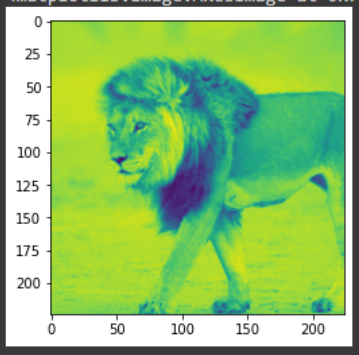
This project is inspired by https://savan77.github.io/blog/imagenet_adv_examples.html
Source code from github:
https://github.com/soumyac1999/FGSM-Keras/blob/master/targeted_attack.ipynb
The source code choose keras as target model. My program choose pytorch as
The program choose pytorch deep learning model as my adversarial target. As we know, deep learning can be used to classify images. This program can generate adversarial examples that can be misclassified by the deep learning model without making human distinguishable change to the original picture. For example, given a picture of lion, run the deep leaning model on the picture, the model will classify the picture correctly.

```
[ ]  plt.imshow(input_batch.cpu().detach().numpy()[0][0])
```

```
<matplotlib.image.AxesImage at 0x7f22bc061090>
```



first prediction

```
# Show top categories per image
top5_prob, top5_catid = torch.topk(probabilities, 5)
for i in range(top5_prob.size(0)):
    print(categories[top5_catid[i]], top5_prob[i].item())

print('-----------------------------------------------------')
```

```
291, lion 0.99729984998703
231, collie 0.0006902170134708285
230, Shetland_sheepdog 0.0004986550775356591
354, Arabian_camel 0.000373626098735258
676, muzzle 0.0002362161030760035
-----------------------------------------------------
```

Generating adversarial examples, target is cucumber.

```
start_time = time.time()
for i in range(500):
  output = model(input_batch)
  print('----------------------------------------------------')
  y = 943   #cucumber, or change to whatever you like
  target = Variable(torch.LongTensor([y]), requires_grad=False)
  target=target.to('cuda') #comment this line out if CPU only
  loss = torch.nn.CrossEntropyLoss()
  loss_cal = loss(output, target)
  loss_cal.backward(retain_graph=True)
  eps = 0.03 # works for every model
  x_grad = torch.sign(input_batch.grad.data)               #calculate the sign of gradient of the loss func (with respect to input X) (adv
  input_batch.data = input_batch.data - eps * x_grad         #find adv example using formula shown above
  output_adv = model.forward(Variable(input_batch.data))   #perform a forward pass on adv example

  op_adv_probs = F.softmax(output_adv[0], dim=0)               #get probability distribution over classes
  top5_prob, top5_catid = torch.topk(op_adv_probs, 5)

  for j in range(top5_prob.size(0)):
      print(i, categories[top5_catid[j]], top5_prob[j].item())
  if(categories[top5_catid[0]] == categories[y]):       #you can comment this if you want to increase the prediction result.
    break              #you can comment this if you want to further increase the prediction result.
timeCost = [time.time() - start_time]   #get the time cost of the attack
print("Time cost: ")
print(timeCost)
```

```
----------------------------------------------------
0 230, Shetland_sheepdog 0.15545068681240082
0 154, Pekinese 0.14559978246688843
0 259, Pomeranian 0.10014822334051132
0 185, Norfolk_terrier 0.05047866702079773
0 186, Norwich_terrier 0.05024155974381226
----------------------------------------------------
1 187, Yorkshire_terrier 0.04591984674334526
1 377, marmoset 0.04173104465007782
1 380, titi 0.037488460540771484
1 722, ping-pong_ball 0.03309197351336479
1 434, bath_towel 0.02897103689610958
----------------------------------------------------
2 187, Yorkshire_terrier 0.038982655853033066
```

```
 1 377, marmoset 0.04173104465007782
 1 380, titi 0.037488460540771484
 1 722, ping-pong_ball 0.03309197351336479
 1 434, bath_towel 0.02897103689610958
 ----------------------------------------------------
 2 187, Yorkshire_terrier 0.038982655853033066
 2 722, ping-pong_ball 0.03060135431587696
 2 380, titi 0.027581868693232536
 2 377, marmoset 0.022317377850413322
 2 697, pajama 0.020352039486169815
 ----------------------------------------------------
 3 948, Granny_Smith 0.04507594555616379
 3 380, titi 0.02957764081656933
 3 929, ice_lolly 0.02693130075931549
 3 187, Yorkshire_terrier 0.023635927587747574
 3 722, ping-pong_ball 0.023486550897359848
 ----------------------------------------------------
 4 948, Granny_Smith 0.07430987060070038
 4 794, shower_curtain 0.043278370052576065
 4 929, ice_lolly 0.02884283848106861
 4 943, cucumber 0.02391311526298523
 4 953, pineapple 0.022337811067700386
 ----------------------------------------------------
 5 948, Granny_Smith 0.0811830461025238
 5 952, fig 0.05886020511388779
 5 943, cucumber 0.040006063878536224
 5 794, shower_curtain 0.0363924615085125
 5 939, zucchini 0.03570163995027542
 ----------------------------------------------------
 6 952, fig 0.09596412628889084
 6 948, Granny_Smith 0.06958582997322083
 6 943, cucumber 0.06459217518568039
 6 953, pineapple 0.05076175555586815
 6 939, zucchini 0.04914021119475365
 ----------------------------------------------------
 7 943, cucumber 0.11382688581943512
 7 953, pineapple 0.07999725639820099
 7 952, fig 0.07975942641496658
 7 939, zucchini 0.04828951507806778
 7 948, Granny_Smith 0.04818175733089447
 Time cost:
 [0.1596088409423828]


ult image after attack
```

After 7 iteration, the picture is classified as cucumber but the still looks like a lion by human eyes.

result image after attack

```
plt.imshow(input_batch.cpu().detach().numpy()[0][0])
```

```
<matplotlib.image.AxesImage at 0x7f22b8d4f450>
```