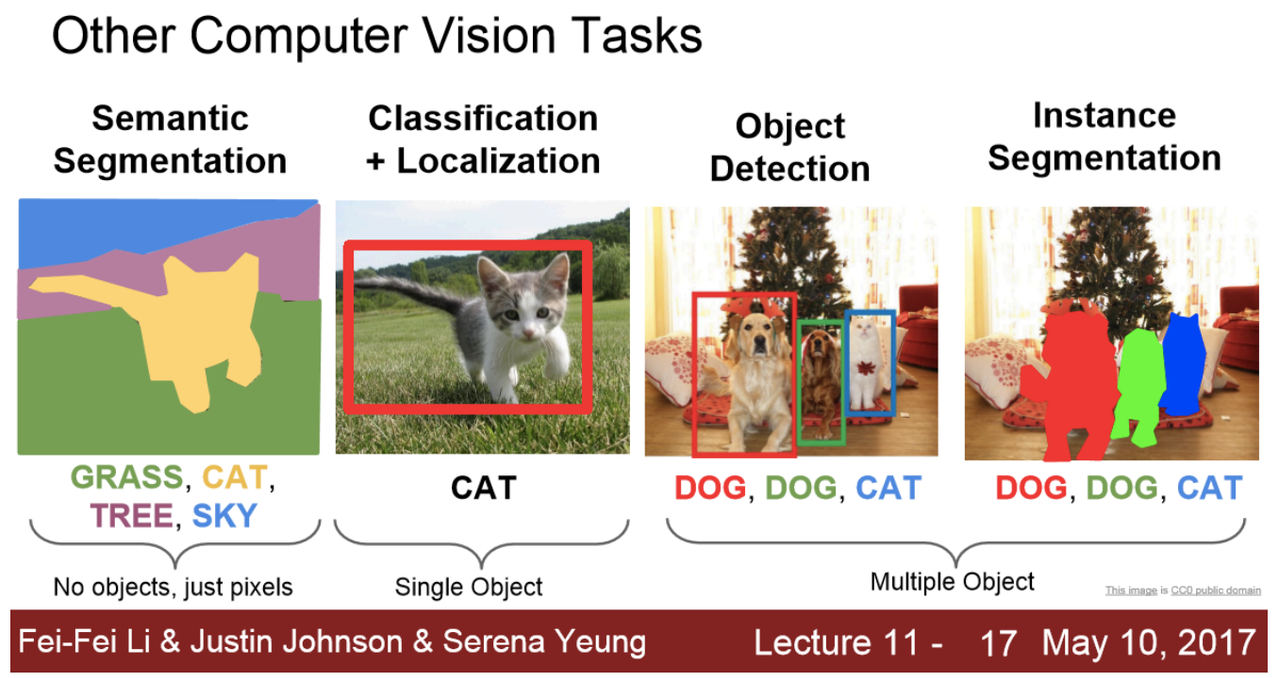
FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

