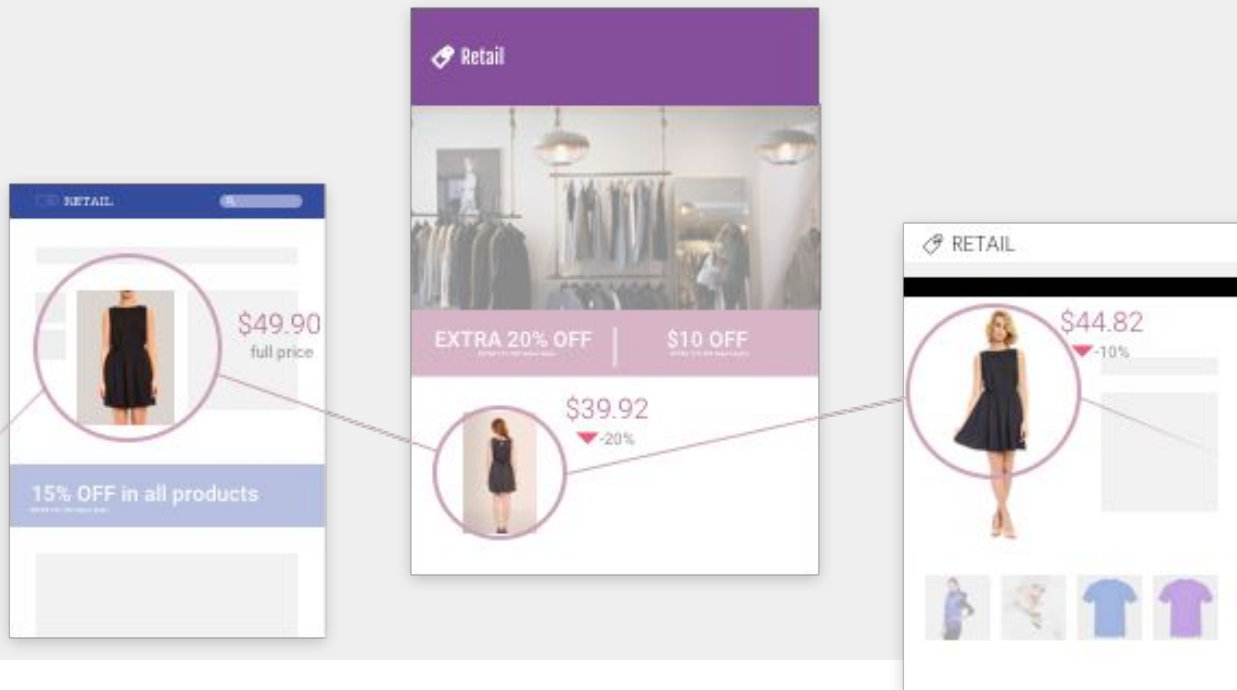


The bad guys in AI

atacando sistemas de machine learning

Javier Ordóñez & Alicia Pérez

Data Scientists @Style_Sage



¿Porqué...?

ABC REPORTAJES

España ▼ Inter ORBYT Tienda SuVivienda Empleo Coches Motor Tendencias Náutica Viajes Yodona Metrópoli Gentes de Verano! Hemeroteca

ABC MOTOR

Publicidad

Los c

• Tres ve

España

Natura Nano

Compartir

Recomm

Twitter

Tuenti

Herramientas

Enviar a un

Valorar

Imprimir

En tu móvil

Rectificar

Video: Vea Rodrigo Mu

EL PAÍS

OPINIÓN

EDITORIALES TRIBUNAS COLUMNAS VIÑETAS VÍDEOS CÓMO COLABORAR CARTAS A LA DIRECTORA NUESTRAS FIRMAS QUIENES SOMOS

TRIBUNA >

'Fake news' y credulidad

Cada vez con más frecuencia pensamos con una trama de datos e ideas facilitados por las redes. Creemos cualquier cosa que se presente con cierta contundencia; somos la población más informada de la historia, pero también la más vulnerable

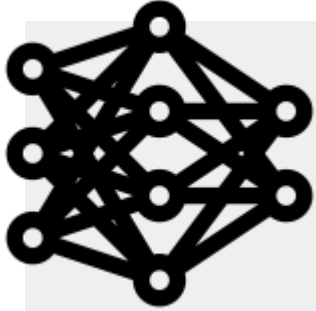
JORDI SOLER

29 ABR 2018 - 00:00 CEST



Marcel Duchamp leyó dos libros de filosofía en su vida, uno de ellos con verdadera devoción. Leyó *El único y su propiedad*, de Max Stirner, y se apasionó con un pequeño volumen, que releyó varias veces cuando trabajaba en la biblioteca de Sainte-Geneviève, en el que se contaba la vida y las ideas de Pirrón de Elis.

El escepticismo radical de Pirrón sirvió



Redes neuronales

*Qué es una red neural
profunda y cómo
aprende a clasificar
datos*

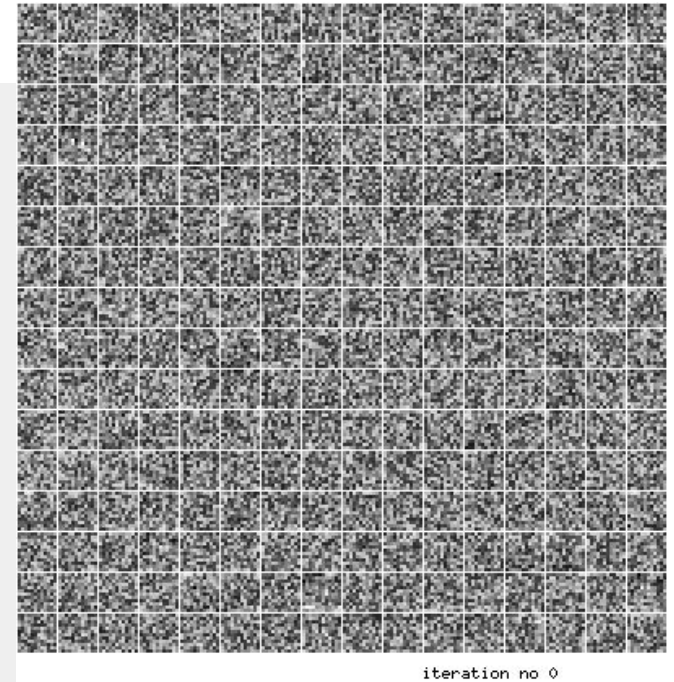
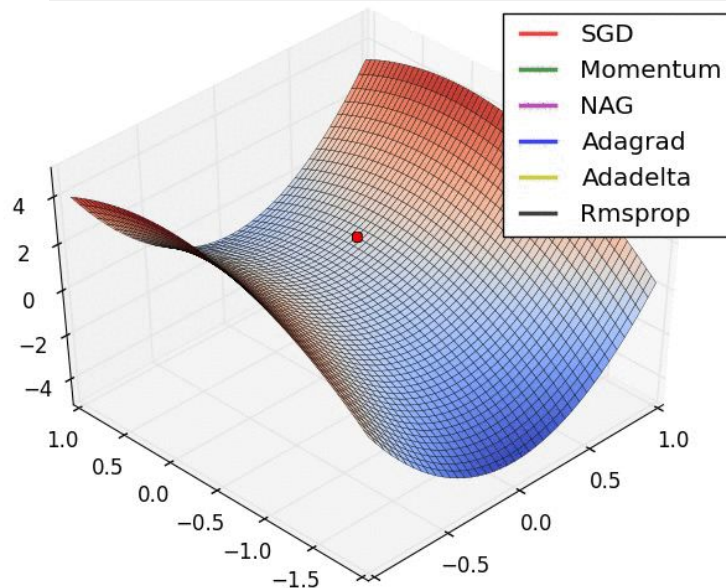
Red neural profunda

La imagen de entrada se transforma aplicando patrones organizados en una estructura jerárquica, extraídos mediante un método de optimización y una señal supervisora, para dar lugar a un vector de probabilidades.



Aprendizaje de características

- Inicialización aleatoria de los parámetros.
- Convergencia gracias a métodos de optimización.



