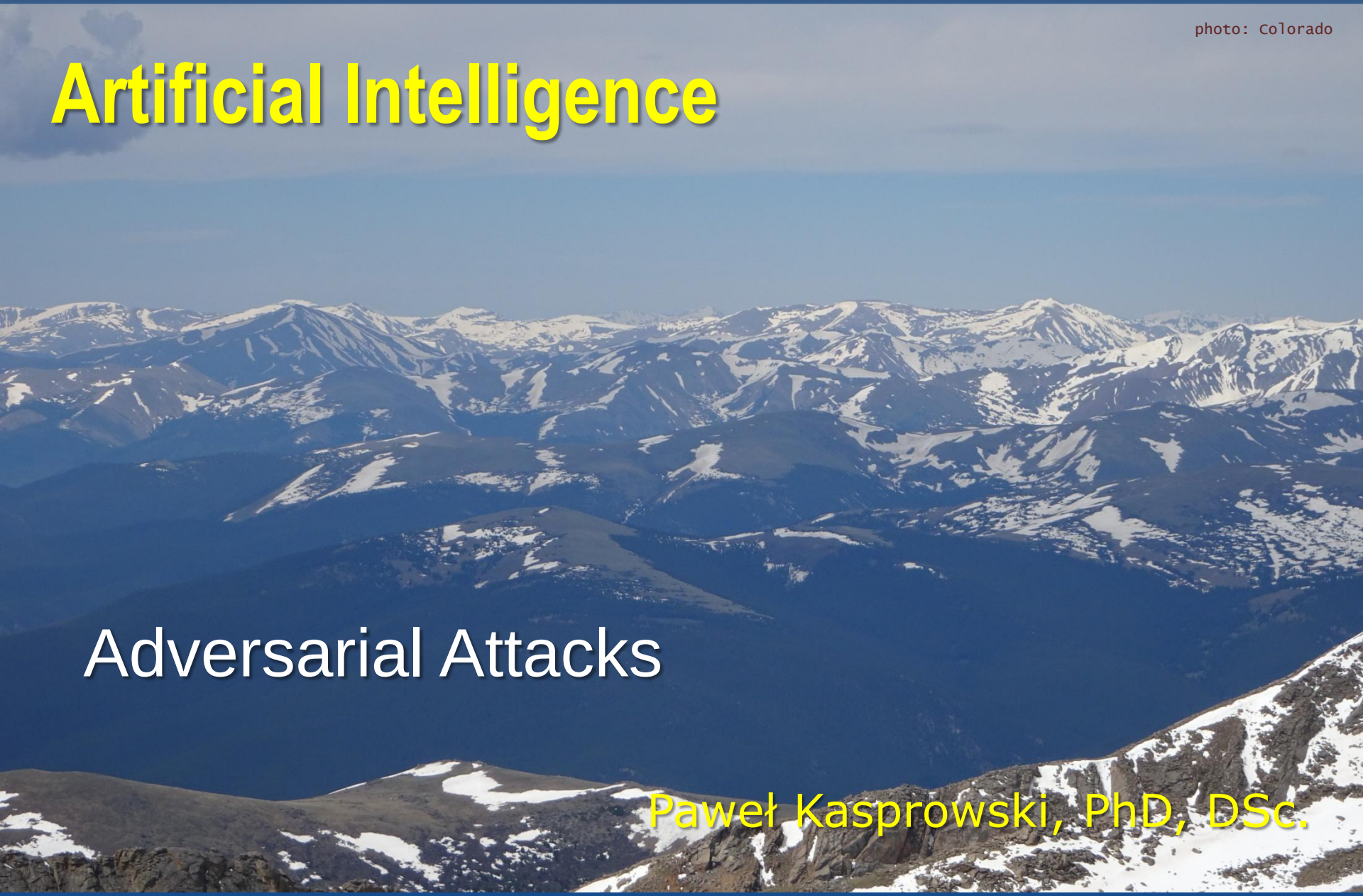
pawel.kasprowski@polsl.pl



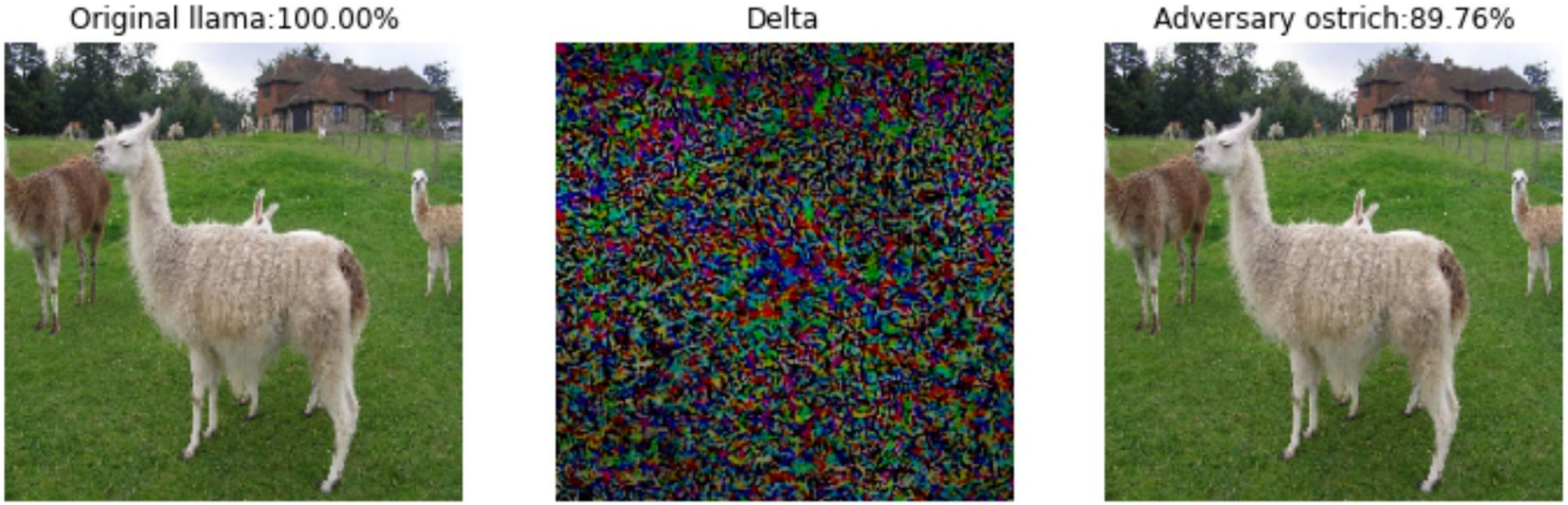
Artificial Intelligence

Adversarial Attacks

Paweł Kasprowski, PhD, DSc.

Adversarial attacks

- After tiny (almost unvisible for human) modifications the network classifies the object differently



Fooling the network

- We have an image that is correctly identified
- We modify the image in a way that it is not visible for humans
- After this modification the image is classified differently!
- Question: how to modify the image?
- Answer: use gradient descent
  - but reverse the loss!

Fooling ResNet50

adversarial/adversarial.ipynb

- Loads ResNet50
- Classifies the given image
- Modifies the image to be misclassified

Preparation

- Initial images as tensors:
  - baseImage = tf.constant(image, dtype=tf.float32)
  - delta = tf.Variable(tf.zeros_like(baseImage), trainable=True)
- Initial objects:
  - optimizer = Adam(learning_rate=0.1) # high rate!
  - lossFunct = SparseCategoricalCrossentropy()
- Loading the model
  - model = ResNet50(weights="imagenet")