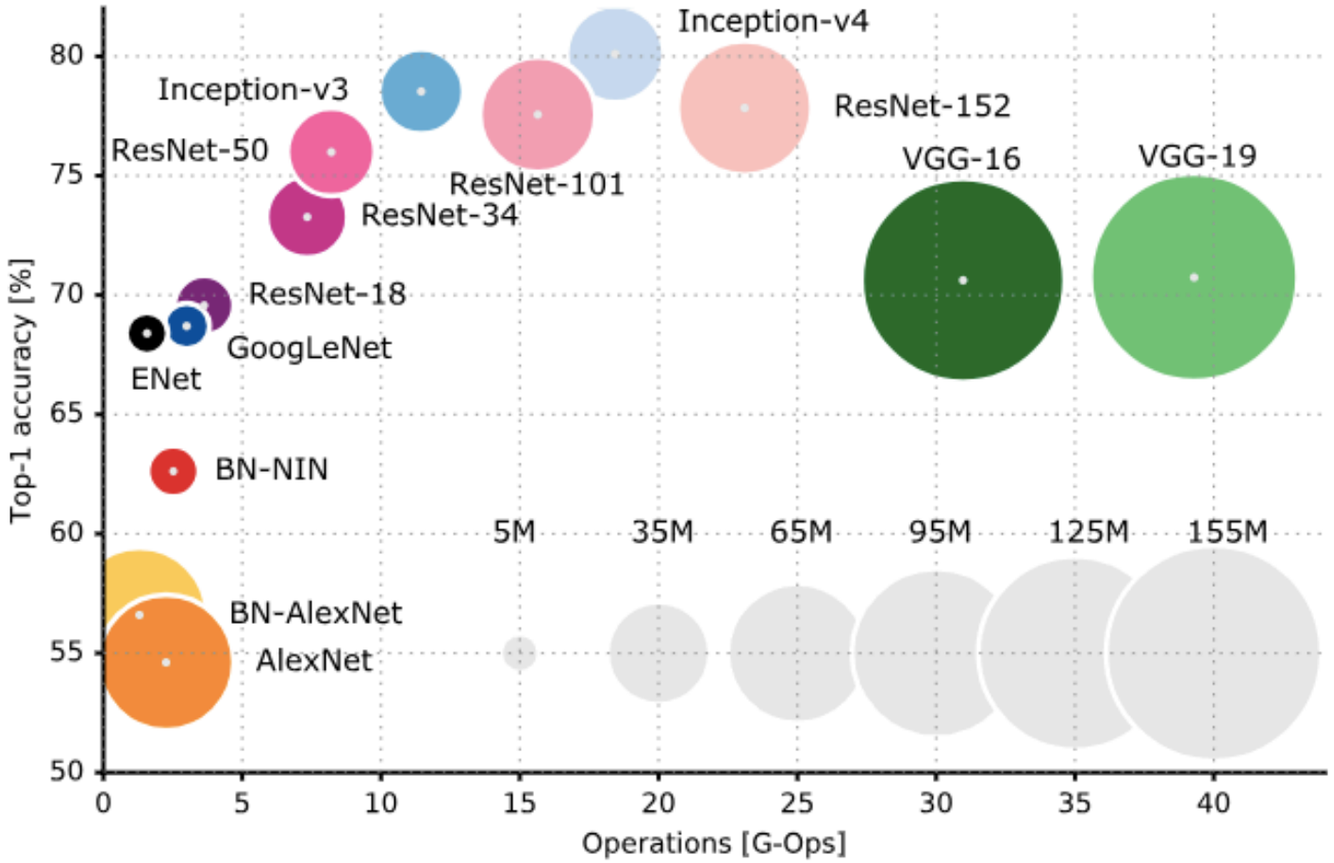
What should be the image size, or is it important? If it is, how can we manage?

- First of all, you should choose the state-of-the-art architecture for your problem.



