Supplementary

No Author Given

No Institute Given

1 System Requirements

1. Operating system : Linux ubuntu 16.04 (`https://www.ubuntu.com/download/desktop`)
2. Python 3.5.2 kernel (`www.python.org`)
3. Anaconda 4.3 data science library (`www.anaconda.com`)
4. GUI interface: pycharm professional 2017 (`https://www.jetbrains.com/pycharm/`)
5. Check available GPU type in Linux terminal (Figure 1).

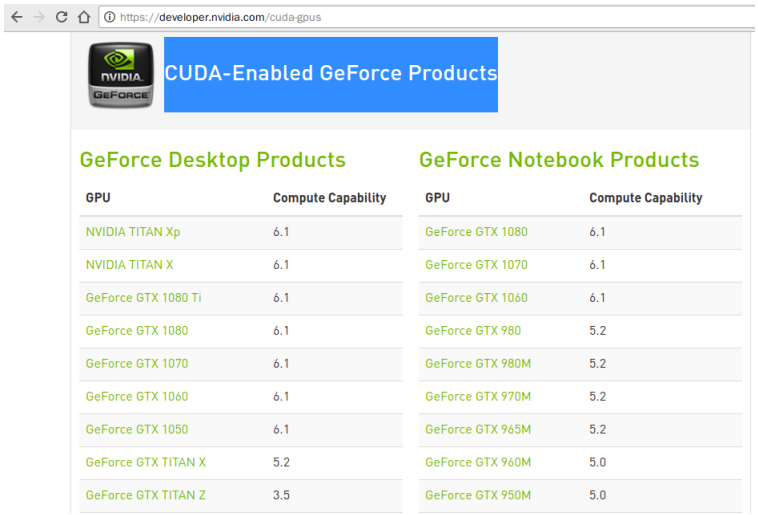
Fig. 1. Check GPU type; Linux terminal

6. whether is CUDA-enabled GPU? (check the list `https://developer.nvidia.com/cuda-gpus`)(Figure 5)

2 Package Installation

2.1 CUDA Installation

1. su root (login as a root user)
2. Download CUDA run-file from `https://developer.nvidia.com/cuda-downloads`)(Figure 2)
3. Copy CUDA run-file in home directory (Figure 3)
4. Ctrl+ Alt+ F1 (Open command line)
5. root
6. apt-get update (update all Linux repositories)
7. service lightdm stop (deactivate display manager)
8. service lightdm status (double check display manager status)
9. chmod 777 cuda.run (if permission denied) (Figure 9)
10. ./cuda.run (execute cuda.run in home directory)
11. Reply following question with "accept" phrase
    "Do you accept the previously read EULA"?



https://developer.nvidia.com/cuda-gpus

CUDA-Enabled GeForce Products

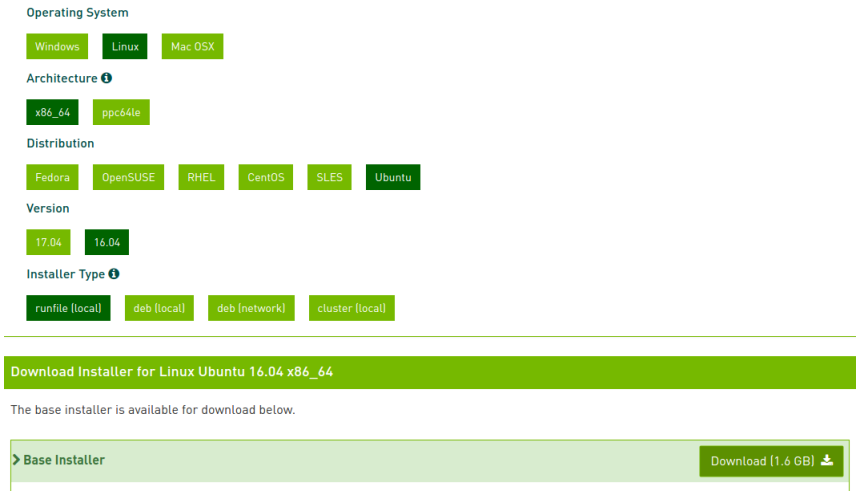
GeForce Desktop Products

GPU	Compute Capability
NVIDIA TITAN Xp	6.1
NVIDIA TITAN X	6.1
GeForce GTX 1080 Ti	6.1
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 1050	6.1
GeForce GTX TITAN X	5.2
GeForce GTX TITAN Z	3.5

GeForce Notebook Products

GPU	Compute Capability
GeForce GTX 1080	6.1
GeForce GTX 1070	6.1
GeForce GTX 1060	6.1
GeForce GTX 980	5.2
GeForce GTX 980M	5.2
GeForce GTX 970M	5.2
GeForce GTX 965M	5.2
GeForce GTX 960M	5.0
GeForce GTX 950M	5.0

Fig. 2. CUDA-enabled NVIDIA products



Operating System

Windows Linux Mac OSX

Architecture

x86_64 ppc64le

Distribution

Fedora OpenSUSE RHEL CentOS SLES Ubuntu

Version

17.04 16.04

Installer Type

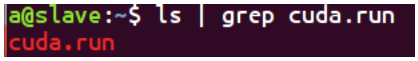
runfile (local) deb (local) deb (network) cluster (local)

Download Installer for Linux Ubuntu 16.04 x86_64

The base installer is available for download below.

Base Installer Download (1.6 GB)

Fig. 3. Download CUDA



```
a@slave:~$ ls | grep cuda.run
cuda.run
```

Fig. 4. CUDA file saved in home directory

12. Reply all other respective questions with "yes" excluding "Do you want to install the OpenGL libraries?
13. service lightdm start (Figure 13)