Learning step

```
with tf.GradientTape() as tape:
    tape.watch(delta)
    adversary = preprocess_input(baseImage + delta)
    predictions = model(adversary , training=False) # use model
    originalLoss = lossFunct(tf.convert_to_tensor([real_class]),predictions)
    loss = - originalLoss # we want to *maximize* the loss
gradients = tape.gradient(loss, delta) # calculate change in delta
optimizer.apply_gradients([(gradients, delta)])
delta.assign_add(delta) # apply new changes
```

The whole loop

- Learning loop:

```
for step in range(0, steps):
    delta, loss = do_step(baseImage, delta)
    if step % 10 == 0:
      show_ step(delta,loss)
```

- Check results so far:

```
adverImage = (baseImage + delta).numpy().squeeze()
adverImage = np.clip(adverImage, 0, 255).astype("uint8")
model.predict(adverImage)
```

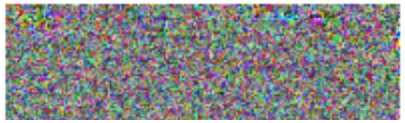
Results

- The image is misclassified just after few iterations
- However the noise is visible very fast

```
step: 12, loss: -0.524643063545227...
12. llama -> (ostrich:0.82, llama:0.12, crane:0.02)
```

Original llama:100.00%

Delta

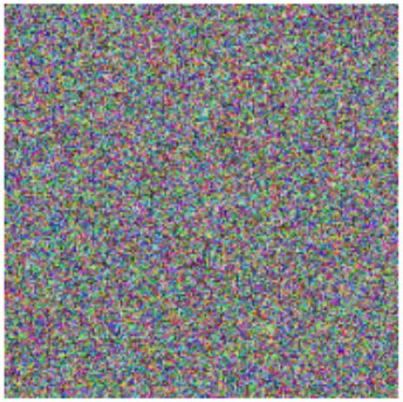


Adversary ostrich:81.58%

```
step: 16, loss: -3.6231160163879395...
16. llama -> (sea_slug:0.38, peacock:0.14, coral_reef:0.11)
```

Original llama:100.00%

Delta



Adversary sea_slug:38.26%

Reducing the noise by clipping

- Instead of:

  optimizer.apply_gradients([(gradients, delta)])

  delta.assign_add(delta)

- We use:

  optimizer.apply_gradients([(gradients, delta)])

  clipped_delta = tf.clip_by_value(delta,
  clip_value_min=-0.01, clip_value_max=0.01)

  delta.assign_add(clipped_delta)

- Result:
  - training lasts longer but the noise is not visible!

Another image

- Initial classification: Egyptian_cat
- After some iterations classification changes to:
  - tiger_cat
  - tiger
  - tabby
- All these classes are similar 'cat-like' classes
- Question: is it possible to modify the image to be similar to the given class?
- Answer: why not? ☺

Targeted attack

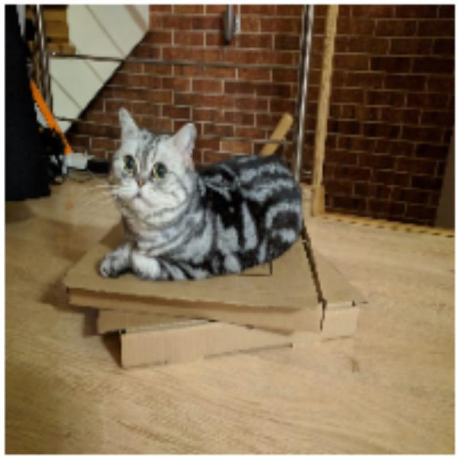
- We want to change the image the way that the network will classify it as the specific class
- Our target class: llama (id=355)
- The change in our loop

```
originalLoss =  # loss for real class
    lossFunct(tf.convert_to_tensor([real_class]),predictions)
targetLoss =  # loss for target class
    lossFunct(tf.convert_to_tensor([target_class]),predictions)
loss = targetLoss - originalLoss  # goal: minimize error to
    targetLoss and maximize error to originalLoss
```

