

Training a Model

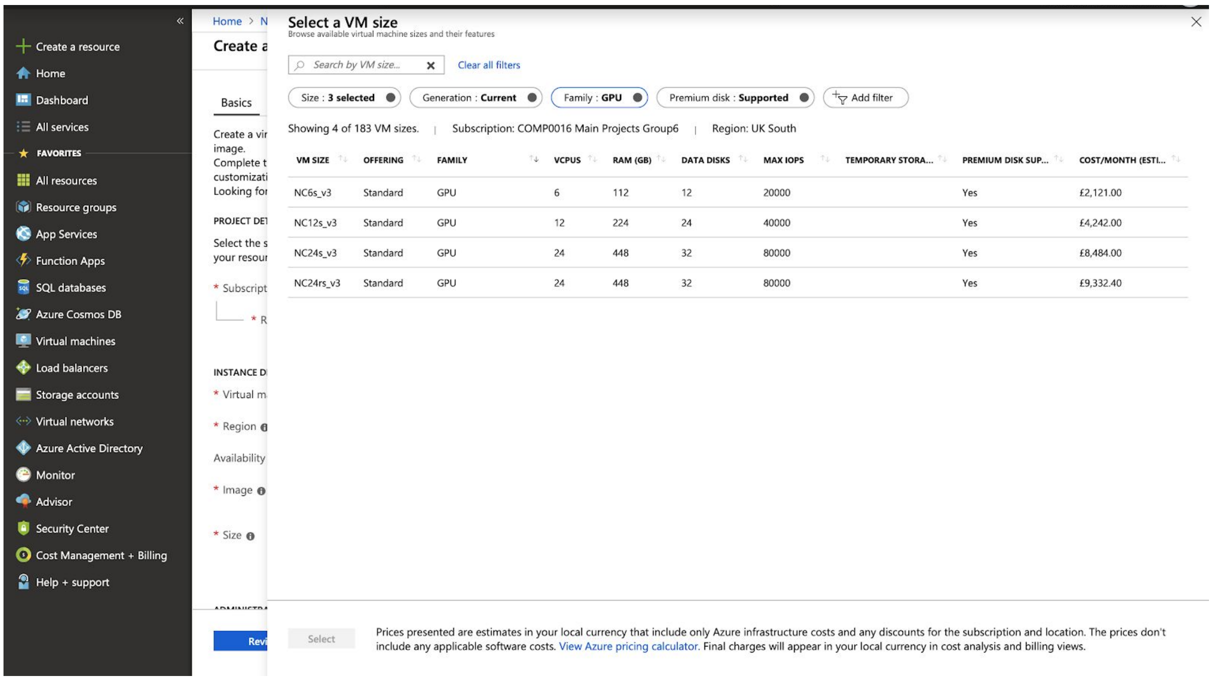
Our application works with any tensorflow inference graph model. Therefore, we will not include the exact method in which we used to make our model as it has become redundant and the scripts we used have been moved to the legacy section. To make our model we used the Google TensorFlow Object Detection API and trained our own model using the faster_rcnn model configuration (https://github.com/tensorflow/models/tree/master/research/object_detection). All the details in training your own model can be found there in the their git repository.

If you want to use our own pre trained model that we used for our project, it can be found in our sysEngWebAPI git repository (https://github.com/gohchanb/sysEngWebAPI) under *object_detection/instruments_graph.* The graph has been trained using around 1500 images of 3 different medical instruments categories (forceps_straight, forceps_curved and tweezers). For each instrument category we only used 1 specific instrument from which we took the images. The instruments can be found at GOSH DRIVE. If you decided to train your own model, the model currently in the *object_detection/instruments_graph* folder can be replaced and the Web API will still run as intended using the new graph.

Web API

All the files needed for the WebAPI are found in the sysEngWebAPI git repository (https://github.com/gohchanb/sysEngWebAPI). **If you want to change the model**, make a copy of the repository and make changes to your own repository. Then instead of cloning our repository onto the VM, you should clone your own.

