

✓ 적대적 공격(Adversarial Attack)

✓ 적대적 공격

- 딥러닝 모델의 취약점을 이용하여 의도적으로 모델을 속이거나 오작동하게 만드는 기술
 - 주로 입력 데이터에 미세한 변화를 주어 모델의 출력을 조작하는 방식으로 이루어짐
 - 예1) 이미지 분류 모델에서 고양이 사진에 사람 눈으로는 거의 감지할 수 없는 노이즈를 추가하여, 모델이 이를 개로 잘못 분류하게 만드는 것이 대표적인 적대적 공격의 예시임
 - 예2) 자율주행 차량의 이미지 인식 시스템을 속여 교통 사고를 유발하거나, 얼굴 인식 기반의 보안 시스템을 우회하여 불법적인 접근을 시도하는 등의 위험이 있을 수 있음
 - 인공지능 모델에 대한 보안적 검토의 중요성 대두. AI 보안 측면의 연구 필요성도 높아지고 있는 추세임
-

✓ 적대적 공격의 유형

- 백박스 공격(White-box Attack)
 - 공격자가 모델의 구조, 파라미터, 학습 데이터 등 모든 정보를 알고 있는 상태에서 수행하는 공격
 - 이는 가장 강력한 형태의 공격이지만, 현실에서는 이런 정보를 모두 얻기 어려움

- 블랙박스 공격(Black-box Attack)

- 모델의 내부 구조나 파라미터를 모르는 상태에서 수행하는 공격
- 주로 모델의 입출력 관계만을 관찰하여 공격을 설계

- 표적 공격(Targeted Attack)

- 특정한 오분류 결과를 목표로 하는 공격
- 예를 들어, 고양이 이미지를 개로 분류하게 만드는 것이 목표일 수 있음

- 비표적 공격(Untargeted Attack)

- 단순히 모델의 예측을 틀리게 만드는 것이 목표인 공격
 - 어떤 오분류 결과가 나오든 상관없음
-

✓ 적대적 공격의 유형과 기법 - Fast Gradient Sign Method(FGSM)

- 가장 기본적이면서도 효과적인 적대적 공격 방법 중 하나

- 모델의 그래디언트를 이용하여 입력 데이터를 조작

- FGSM의 핵심 아이디어는 다음과 같음

- 모델의 손실 함수에 대한 입력의 그래디언트를 계산합니다.
 - 이 그래디언트의 방향으로 입력을 조금씩 변화시킵니다.
 - 이렇게 변화된 입력이 모델을 속일 수 있는 적대적 예제가 됩니다.
-

```
x_adv = x + ε * sign(∇x J(θ, x, y))
```

여기서,

x_{adv}: 적대적 예제

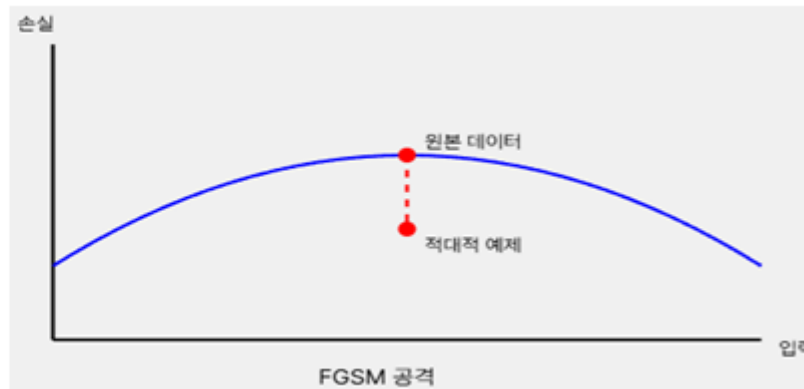
x: 원본 입력

ε: 섭동의 크기를 제어하는 파라미터

J: 손실 함수

θ: 모델 파라미터

y: 실제 레이블



```
1 #https://www.tensorflow.org/tutorials/generative/adversarial_fgsm?hl=ko
2 #사전 훈련된 MobileNetV2 모델과 ImageNet의 클래스(class) 이름들을 불러옵니다.
3 import tensorflow as tf
4 import matplotlib as mpl
5 import matplotlib.pyplot as plt
6
7 mpl.rcParams['figure.figsize'] = (8, 8)
8 mpl.rcParams['axes.grid'] = False
```

```
1 pretrained_model = tf.keras.applications.MobileNetV2(include_top=True,
2                                                         weights='imagenet')
3 pretrained_model.trainable = False
4
5 # ImageNet labels
6 decode_predictions = tf.keras.applications.mobilenet_v2.decode_predictions
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kern14536120/14536120 ————— 0s 0us/step