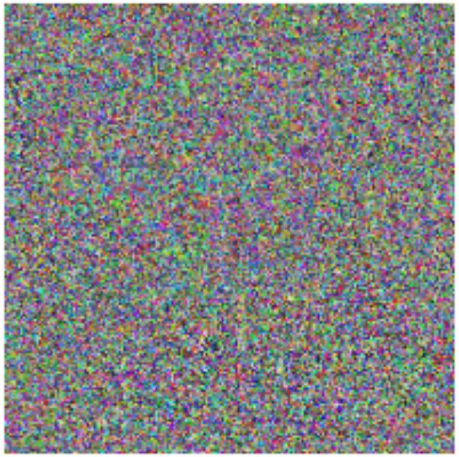
Turning a cat to llama in 25 steps!

25. Egyptian_cat -> (llama:0.93, plunger:0.01, hen:0.01)

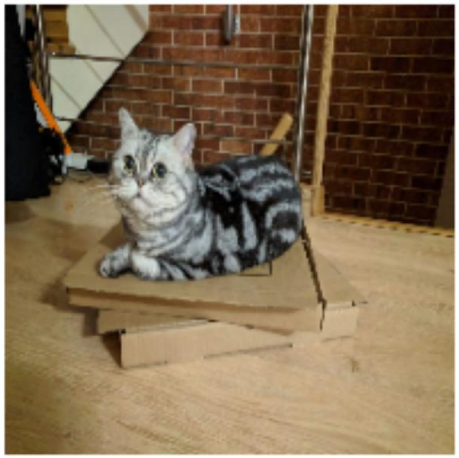
Original Egyptian_cat:95.43%



Delta



Adversary llama:92.71%



- Additionally, save the images:

```
cv2.imwrite(f"outdir/{image_name}_{step:02d}.jpg",
    cv2.cvtColor(adverImage.squeeze(),
    cv2.COLOR_RGB2BGR))
```

Testing the images:

check_image_resnet50.ipynb

- Loads all images from the given directory and classifies them:

kot-frusia_00.jpg -> (Egyptian_cat:0.73, Norwegian_elkhound:0.05, tiger_cat:0.04)
kot-frusia_05.jpg -> (Egyptian_cat:0.73, Norwegian_elkhound:0.04, tiger_cat:0.04)
kot-frusia_10.jpg -> (Egyptian_cat:0.70, tiger_cat:0.05, tabby:0.04)
kot-frusia_15.jpg -> (Norwegian_elkhound:0.21, Egyptian_cat:0.17, standard_schnauzer:0.09)
kot-frusia_20.jpg -> (llama:0.16, Norwegian_elkhound:0.16, swing:0.10)
kot-frusia_25.jpg -> (llama:0.30, Norwegian_elkhound:0.13, swing:0.09)
kot-frusia_30.jpg -> (llama:0.73, swing:0.04, Norwegian_elkhound:0.03)
kot-frusia_35.jpg -> (llama:0.94, swing:0.01, standard_schnauzer:0.01)
kot-frusia_40.jpg -> (llama:0.98, swing:0.00, ostrich:0.00)
kot-frusia_45.jpg -> (llama:0.99, swing:0.00, ostrich:0.00)
kot-frusia_50.jpg -> (llama:0.99, swing:0.00, ostrich:0.00)

Adversarial attacks

- Serious problem for neural network models
- May be used for cyber attacks
- May mislead the network

Eykholt, Kevin, et al. Robust physical-world attacks on deep learning visual classification.

Summary

- In previous versions:
  - Keras gave access only to basic functionalities
  - Tensorflow was quite complicated
- From version 2.0 Keras gives low level access to learning algorithms via GradientTape
- Creation of complex architecture is simple with few lines of code
  - hypernetworks
  - adversarial attacks
  - generative adversarial networks (GANs...)