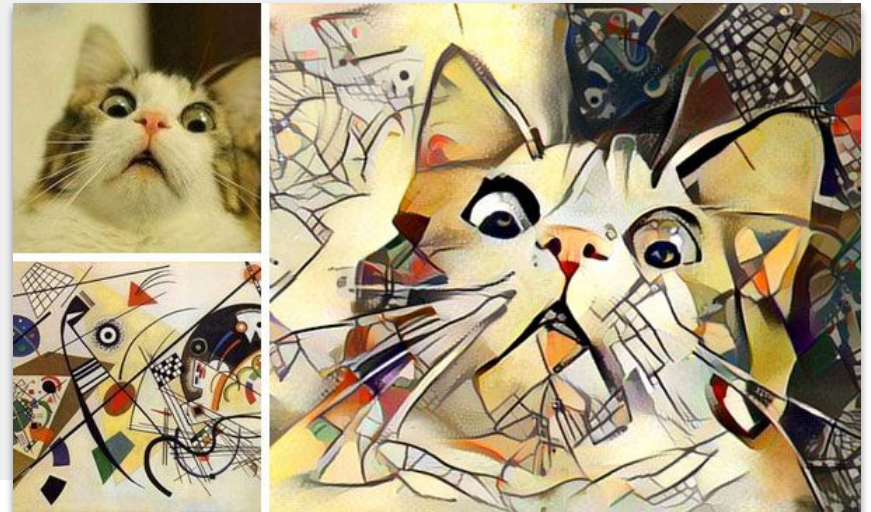
- Optimización basada en métodos de descenso del gradiente.
- Los parámetros se van moviendo hacia un punto que minimice el error del sistema.

Modelos discriminativos

- Dado un vector de características, predice una etiqueta.
- Probabilidad condicional.
- Modelan la dependencia entre una etiqueta (salida) y las características (datos de entrada)

Modelos generativos

- Crea ejemplos en vez de evaluarlos.
- Probabilidad conjunta.
- Modela cómo se distribuyen las características (datos de entrada) de cada tipo de etiqueta.



Evaluación de modelos discriminativos



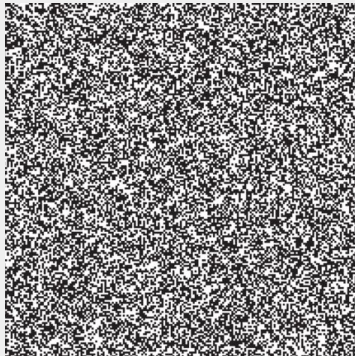
$p(\text{Mopa}) = 7\%$
 $p(\text{Perro}) = 93\%$

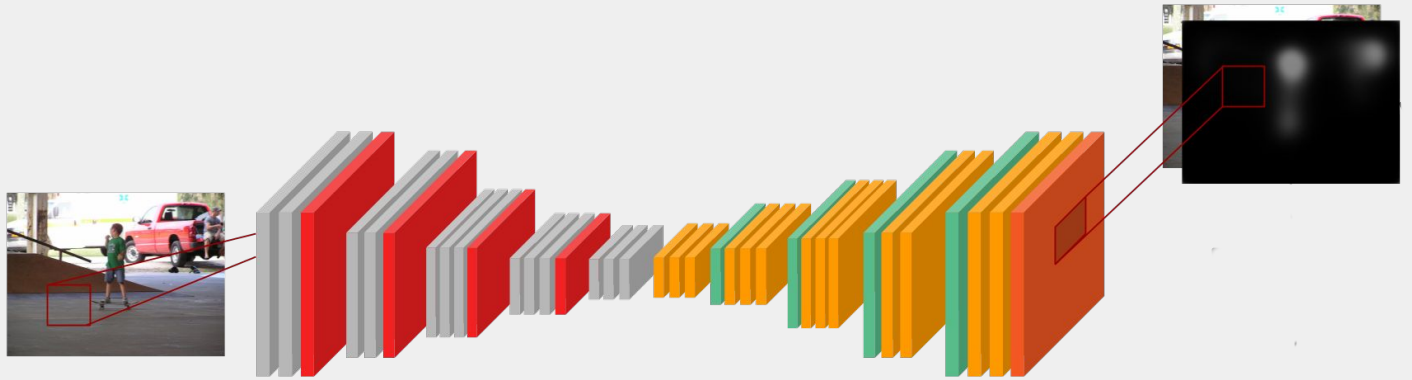


$p(\text{Mopa}) = 87\%$
 $p(\text{Perro}) = 13\%$



Evaluación de modelos generativos

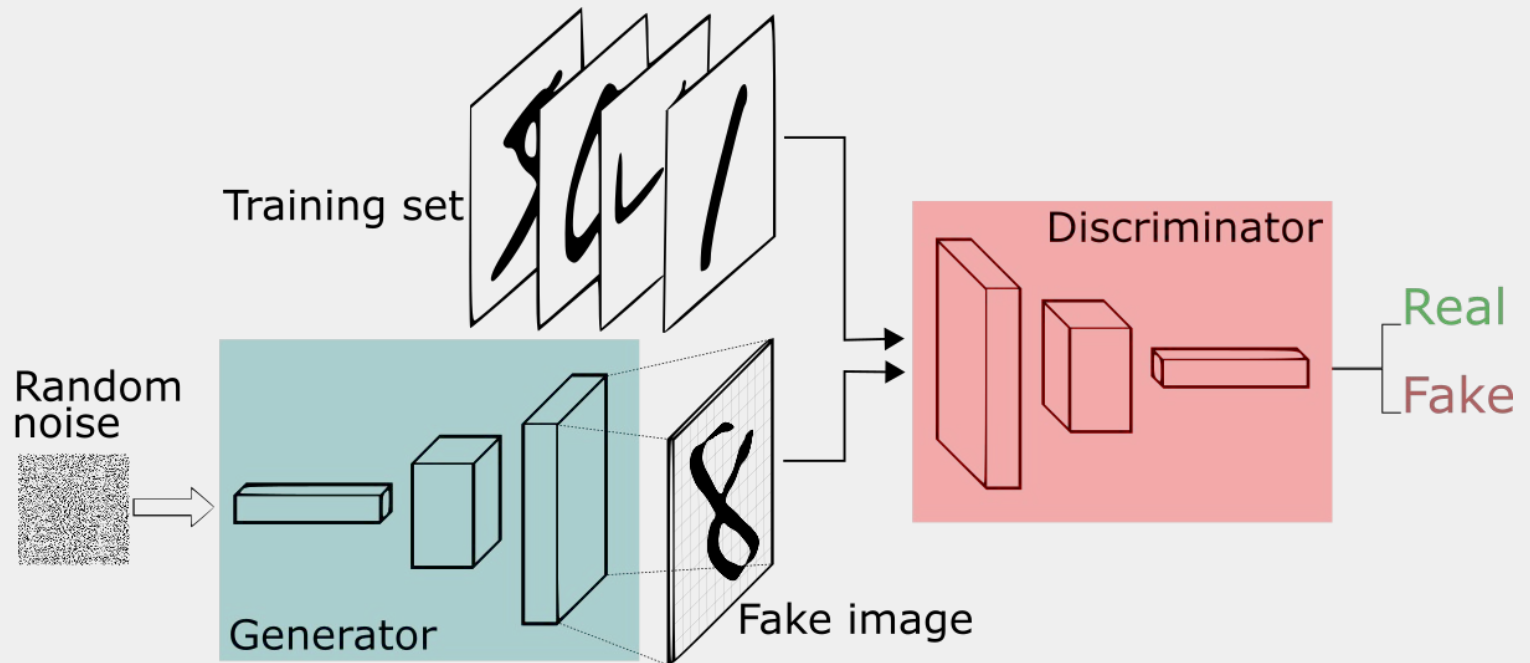




Arquitecturas adversarias

Redes generativas adversarias

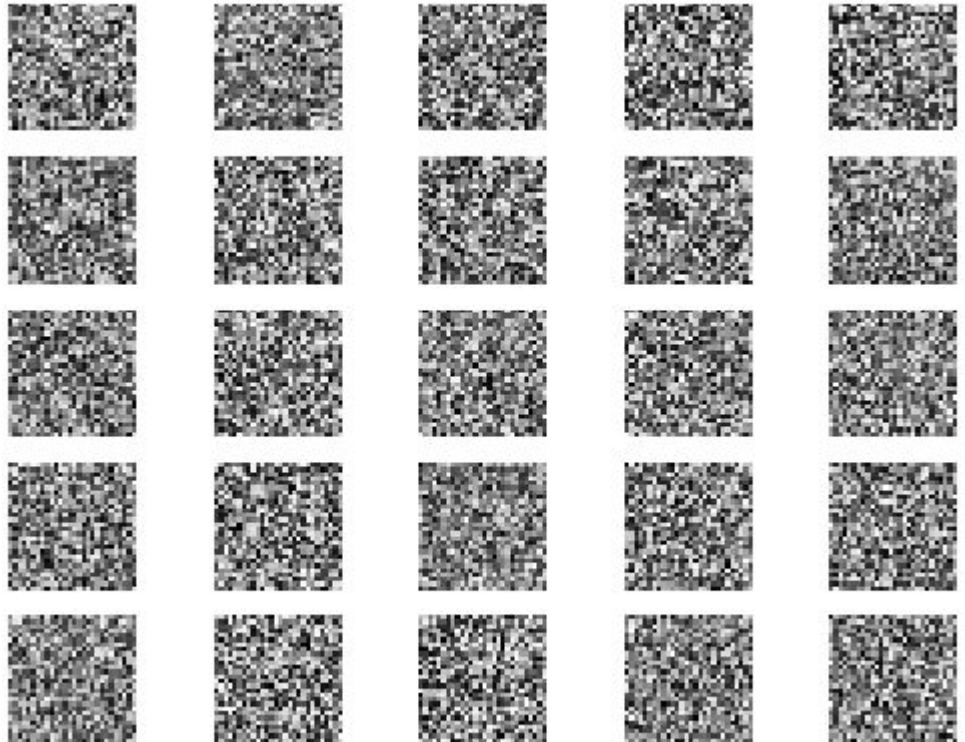
- La salida del bloque generativo es una muestra o ejemplo.
- La salida del bloque discriminativo es la clasificación de la muestra. **La señal supervisora de la que se aprende.**