In this manual i will be showing how to deploy the API onto a Microsoft Azure VM however you can use any cloud hosting service you want. The first thing you need to do is create a Ubuntu VM on Azure. When creating the VM, we recommend to choose a size under the GPU family as this will enable us to use TensorFlow GPU which is significantly faster than TensorFlow CPU for inferencing.



Select a VM size

Browse available virtual machine sizes and their features

Showing 4 of 183 VM sizes. | Subscription: COMP0016 Main Projects Group6 | Region: UK South

VM SIZE	OFFERING	FAMILY	VCPUS	RAM (GB)	DATA DISKS	MAX IOPS	TEMPORARY STORA...	PREMIUM DISK SUP...	COST/MONTH (ESTI...
NC6s_v3	Standard	GPU	6	112	12	20000		Yes	£2,121.00
NC12s_v3	Standard	GPU	12	224	24	40000		Yes	£4,242.00
NC24s_v3	Standard	GPU	24	448	32	80000		Yes	£8,484.00
NC24rs_v3	Standard	GPU	24	448	32	80000		Yes	£9,332.40

Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. View Azure pricing calculator. Final charges will appear in your local currency in cost analysis and billing views.

Picking a GPU compatible VM

However this is not necessary. And i will show how to setup the Web API using both methods. The first thing to do once you create the VM is to ssh into the machine or use the 'Serial console' located in the 'Support + Troubleshooting' section. Once you are into the VM you must run this command.

```
$sudo apt-get update
```

Next you must install python3 and pip for that version, for our project we used specifically python3.6.7 so we would recommend you to use the same version or above. Depending on the version of Ubuntu, this may be already installed but you will have to check for your specific version. For the purpose of this manual, whenever i write `$python` or `$pip` it means those respective programs for version 3.6.7 of python. Next we need to install TensorFlow onto the machines, this is where the process differs depending on that type of VM you are using.

TensorFlow GPU (VMs in the GPU category/with a CUDA compatible GPU)

<u>Install CUDA</u>

```
Wget
https://developer.nvidia.com/compute/cuda/9.0/Prod/local_installer
s/cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64-deb
sudo dpkg -i cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64-deb
sudo apt-key add /var/cuda-repo-9-0-local/7fa2af80.pub
sudo apt-get update
sudo apt-get install -y cuda
rm cuda-repo-ubuntu1604-9-0-local_9.0.176-1_amd64-deb
```

Install CUDnn

When installing CUDnn we need to have a NVDIA account and login in order to access CUDnn (which is free to create). The system requires CUDnn v7.3.0 in Linux by following these steps below.

Please note that the link in the first line may be invalid when using as each user must be verified therefore you may get the correct link after logging into your specific NVIDIA account copying the link for the cuDNN v7.3.0 Library for Linux for CUDA 9.0

```
wget
https://developer.download.nvidia.com/compute/machine-learning/cud
nn/secure/v7.3.0/prod/9.0_2018920/cudnn-9.0-linux-x64-v7.3.0.29.tg
z
sudo tar -xzf cudnn-9.0-linux-x64-v7.tgz -C /usr/local
rm cudnn-9.0-linux-x64-v7.tgz
sudo ldconfig
```

Environment Variables

and add the following exports to `~/.bashrc`

```
export CUDA_HOME=/usr/local/cuda-9.0

export PATH=${CUDA_HOME}/bin:${PATH}

export
LD_LIBRARY_PATH=${CUDA_HOME}/lib64:/usr/local/cuda/lib64:${LD_LIBR
ARY_PATH}
```

Install tensorflow-gpu

```
sudo apt-get install -y python-pip python-dev
sudo pip install tensorflow-gpu
```

TensorFlow CPU (VMs without a CUDA compatible GPU)

