

adversarial_crafts说明文档

161250098 彭俊杰

1.分类模型训练 (Classifier)

为了考虑攻击算法在不同模型间的迁移性，我针对FashionMNIST数据集训练了两个层次结构存在差异的分类模型，其中fashionMNIST模型识别准确率差不多92%（训练数据未增强），cnn5模型识别准确率大约93%（对训练数据进行数据增强），其结构分别如下：

fashionMNIST:

Layer (type)	Output Shape	Param #
Flatten_1 (Flatten)	(None, 784)	0
Reshape (Reshape)	(None, 28, 28, 1)	0
BatchNorm_1 (BatchNormalizat	(None, 28, 28, 1)	4
Conv2D_1 (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 32)	0
dropout_1 (Dropout)	(None, 14, 14, 32)	0
Conv2D_2 (Conv2D)	(None, 12, 12, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 64)	0
dropout_2 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 256)	590080
dropout_3 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 64)	16448
BatchNorm_2 (BatchNormalizat	(None, 64)	256
dense_last (Dense)	(None, 10)	650
Total params: 626,254		
Trainable params: 626,124		
Non-trainable params: 130		

cnn5:

conv2d_11 (Conv2D)	(None, 28, 28, 32)	320
batch_normalization_17 (Batch Normalization)	(None, 28, 28, 32)	128
conv2d_12 (Conv2D)	(None, 28, 28, 32)	9248
batch_normalization_18 (Batch Normalization)	(None, 28, 28, 32)	128
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 32)	0
dropout_18 (Dropout)	(None, 14, 14, 32)	0
conv2d_13 (Conv2D)	(None, 12, 12, 64)	18496
batch_normalization_19 (Batch Normalization)	(None, 12, 12, 64)	256
dropout_19 (Dropout)	(None, 12, 12, 64)	0
conv2d_14 (Conv2D)	(None, 10, 10, 128)	73856
batch_normalization_20 (Batch Normalization)	(None, 10, 10, 128)	512
dropout_20 (Dropout)	(None, 10, 10, 128)	0
conv2d_15 (Conv2D)	(None, 8, 8, 256)	295168
batch_normalization_21 (Batch Normalization)	(None, 8, 8, 256)	1024
max_pooling2d_8 (MaxPooling2D)	(None, 4, 4, 256)	0
dropout_21 (Dropout)	(None, 4, 4, 256)	0
flatten_4 (Flatten)	(None, 4096)	0
dense_9 (Dense)	(None, 512)	2097664
batch_normalization_22 (Batch Normalization)	(None, 512)	2048
dropout_22 (Dropout)	(None, 512)	0
dense_10 (Dense)	(None, 128)	65664
batch_normalization_23 (Batch Normalization)	(None, 128)	512
dropout_23 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 64)	8256
batch_normalization_24 (Batch Normalization)	(None, 64)	256
dropout_24 (Dropout)	(None, 64)	0
dense_12 (Dense)	(None, 10)	650

2.失败探索 (attack(failed).py)

在学习了GAN相关知识之后，准备借鉴其原理，尝试通过深度神经网络构建图片生成器，Generator直接向原始图片添加扰动生成对抗样本。大体设想如下：首先将之前训练好的分类模型作为Discriminator，冻结其各个层的权重（不在训练过程中变化），然后在输入之前添加若干个全连接层，用于调整原图片像素（添加扰动）；然后关键在于保持对抗样本与原始图片的相近性，最初我的做法是在作为generator的若干个全连接层中间均添加Lambda层（用clip对各个像素点的变化进行限制），不过由于keras模型构建的僵硬性，Lambda层在训练过程中被直接忽视了.....于是我决定采取另一种做法，在自定义loss函数中添加ssim相关的项，不过又苦于无法取得ssim相关项与错误分类项的平衡，又只能放弃.....