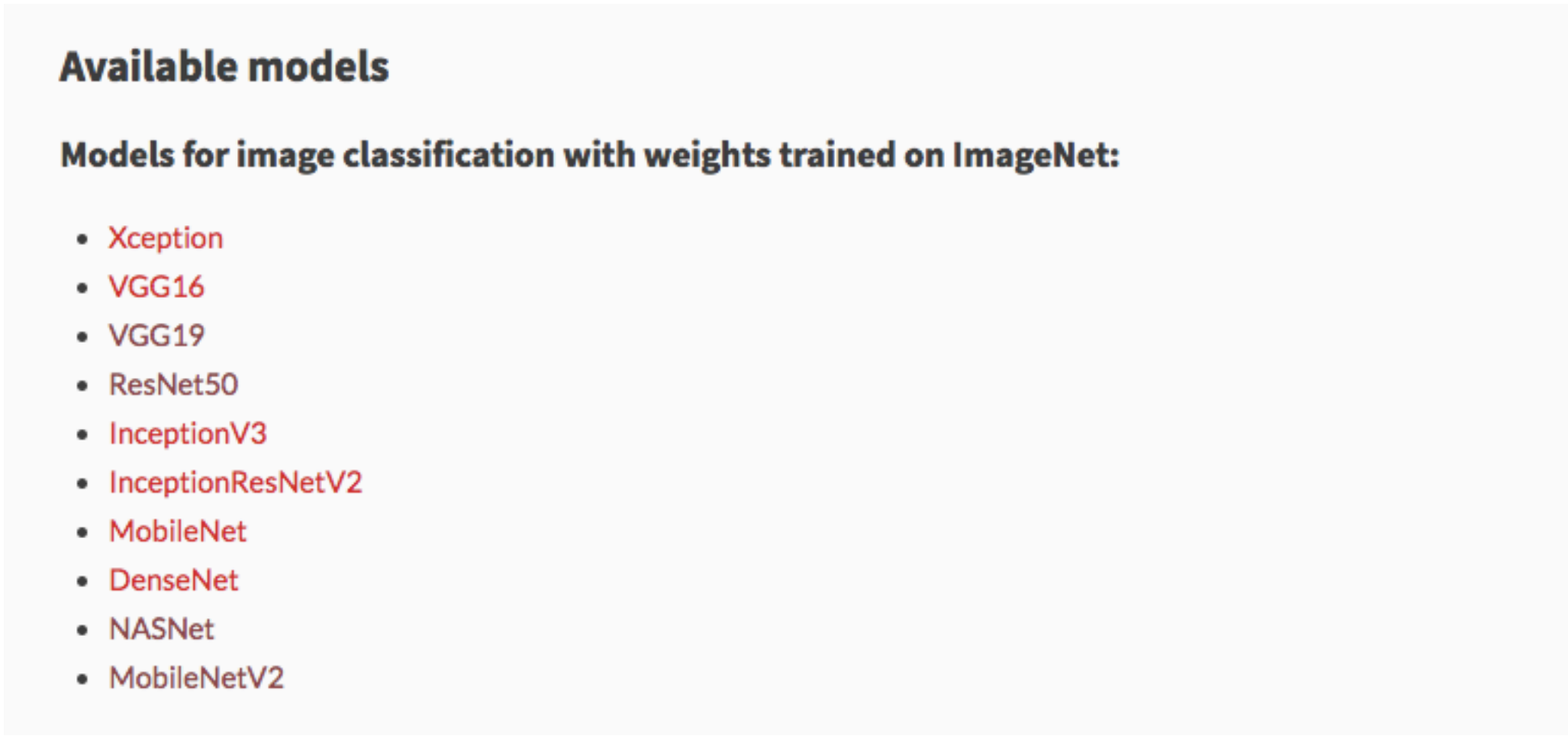
- YOLO for object detection, VGG19(a good one for small projects) for object classification

- Because there are advantages of choosing the state-of-the-art architectures:
- These models are proven methods. Their performance are approved by the articles and competitions.



- https://culurciello.github.io/tech/2016/06/20/training-enet.html

- In deep learning frameworks, there are ready to use codes and weights to apply them(crucial for transfer learning,finetuning).



Available models

Models for image classification with weights trained on ImageNet:

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3
- InceptionResNetV2
- MobileNet
- DenseNet
- NASNet
- MobileNetV2

➤ https://keras.io/applications/

- Thus, you should choose the satisfying architecture for your problem. Then you should check the “input sizes”.
- VGG16-VGG19 : 224x224
- InceptionV3 : 299x299
- NasNetMobile : 224x224

- You don’t have to find 224x224 images, you can just resize bunch of images with few lines of code of:
- SciPy
- OpenCV
- Keras
- PIL
- What should be the data size? How many images per class?

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

What should be the data size? How many images per class?

- Every small piece of data is valuable. You should add as much as you can.
- But the question is, how large the data should be for "satisfying" results?

- Let’s try to find answer for this question, theoretically and practically. But I should add that practical answers are more. These are the answers of practitioners.