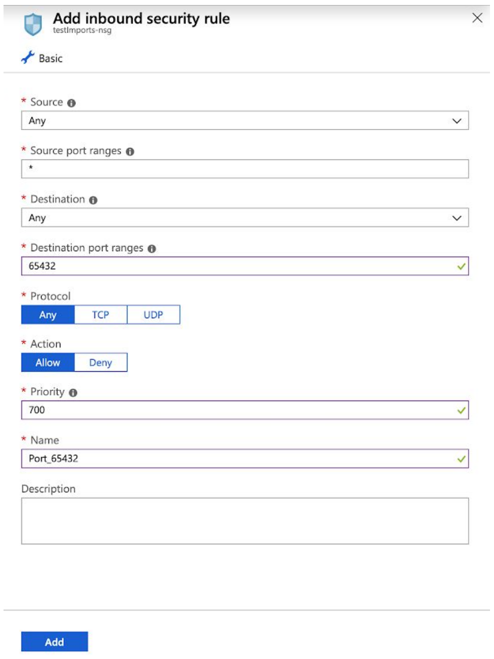
```
$pip install tensorflow
```

All the libraries needed by our application should be installed however is they aren't they can be installed separately. This can be done by entering into the line below into the terminal but replacing `library` with the library you want to download.

```
$pip install library
```

The full list of libraries currently on our VM can be found in the appendix (note: not all of them are used).

To allow the application to communicate with the internet we need to add an inbound security group. To do this, navigate to the network security group for the VM's resource group. Then go to ‘Inbound Security Rules’ and add the rule corresponding to the app’s port. In our example we use port 65432.



Add inbound security rule
testImports-nsg

Basic

* Source: Any

* Source port ranges: *

* Destination: Any

* Destination port ranges: 65432

* Protocol: [✓ Any] TCP UDP

* Action: [✓ Allow] Deny

* Priority: 700

* Name: Port_65432

Description:

Adding Inbound Security Rule

Next we need to download the git repository and navigate to the folder and run the API

```
$git clone https://github.com/gohchanb/sysEngWebAPI
$cd sysEngWebAPI
$python app.py
```

If all is successful, the console should show some TensorFlow information and then print

```
listening on (‘0.0.0.0’,65432)
```

Client Application

All the files needed for the client application are found in the ObjectDetectionAPI git repository (https://github.com/gohchanb/ObjectDetectionAPI). Like the Web API, we have been using python3.6.7 so we would recommend you to use the same version or above. Therefore, for the purpose of this manual, whenever i write `$python` or `$pip` it means those respective programs for version 3.6.7 of python. To setup the environment we highly recommend you to use virtualenv or some other python virtual environment manager.

Once you have activate the virtual environment you need to download the libraries needed by the application. This can be done by entering into the line below into the terminal but replacing `library` with the library you want to download.

```
$pip install library
!!!!! ta een ja
```

The list of libraries that need to be downloaded are listed below, with the versions we used. This should download all the necessary sub libraries as well however the full list can be found in the Appendix.

- pyqt5 (5.12.1)
- numpy (1.16.2)
- matplotlib (3.0.3)
- pillow (6.0.0)
- Tensorflow (1.13.1) - Only CPU needed by client
- mysql-connector-python (8.0.15)
- opencv-python (4.0.0.21)
- plotly (3.7.1)

After all the packages have been installed. The client files should be downloaded and you should move into the folder.

```
$git clone https://github.com/gohchanb/ObjectDetectionAPI
$cd ObjectDetectionAPI
```

The application uses a database to log all the operation stats, so you have to add a database to your local machine. The schema of the database can be found in the folder under *schema.sql* and the database with some sample data already inside and a user can be found under *dbexport.sql.* Either file can be used to import the database onto your local machine. First you need to create the database called systemsEngineering. You might also want to create a user that has access to only the systemsEngineering table. In our project we used ‘sysEng’ as the users username and password.