```
1 # Helper function to preprocess the image so that it can be inputted in MobileNetV2
2 def preprocess(image):
3     image = tf.cast(image, tf.float32)
```

```


4 image = tf.image.resize(image, (224, 224))
5 image = tf.keras.applications.mobilenet_v2.preprocess_input(image)
6 image = image[None, ...]
7 return image
8
9 # Helper function to extract labels from probability vector
10 def get_imagenet_label(probs):
11     return decode_predictions(probs, top=1)[0][0]

```

```

1 #Mirko CC-BY-SA 3.0의 래브라도 리트리버 샘플 이미지를 이용해 적대적 샘플을 생성합니다.
2 #첫 단계로, 원본 이미지를 전처리하여 MobileNetV2 모델에 입력으로 제공합니다.
3 image_path = tf.keras.utils.get_file('YellowLabradorLooking_new.jpg', 'https://storage.googleapis.com/download.tensorflow.org/example_images/
4 image_raw = tf.io.read_file(image_path)
5 image = tf.image.decode_image(image_raw)
6
7 image = preprocess(image)
8 image_probs = pretrained_model.predict(image)

```

 Downloading data from https://storage.googleapis.com/download.tensorflow.org/example_images/YellowLabradorLooking_new.jpg
 83281/83281 ————— 1s 10us/step
 1/1 ————— 6s 6s/step

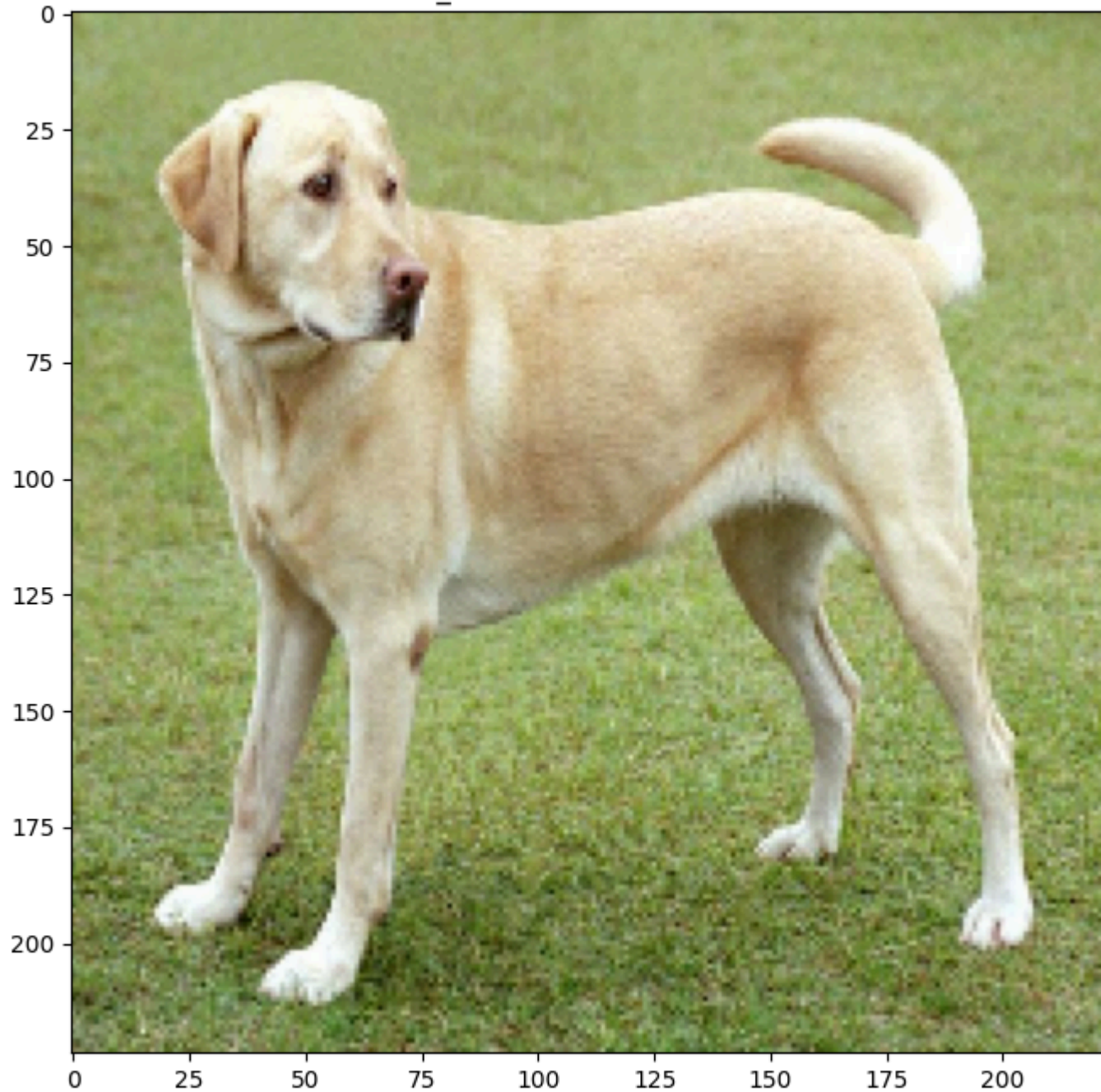
```

