DSAIT 4015 2025-2026
Project assignment 2
Black-box adversarial testing of the VGG16 model (Image Detection)

Instructors: Annibale Panichella and Cynthia Liem

Background

Deep neural networks used for image classification, such as VGG-16 trained on ImageNet, are known to be vulnerable to adversarial examples: tiny perturbations to an input image that can drastically change the model's prediction, even when the perturbation is visually imperceptible.

In practice, many adversarial attacks require white-box access (model parameters and gradients). However, in real-world systems we often face black-box models: we can query them for predictions but do not know their internal structure.

This assignment focuses on testing the robustness of VGG-16 by designing a black-box Hill-Climbing (HC) search that generates adversarial attacks *without access to gradients*.
You will compare your results against white-box baseline attacks (FGM and PGD) implemented for you using CleverHans.

Overall purpose of the assignment

In this assignment, you will:

1. Implement a black-box Hill-Climbing algorithm that generates adversarial attacks for the VGG-16 model, following a provided template.
2. Ensure that all pixel-level mutations comply with the given ε-bound (same epsilon used by the baseline PGD/FGM attacks).
3. Analyze and evaluate the effectiveness of your black-box attack in comparison to the provided white-box baselines in terms of:
    - Attack success rate
    - Magnitude of perturbations
    - Efficiency (time per attack)
    - Observed patterns (which images are easier/harder to fool)
    - Strengths and weaknesses of black-box hill-climbing compared to gradient-based attacks

At grading time, we will test your code on **new images not included in the assignment**, to verify that your HC implementation is *generalizable* and *not hard coded* on the image we provide you for generating the attacks.

What we provide to you: To support your work in this assignment, we provide a Python repository containing all required baseline material. Please carefully read the accompanying README.md file, which explains the file structure and how to run the provided scripts.

The repository includes:

- **baselines.py** - A fully working implementation of two *white-box* adversarial attacks (FGM and PGD) applied to VGG-16.
  You will use these as baselines to compare against your own black-box Hill-Climbing (HC) attack.
- **hill_climbing.py** (template) - A **partially implemented** script where you must complete the Hill-Climbing search algorithm. *Do not modify the provided interfaces*. Your implementation must remain compatible with the existing structure so we can test your solution on unseen images.
- **images/ directory** - A set of input images used as seeds for both the baseline attacks and your HC method.
- **data/image_labels.json** - Contains the human-assigned semantic label of each image (e.g., "goldfish", "viaduct").
  These labels indicate what the model *should* predict.
- **data/imagenet_classes.txt** - The canonical list of 1000 ImageNet category names used by VGG-16.
- **requirements.txt** - Describes the Python environment and dependency versions used for our reference solution.

Compared to Assignment 1, **your entire group will work together as a single team** (i.e., no sub-groups). All steps of this assignment (i.e, implementation, experimentation, and reporting) must be completed collaboratively.

1. Implement the Hill Climber

Your main task is to implement the hill climbing algorithm that generate adversarial attacks by inspecting only the input (images) output (predicted labels with confidence values) of the VGG16 models.

We provide the template of the hill climber in the **hill_climbing.py** script and you have to complete the implementation for the placeholder methods within the file.
Your implementation must:

a. Load and evaluate images

1. Read image paths and human labels from data/image_labels.json
2. Load the actual image files from images/
3. Use VGG-16 in evaluation mode to obtain predictions

b. Implement mutation operators (“neighborhood generation”)

- Mutations must modify pixel values **without exceeding the ε-bound in L∞ norm**
- Examples include (but are not limited to):
    - Additive bounded noise
    - Localized masks or patches
    - Channel-specific perturbations
    - Search over structured patterns

c. Implement a fitness/evaluation function

Your fitness function must analyze only:

- The **input image** (pixels)
- The **output prediction** (probabilities, label)
-

Recall, the fitness function should measure how close is the HC from “fooling” the model, i.e., leading to a wrong predicted top-1 label.

You may NOT use gradients or internal model state.

d. Implement the Hill-Climbing loop

- Start from the clean image
- Iteratively generate neighbor candidates
- Accept neighbors according to your strategy (e.g., first improvement, best improvement, stochastic)
- Terminate when:
    - An adversarial example is found, or
    - The iteration budget is exhausted (number of iterations, or mutated images used for the attack evaluation)

e. Save successful adversarial images

Your script should store results in a folder (e.g., hc_results/), including:

- Original image & prediction
- Final HC adversarial example & prediction
- Information such as number of iterations, mutation type, etc.

f. Constraints

All your mutation operators **must enforce**:

$$|x_{\{adv\}} - x_{\{orig\}}|_{\{\infty\}} \leq \epsilon$$

Where $x_{\{adv\}}$ is a mutated image, $x_{\{orig\}}$ is the original image (without any pixel change) and ϵ is the max pixel value delta allowed for all pixel in the original image.

Note that this is the **same epsilon** used in the baseline PGD/FGM scripts.

2. Experimental Evaluation and Comparison with Baselines

In this second part, you will compare your black-box Hill-Climbing attacks against the white-box baselines (FGM and PGD). Using baselines.py, run FGM and PGD attacks on the same images used by your hill climber.

Perform a structured comparison covering the following aspects:

1. **Attack success**: For each image, report whether FGM, PGD, and your HC were able to produce a successful adversarial attack. Summaries may include tables or aggregated statistics (e.g., number of successful attacks per method). **Notice**: *a successful adversarial attack is achieved when the model's top-1 prediction is different from the original image's ground-truth class.*

2. **Perturbation characteristics**: Compare how the perturbations produced by each method differ. Your discussion should include at least:

    - The average number of pixels that changed.
    - The maximum observed perturbation magnitude (L∞ distance).
    - Optional visual comparison (e.g., showing clean vs. adversarial images in an appendix).

3. **Efficiency**: Evaluate and compare you Hill Climber vs. the white-box baseline in terms runtime (wall-clock time in seconds). Include a brief discussion of changes in scalability when using different values for the epsilon.

4. **Quality observations**: Reflect on your empirical findings:

    - Which images were easiest to attack, and which were more difficult?
    - Did you notice patterns in vulnerability (e.g., smooth textures, background–foreground contrast, object size)?
    - Does your HC tend to discover different failure modes than PGD/FGM?

4. Pros and cons of black-box HC vs. white-box baselines: Discuss the relative strengths and weaknesses of the two families of methods.

What to deliver

As final deliverable for this assignment, deliver a zip file containing:

1. *A project report, with:*
    - **Description of your hill-climbing design and rationale**, including the mutation operators, fitness function, and any other decision implemented for the hill climber *(max 1.5 pages)*
    - **Evaluation comparing HC vs. FGM/PGD** *(max 1.5 pages)*
    - **Reflection and insights:** strengths/weaknesses of black-box vs. white-box, surprising behaviors or observations *(max 1 page)*
    - **Optional appendix**

    Total report length (excluding appendix): ***max 4 pages***.

2. *A zip file with:*
    - A **requirements.txt** file listing:
        - the Python version you used (if different from the version we recommend)
        - all Python dependencies (libraries) with specific versions you may have added to implement your HC
    - The **completed implementation** of your hill-climbing attack:
        - Your hill_climbing.py file (**without changing the provided interface / function signatures**).
        - Any additional Python files/modules you created that are needed by your HC (if applicable).
    - A README.md file indicating:
        - The command required to re-produce the numerical results (and, if relevant, plots/tables) that appear in your report.
        - Any special steps needed to set up the environment or data.

Final Notes

- Your HC will be executed on images **not included** in the assignment, to verify generality.
- Creativity in mutation operators and fitness design is encouraged.