Once the database has been created, you can import the the database with:

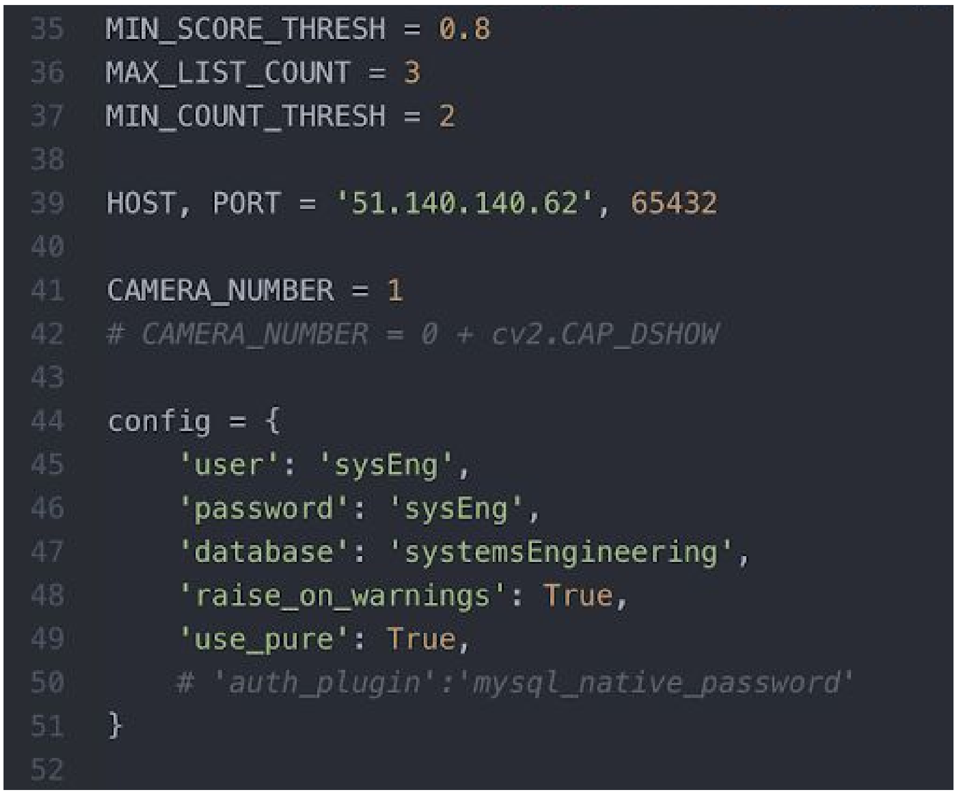
```
$mysql -u sysEng -p systemsEngineering < schema.sql
```

or

```
$mysql -u sysEng -p systemsEngineering < dbexport.sql
```

depending on which file you would like to use.

Next you need to configure the client application. All the relevant parameters are near the top of the code.



```python
MIN_SCORE_THRESH = 0.8
MAX_LIST_COUNT = 3
MIN_COUNT_THRESH = 2

HOST, PORT = '51.140.140.62', 65432

CAMERA_NUMBER = 1
# CAMERA_NUMBER = 0 + cv2.CAP_DSHOW

config = {
    'user': 'sysEng',
    'password': 'sysEng',
    'database': 'systemsEngineering',
    'raise_on_warnings': True,
    'use_pure': True,
    # 'auth_plugin':'mysql_native_password'
}

```

Client application configuration settings

MIN_SCORE_THRESH is the minimum score needed for an instrument to be registered by the application. We recommend you to keep it at 0.8 but you can change it. MAX_LIST_COUNT and MIN_COUNT_THRESH are variables that describe how the ListObject registers objects. Please refer to the ListObject section in the ‘Key Functionalities’ section to learn how they work, but again we recommend you to keep them unchanged. HOST and PORT refer to the host and port of the WebAPI you should have set up earlier. The PORT number should always be 65432 by default, while the HOST number should be changed depending on the ip-address of your VM. CAMERA_NUMBER refers to the number of the camera on your system that you want to use for your input. If your system only has 1 camera it will probably just be 0 however it will differ for different machines. Lastly we have ‘config’ which is the configuration for the database. Here we are using the login of our custom user we made however u can change the ‘user’ and ‘password’ to any combination which has access to the systemsEngineering database.