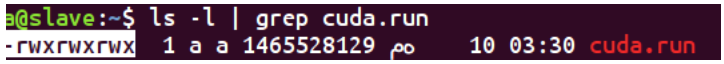


Fig. 5. Permission installation

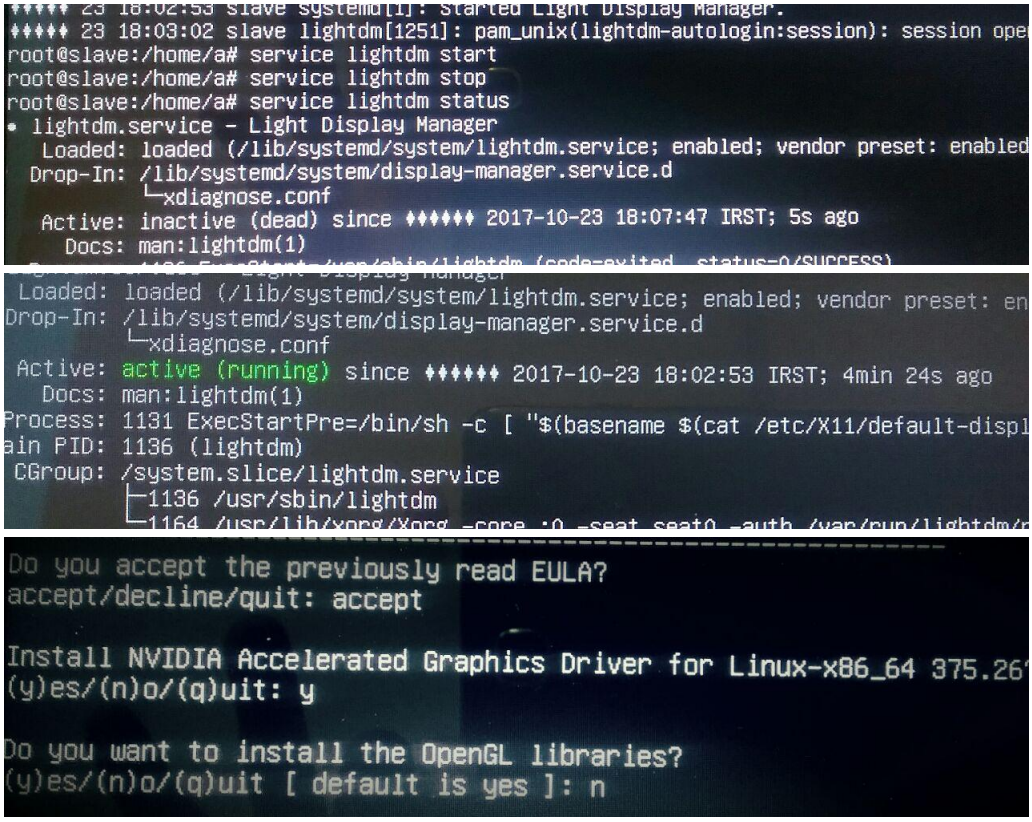


Fig. 6. CUDA installation steps

14. Ctrl+ Alt+ T (open TTY)
15. gedit /.bashrc
16. Add these lines :
    export PATH="/usr/local/cuda-8.0/:$PATH"
    export PATH="/usr/local/cuda-8.0/bin/:$PATH"
    export LD_LIBRARY_PATH="/usr/local/cuda-8.0/lib64/:$LD_LIBRARY_PATH"

17. ldconfig (Create the necessary links and caches to share libraries)
18. reboot (restart operating system)
19. Ctrl + Alt + T (open TTY)
20. nvidia-smi (monitoring information for each NVIDIA devices)

2.2 cuDNN Installation

1. `https://developer.nvidia.com/cudnn`
2. Register and download compressed (*.tgz) cuDNN v5.1 Library for Linux (Figure 2

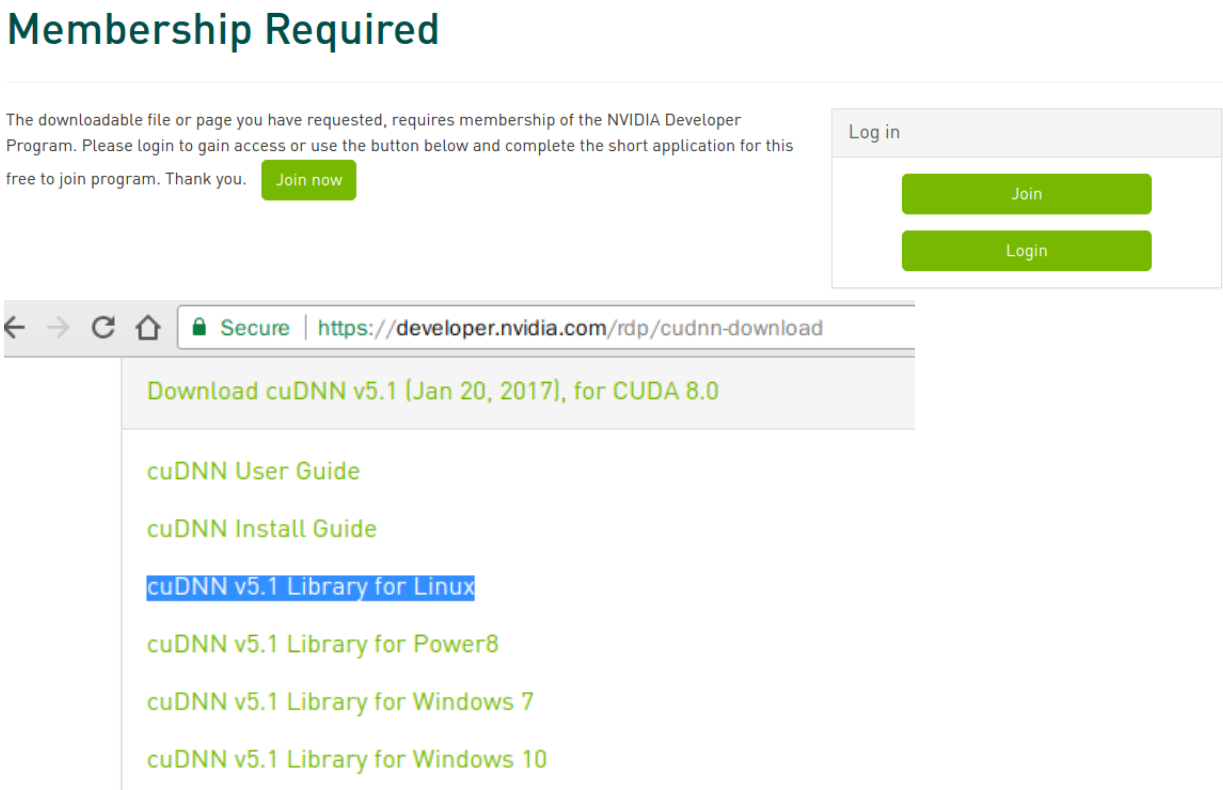


Fig. 7. cuDNN v5.1, for CUDA 8.0 installation

3. Installing from a Tar File: (`http://docs.nvidia.com/deeplearning/sdk/cudnn-install/index.html`)

   – tar -xzvf cudnn-8.0-linux-x64-v7.tgz (Unzip the cuDNN package)

   Hint: Extracted file will be saved as "cuda" directory (3)
   – Copy the following files into the CUDA Toolkit directory.

   (a) sudo cp cuda/include/cudnn.h /usr/local/cuda/include
   (b) sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
   (c) sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

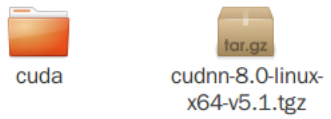


Fig. 8. Unzip cudNN

2.3 Tensorflow with GPU Support Installation

In regard to Tensorflow with GPU Support installation and documentations navigate `https://www.tensorflow.org/install/install$_$linux#InstallingAnaconda`

3 Data Preparation

3.1 Data Acquisition

1. Navigate "Stanford Tissue Microarray Database (TMA)" (`https://tma.im/cgi-bin/viewArrayBlockList.pl`)
2. "Breast, Bladder, Lung, Lymphoma" data crawling with (Bioinfomatics/Codes/1.TMA_crawl TMA_crawl.py) (Figure 2)
   - output 1: Image-link address saved in text-file.
   - output 2: All separated images saved in current folder.
2. Navigate Breast Cancer Histopathological Database (BreakHis): `https://web.inf.ufpr.br/vri/databases/breast-cancer-histopathological-database-breakhis/`
   - Go to each individual subtype folders.
   - combine different magnification levels into one folder for a specific class

3.2 Augmented Data

1. pip install Augmentor (Augmentation library: `https://github.com/mdbloice/Augmentor)`
   1. Table (**??**)
2. Run Bioinfomatics/Codes/img-augmentation.py
3. out put: create a folder called (output) to save all augmented data 1. Note : copy all images in output folder excluding (0) folder that is software bug.