It really is pretty hard (if not impossible) to give a 'one size fits all' answer. The amount of training data you require is dependent on many different aspects of your experiment:

- how different are the classes that you're trying to separate? e.g. if you were just trying to classify black versus white images then you'd need very few training examples! But if you're trying to solve ImageNet then you need training data on the order of 1000 examples per class.
- how aggressively can you augment the training data? (Sander Dieleman's blog post on 'Classifying plankton with deep neural networks' gives a great introduction to data augmentation; and he talks a little about the question of 'how much data do deep neural nets require')
- can you use pre-trained weights to initialise the lower layers of your net? (e.g. using weights trained from ImageNet)
- do you plan to use batch normalisation? It can help reduce the amount of data required.
- There are loads of other 'tricks' discussed in the literature on ways to squeeze maximum advantage from a small training dataset.

http://benanne.github.io/2015/03/17/plankton.html

- https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN

case by case it could be different, there are some cases you may not be able to even collect 30 sample for training the system like if the research is on tsunami. Hence, there is no minimum or maximum for training sample size. Generally the more sample for training can ensure better performance of system but make sure dont get over trained your network.

- https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN

In practical cases, more data is needed for training a deep model since you want to see the model's ability to generalize. No one can tell you how much data you need because the answer is " It is unknowable" the reason why of the complexity of the problem and the model implemented.

The size of the data is not very important. What is important is the variations in the samples you have. That is why data augmentation increases the accuracy and the generalization of a CNN model. However, you need to have additional training data if your model is over-fitting. If your model does not over-fit with n samples, you only can get benefits of additional good quality data in improving the generalization of the model. In other words, you need additional data if you need more accurate model. If you are just fine with an accuracy of 80%, thus just get n samples of training data that achieves that accuracy.

- https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN

- **You don't need Google-scale data to use deep learning.** Using all of the above means that even your average person with only a 100-1000 samples can see some benefit from deep learning. With all of these techniques you can mitigate the variance issue, while still benefitting from the flexibility. You can even build on others work through things like transfer learning.

➤ https://beamandrew.github.io/deeplearning/2017/06/04/deep_learning_works.html

The Fundamental Theorem of Statistical Learning Theory:

Let H be a hypothesis class of functions from a domain X to {0, 1} and let the loss function be the 0 − 1 loss. Assume that VCdim(H) = d < ∞. Then, there are absolute constants C1, C2 such that: H is agnostic PAC learnable with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_{\mathcal{H}}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

the hypothesis class H will probably(with confidence 1-δ) approximately(up to an error of e(epsilon)) source pp.73

So if we want an error of 0.05 with a confidence of 0.95 and if we assume d =1 (just for approximating although d in practical situations will be way more than 1) then m = 1600 approximately.

But as we know that VC dimension of neural network depends on the neural net selected and on the data and hence on the task selected(as mentioned by Ian Goodfellow). But if i were you I will definitely give 10 times more examples than the above estimate.

- https://www.quora.com/How-many-images-data-do-I-need-to-start-training-a-deep-neural-network-from-scratch-1

How much data is needed to train a medical image deep learning system to achieve necessary high accuracy?

Junghwan Cho, Kyewook Lee, Ellie Shin, Garry Choy, and †Synho Do *
Department of Radiology
Massachusetts General Hospital and Harvard Medical School
Boston, MA, USA