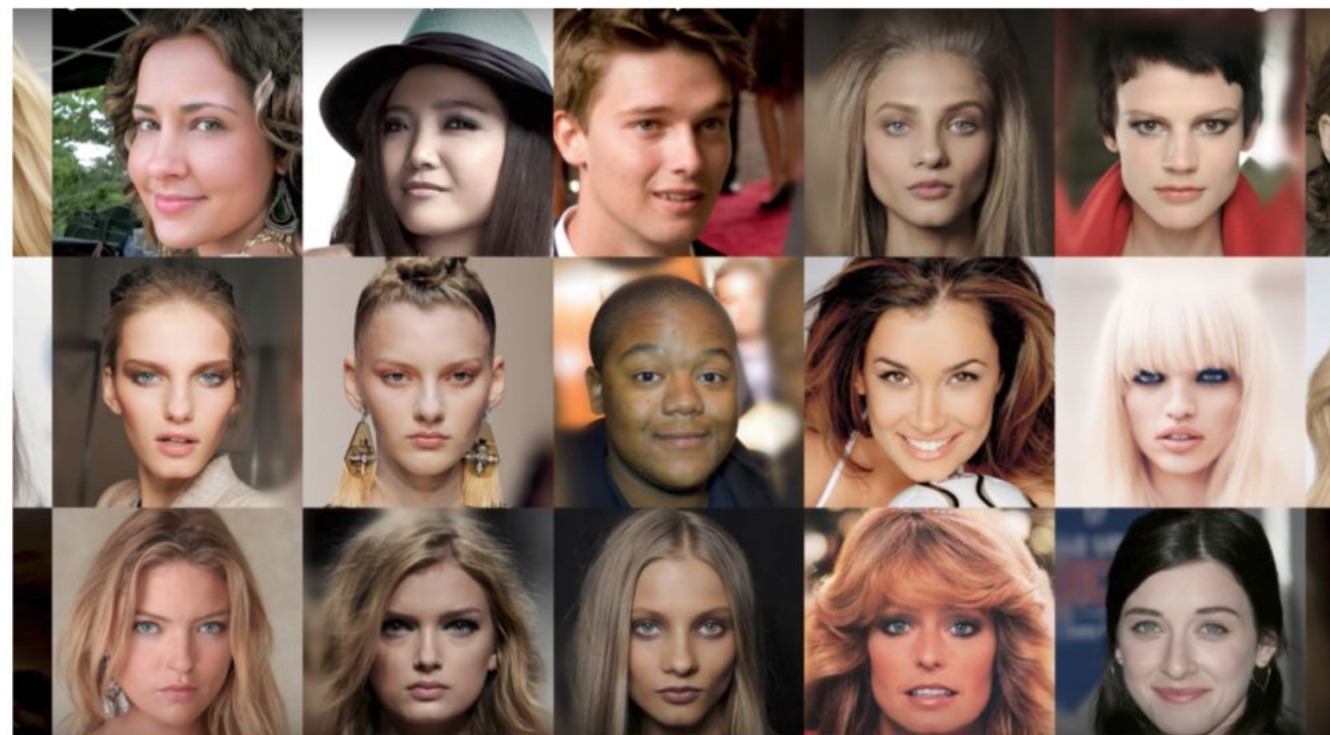
Generación de datos sintéticos

El modelo generativo comienza aleatoriamente y converge hacia representaciones que se asemejan a los datos usados para entrenar al modelo discriminativo.

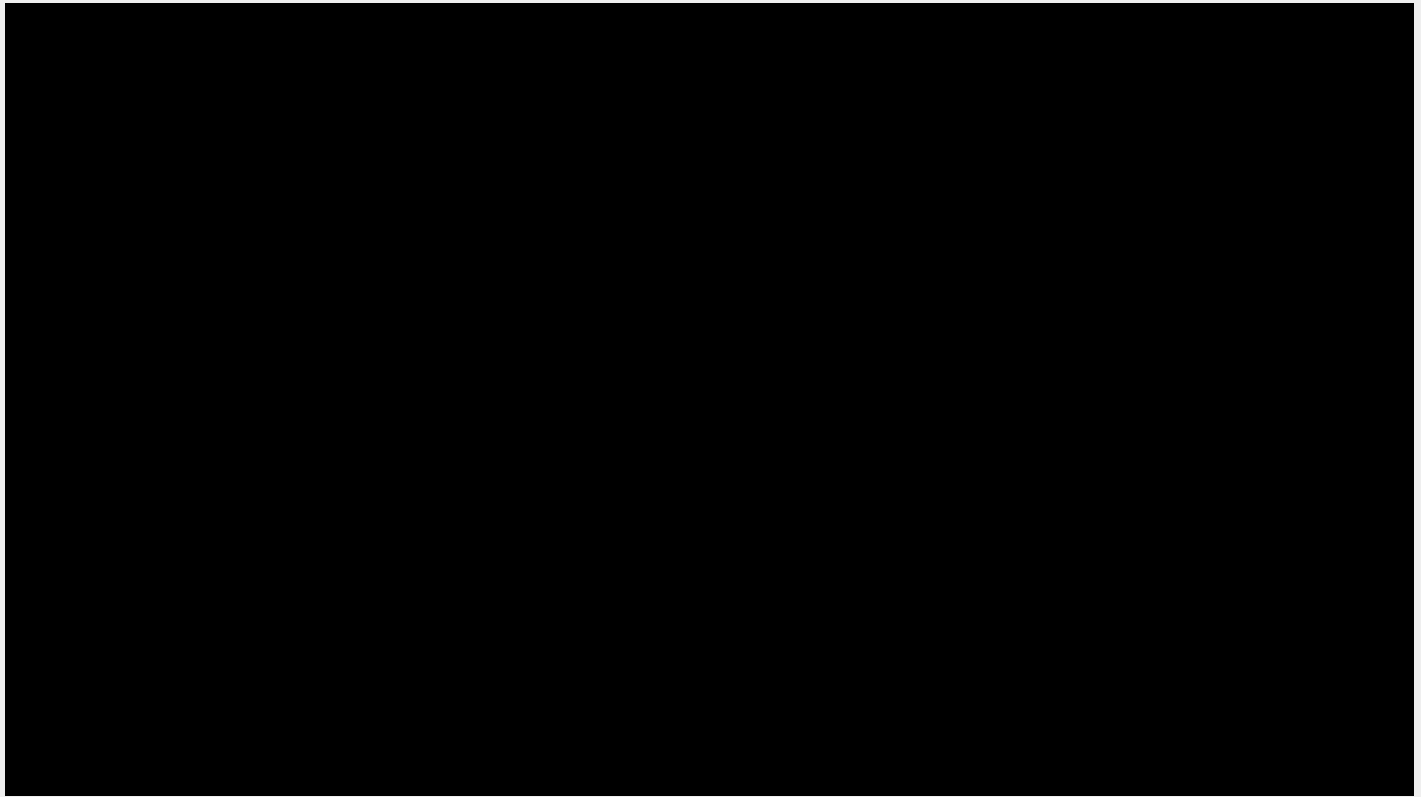
No aprende a **clasificar**
Aprende a **engañar**



Ejemplo de generación de imágenes



Ejemplo de generación de imágenes





a car

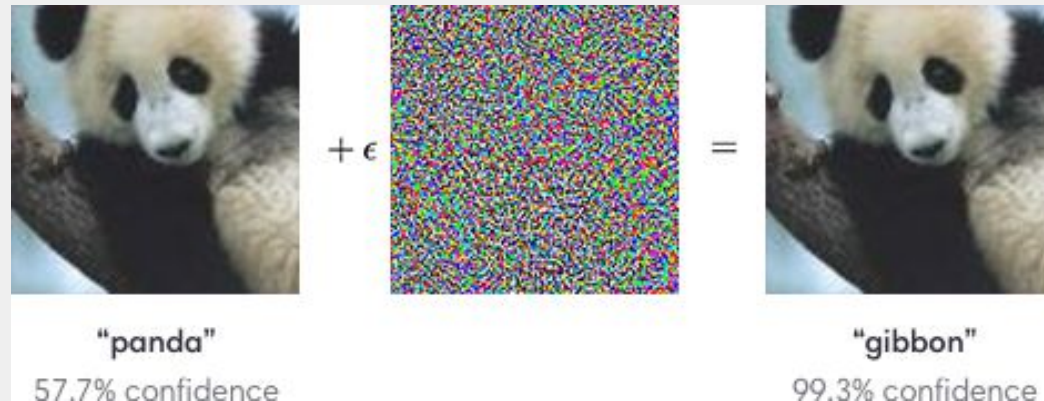


a cat

Concepto de ataque

Ejemplo antagónico

Muestra perteneciente a la distribución de los datos alterada mínimamente para que sea clasificada incorrectamente.