3.3 Convert to TFRecord

Augmented data for each individual class copied into (Codes/3.Converted/ Tfrecord/BC_malignant_data /cancer_-photos

?????3 images For example; malignant classification is followed as bellow:

1. Open to edit (Codes/3.Converted Tfrecord/BC convert_to_TFrecord.py)

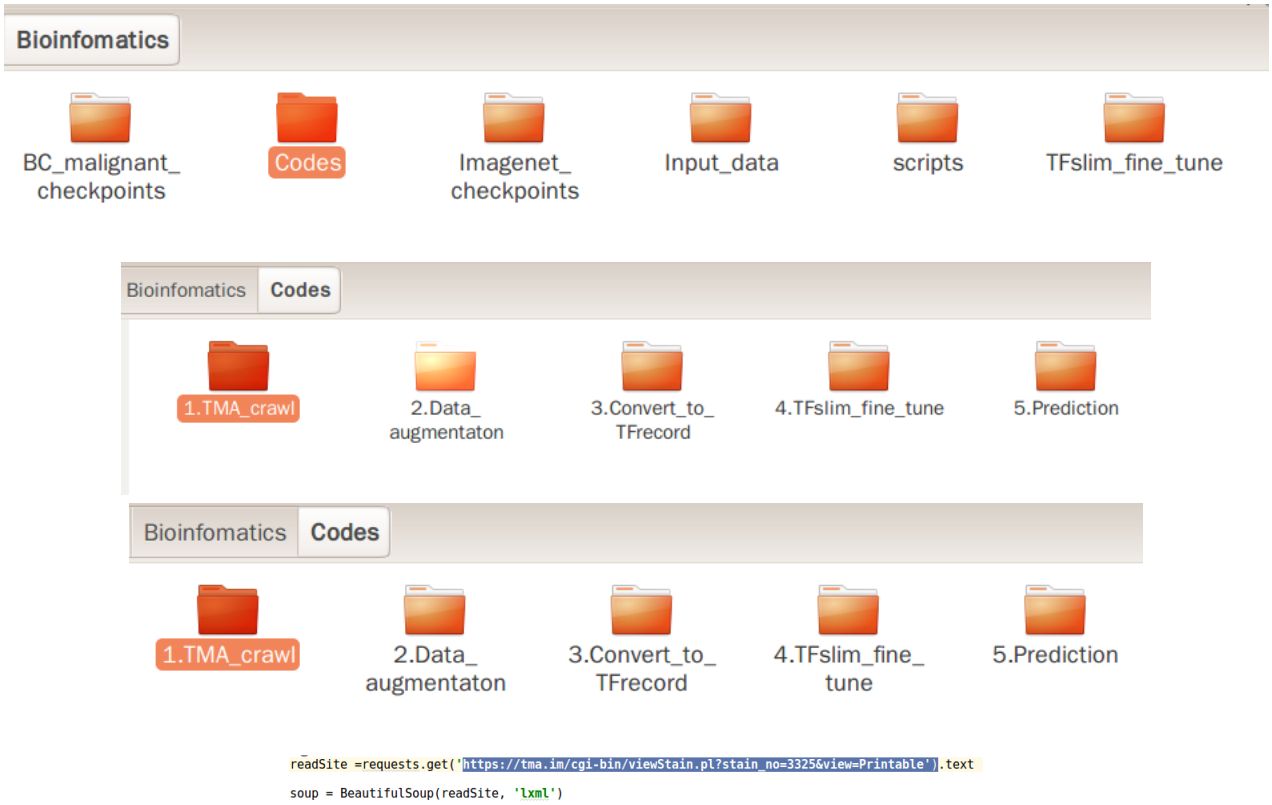


Fig. 9. TMA crawling steps

2. Modify following parameters:
   – Number of validation == 1000?? (almost 10 % of 9394 augmented samples)
   – Number of shard == 4 (number of Malignant sub-types)
   – Seed of repeatability == 0 (shufflingthe data with initial 0 seed)
   4. input parent Data set directory == ./BC-Malignant-data-set 1. Augmented or original (per case) input data saved into data-dir
3. (data-dir = './BC-Malignant-data-set') 5. run Convert-to-Tfrecord.py
4. TFRecord (output) will be saved BC-Malignant-data-set

4 Fine-tune pre-trained models

4.1 Checkpoints for individual deep learning Models

: 1. Download Pre-trained models checkpoints (Learned on ImageNet `http://www.image-net.org`) from (`https://github.com/tensorflow/models/tree/master/research/slim`)

2.Copy downloaded file into an optional directory (in our case: Desktop/checkpoints)
3. Copy "BC-Malignant-data-set" directory included original images and respective TFRcord files into an optional directory (i.e. ./Desktop/cancers)
4.Open and edit parameters for example (Inception V1) scripts/ finetune_inception_v1_on_cancer.sh.

4.2 Server Setting

14. Fine tuning last and all layer(s) 1. open script folder and choose script for tuning any of pre-trained models. 2. set path for checkpoint, input data-set and output checkpoint. 3.open terminal and copy script file in terminal.

1. Where the pre-trained InceptionV1 checkpoint is saved to.
   PRETRAINED_CHECKPOINT_DIR=/Desktop/checkpoints
2. Where the training (fine-tuned) checkpoint and logs will be saved to.
   TRAIN_DIR=/Desktop/BC_malignant_ckpt/inception_v1
3. Where the dataset is saved to.
   DATASET_DIR=/Desktop/cancers
4. Fine-tune only the new layers for 3000 steps.
   cd /Desktop/Tfslim_fine_tune (path of train_image_classifier.py file)
   python train_image_classifier.py
   –train_dir=$TRAIN_DIR
   –dataset_name=cancers
   –dataset_split_name=train
   –dataset_dir=$DATASET_DIR
   –model_name=inception_v1 #(pre_trained model name)
   –checkpoint_path=$PRETRAINED_CHECKPOINT_DIR/inception_v1.ckpt
   –checkpoint_exclude_scopes=InceptionV1/Logits (name of last layer for fine-tuning)
   –trainable_scopes=InceptionV1/Logits (name of last layer for fine-tuning)
   –max_number_of_steps=3000 (number of epochs)
   –batch_size=32
   –learning_rate=0.01
   –save_interval_secs=60
   –save_summaries_secs=60
   –log_every_n_steps=100
   –optimizer=rmsprop (optimizer) –weight_decay=0.00004
5. Run evaluation after fine-tuning last layer.
   cd /Desktop/Tfslim_fine_tune (path of eval_image_classifier.py file) python eval_image_classifier.py
   –checkpoint_path=$TRAIN_DIR
   –eval_dir=$TRAIN_DIR
   –dataset_name=cancers
   –dataset_split_name=validation
   –dataset_dir=$DATASET_DIR
   –model_name=inception_v1 (pre_trained model name)

6. Fine-tune all the new layers for 3000 steps. cd /Desktop/Tfslim_fine_tune (path of train_image_classifier.py file)
   python train_image_classifier.py
   –train_dir=$TRAIN_DIR/all (fine-tuning all layers) –dataset_name=cancers
   –dataset_split_name=train
   –dataset_dir=$DATASET_DIR
   –checkpoint_path=$TRAIN_DIR
   –model_name=inception_v1 (pre_trained model name) –max_number_of_steps=3000 (number of epochs) –batch_size=32
   –learning_rate=0.001
   –save_interval_secs=60
   –save_summaries_secs=60
   –log_every_n_steps=100
   –optimizer=rmsprop (optimizer)
   –weight_decay=0.00004
7. Run evaluation after fine-tuning all layers.
   cd /Desktop/Tfslim_fine_tune (path of eval_image_classifier.py file)
   python eval_image_classifier.py
   –checkpoint_path=$TRAIN_DIR/all(fine-tuning all layers) –eval_dir=$TRAIN_DIR/all
   –dataset_name=cancers
   –dataset_split_name=validation
   –dataset_dir=$DATASET_DIR –model_name=inception_v1(pre_trained model name)

for showing tensorboard, open a terminal in path of event file and type tensorboard logdir=. By openning local host http://slave:6006/ tensorboard was observable.