1 #이미지를 살펴봅시다.
2
3
4 plt.figure()
5 plt.imshow(image[0] * 0.5 + 0.5) # To change [-1, 1] to [0,1]
6 _, image_class, class_confidence = get_imagenet_label(image_probs)
7 plt.title('{} : {:.2f}% Confidence'.format(image_class, class_confidence*100))
8 plt.show()

```

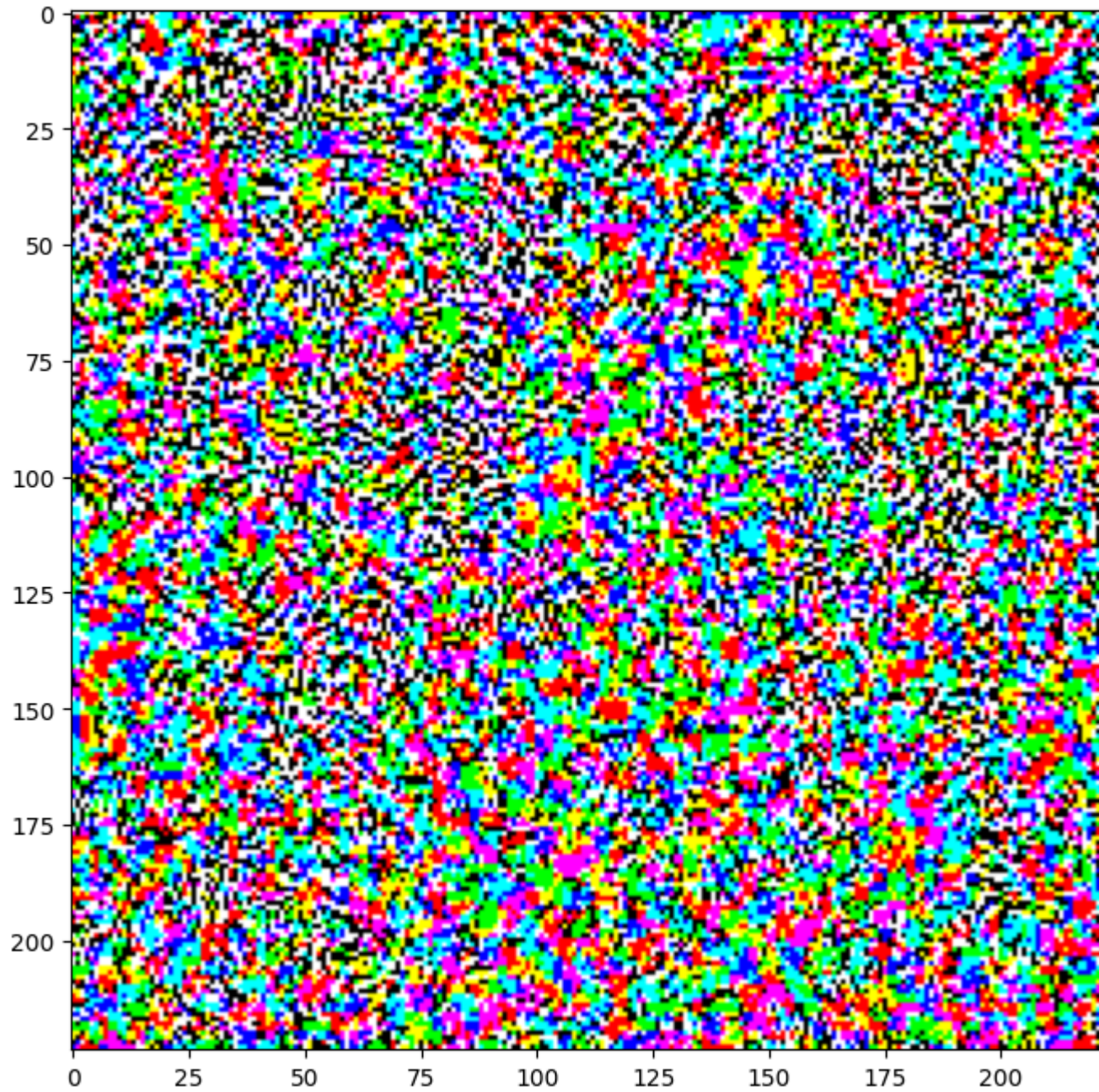
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 ————— 0s 5us/step

Labrador_retriever : 41.82% Confidence