Ataque antagónico

Entrenamiento de modelo generativo adversario que es capaz de crear muestras sintéticas para engañar a un modelo discriminativo objetivo.

Tipos de ataque

No orientado

El más habitual

Sólo se busca que el clasificador ofrezca un resultado incorrecto

Orientado

Más difícil

Busca obtener una clase específica.

Caja blanca

Tenemos acceso al modelo:

- ☐ Arquitectura
- ☐ Parametrización
- ☐ Datos

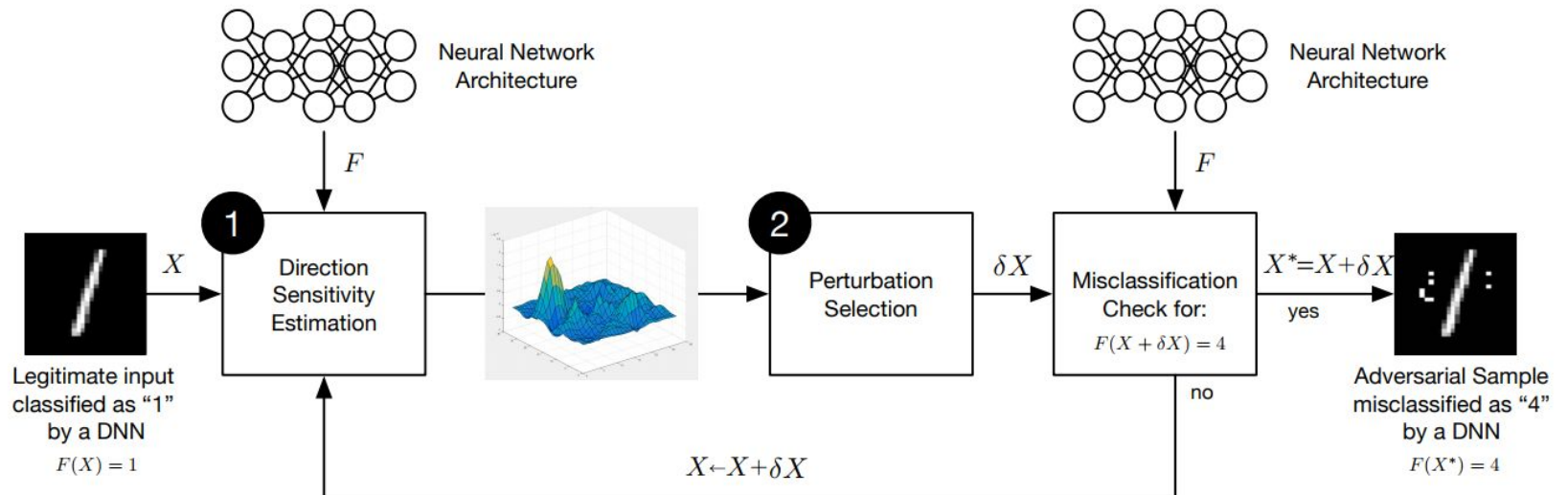
Caja negra

Sólo disponemos de la salida del modelo a atacar

Ataque no orientado de caja negra



Ataque no orientado de caja negra





Demo time

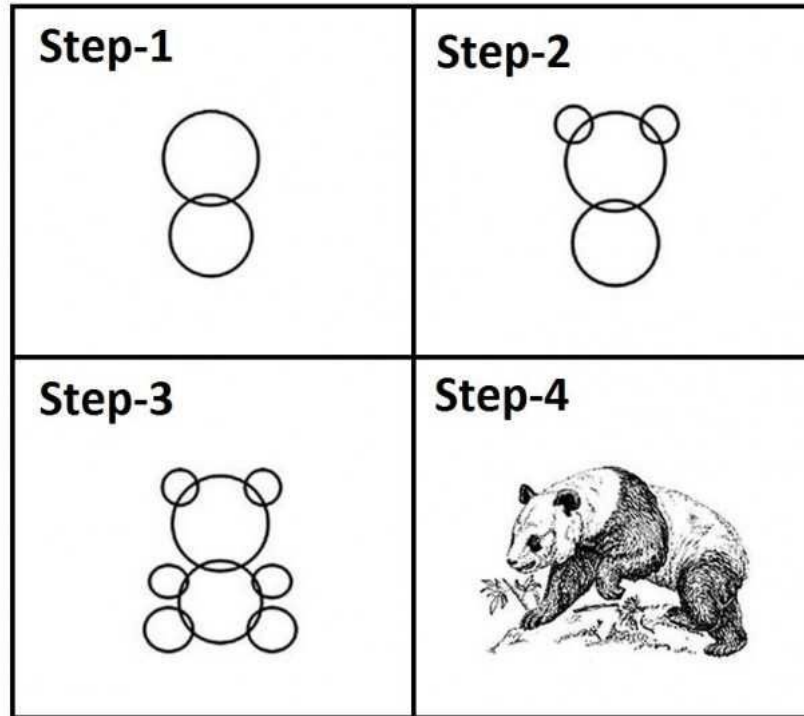
*Cómo atacar un sistema
de clasificación de
imágenes de comida*

Antes de empezar...

How to draw a panda

Ataque

1. E
2. E
3. A



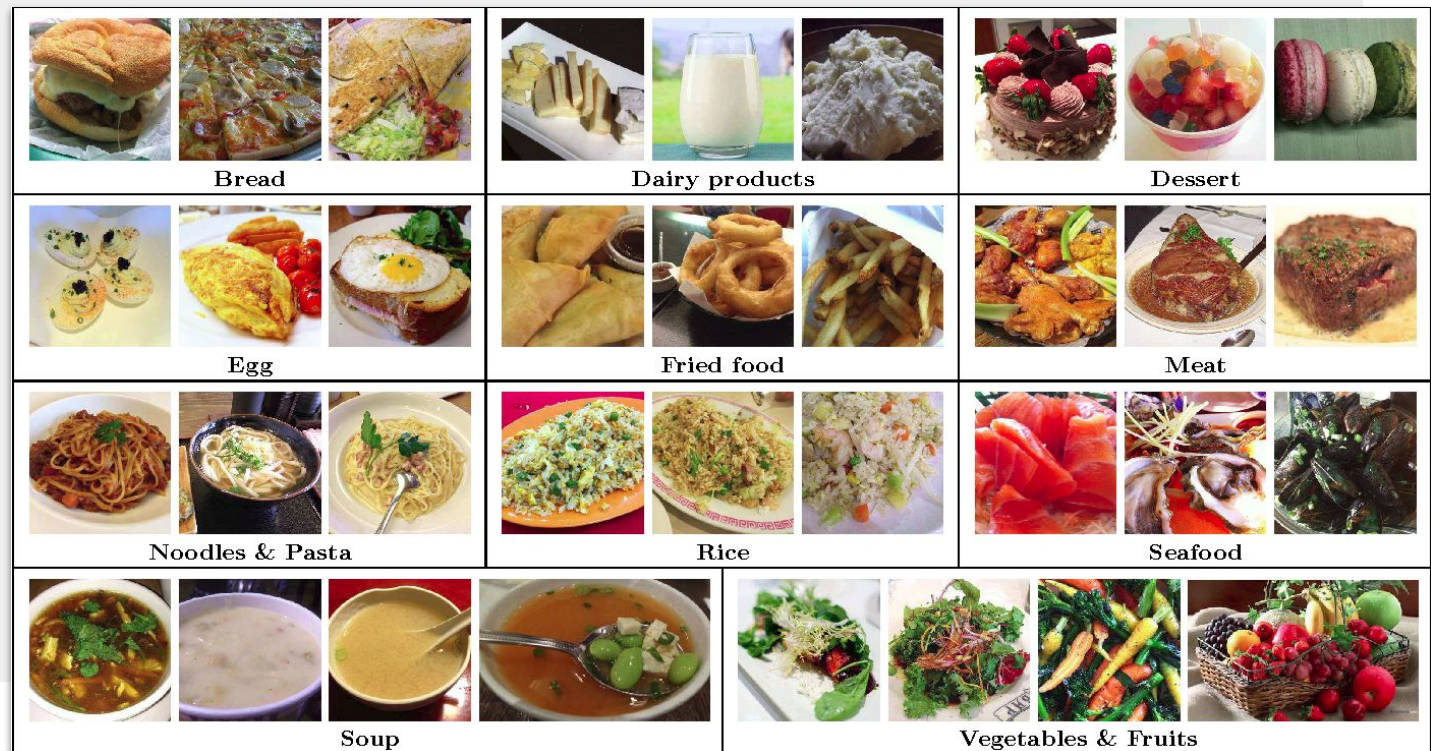
clasificar

!!No es trivial!!

