Exercise 8

Explanation-by-Example Methods

- You can get a bonus point for this exercise if you pass at least 85% of the tests. Code is automatically tested after pushing. If your tests fail, either your code is wrong, or you solved the task differently. In the second case, we will manually check your solution.
- Three collected bonus points result in a 0.33 increase of the final grade.
- You are allowed to work in groups with up to three people. You do not need to specify your ErgebnisPIN as it is already collected from the entrance test.
- Follow the 'README.md' for further instructions.
- Finish this exercise by 8th December, 2021 at 11:59 pm.

Adversarial Examples

Adversarial Examples are carefully constructed inputs that "fool" a trained model. In this exercise, your task is to generate adversarial examples using the fast gradient sign method:

$$x \leftarrow x + \delta \tag{1}$$

where

$$\delta = \epsilon \cdot sign(\nabla_x \mathcal{L}(x, y; \theta)) \tag{2}$$

This is an untargeted method, which means the resulting class label is not specified in advance. The goal is to create an adversarial input that is indistinguishable (to the human eye) from an original input, but confuses the model.
In this exercise, the goal is to create adversarial examples to fool an image classifier trained on the ImageNet dataset. To accomplish this, complete the functions *get_gradient*, *perturb_image*, *create_adversarials*. Finally, complete the *plot_adversarials* function to visualize the original image and the generated adversarial examples for different values of ϵ.

Associated file: *adversarial.py*.

Counterfactual Examples

Counterfactual examples are instances that with minimal feasible changes result in a different, used-defined prediction outcome. They tells us exactly what needs to be done to change a prediction and thus provide (actionable) recourse. Complete the functions *compute_distance, compute_output_difference, compute_loss and create_counterfactual* to create counterfactual instances based on the method proposed by Wachter et al.:

$$argmin_{x'} d(x, x') + \lambda \cdot (f(x') - y')^2 \tag{3}$$

where d is the L1 distance.
See also W08 T03 slide 21.

Associated file: *counterfactual.py*.