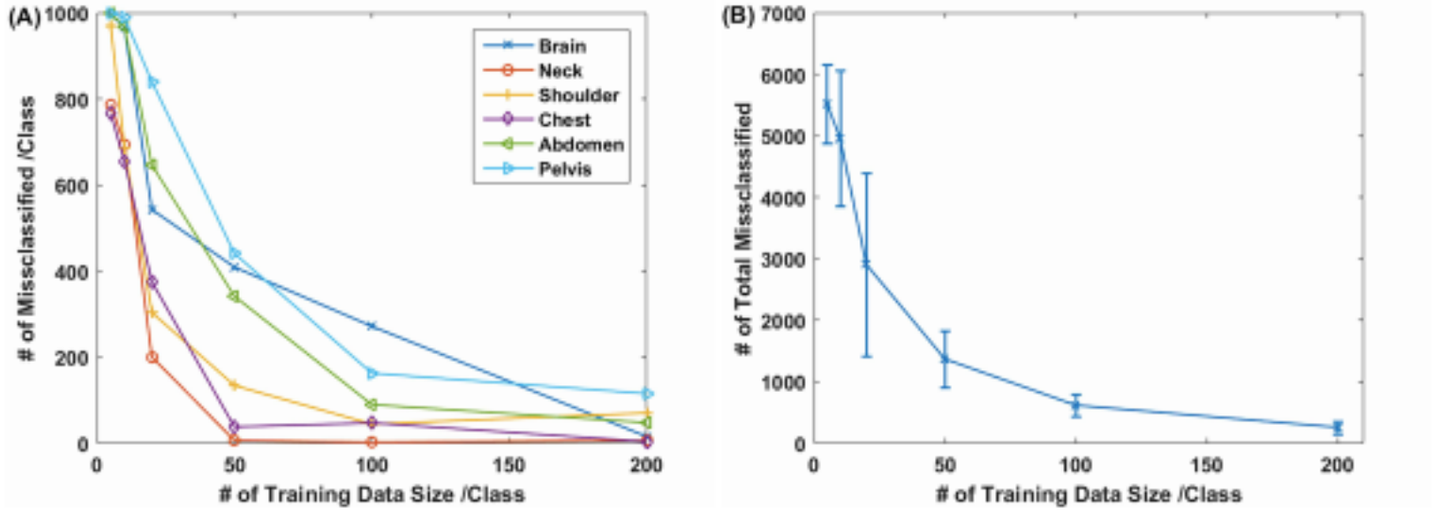


Figure 4: (A) The number of misclassified images on each body part class and (B) of total misclassified ones on whole body in increasing number of training data sets.

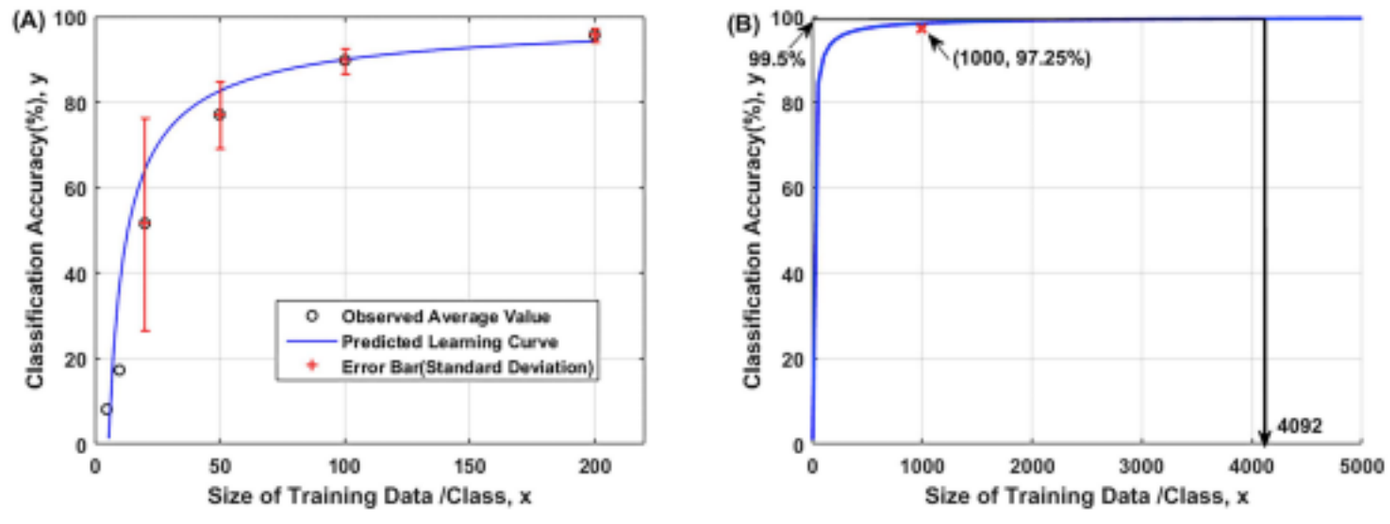


Figure 5: (A) The predicted learning curve and (B) tested result at large data set.

- Nearly 1000 images for starting to get satisfying results,
- Of course, it’s not a rule of thumb.
- Of course, it depends on complexity, variance of dataset.