Dataset

Food-11- Etiquetas con 11 tipos de comida. Imágenes: 16643

Food-5K - Etiquetas binarias. Imágenes: 2500 comida, 2500 no comida



Dataset

Food-12

- Incluye 12 etiquetas
- Tamaño: 19125 imágenes
- Dividido en:
 - Entrenamiento
 - Validación
 - Test
- Tamaño de entrenamiento aumentado

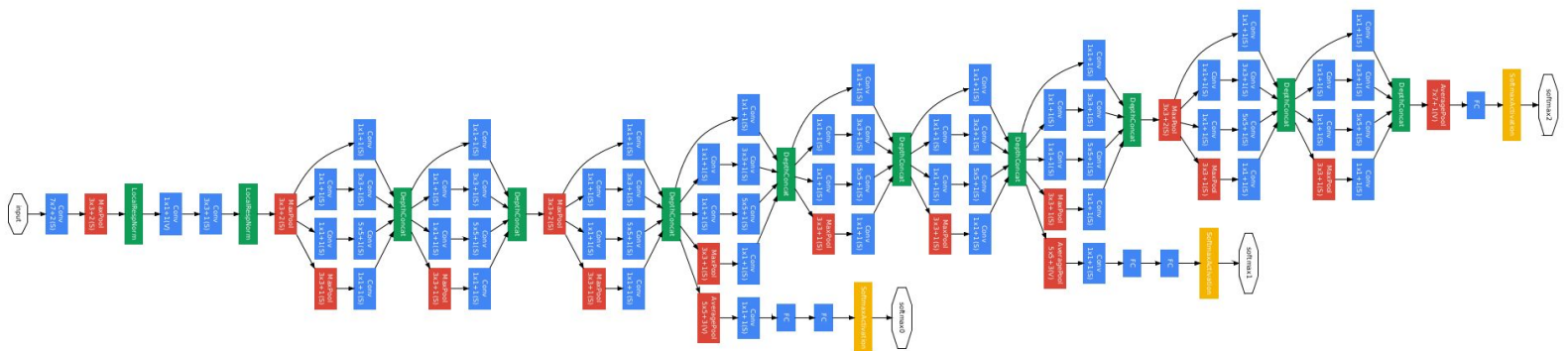
Categoria	Imágenes
Bread	1724
Dairy product	721
Dessert	2500
Egg	1648
Fried food	1461
Meat	2206
Noodles/Pasta	734
Rice	472
Seafood	1505
Soup	2500
Vegetable/Fruit	1154
Non food	2500

Modelo discriminativo

Red neuronal InceptionV3



- 23.851.784 parámetros
- 159 capas
- Pre-entrenada en Imagenet
- Reentrenamiento de capas profundas



StyleSage Image Classification Demo

Food Type Classification. Please select an image of food.



Predictions

Meat	0.96394
Seafood	0.03529
Fried food	0.00073
Bread	0.00001
Vegetable/Fruit	0.00001

Model took 0.024 seconds.

Provide an image URL

Classify URL

Or upload an image:

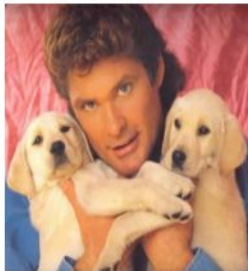
Choose File No file chosen

Classify file

Rendimiento - **91% exactitud**

StyleSage Image Classification Demo

Food Type Classification. Please select an image of food.



Predictions

Non food	0.99773
Dessert	0.00216
Seafood	0.00009
Soup	0.00002
Egg	0.00000

Model took 0.023 seconds.

Provide an image URL

Classify URL

Or upload an image:

Choose File No file chosen

Classify file

Rendimiento - **91% exactitud**



Modelo generativo

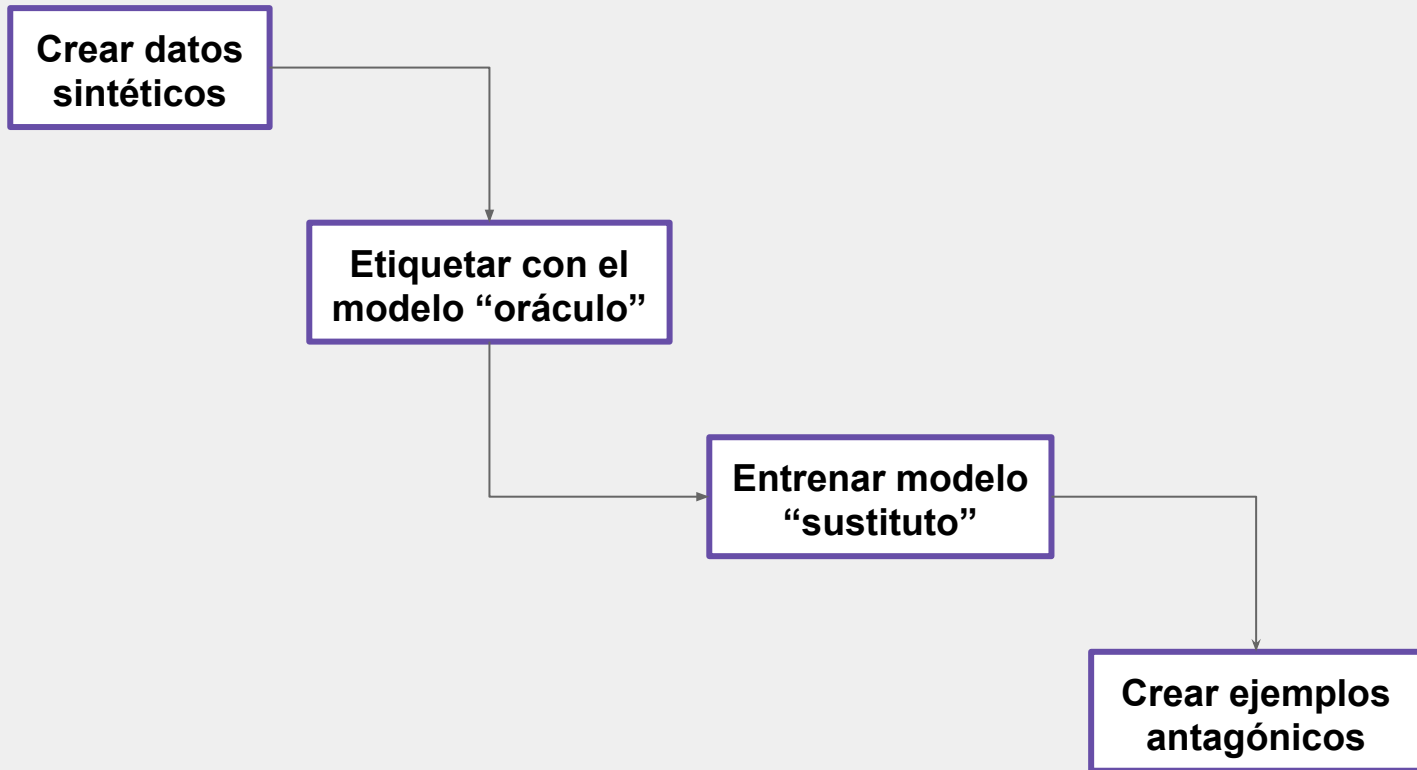
“An adversarial example library for constructing attacks, building defenses, and benchmarking both”

