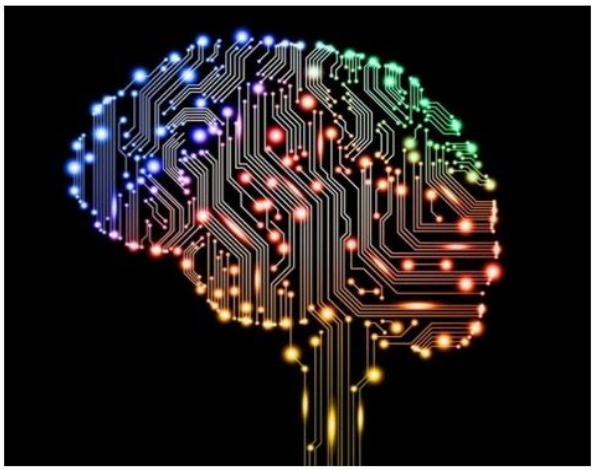
Deep learning



- **Deep Learning의 발전**

영상처리, 자연어처리, 강화학습 등 성능 향상

특히 **컴퓨터비전 분야에서의 성장이 두드러짐**

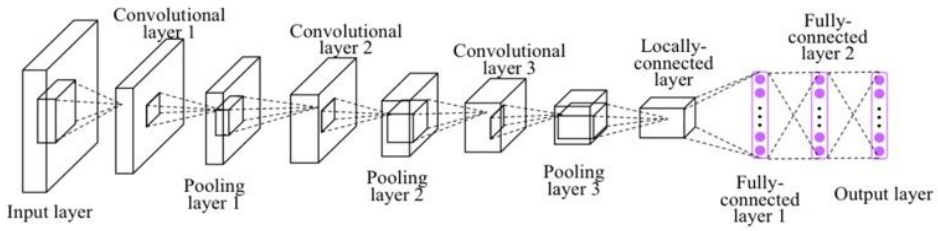
CNN

- ConvNet은 학습 가능한 **가중치** 와 **바이어스**로 구성되어 있다.

- **각** 뉴런은 입력을 받아 내적 연산을 한 뒤 선택에 따라 비선형 연산

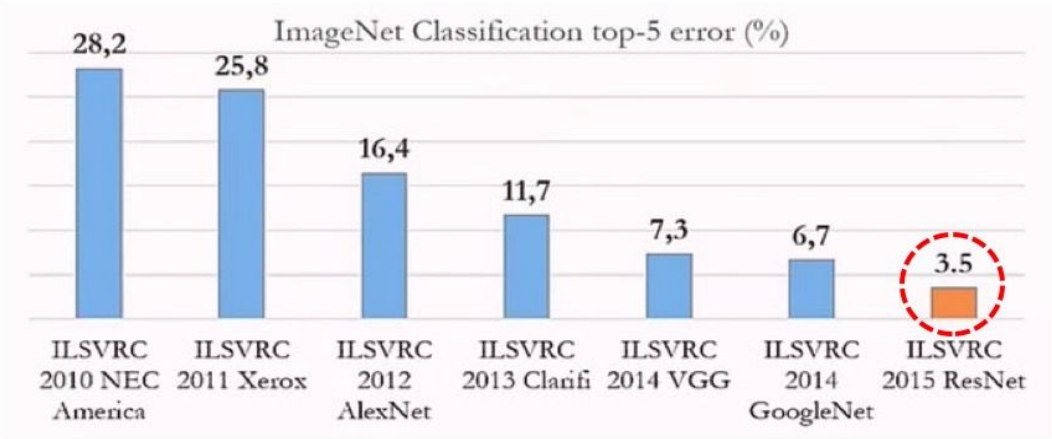
- **전**체 네트워크는 일반 신경망과 마찬가지로 미분 가능한 하나의 스코어 함수를 갖게 된다.

- ConvNet은 마지막 레이어에 손실 함수를 가지며, 우리가 일반 신경망을 학습시킬 때 사용하던 각종 기법들을 동일하게 적용할 수 있다.



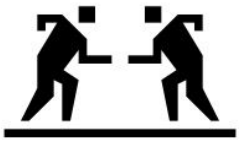
CNN

Andrej Karpathy에 의하면 사람이 해당 이미지셋을 분류했을 때 오차율이 약 5%인데,
2015년 ResNet은 분류율이 3.5%로 이미 인간의 눈 보다 뛰어난 성능을 보이고 있다.