- However, whatever the data size is, if you can augment the data and get closer to the 1000, you will also get good results.

- Classify grayscale images of plankton into one of 121 classes.
- Some classes have nearly 20 images.

- “Our best models achieved an accuracy of over 82% on the validation set, and a top-5 accuracy of over 98%. ”

- How?
- http://benanne.github.io/2015/03/17/plankton.html
- Hint: Balancing, Augmentation, Carefully Prepared CNN

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

Our Sample Dataset : Cats and Dogs Dataset

➤ https://www.kaggle.com/c/dogs-vs-cats

In this competition, you'll write an algorithm to classify whether images contain either a dog or a cat. This is easy for humans, dogs, and cats. Your computer will find it a bit more difficult.

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	**Pierre Sermanet**			0.98914	5	5y
2	▲4	**orchid**			0.98308	17	5y
3	—	**Owen**			0.98171	15	5y
4	—	**Paul Covington**			0.98171	3	5y
5	▼3	**Maxim Milakov**			0.98137	24	5y
6	▼1	**we've been in KAIST**			0.98102	8	5y
7	▲1	**Doug Koch**			0.98057	6	5y
8	▲2	**fastml.com/cats-and-dogs**			0.98000	6	5y
9	▲3	**Luca Massaron**			0.97965	23	5y
10	▼1	**Daniel Nouri**			0.97851	13	5y
11	▲2	**anton**			0.97771	6	5y
12	▲2	**miacis**			0.97748	7	5y
13	▲6	**DaggerFS**			0.97691	4	5y
14	▼7	**jun**			0.97645	13	5y
15	—	**Matt**			0.97542	47	5y
16	▲2	**Boost.Wu**			0.97440	2	5y
17	▲3	**Charlie**			0.97417	10	5y
18	▼7	**Kien Trinh**			0.97405	10	5y

- We will edit this data for the purpose of facing with imbalanced dataset problem and image augmentation, but not now.

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

How to arrange the data? Why is it a subject of this lecture?

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

How to load images to Python? Which libraries will be used?

- There many ways to load photos to the Python environment.
- We will apply them, then we will talk about the advantages, and speed comparisons.
- SciPy
- OpenCV
- PIL
- NumPy
- Keras' function

- !You should firstly identify your CNN model.
- !You should load images for prediction with the same method of loading images for training.

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

Methodology to Load Images

- Our methodology for non-Keras Method:
- First identify the CNN.
- Identify its input shape(image size)
- Take the path of the images
- Load class by class
- Note the number of images per class
- Save it
- Then prepare output according to (Note the number of images per class)
- Load and train

OPENCV

```
"""
1- Firstly find CNN
2- Identify the image size
3- To load the dataset, prepare the data:
    Take the adress of images, then load it with opencv
4- Load images with  OPENCV/SCİPY
5- Resize 224,224
6- Save it to numpy format
7- We need to prepare output()
"""


import glob
import cv2
import numpy as np


catimagenumber = 0


catdata = np.array([])

for imagepath in glob.glob("kaggledata/PetImages/cat2/*"):
    try:
        image = cv2.imread(imagepath)
        resizedimage = cv2.resize(image,(224,224))
        catdata = np.append(catdata,resizedimage)

        catimagenumber = catimagenumber +1
    except:
        print(imagepath)
print(catimagenumber)
np.save("catdata",catdata)


#####--------

dogimagenumber = 0


dogdata = np.array([])

for imagepath in glob.glob("kaggledata/PetImages/dof2/*"):
    try:
        image = cv2.imread(imagepath)
        resizedimage = cv2.resize(image,(224,224))
        dogdata = np.append(dogdata,resizedimage)

        dogimagenumber = dogimagenumber +1
    except:
        print(imagepath)
print(dogimagenumber)
np.save("dogdata",dogdata)

➤
```