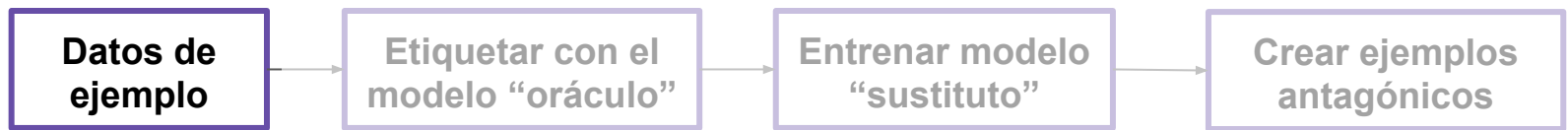
by Goodfellow and Papernot



- La librería más actual y completa (v2.1.0):
 - Modelos (en Tensorflow, Keras, Pytorch...)
 - Ataques (Fast gradient sign method, Carlini-Wagner attack...)
 - Defensas (Adversarial training)
 -

Ataque no orientado de caja negra





```
train_generator = train_datagen.flow_from_directory(
    '/data/pycon18/data/randomfood/training' ,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=BATCH_SIZE,
)

validation_generator = test_datagen.flow_from_directory(
    '/data/pycon18/data/randomfood/evaluation' ,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=BATCH_SIZE,
)

x_train, y_train = train_generator.next()
x_test, y_test = validation_generator.next()

# Initialize substitute training set reserved for adversary
X_sub = x_test
Y_sub = np.argmax(y_test, axis=1)
```

Datos de ejemplo

Etiquetar con el modelo "oráculo"

Entrenar modelo "sustituto"

Crear ejemplos antagónicos

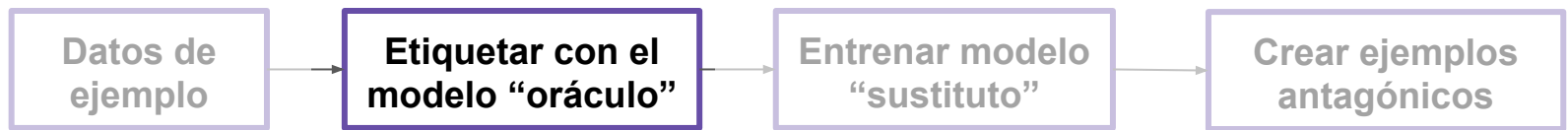
```
train_generator = train_datagen.flow_from_directory(
    '/data/pycon18/data/randomfood/training' ,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=BATCH_SIZE,
)
validation_generator = test_datagen.flow_from_directory(
    '/data/pycon18/data/randomfood/evaluation' ,
    target_size=(IM_WIDTH, IM_HEIGHT),
    batch_size=BATCH_SIZE,
)
```

```
x_train, y_train = train_generator.next()
x_test, y_test = validation_generator.next()
```

Carga de imágenes

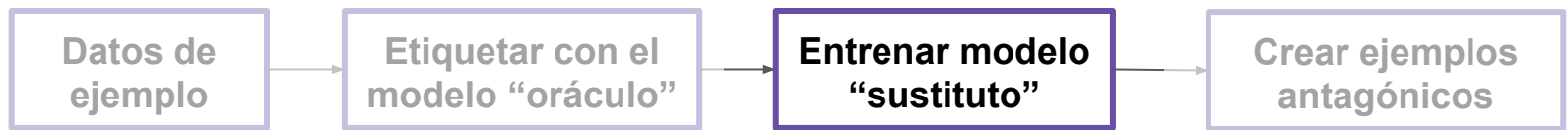
```
# Initialize substitute training set reserved for adversary
X_sub = x_test
Y_sub = np.argmax(y_test, axis=1)
```

Necesitamos separar
train y eval para el
modelo sustituto



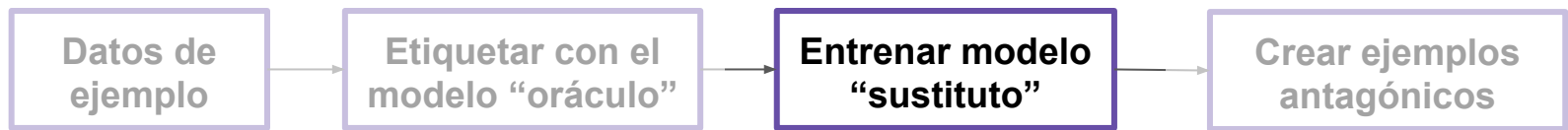
```
model = load_model('/data/pycon18/inceptionv3-ft120_910acc.model')
kmodel = KerasModelWrapper(model)
bbox_preds = kmodel.get_probs(x)
```

You could replace this by a remote labeling API for instance



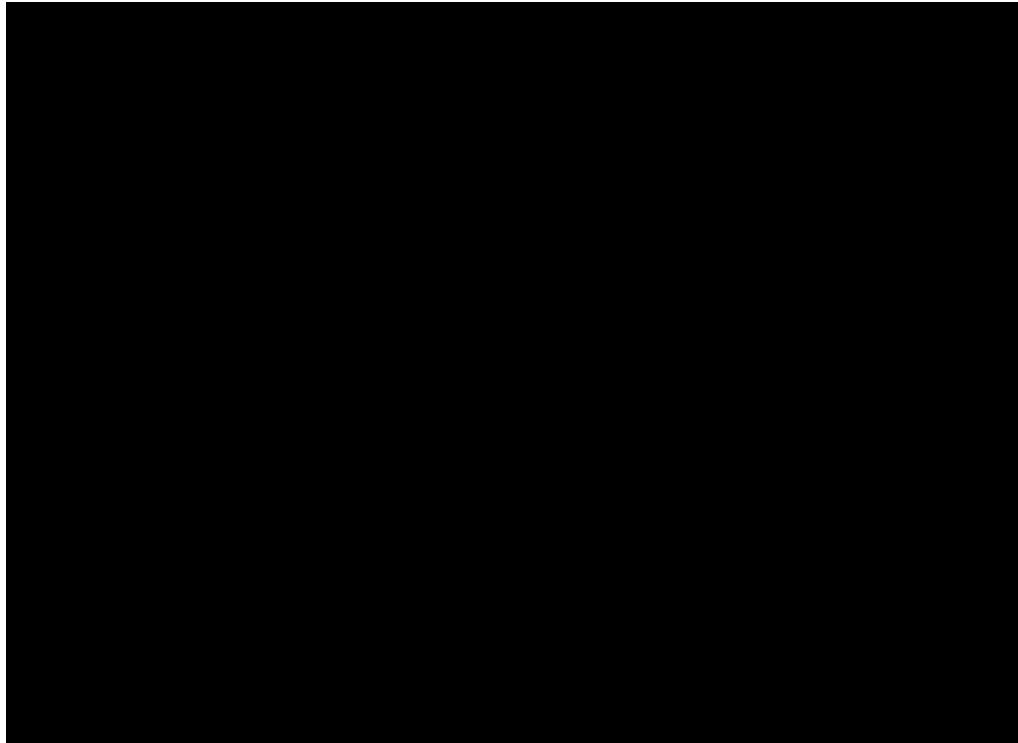
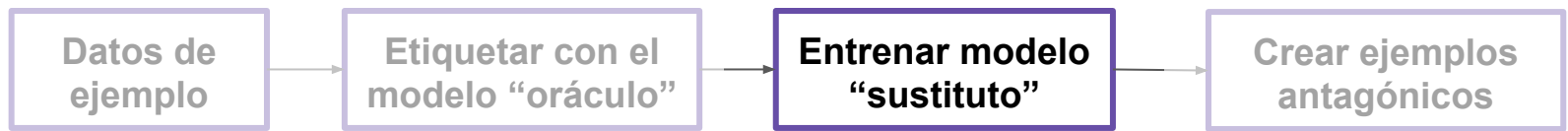
```
class ModelSubstitute (Model):
    def __init__(self, scope, nb_classes, nb_filters=200, **kwargs):
        del kwargs
        Model.__init__(self, scope, nb_classes, locals())
        self.nb_filters = nb_filters

    def fprop(self, x, **kwargs):
        del kwargs
        my_dense = functools.partial(
            tf.layers.dense, kernel_initializer=HeReLuNormalInitializer)
        with tf.variable_scope(self.scope, reuse=tf.AUTO_REUSE):
            y = tf.layers.flatten(x)
            y = my_dense(y, self.nb_filters, activation=tf.nn.relu)
            y = my_dense(y, self.nb_filters, activation=tf.nn.relu)
            logits = my_dense(y, self.nb_classes)
            return {self.O_LOGITS: logits,
                    self.O_PROBS: tf.nn.softmax(logits=logits)}
```

```
class ModelSubstitute(Model):  
    def __init__(self, scope, nb_classes, nb_filters=200, **kwargs):  
        del kwargs  
        Model.__init__(self, scope, nb_classes, locals())  
        self.nb_filters = nb_filters  
  
    def fprop(self, x, **kwargs):  
        del kwargs  
        my_dense = functools.partial(  
            tf.layers.dense, kernel_initializer=HeReLuNormalInitializer)  
        with tf.variable_scope(self.scope, reuse=tf.AUTO_REUSE):  
            y = tf.layers.flatten(x)  
            y = my_dense(y, self.nb_filters, activation=tf.nn.relu)  
            y = my_dense(y, self.nb_filters, activation=tf.nn.relu)  
            logits = my_dense(y, self.nb_classes)  
        return {self.O_LOGITS: logits,  
                self.O_PROBS: tf.nn.softmax(logits=logits)}
```