```
1 #적대적 이미지 생성하기
2 #FGSM 실행하기
3 #첫번째 단계는 샘플 생성을 위해 원본 이미지에 가하게 될 왜곡을 생성하는 것입니다. 앞서 살펴보았듯이, 왜곡을 생성할 때에는 입력 이미지에 대한
4 loss_object = tf.keras.losses.CategoricalCrossentropy()
5
6 def create_adversarial_pattern(input_image, input_label):
7     with tf.GradientTape() as tape:
8         tape.watch(input_image)
9         prediction = pretrained_model(input_image)
10        loss = loss_object(input_label, prediction)
11
12    # Get the gradients of the loss w.r.t to the input image.
13    gradient = tape.gradient(loss, input_image)
14    # Get the sign of the gradients to create the perturbation
15    signed_grad = tf.sign(gradient)
16    return signed_grad


1 #생성한 왜곡을 시각화해 볼 수 있습니다.
2
3
4 # Get the input label of the image.
5 labrador_retriever_index = 208
6 label = tf.one_hot(labrador_retriever_index, image_probs.shape[-1])
7 label = tf.reshape(label, (1, image_probs.shape[-1]))
8
9 perturbations = create_adversarial_pattern(image, label)
10 plt.imshow(perturbations[0] * 0.5 + 0.5); # To change [-1, 1] to [0,1]
```

```

1 #왜곡 승수 엡실론(epsilon)을 바꿔가며 다양한 값들을 시도해봅시다.
2 #위의 간단한 실험을 통해 엡실론의 값이 커질수록 네트워크를 혼란시키는 것이 쉬워짐을 알 수 있습니다.
3 #하지만 이는 이미지의 왜곡이 점점 더 뚜렷해진다는 단점을 동반합니다.
4
5 def display_images(image, description):
6     _, label, confidence = get_imagenet_label(pretrained_model.predict(image))
7     plt.figure()
8     plt.imshow(image[0]*0.5+0.5)
9     plt.title('{ } Wn { } : {:.2f}% Confidence'.format(description,
10                                                         label, confidence*100))
11     plt.show()

1 epsilons = [0, 0.01, 0.1, 0.15]
2 descriptions = [('Epsilon = {:.3f}'.format(eps) if eps else 'Input')
3                 for eps in epsilons]
4
5 for i, eps in enumerate(epsilons):
6     adv_x = image + eps*perturbations
7     adv_x = tf.clip_by_value(adv_x, -1, 1)
8     display_images(adv_x, descriptions[i])

```



✓ <<<참조자료 사이트>>>

1. [데이터 유형: 숫자, 범주 및 순서 데이터의 이해](#)
2. [IQR\(사분위수 범위\)](#)
3. [통계분석-표준화\(Standardization\)란?](#)
4. [정규화 vs 표준화](#)
5. [데이터의 정규화\(normalization\) 또는 표준화\(standardization\)이 필요한 이유](#)

6. [교차 검증\(Cross Validation\)](#)
 7. [표본추출방법 - 계통 추출, 층화 추출](#)
 8. [인공지능보안과 적대적공격](#)
 9. [MLOps가 무엇인가?](#)
 10. [정규화, 표준화 - 수치형데이터 스케일링-MinMaxScaler StandardScaler](#)
 11. [적대적 공격이란?](#)
 12. [인공지능보안과 적대적공격](#)
 13. [Adversarial Attacks and Defenses in Machine Learning](#)
 14. [FGSM을 이용한 적대적 샘플 생성](#)
-

1. [딥러닝 적대적 공격 막는 방어 프레임 개발 / YTN 사이언스](#)
 2. [\(KAIST 연구\) 인공지능이 무력화 공격을 받으면? 창과 방패의 싸움! Feat. 적대적 패턴 공격과 방어 프레임](#)
-