5 Prediction

In each epoch, one checkpoint file (model.ckpt-****.meta), (model.ckpt-****.index) and (model.ckpt-****.data-****-of-****), was created and five of them was kept. In addition to these files, there are checkpoint, events.out.tfevents.**** for showing tensorboard and graph.pbtxt file. All related checkpoints and files to fine-tuning all layars were saved in a folder called all.

model.ckpt.data-00000-of-00001
model.ckpt.index
model.ckpt.meta

meta file: describes the saved graph structure, includes GraphDef, SaverDef, and so on; index file: it is a string-string immutable table(tensorflow::table::Table). Each key is a name of a tensor and its value is a serialized BundleEntryProto. Each BundleEntryProto describes the metadata of a tensor: which of the "data" files contains the content of a tensor, the offset into that file, checksum, some auxiliary data, etc. data file: it is TensorBundle collection, save the values of all variables

After opening this link, you can see some information including histograms, tensorboard for pre-trained model and confusion matrix.

Prediction class for new image:
1. Set checkpoint from fine-tune last or all layer(s) in a directory and put it in code as input. In our example this path is:/home/habibzadeh/Desktop/BC_malignant_checkpoints/inception_v1
2. check path in checkpoint file in /Bioinfomatics/BC_malignant_checkpoints/inception_v* and /Bioinfomatics/BC_malignant_checkpoints/inception_v*/all.(see fig ..) 3. Set path of new image in code as input. 4. Run prediction code, results were the probability for each class.