Once you have set all this up, you should be able to run the application by running.

```
$python object_detection_client.py
```

Appendix

VM list of libraries
absl-py (0.7.1)

```
astor (0.7.1)
beautifulsoup4 (4.4.1)
blinker (1.3)
chardet (2.3.0)
cloud-init (18.5)
command-not-found (0.3)
configobj (5.0.6)
cryptography (1.2.3)
defer (1.0.6)
gast (0.2.2)
grpcio (1.19.0)
gunicorn (19.4.5)
h5py (2.9.0)
html5lib (0.999)
idna (2.0)
Jinja2 (2.8)
jsonpatch (1.10)
jsonpointer (1.9)
Keras-Applications (1.0.7)
Keras-Preprocessing (1.0.9)
language-selector (0.1)
lxml (3.5.0)
Markdown (3.1)
MarkupSafe (0.23)
mock (2.0.0)
numpy (1.16.2)
oauthlib (1.0.3)
pbr (5.1.3)
pip (8.1.1)
prettytable (0.7.2)
protobuf (3.7.1)
pyasn1 (0.1.9)
pycups (1.9.73)
pycurl (7.43.0)
pygobject (3.20.0)
PyJWT (1.3.0)
pyserial (3.0.1)
python-apt (1.1.0b1+ubuntu0.16.4.2)
python-debian (0.1.27)
python-systemd (231)
pyxdg (0.25)
PyYAML (3.11)
requests (2.9.1)
screen-resolution-extra (0.0.0)
setuptools (20.7.0)
six (1.10.0)
```

```
ssh-import-id (5.5)
stevedore (1.30.1)
system-service (0.3)
tensorboard (1.12.2)
tensorflow-estimator (1.13.0)
tensorflow-gpu (1.12.0)
termcolor (1.1.0)
ufw (0.35)
unattended-upgrades (0.1)
urllib3 (1.13.1)
virtualenv (16.4.3)
virtualenv-clone (0.5.1)
virtualenvwrapper (4.8.4)
WALinuxAgent (2.2.32.2)
Werkzeug (0.15.1)
wheel (0.29.0)
xkit (0.0.0)
```

Client application Packages

```
Package                 Version
---------------------- --------
absl-py                 0.7.1
astor                   0.7.1
attrs                   19.1.0
certifi                 2019.3.9
chardet                 3.0.4
cycler                  0.10.0
decorator               4.4.0
gast                    0.2.2
grpcio                  1.19.0
h5py                    2.9.0
idna                    2.8
ipython-genutils        0.2.0
jsonschema              3.0.1
jupyter-core            4.4.0
Keras-Applications      1.0.7
Keras-Preprocessing     1.0.9
kiwisolver              1.0.1
Markdown                3.1
matplotlib              3.0.3
mock                    2.0.0
mysql-connector         2.2.9
mysql-connector-python 8.0.15
nbformat                4.4.0
numpy                   1.16.2
opencv-python           4.0.0.21
pbr                     5.1.3
Pillow                  6.0.0
pip                     19.0.3
plotly                  3.7.1
protobuf                3.7.1
pyparsing               2.4.0
PyQt5                   5.12.1
```

```
PyQt5-sip             4.19.15
pyrsistent            0.14.11
python-dateutil       2.8.0
pytz                  2018.9
requests              2.21.0
retrying              1.3.3
setuptools            41.0.0
six                   1.12.0
tensorboard           1.13.1
tensorflow            1.13.1
tensorflow-estimator  1.13.0
termcolor             1.1.0
traitlets             4.3.2
urllib3               1.24.1
Werkzeug              0.15.2
wheel                 0.33.1
```