3.Github开源项目foolbox（依赖Python3,NumPy和SciPy库）

在研究对抗样本生成相关的论文及github开源项目时，偶然发现foolbox——IBM's Python toolbox to create adversarial examples (<https://github.com/bethgelab/foolbox>)，通过查阅其使用手册，发现其提供了我之前看到的各种图像攻击算法的实现，截图如下：

Gradient-based attacks

GradientAttack	Perturbs the input with the gradient of the loss w.r.t.
GradientSignAttack	Adds the sign of the gradient to the input, gradually increasing
FGSM	alias of <code>foolbox.attacks.gradient.GradientSignAttack</code>
LinfinitBasicIterativeAttack	The Basic Iterative Method introduced in [R37dbc8f24aee-1].
BasicIterativeMethod	alias of <code>foolbox.attacks.iterative_projected_gradient.LinfinitBas</code>
BIM	alias of <code>foolbox.attacks.iterative_projected_gradient.LinfinitBas</code>
L1BasicIterativeAttack	Modified version of the Basic Iterative Method that minimizes
L2BasicIterativeAttack	Modified version of the Basic Iterative Method that minimizes
ProjectedGradientDescentAttack	The Projected Gradient Descent Attack introduced in [R367e8
ProjectedGradientDescent	alias of <code>foolbox.attacks.iterative_projected_gradient.ProjecteGra</code>
PGD	alias of <code>foolbox.attacks.iterative_projected_gradient.ProjecteGra</code>
RandomStartProjectedGradientDescentAttack	The Projected Gradient Descent Attack introduced in [Re6066
RandomProjectedGradientDescent	alias of <code>foolbox.attacks.iterative_projected_gradient.RandomStartP</code>
RandomPGD	alias of <code>foolbox.attacks.iterative_projected_gradient.RandomStartP</code>
MomentumIterativeAttack	The Momentum Iterative Method attack introduced in [R86d3
MomentumIterativeMethod	alias of <code>foolbox.attacks.iterative_projected_gradient.MomentumIter</code>
LBFGSAttack	Uses L-BFGS-B to minimize the distance between the input an
DeepFoolAttack	Simple and close to optimal gradient-based adversarial attack.
NewtonFoolAttack	Implements the NewtonFool Attack.
DeepFoolL2Attack	
DeepFoolLinfinitAttack	
ADefAttack	Adversarial attack that distorts the image, i.e.
SLSQPAttack	Uses SLSQP to minimize the distance between the input and t
SaliencyMapAttack	Implements the Saliency Map Attack.
IterativeGradientAttack	Like GradientAttack but with several steps for each epsilon.
IterativeGradientSignAttack	Like GradientSignAttack but with several steps for each epsilon
CarliniWagnerL2Attack	The L2 version of the Carlini & Wagner attack.
DecoupledDirectionNormL2Attack	The Decoupled Direction and Norm L2 adversarial attack from
SparseFoolAttack	A geometry-inspired and fast attack for computing sparse adv

Score-based attacks

SinglePixelAttack	Perturbs just a single pixel and sets it to the min or max.
LocalSearchAttack	A black-box attack based on the idea of greedy local search.
ApproximateLBFGSAttack	Same as LBFGSAttack with approximate_gradient set to True.

Decision-based attacks

BoundaryAttack	A powerful adversarial attack that requires neither gradients nor probabilities.
SpatialAttack	Adversarially chosen rotations and translations [1].
PointwiseAttack	Starts with an adversarial and performs a binary search between the adversarial and the original input.
GaussianBlurAttack	Blurs the input until it is misclassified.
ContrastReductionAttack	Reduces the contrast of the input until it is misclassified.
AdditiveUniformNoiseAttack	Adds uniform noise to the input, gradually increasing the standard deviation until it is misclassified.
AdditiveGaussianNoiseAttack	Adds Gaussian noise to the input, gradually increasing the standard deviation until it is misclassified.
SaltAndPepperNoiseAttack	Increases the amount of salt and pepper noise until the input is misclassified.
BlendedUniformNoiseAttack	Blends the input with a uniform noise input until it is misclassified.
BoundaryAttackPlusPlus	A powerful adversarial attack that requires neither gradients nor probabilities.

Other attacks

BinarizationRefinementAttack	For models that preprocess their inputs by binarizing the inputs, this attack refines the adversarial until it is misclassified.
PrecomputedAdversarialsAttack	Attacks a model using precomputed adversarial candidates.