Deep learning

VS



그렇다면 인간과 컴퓨터는 시각적인 부분에서 어떤 차이가 있을까?

관련 이론

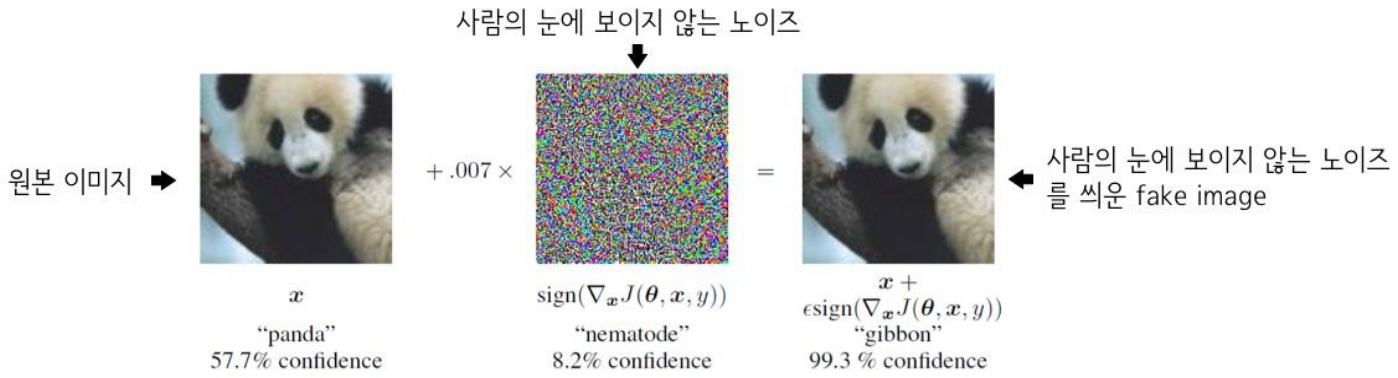
사람 눈에는 똑같은 코끼리 사진으로 보이지만, 컴퓨터는 각각 코끼리와 코알라, 다른 동물 사진으로 인식한다.

elephant **elephant**


elephant **koala**


관련 이론

원본 이미지에 사람 눈으로 볼때는 식별이 불가능한 노이즈를 넣어 fake image를 만들어 컴퓨터(NN 모델)를 속이는 연구가 진행되었다.



Explaining and Harnessing Adversarial Examples

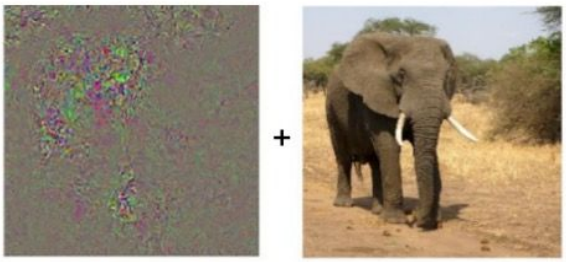
Goodfellow, Ian J., Jonathon Shelns, and Christian Szegedy(ICLRK 2015)

관련 이론

원본 이미지는 정상적으로 코끼리로 분류하지만, 노이즈를 첨가하여, 컴퓨터는 코알라로 인식을 하게끔 이미지에 변형을 가한 것이다.

elephant로 분류

사람의 눈에 보이지 않는 노이즈 원본 이미지



koala로 분류


도입
구현
결론
구현

Data

QR 코드 이미지와 CIFAR-10 이미지를 이용하여, CNN 모델을 속여보기로 하였다.

<일반 사진 - CIFAR-10>
1100장

<QR code>
1100장

Model

사용된 CNN 모델은 아래와 같다.

[32x32x3] INPUT
[32x32x32] CONV1: 32 3x3 filters at stride 2 with pad 1
[16x16x32] MAX POOL1: 2x2 filters at stride 2
[16x16x64] CONV2: 64 3x3 filters at stride 2 with pad1
[8x8x64] MAX POOL2: 2x2 filters at stride 2
[1024] FC3: 1024 neurons
[2] FC4: 2neurons(class scores)

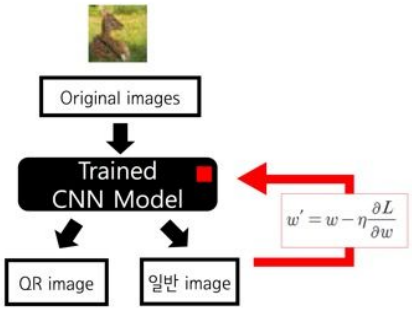
HYPER PARAMETER:

RELU
Batch size: 64
Dropout: 0.5
Adam Optimizer
Learning rate: 1e-4
Training model Epoch: 1000
Fooling model Epoch: 10000(MAX)

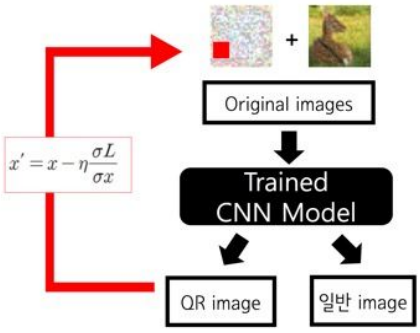
모델 overview

전체 과정

1. 속일 대상이 될 CNN 모델 생성 후 학습
2. 학습된 CNN 모델을 속이는 fake 이미지 생성



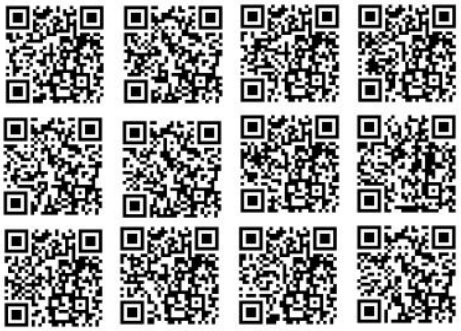
< CNN 모델 학습 과정 >



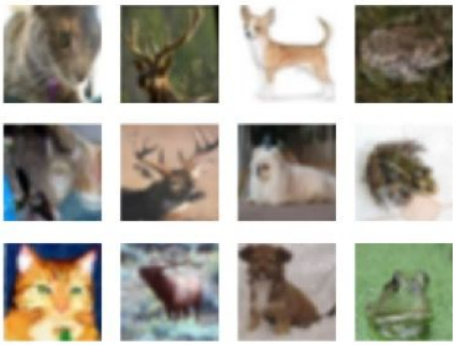
< CNN 모델을 속이는 이미지 생성 과정 >

모델 학습

사용한 데이터 셋 : CIFAR - 10 1100장, QR코드 1100장, 총 2200장
(size : 32 * 32, train set : 각 900장, validation set : 각 100장, test set : 각 100장)



< QR코드 >



< CIFAR - 10 >

데이터 수집

QR코드 이미지 세트 수집 과정

1. QR코드를 만들 url을 우선적으로 수집(이미지 내 file_name)
2. url을 QR코드로 변환해주는 google API를 QR코드 이미지로 변환하는 과정을 R로 구현

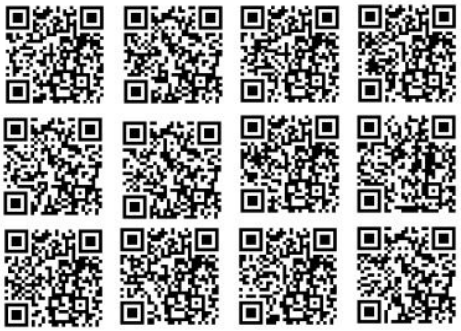
```
install.packages("RCurl")
library(RCurl)
setwd(directory)
csv <- read.csv(file_name, stringsAsFactors = FALSE, header = FALSE)
count <- 0
for(i in csv$V1){
  count <- count + 1
  qr <- paste0("http://chart.apis.google.com/chart?cht=qr&chs=144x144&choe=UTF-8&chl=", i)
  re <- paste0("starLink144/QR_star_", count, ".png")
  download.file(qr, re)
}

getURLContent(qr, re)
```

< QR코드 >

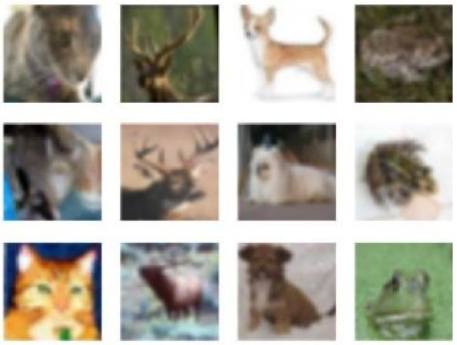
데이터 수집

이미지 세트 수집 과정

1. CIFAR-10 데이터 세트 다운로드
   (https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz)
2. CIFAR-10 데이터 세트 중 1100장을 임의 추출하여 QR 코드 1100장과 합침
3. CIFAR-10 이미지에 Label = 0, QR코드 이미지에 Label = 1을 매김