Configuración de una red "sencilla"

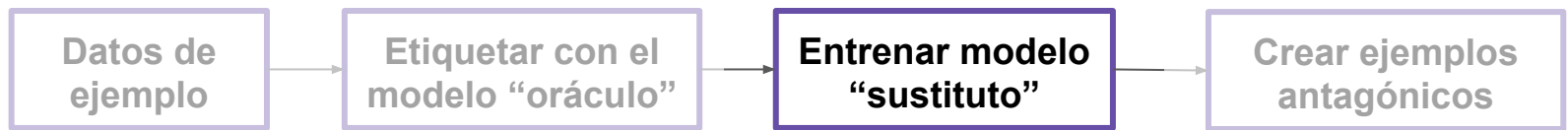




```
# Train the substitute and augment dataset alternatively
for rho in xrange(data_aug):
    print("Substitute training epoch #" + str(rho))
    train_params = {'nb_epochs': nb_epochs_s, 'batch_size': batch_size,
                    'learning_rate': learning_rate}
    train(sess, loss_sub, x, y, X_sub, to_categorical(Y_sub, nb_classes),
          init_all=False, args=train_params, rng=rng,
          var_list=model_sub.get_params())

    if rho < data_aug - 1:
        lambda_coef = 2 * int(int(rho / 3) != 0) - 1
        X_sub = jacobian_augmentation(sess, x, X_sub, Y_sub, grads,
                                      lambda_coef * lambda, aug_batch_size)

        Y_sub = np.hstack([Y_sub, Y_sub])
        X_sub_prev = X_sub[int(len(X_sub)/2):]
        eval_params = {'batch_size': batch_size}
        bbox_val = batch_eval(sess, [x], [bbox_preds], [X_sub_prev],
                              args=eval_params)[0]
        Y_sub[int(len(X_sub)/2):] = np.argmax(bbox_val, axis=1)
```



Train the substitute and augment dataset alternatively

```
for rho in xrange(data_aug):
```

```
    print("Substitute training epoch #" + str(rho))
```

```
    train_params = {'nb_epochs': nb_epochs_s, 'batch_size': batch_size_s,
                    'learning_rate': learning_rate}
```

```
    train(sess, loss_sub, x, y, X_sub, to_categorical(Y_sub, nb_classes),
          init_all=False, args=train_params, rng=rng,
          var_list=model_sub.get_params())
```

```
if rho < data_aug - 1:
```

```
    lambda_coef = 2 * int(int(rho / 3) != 0) - 1
```

```
    X_sub = jacobian_augmentation(sess, x, X_sub, Y_sub, grads,
                                  lambda_coef * lambda, aug_batch_size)
```

```
    Y_sub = np.hstack([Y_sub, Y_sub])
```

```
    X_sub_prev = X_sub[int(len(X_sub)/2):]
```

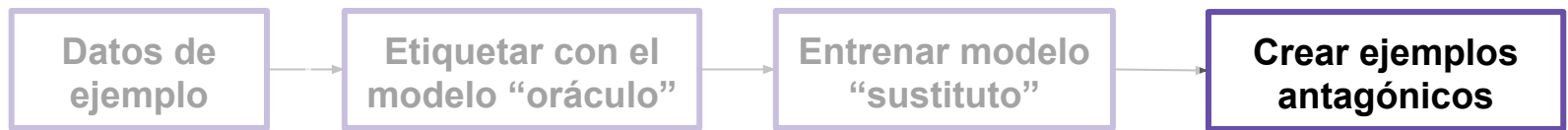
```
    eval_params = {'batch_size': batch_size}
```

```
    bbox_val = batch_eval(sess, [x], [bbox_preds], [X_sub_prev],
                           args=eval_params)[0]
```

```
    Y_sub[int(len(X_sub)/2):] = np.argmax(bbox_val, axis=1)
```

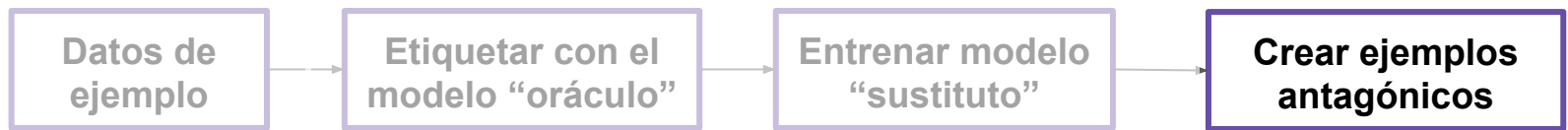
Entrenamiento del
sustituto

Generación de datos
sintéticos (opcional)



```
# Initialize the Fast Gradient Sign Method (FGSM) attack object.
fgsm_par = {'eps': 0.3, 'ord': np.inf, 'clip_min': 0., 'clip_max': 1.}
fgsm = FastGradientMethod(model_sub, sess=sess)

# Generate adversarial images
x_adv_sub = fgsm.generate(x, **fgsm_par)
adv_images = sess.run(x_adv_sub, feed_dict={x: x_test})
```



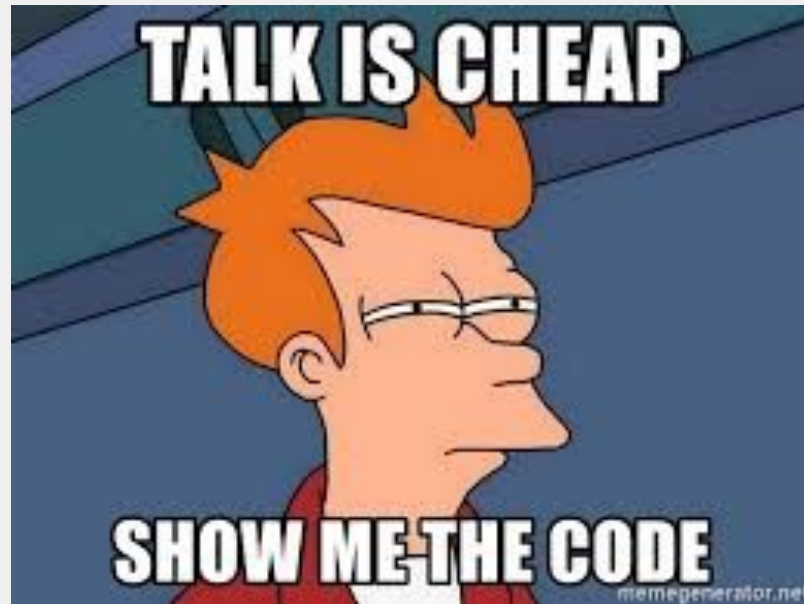
```
# Initialize the Fast Gradient Sign Method (FGSM) attack object.
fgsm_par = {'eps': 0.3, 'ord': np.inf, 'clip_min': 0., 'clip_max': 1.}
fgsm = FastGradientMethod(model_sub, sess=sess)
```

```
# Generate adversarial images
```

```
x_adv_sub = fgsm.generate(x, **fgsm_par)
```

```
adv_images = sess.run(x_adv_sub, feed_dict={x: x_test})
```

Generación de ejemplos
antagónicos



<https://github.com/aliciapj/pycon18-attack>

Resultados



Predictions

Bread	0.99288
Soup	0.00663
Dessert	0.00046
Egg	0.00001
Dairy product	0.00001

Predictions

Dessert	0.99995
Bread	0.00004
Soup	0.00001
Non food	0.00000
Fried food	0.00000

Resultados



Predictions

Dessert	0.45860
Meat	0.42155
Fried food	0.11669
Dairy product	0.00191
Seafood	0.00062

Predictions

Dairy product	0.96893
Dessert	0.02798
Fried food	0.00102
Meat	0.00075
Soup	0.00056

Resultados



Predictions	
Dairy product	0.97058
Dessert	0.02939
Egg	0.00003
Seafood	0.00000
Soup	0.00000