通过对以上各种攻击算法的尝试，发现：

1. 尽管Gradient-based类算法（FGSM, DeepFool...）具有极高的ssim(80~90)，但是对抗样本的迁移性极差，基于模型fashionMNIST生成的对抗样本就算能让fashionMNIST的正确预测类别概率降到0.1以下，却无法对模型cnn5的预测产生影响，依旧高达0.9.....
2. Score-based类算法由于基于预测的梯度，在我梯度差异悬殊的两个分类模型fashionMNIST和cnn5上同样表现差强人意，对抗样本迁移性依旧不太行.....
3. Other attacks是一些特殊情况下的攻击算法，同样不符合我的预期目标.....
4. 所幸Decision-based类算法对模型的依赖性并不强，对抗样本能较好地在模型间迁移。通过对比各个Decision-based算法生成的对抗样本的攻击成功率以及ssim指标，发现**AdditiveUniformNoiseAttack**及**AdditiveGaussianNoiseAttack**表现较好，最后通过反复对比，选择通过**AdditiveUniformNoiseAttack**实现我的adversarial_crafts，**AdditiveUniformNoiseAttack**通过向输入添加高斯噪声，逐渐增加标准偏差，直到输入被错误分类。在参考foolbox源码后，即将**AdditiveUniformNoiseAttack**代码实现整合为util包供项目使用。

4.项目本地运行情况

基于fashionMNIST模型利用AdditiveUniformNoiseAttack生成对抗样本，并在cnn5模型上进行迁移攻击：

生成100张对抗样本结果：

```
# self-test
import time
import skimage

(train_images, train_labels), (test_images, test_labels) = keras.datasets.fashion_mnist.load_data()
train_images = np.expand_dims(train_images, axis=3)
test_images = np.expand_dims(test_images, axis=3)

testNum = 100
randArray = np.random.choice(10000, testNum, replace=False)
testImages = []
for i in randArray:
    testImages.append(test_images[i])

startTime = time.time()
generate_images = allTest(np.array(testImages), (testNum, 28, 28, 1))
endTime = time.time() - startTime

cnn5 = keras.models.load_model('cnn5.h5')
trained_model = keras.models.load_model('fashionMNIST.h5')
attackRate = 0.0
attackSSIM = 0.0

14 15 warnings.warn('Not running the attack because the original input'
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
D:\Python3.7.0\lib\site-packages\skimage\measure\structural_similarity.py:155: UserWarning: Inputs have mismatched dtype. Setting data_range based on X.dtype.
warn('Inputs have mismatched dtype. Setting data_range based on ')
-----
attackRate: 0.74
attackSSIM: 0.5280822814776047
endTime: 152.500244140625 sec

Process finished with exit code 0
```

测试1000张对抗图片结果：

```
startTime = time.time()
generate_images = allTest(np.array(testImages), (testNum, 28, 28, 1))
endTime = time.time() - startTime

cnn5 = keras.models.load_model('cnn5.h5')
trained_model = keras.models.load_model('fashionMNIST.h5')
attackRate = 0.0
attackSSIM = 0.0

for i in range(testNum):
    origin_label = np.argmax(cnn5.predict(np.expand_dims(testImages[i] / 255.0, 0))[0])
    fake_label = np.argmax(cnn5.predict(np.expand_dims(generate_images[i] / 255.0, 0))[0])
    if origin_label != fake_label:
        attackRate = attackRate + 1
        attackSSIM = attackSSIM + skimage.measure.compare_ssim(testImages[i].reshape([28, 28]),
            generate_images[i].reshape([28, 28]))

attackSSIM = attackSSIM / attackRate
attackRate = attackRate / testNum

print("-----")
print("attackRate:", attackRate)
print("attackSSIM:", attackSSIM)
print("endTime:", endTime, "sec")

allTest() for i in range(shape[0])

18: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
19: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
D:\Python3.7.0\lib\site-packages\skimage\measure\structural_similarity.py:155: UserWarning: Inputs have mismatched dtype. Setting data_range based on X.dtype.
warn('Inputs have mismatched dtype. Setting data_range based on ')
-----
attackRate: 0.751
attackSSIM: 0.5306040675566926
endTime: 1463.4125199317932 sec

Process finished with exit code 0
```

ps: 对抗样本生成时间约为 1.5s / 张