3. Feature Extraction (Stanford dogs dataset)

Before running, please visit Stanford dogs dataset (http://vision.stanford.edu/aditya86/ImageNetDogs/) and download images.tar (http://vision.stanford.edu/aditya86/ImageNetDogs/images.tar). Create a folder named "data" in the directory storing this Python notebook (if you haven't done so in step 1) and unzip the content inside "data" folder.

Loading packages

In [1]:
```
import mxnet as mx
from mxnet import init, gluon, nd, autograd, image
from mxnet.gluon import nn
from mxnet.gluon.data import vision
from mxnet.gluon.model_zoo import vision as models
import numpy as np
import pandas as pd
from tqdm import tqdm
import cv2
import h5py
import os
from glob import glob
import matplotlib.pyplot as plt

%matplotlib inline
%config InlineBackend.figure_format = 'retina'

# Change the following to mx.cpu() if you don't have GPU in your compute
# To use different GPU, you can try "ctx = mx.gpu(1)", where 1 is the fi
ctx = mx.gpu()
```

Setting parameters

In [2]:
```
data_dir = "data"

#288 = 224 + 32 *2, 352 = 224 + 32 * 4
imageSize_resnet = 288

# 363 = 299 + 32 *2, 427 = 299 + 32 * 4
imageSize_inception = 363
```

Code

In [3]:
```
def accuracy(output, labels):
    return nd.mean(nd.argmax(output, axis=1) == labels).asscalar()


def evaluate(net, data_iter):
    loss, acc, n = 0., 0., 0.
    steps = len(data_iter)
    for data, label in data_iter:
        data, label = data.as_in_context(ctx), label.as_in_context(ctx)
        output = net(data)
        acc += accuracy(output, label)
        loss += nd.mean(softmax_cross_entropy(output, label)).asscalar()
    return loss/steps, acc/steps
```

In [4]:
```
n = len(glob(os.path.join('.', data_dir, "Images", "*", "*.jpg")))

mean = np.array([0.485, 0.456, 0.406])
std = np.array([0.229, 0.224, 0.225])
```

In [5]:
```
%%time

net = models.get_model('resnet152_v1', pretrained=True, ctx=ctx)
features = []
for j in tqdm(range(0,161)):
    i = 0
    temp = nd.zeros((128, 3, imageSize_resnet, imageSize_resnet))
    for file_name in glob(os.path.join(data_dir, "Images", "*", "*.jpg")
        img = cv2.imread(file_name)
        img_224 = ((cv2.resize(img, (imageSize_resnet, imageSize_resnet)
                    / 255.0 - mean) / std).transpose((2, 0, 1))
        temp[i] = nd.array(img_224)
        nd.waitall()
        i += 1
    if j == 160:
        temp = temp[0:100]
    data_iter_224 = gluon.data.DataLoader(gluon.data.ArrayDataset(temp),
    for data in data_iter_224:
        feature = net.features(data.as_in_context(mx.gpu()))
        feature = gluon.nn.Flatten()(feature)
        features.append(feature.as_in_context(mx.cpu()))
    nd.waitall()
features = nd.concat(*features, dim=0)
print(features.shape)
nd.save(os.path.join(data_dir, 'features_res.nd'), features)
```

```
100%|██████████| 161/161 [12:27<00:00,  9.36s/it]

(20580, 2048)
CPU times: user 10min 32s, sys: 2min 9s, total: 12min 41s
Wall time: 12min 29s
```

In [6]:
```
%%time

net = models.get_model('inceptionv3', pretrained=True, ctx=ctx)
features = []
for j in tqdm(range(0,161)):
    i = 0
    temp = nd.zeros((128, 3, imageSize_inception, imageSize_inception))
    for file_name in glob(os.path.join(data_dir, "Images", "*", "*.jpg")
        img = cv2.imread(file_name)
        img_299 = ((cv2.resize(img, (imageSize_inception, imageSize_ince
                    / 255.0 - mean) / std).transpose((2, 0, 1))
        temp[i] = nd.array(img_299)
        nd.waitall()
        i += 1
    if j == 160:
        temp = temp[0:100]
    data_iter_299 = gluon.data.DataLoader(gluon.data.ArrayDataset(temp),
    for data in data_iter_299:
        feature = net.features(data.as_in_context(mx.gpu()))
        feature = gluon.nn.Flatten()(feature)
        features.append(feature.as_in_context(mx.cpu()))
    nd.waitall()
features = nd.concat(*features, dim=0)
print(features.shape)
nd.save(os.path.join(data_dir, 'features_incep.nd'), features)
```

```
100%|██████████| 161/161 [12:17<00:00, 15.75s/it]

(20580, 2048)
CPU times: user 8min 55s, sys: 4min 34s, total: 13min 30s
Wall time: 12min 17s
```