< QR코드, Label = 1 >



< CIFAR – 10, Label = 0 >

모델 학습

속일 모델 구현

1. tensorflow를 이용하여, CNN 모델 및 학습 과정 구현
   (tensorflow 1.2.0)

```
def weight_variable(shape):
 initial = tf.truncated_normal(shape, stddev=0.1)
 return tf.Variable(initial)

def bias_variable(shape):
  initial = tf.constant(0.1, shape=shape)
  return tf.Variable(initial)

def conv2d(x, W):
  return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')

def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')
W_conv1 = weight_variable([3, 3, 3, 32])
b_conv1 = bias_variable([32])

x_image = tf.reshape(x, [-1,32,32,3])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([3, 3, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)

W_fc1 = weight_variable([8 * 8 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 8*8*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 2])
b_fc2 = bias_variable([2])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

모델 학습 결과

결과

1. 최종 정확도 100%
   → QR코드와 CIFAR-10 이미지는 각 픽셀 값이 극단적으로 차이가 나기 때문에 쉽게 100%를 달성할 수 있음

```
Training Neural Net
step 0, training accuracy 0.56
step 100, training accuracy 1
step 200, training accuracy 1
step 300, training accuracy 1
step 400, training accuracy 1
test accuracy 1
```

< 학습 결과 >

Fake 이미지 생성

이미지 생성

1. 모든 픽셀이 (0, 0, 0)의 값을 가지는 초기 노이즈 이미지로 학습을 시작
2. 원본 이미지와 노이즈 이미지 합하여, 학습된 CNN 모델을 통하여 분류
3. 위의 분류 결과를 기반으로한 gradient를 CNN 모델에 학습시키는 것이 아닌, 노이즈 이미지에 학습을 시킴
   (loss 측정에는 sotfmax_cross_entropy_with_logits 사용, 학습에는 AdamOptimizer(lr = 1e-4) 사용)

```
def conv2d(x_images, W):
  return tf.nn.conv2d(x_images, W, strides=[1, 1, 1, 1], padding='SAME')
def max_pool_2x2_n32(x_images):
  return tf.nn.max_pool(x_images, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

W_conv1 = tf.constant(weights[0])
b_conv1 = tf.constant(weights[1])
x_image = tf.reshape(x_images, [-1,32,32,3])
h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2_n32(h_conv1)

W_conv2 = tf.constant(weights[2])
b_conv2 = tf.constant(weights[3])
h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2_n32(h_conv2)

W_fc1 = tf.constant(weights[4])
b_fc1 = tf.constant(weights[5])

h_pool2_flat = tf.reshape(h_pool2, [-1, 8*8*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = tf.constant(weights[6])
b_fc2 = tf.constant(weights[7])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2

cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_, logits=y_conv))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy + 10 * np.sum(np.multiply(x_image, x_image)))
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

<이미지 생성시 사용된 코드>

<original image> <noise image> <fake image>

하이퍼 파라미터 튜닝

튜닝의 필요성

- fake 이미지를 생성하면, 이미지가 깨져보이는 결과가 있었음(아래 그림 참조)
- 좀 더 나은 이미지를 생성하기 위하여 하이퍼 파라미터를 조정하면서 좀 더 나은 결과값을 찾기로 함