Fig. 10. TMA crawling steps

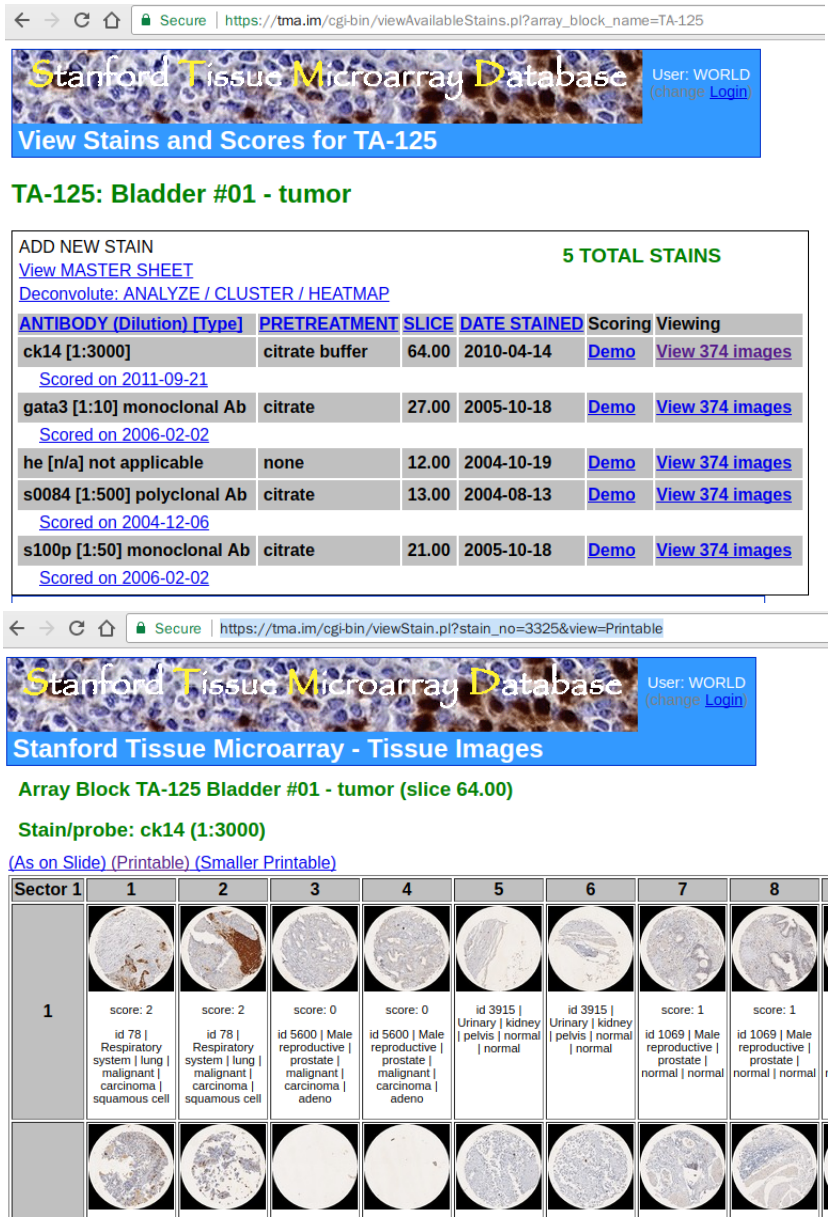


Fig. 11. TMA crawling steps

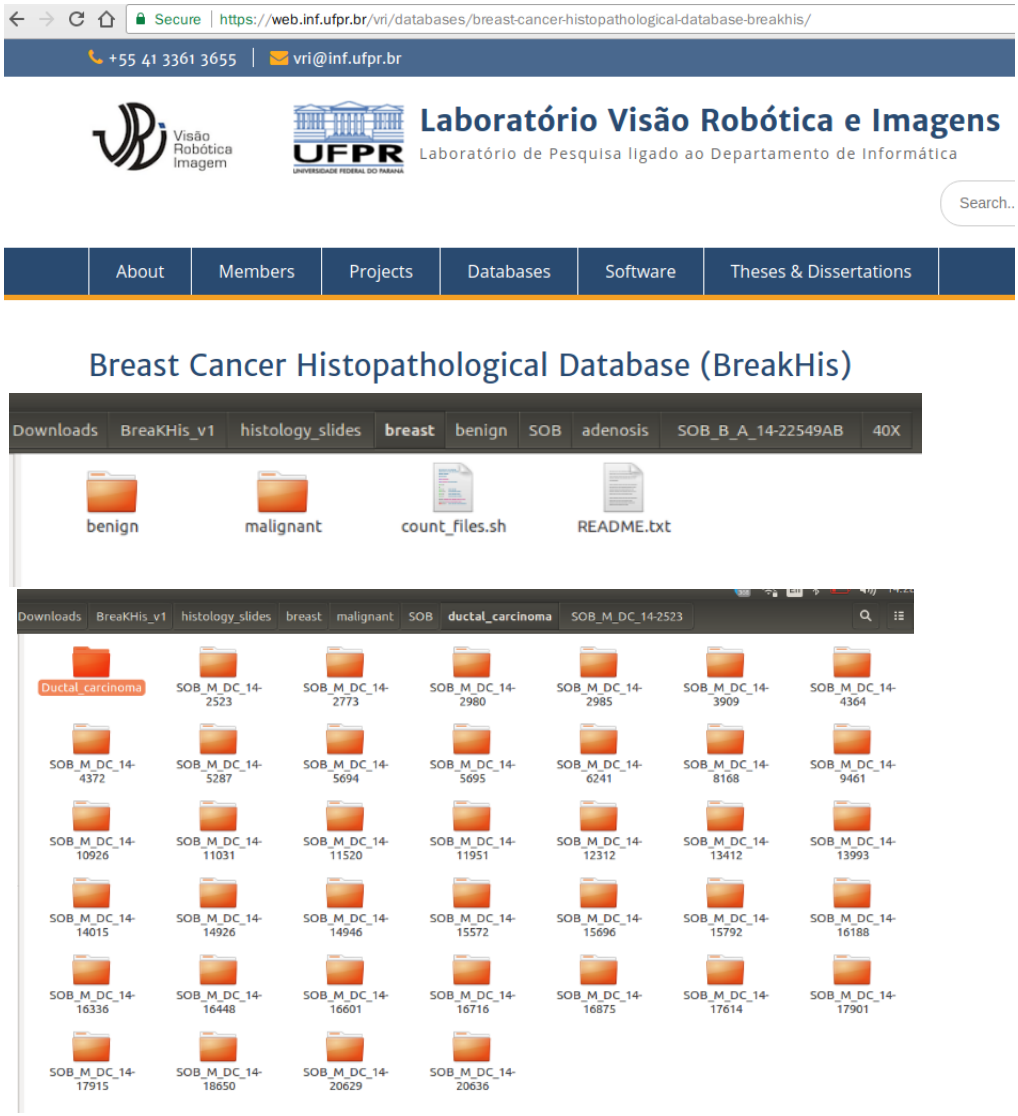
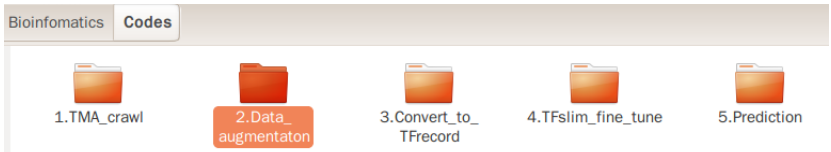
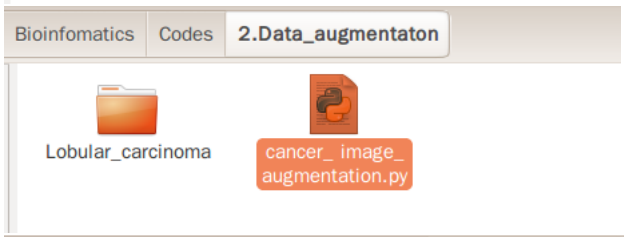
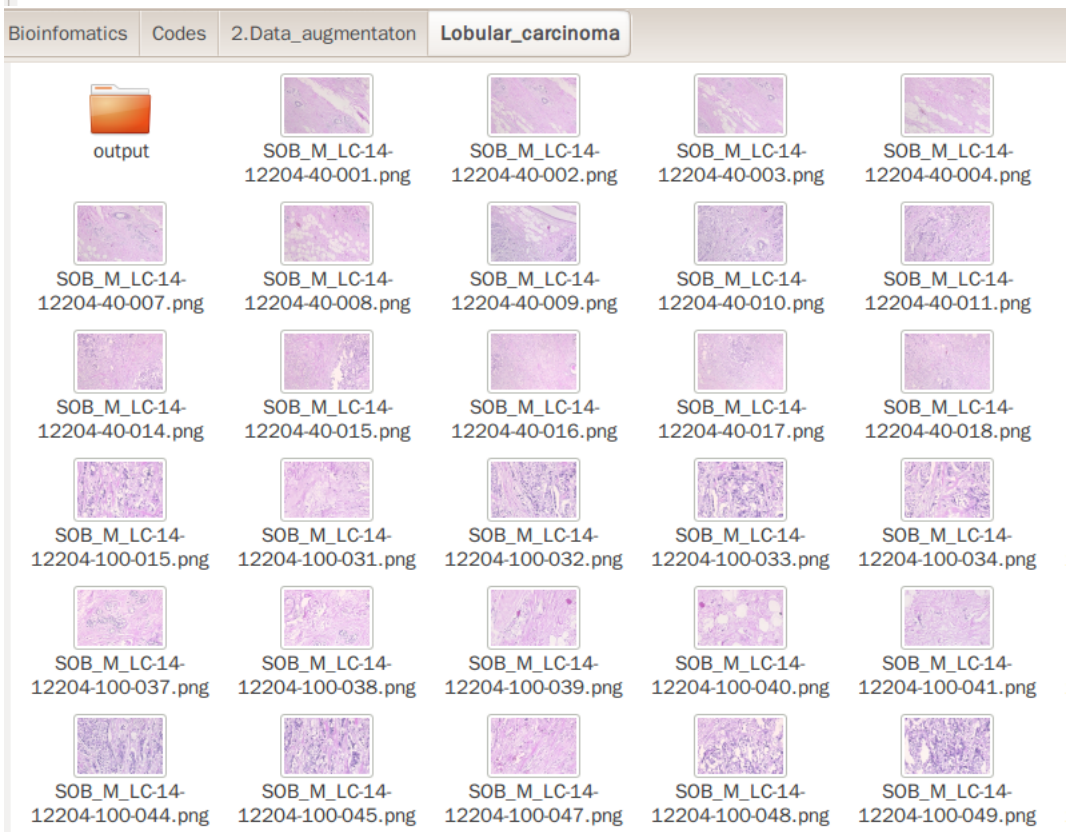
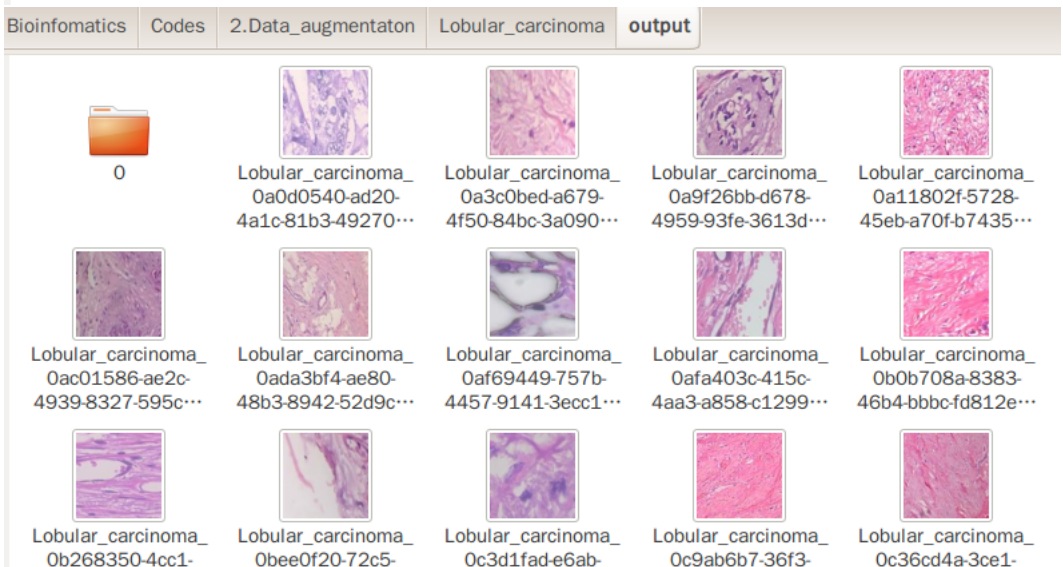


Fig. 12. BreakHis data


Bioinfomatics
Codes
1.TMA_crawl
2.Data_ augmentaton
3.Convert_to_ TFrecord
4.TFslim_fine_tune
5.Prediction


Bioinfomatics
Codes
2.Data_augmentaton
Lobular_carcinoma
cancer_ image_ augmentation.py