SCIPY

```
"""
1- Firstly find CNN
2- Identify the image size
3- To load the dataset, prepare the data:
    Take the adress of images, then load it with opencv
4- Load images with  OPENCV/SCİPY
5- Resize 224,224
6- Save it to numpy format
7- We need to prepare output()
"""


import glob
from scipy     import misc
import numpy as np


catimagenumber = 0


catdata = np.array([])

for imagepath in glob.glob("kaggledata/PetImages/cat2/*"):
    try:
        image = misc.imread(imagepath)
        resizedimage = misc.imresize(image,(224,224))
        catdata = np.append(catdata,resizedimage)

        catimagenumber = catimagenumber +1
    except:
        print(imagepath)
print(catimagenumber)
np.save("catdata",catdata)


#####--------

dogimagenumber = 0


dogdata = np.array([])

for imagepath in glob.glob("kaggledata/PetImages/dog2/*"):
    try:
        image = misc.imread(imagepath)
        resizedimage = misc.imresize(image,(224,224))
        dogdata = np.append(dogdata,resizedimage)

        dogimagenumber = dogimagenumber +1
    except:
        print(imagepath)
print(dogimagenumber)
np.save("dogdata",dogdata)

➤
```

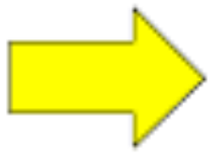
PIL

```
"""
1- Firstly find CNN
2- Identify the image size
3- To load the dataset, prepare the data:
    Take the adress of images, then load it with opencv
4- Load images with  OPENCV/SCİPY
5- Resize 224,224
6- Save it to numpy format
7- We need to prepare output()
"""


import glob
from PIL import Image
import numpy as np


catimagenumber = 0


catdata = np.array([])



for imagepath in glob.glob("kaggledata/PetImages/cat2/*"):
    try:
        image = Image.open(imagepath)

        image.thumbnail((224,224),Image.ANTIALIAS)
        catdata = np.append(catdata,image)



        catimagenumber = catimagenumber +1
    except:
        print(imagepath)


print(catimagenumber)
np.save("catdata",catdata)


#####--------

dogimagenumber = 0


dogdata = np.array([])

for imagepath in glob.glob("kaggledata/PetImages/dog2/*"):
    try:
        image = Image.open(imagepath)
        image.thumbnail((224,224),Image.ANTIALIAS)
        dogdata = np.append(dogdata,image)

        dogimagenumber = dogimagenumber +1
    except:
        print(imagepath)
print(dogimagenumber)
np.save("dogdata",dogdata)

➤
```

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

One Hot Encoding

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

- https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding
-

FEATURES OF APPROPRIATE DATASET, HOW TO LOAD IMAGE DATASETS TO THE PYTHON?

Keras' Image Data Generator

```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
        #shear_range=0.2,
        #zoom_range=0.5,
        #width_shift_range=0.5,
        #height_shift_range=?,
        #rotation_range = 10,
        #horizontal_flip=True,
        fill_mode = 'constant',
        cval = 0., # value to fill input images when fill_mode='constant'
        label_cval = 11. # value to fill labels when fill_mode='constant'
        )
val_datagen = ImageDataGenerator(
        fill_mode = 'constant',
        cval = 0.,
        label_cval = 11.
        )
```

```
train_flow = train_datagen.flow_from_directory(
        X_path, Y_path,
        target_size=(h, w),
        batch_size=batch_size,
        shuffle = True,
        #save_to_dir = os.path.join('camvid', 'debugs'), # uncomment to save (image, label) to dir for debuging mode
        #save_prefix = 'd',
        #save_format = 'png',
        class_mode=None
        )

val_flow = val_datagen.flow_from_directory(
        val_X_path, val_Y_path,
        target_size=(h, w),
        batch_size=batch_size,
        shuffle= False,
        #save_to_dir = os.path.join('camvid', 'debugs'),
        #save_prefix = 'd',
        #save_format = 'png',
        class_mode=None
        )

model.fit_generator(train_flow,
                    steps_per_epoch = len(train_flow)/batch_size,
                    validation_data=val_flow,
                    validation_steps =len(val_flow)/batch_size,
                    epochs=epochs,
                    #callbacks=[reduce, tb, early],
                    verbose=1
                    )
```

➤