Predictions	
Dessert	0.99279
Dairy product	0.00668
Egg	0.00030
Seafood	0.00021
Soup	0.00002

Resultados



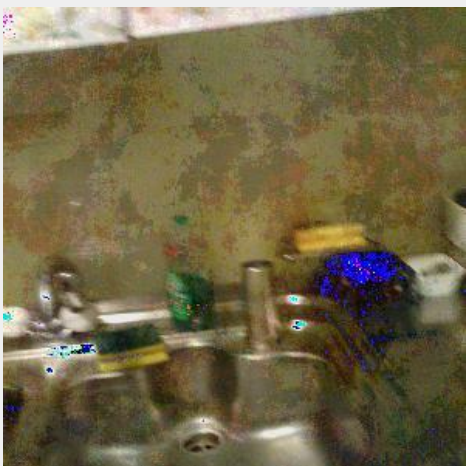
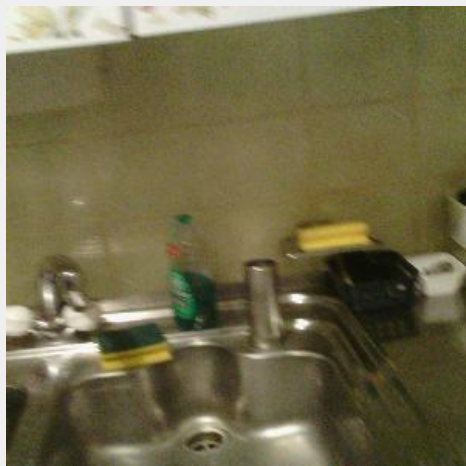
Predictions

Non food	0.89039
Seafood	0.05469
Dessert	0.03103
Fried food	0.01078
Meat	0.00689

Predictions

Seafood	0.91692
Dessert	0.07600
Fried food	0.00168
Non food	0.00164
Bread	0.00126

Resultados



Predictions

Non food	0.59072
Dessert	0.34936
Soup	0.05911
Seafood	0.00035
Egg	0.00026

Predictions

Soup	0.36556
Seafood	0.28649
Bread	0.11913
Meat	0.10086
Dairy product	0.08910

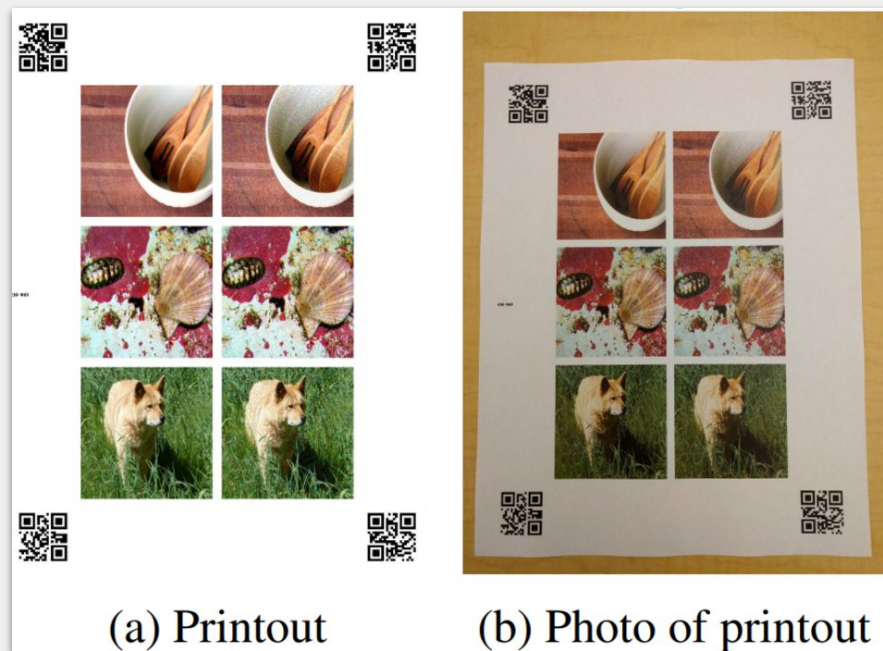


Discusión

¿Cómo afecta a nuestros sistemas y algoritmos?

Ejemplos reales

¡Los ejemplos antagónicos funcionan incluso cuando son impresos y fotografiados a resoluciones estándar!



Ejemplos reales

Original - library (76%)

Antagónica -prison (52%)

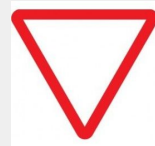


Original - washer (54%)

Antagónica - doormat (35%)

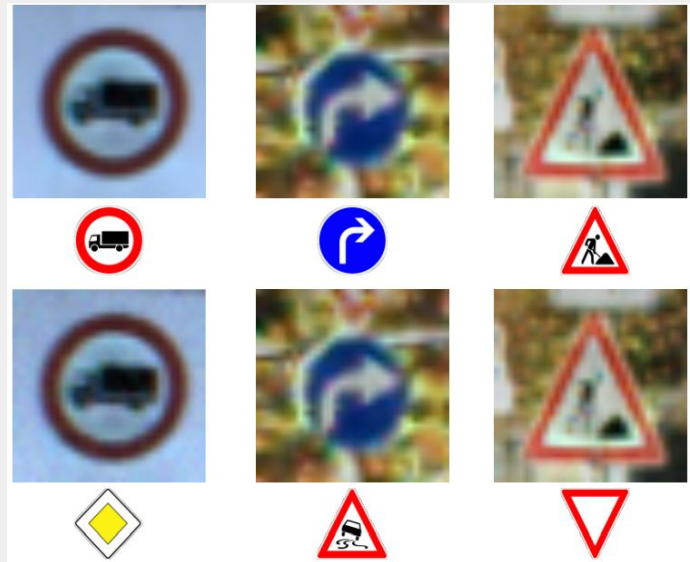


Ejemplos reales



Señales de tráfico

- Dataset GTSRD





Cómo defendernos

¿Qué podemos hacer?

1. **No es fácil. Vuestras APIs son vulnerables.**
2. Todavía no existe un método infalible
3. La propuestas actuales se basan en las redes neuronales

Estrategia reactiva

- Intentar detectar y anular el ataque.
- Aumentar el tamaño de las entradas o la complejidad de la red para suponer mayor esfuerzo al sistema atacante.

Estrategia proactiva

- Crear modelos más robustos a los ataques.
- Técnicas relacionadas con el entrenamiento de la red.

Estrategias proactivas

Adversarial training

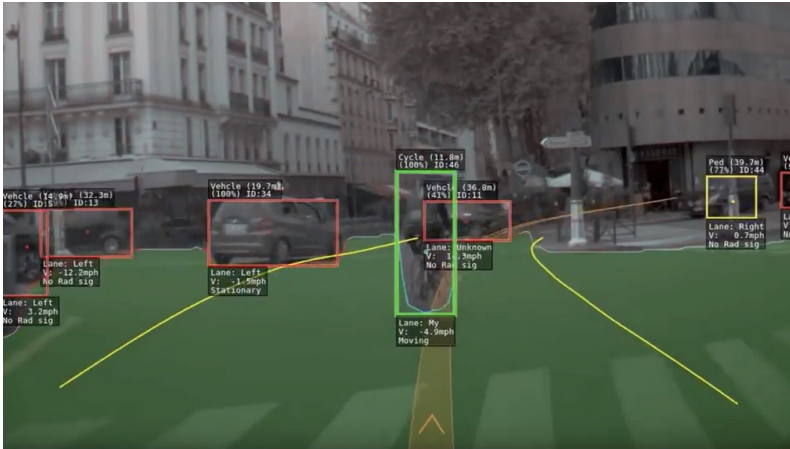
El dataset de entrenamiento del modelo se enriquece añadiendo ejemplos antagónicos creados por nosotros. Aumenta la robustez del modelo y sirve como factor regularizador.

Defensive distillation

Aplica el principio de entrar un modelo sustituto para reducir la confianza de las predicciones del sistema. El modelo se entrena sobre distribuciones de probabilidad en vez de etiquetas.

Gradient masking

Intenta ocultar el gradiente. Se ha demostrado que no es válido. El modelo sustituto lo hace inútil.



- La IA ya salido del ámbito teórico y la investigación.
- **Technical AI safety** es un campo nuevo, pero con mucho potencial.
- Ya presente en empresas como Google DeepMind, con su *DeepMind safety team*.

- Agente del juego CoastRunners (OpenAI). ¿Buen diseño de métricas?
- Los cimientos de muchas aplicaciones del futuro.

Reflexiones

gracias!

¿Alguna pregunta?

@alipeji | alicia@stylesage.co
@fjordonz | javier.ordonez@stylesage.co

We are hiring!!!