Bioinfomatics
Codes
2.Data_augmentaton
Lobular_carcinoma
output
SOB_M_LC-14-12204-40-001.png
SOB_M_LC-14-12204-40-002.png
SOB_M_LC-14-12204-40-003.png
SOB_M_LC-14-12204-40-004.png
SOB_M_LC-14-12204-40-007.png
SOB_M_LC-14-12204-40-008.png
SOB_M_LC-14-12204-40-009.png
SOB_M_LC-14-12204-40-010.png
SOB_M_LC-14-12204-40-011.png
SOB_M_LC-14-12204-40-014.png
SOB_M_LC-14-12204-40-015.png
SOB_M_LC-14-12204-40-016.png
SOB_M_LC-14-12204-40-017.png
SOB_M_LC-14-12204-40-018.png
SOB_M_LC-14-12204-100-015.png
SOB_M_LC-14-12204-100-031.png
SOB_M_LC-14-12204-100-032.png
SOB_M_LC-14-12204-100-033.png
SOB_M_LC-14-12204-100-034.png
SOB_M_LC-14-12204-100-037.png
SOB_M_LC-14-12204-100-038.png
SOB_M_LC-14-12204-100-039.png
SOB_M_LC-14-12204-100-040.png
SOB_M_LC-14-12204-100-041.png
SOB_M_LC-14-12204-100-044.png
SOB_M_LC-14-12204-100-045.png
SOB_M_LC-14-12204-100-047.png
SOB_M_LC-14-12204-100-048.png
SOB_M_LC-14-12204-100-049.png


Bioinfomatics
Codes
2.Data_augmentaton
Lobular_carcinoma
output
0
Lobular_carcinoma_0a0d0540-ad20-4a1c-81b3-49270…
Lobular_carcinoma_0a3c0bed-a679-4f50-84bc-3a090…
Lobular_carcinoma_0a9f26bb-d678-4959-93fe-3613d…
Lobular_carcinoma_0a11802f-5728-45eb-a70f-b7435…
Lobular_carcinoma_0ac01586-ae2c-4939-8327-595c…
Lobular_carcinoma_0ada3bf4-ae80-48b3-8942-52d9c…
Lobular_carcinoma_0af69449-757b-4457-9141-3ecc1…
Lobular_carcinoma_0afa403c-415c-4aa3-a858-c1299…
Lobular_carcinoma_0b0b708a-8383-46b4-bbbc-fd812e…
Lobular_carcinoma_0b268350-4cc1-
Lobular_carcinoma_0bee0f20-72c5-
Lobular_carcinoma_0c3d1fad-e6ab-
Lobular_carcinoma_0c9ab6b7-36f3-
Lobular_carcinoma_0c36cd4a-3ce1-

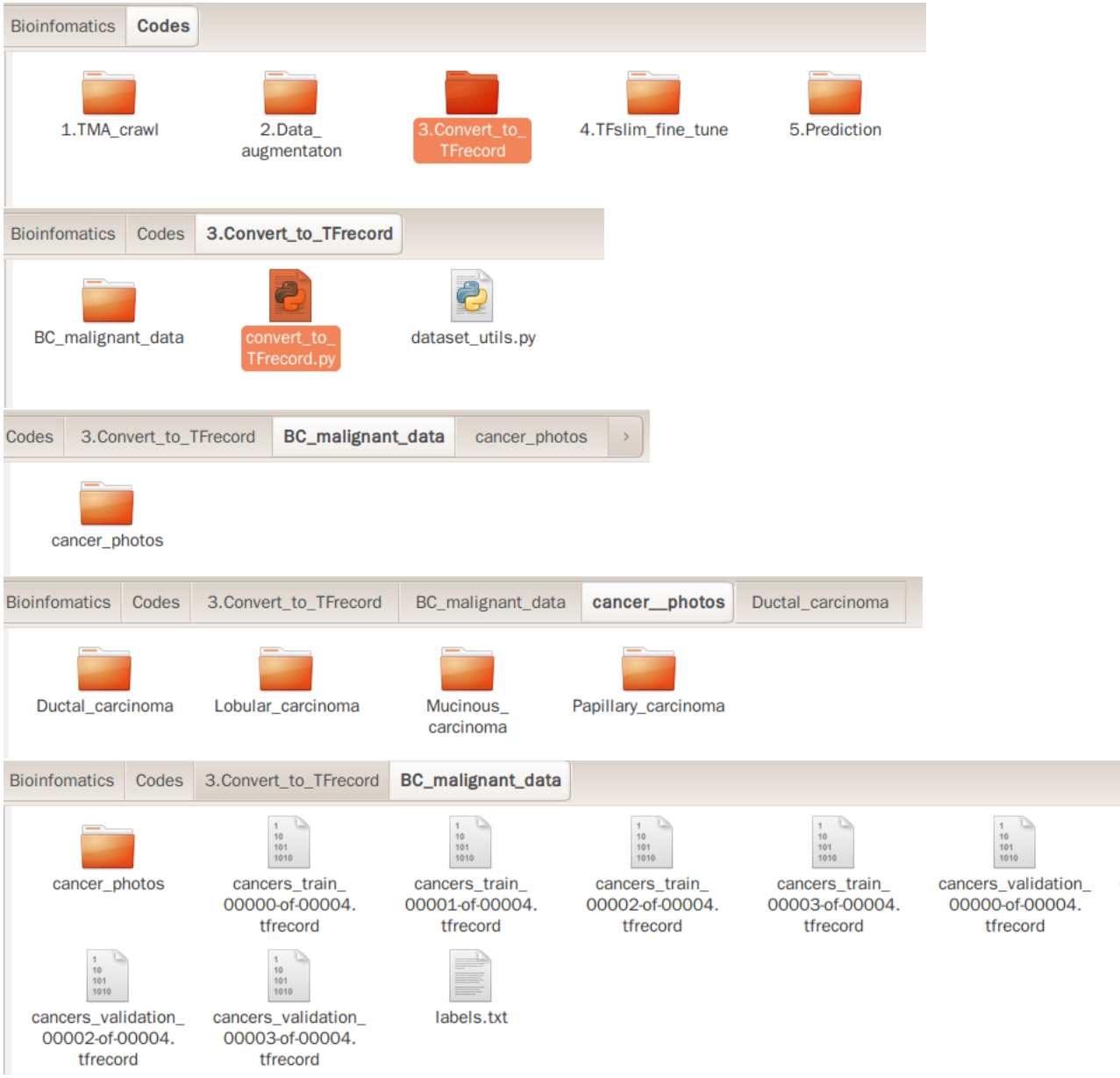
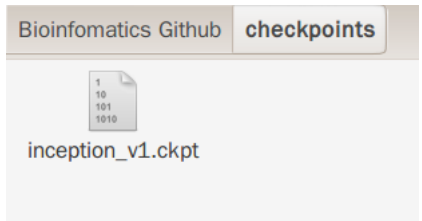