<original image>

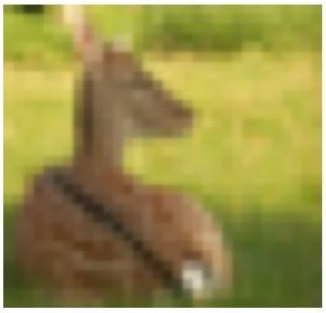
<fake image>

최종 결과

최종 결과

- 사람의 눈으로 보았을 때는 원본 이미지와 유사하게 보이나, CNN 모델은 QR코드로 분류하는 이미지 생성

<Real QR image>



<일반 image>

<Fake QR image>

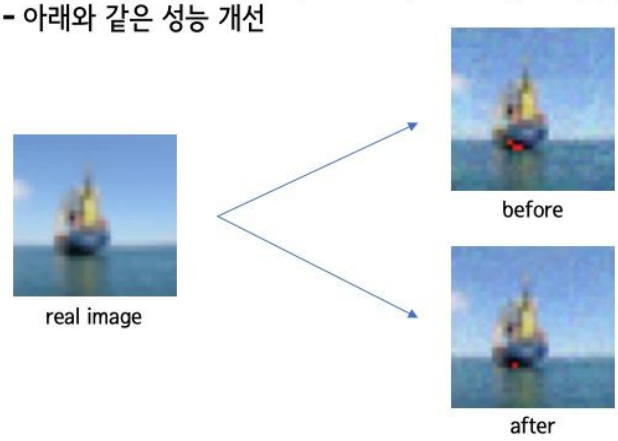

도입
구현
결론
결론

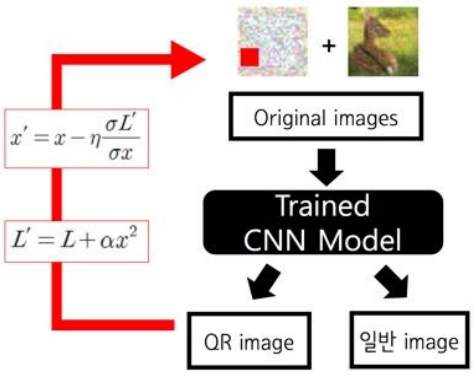
하이퍼 파라미터 튜닝

하이퍼 파라미터 튜닝 과정

- 이미지 생성 과정에 사용된 loss에 원본 이미지와의 차이를 추가하기로 함

  loss = cross entropy → loss = cross entropy * a + noise ^ 2 * b, a와 b는 기존 loss와 추가된 loss의 비율로, 이 역시 조정가능
- 학습에 사용되는 AdamOptimizer의 learning rate를 수정
- 아래와 같은 성능 개선

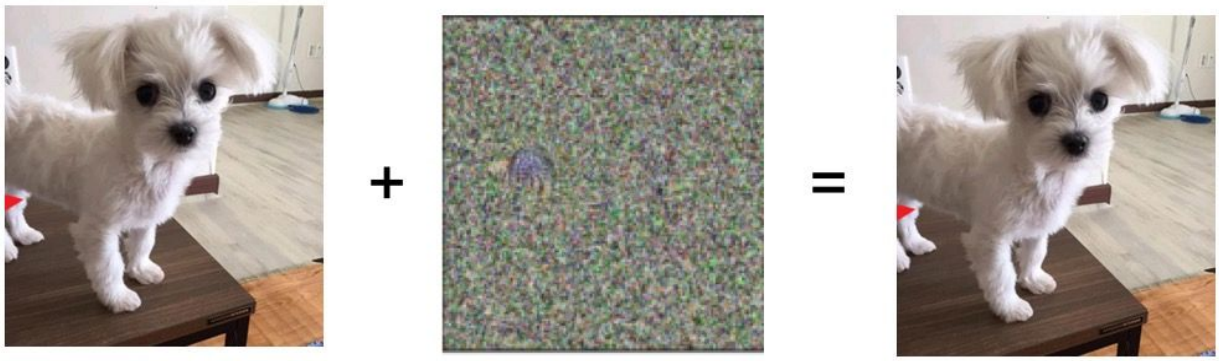




활용 방안

이러한 특성을 어떻게 이용할 수 있을까요?
이미지와 영상의 저작권과 관련하여 생각해 봤습니다.

활용 방안



업로드하는 원본 사진에 티가 안나는 노이즈를 더합니다.
이 사진을 누군가가 사용하려 할 때 CNN으로 만들어진 필터를 거칩니다.
필터가 이 사진을 원본이 아닌 사진으로 분류하여 사용을 막습니다.