Fig. 14. Convert to TFRecord procedure




```bash
#!/bin/bash
#
# This script performs the following operations:
# 1. Fine-tunes an InceptionV1 model on the Breast Cancer (Malignant) training set.
# 2. Evaluates the model on the Breast Cancer (Malignant) validation set.
#
# Usage:
# ./scripts/finetune_inceptionv1_on_cancer.sh

# Where the pre-trained InceptionV1 checkpoint is saved to.
PRETRAINED_CHECKPOINT_DIR=/Desktop/checkpoints

# Where the training (fine-tuned) checkpoint and logs will be saved to.
TRAIN_DIR=/Desktop/BC_malignant_ckpt/inception_v1

# Where the dataset is saved to.
DATASET_DIR=/Desktop/cancers
```

https://github.com/tensorflow/models/tree/master/research/slim

Model	TF-Slim File	Checkpoint	Top-1 Accuracy	Top-5 Accuracy
Inception V1	Code	inception_v1_2016_08_28.tar.gz	69.8	89.6
Inception V2	Code	inception_v2_2016_08_28.tar.gz	73.9	91.8
Inception V3	Code	inception_v3_2016_08_28.tar.gz	78.0	93.9
Inception V4	Code	inception_v4_2016_09_09.tar.gz	80.2	95.2
Inception-ResNet-v2	Code	inception_resnet_v2_2016_08_30.tar.gz	80.4	95.3
ResNet V1 50	Code	resnet_v1_50_2016_08_28.tar.gz	75.2	92.2
ResNet V1 101	Code	resnet_v1_101_2016_08_28.tar.gz	76.4	92.9
ResNet V1 152	Code	resnet_v1_152_2016_08_28.tar.gz	76.8	93.2
ResNet V2 50^	Code	resnet_v2_50_2017_04_14.tar.gz	75.6	92.8
ResNet V2 101^	Code	resnet_v2_101_2017_04_14.tar.gz	77.0	93.7
ResNet V2 152^	Code	resnet_v2_152_2017_04_14.tar.gz	77.8	94.1
ResNet V2 200	Code	TBA	79.9*	95.2*
VGG 16	Code	vgg_16_2016_08_28.tar.gz	71.5	89.8
VGG 19	Code	vgg_19_2016_08_28.tar.gz	71.1	89.8
MobileNet_v1_1.0_224	Code	mobilenet_v1_1.0_224_2017_06_14.tar.gz	70.7	89.5
MobileNet_v1_0.50_160	Code	mobilenet_v1_0.50_160_2017_06_14.tar.gz	59.9	82.5
MobileNet_v1_0.25_128	Code	mobilenet_v1_0.25_128_2017_06_14.tar.gz	41.3	66.2

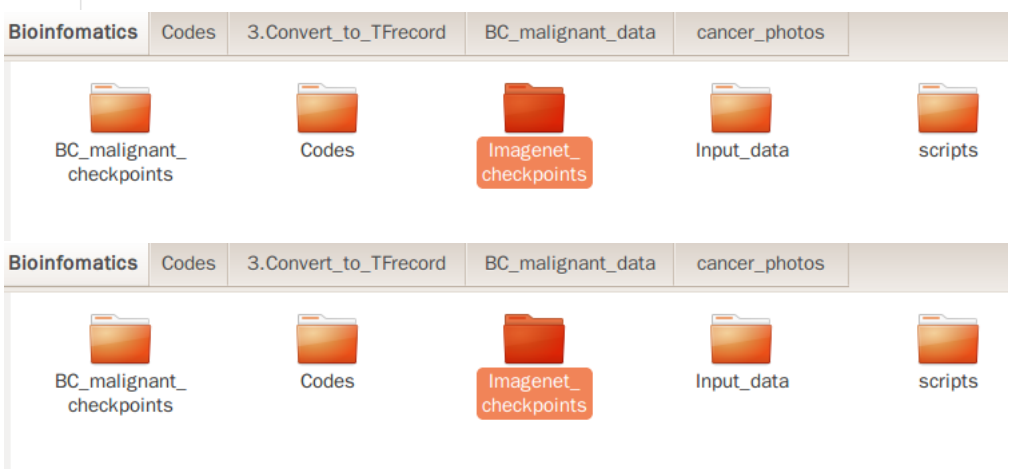


Fig. 15. Convert to TFRecord procedure

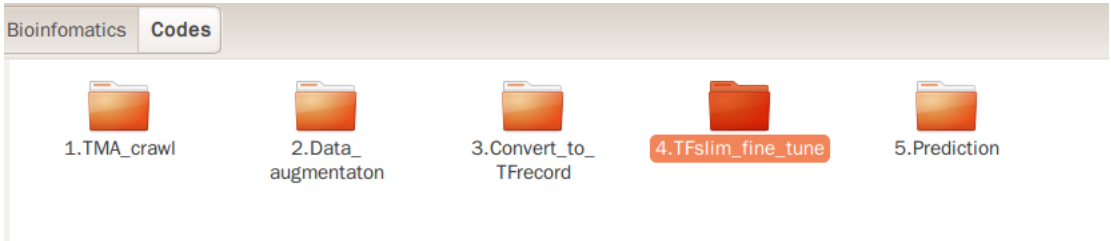


Fig. 16. All files

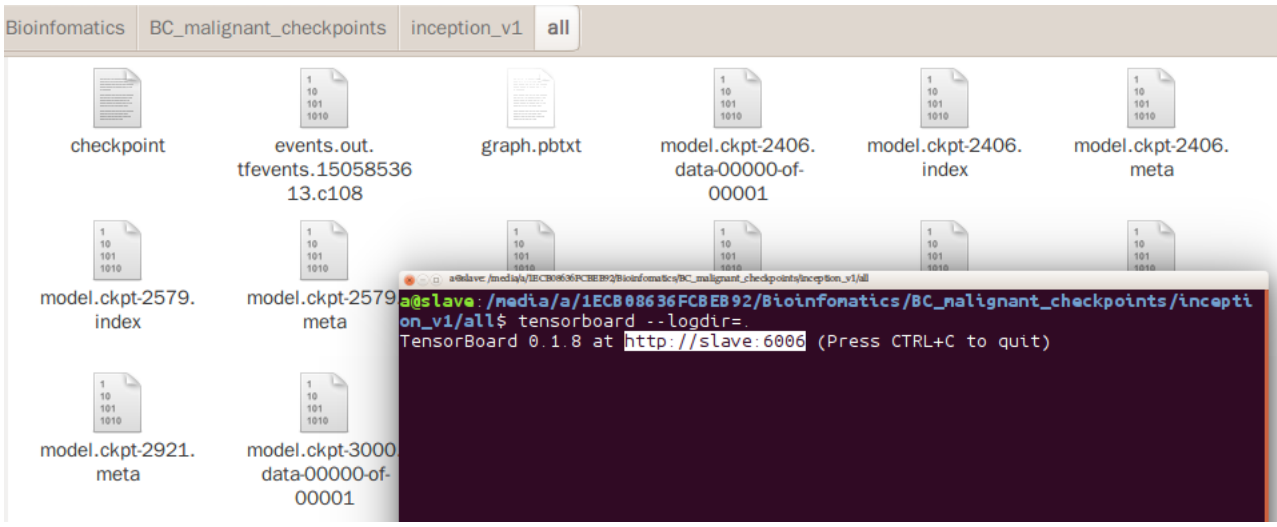


Fig. 17. Open tensorboard command

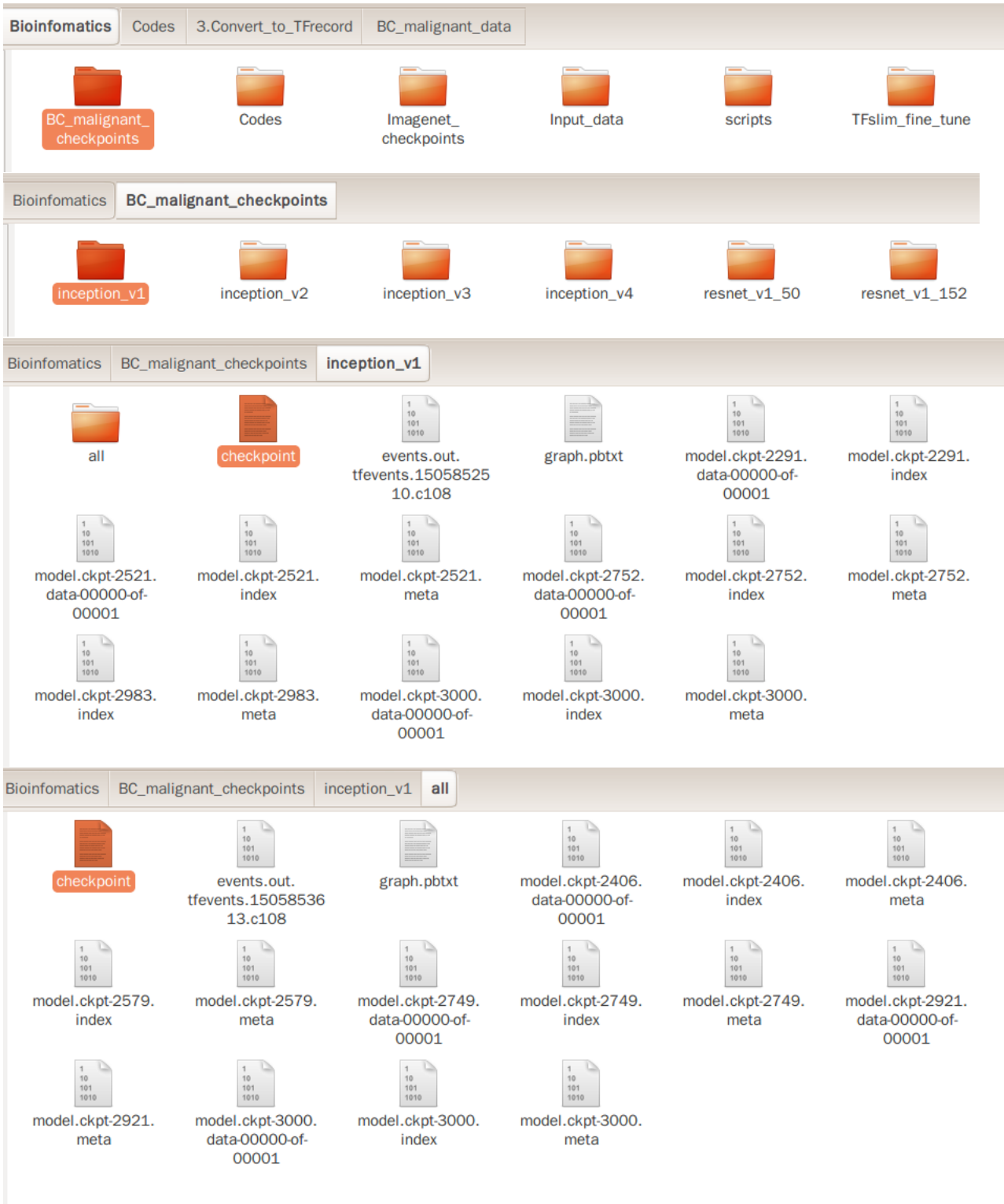


Fig. 18. Convert to TFRecord procedure

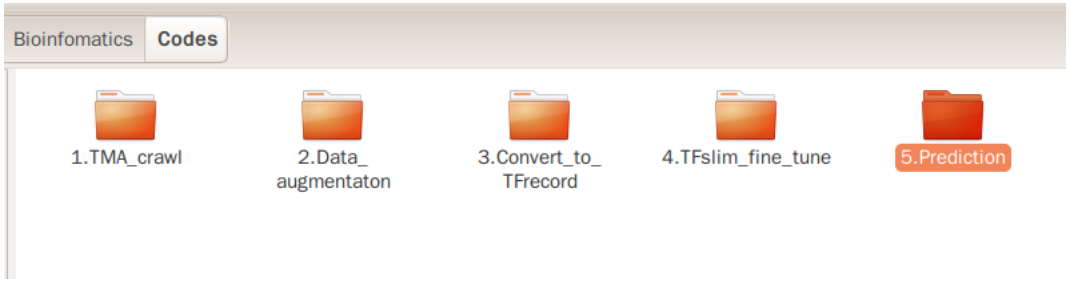


Fig. 19. All files

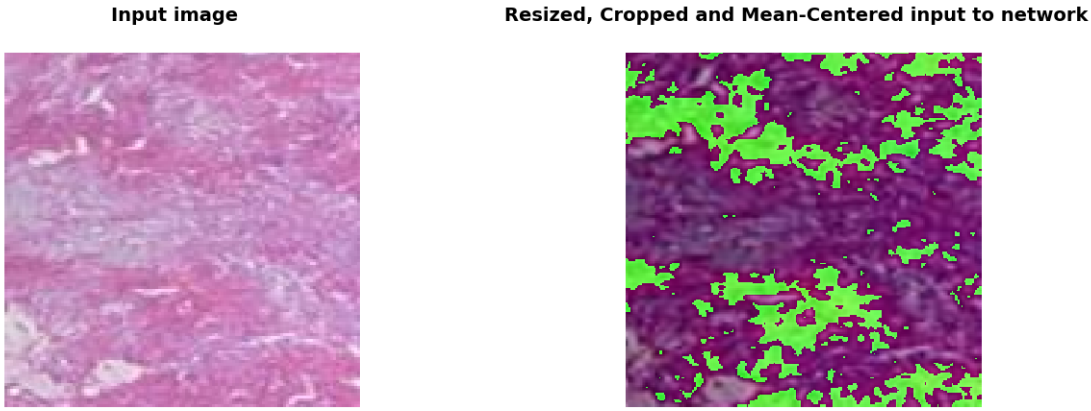
Prediction_InceptionV1.py ×

Python interpreter configured for the project

```
'''
All rights reserved", Royan Institute for Stem Cell Biology and Technology,
Oct 2017  ( Mehdi Habibzadeh et al)
'''


import ...
# from preprocessing import inception_preprocessing

slim = tf.contrib.slim

# checkpoints path of fine-tunning last layer
# checkpoints_path = '/home/habibzadeh/Desktop/BC_malignant_checkpoints/inception_v1'

#checkpoints path of fine-tunning all layers
checkpoints_path = '/home/habibzadeh/Desktop/BC_malignant_checkpoints/inception_v1/all/'
checkpoints_path = tf.train.latest_checkpoint(checkpoints_path)
print("checkpoint path is",checkpoints_path)

image_size = inception.inception_v1.default_image_size

with tf.Graph().as_default():
    image_string=tf.gfile.GFile("./Malignant_Test/Mucinous.JPEG", "rb").read()

    image = tf.image.decode_jpeg(image_string, channels=3)
    processed_image = inception_preprocessing.preprocess_image(image, image_size, image_size, is_trai
```

```
#checkpoints path of fine-tunning all layers
checkpoints_path = '/home/habibzadeh/Desktop/BC_malignant_checkpoints/inception_v1/all/'
#checkpoints path of fine-tunning last layer

checkpoints_path = '/home/habibzadeh/Desktop/BC_malignant_checkpoints/inception_v1'


with tf.Graph().as_default():
    image_string=tf.gfile.GFile("./Malignant_Test/Mucinous.JPEG", "rb").read()
```

Fig. 20. Input of prediction code



Probability 0.9981 => [Mucinous carcinoma]
Probability 0.0019 => [Lobular carcinoma]
Probability 0.0000 => [Papillary carcinoma]
Probability 0.0000 => [Ductal carcinoma]

Fig. 21. Prediction for new image