활용 방안

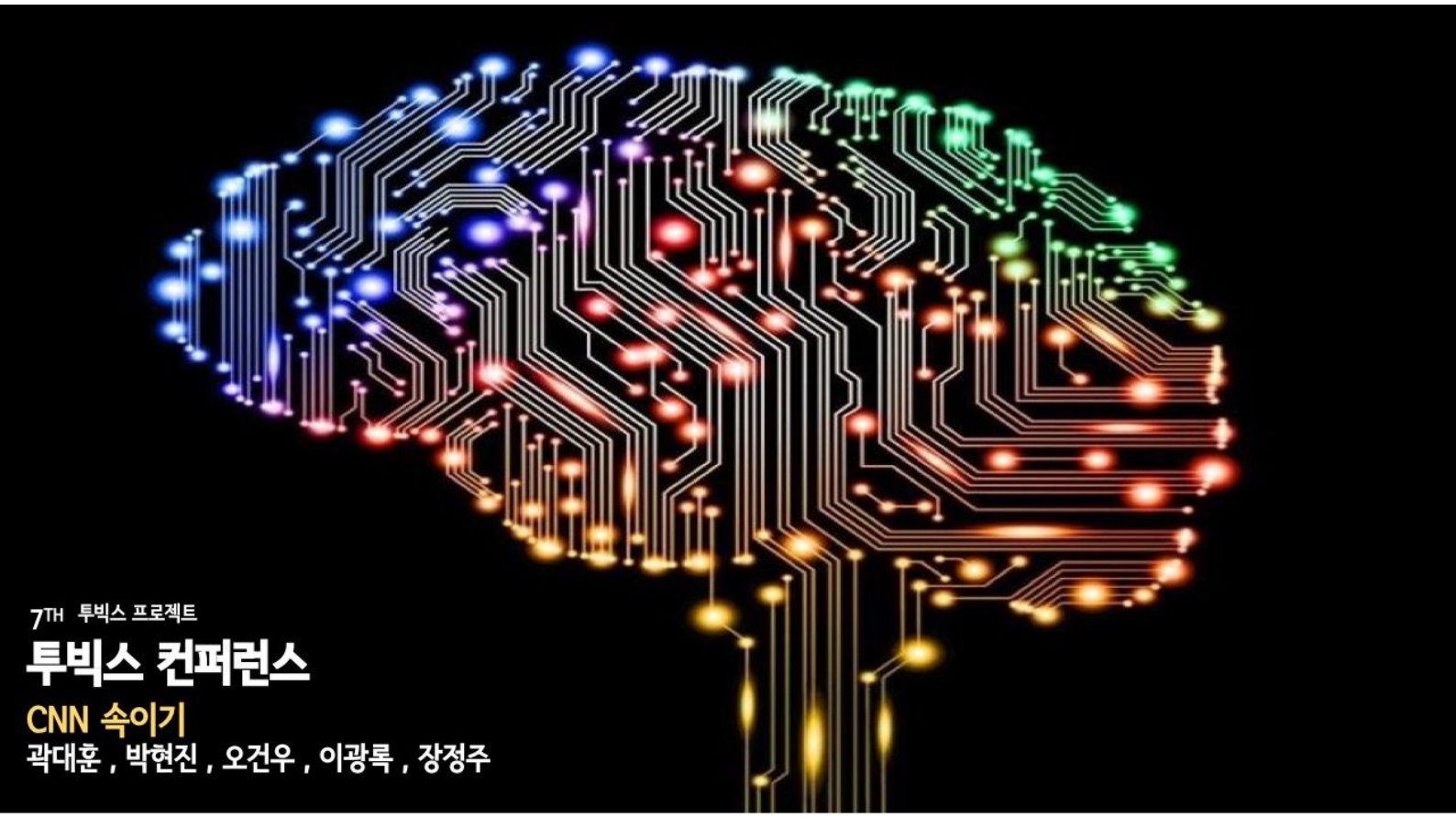
이와 마찬가지로,
영상에도 노이즈를 추가하여 저작권 보호에 도움이 될 것이라고 생각합니다.

활용 방안

하지만 영상 원본을 자르거나 흐리는 식으로 변형 시킬 경우,
불법으로 잘 분류가 되는지 더 실험이 필요합니다.

reference

- I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, Dec. 2014.
- C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.
- A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 427–436, 2015.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: A simple and accurate method to fool deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2016b.
- Metzen, J.H., Genewein, T., Fischer, V., and Bischoff, B. On detecting adversarial perturbations. International Conference on Learning Representations, 2017.
- cifar10 https://www.cs.toronto.edu/~kriz/cifar.html/
- http://personal.ie.cuhk.edu.hk/~ccloy/project_target_code/index.html
- https://twitter.com/randomknow_bot/status/701669740847083521 =http://mainia.tistory.com/5091
- https://www.youtube.com/watch?v=pIMe9qeMp2c
- https://billmuehlenberg.com/2017/06/23/bible-copyright-law



7TH 투빅스 프로젝트

투빅스 컨퍼런스

CNN 속이기

곽대훈 , 박현진 , 오건우 , 이광록 , 장정주

목차


도입
구현
결론
도입