



University of Poitiers — Academic year 2025–2026
U.F.R. Sciences Fondamentales et Appliquées
Master's degree in Signal and Image Processing (TDSI)
EUR Smart IoT

MASTER'S INTERNSHIP REPORT

Submitted for the degree of Master

Generation of universal attacks against deepfake detectors

Internship carried out from 28 April to 30 September 2026

WRITTEN BY

Aymane SNOUSSI

HOST SUPERVISORS

Prof. Christophe CHARRIER
Dr. Emmanuel GIGUET

ACADEMIC SUPERVISOR

Prof. David HELBERT
University of Poitiers

GREYC – UMR CNRS 6072 – SAFE team

Campus 2, 6 boulevard Maréchal Juin, 14032 Caen Cedex



Acknowledgements

I would first like to thank Prof. Christophe Charrier for welcoming me into the SAFE team and for supervising this internship. His questions were consistently the ones I had avoided asking myself, and several of the directions that ended up mattering came straight out of our weekly meetings.

I am equally grateful to Dr. Emmanuel Giguet for co-supervising this work and for the forensic perspective he brought to a subject I would otherwise have treated as a purely algorithmic problem.

I thank Prof. David Helbert, my academic supervisor at the Université de Poitiers, for his follow-up throughout the internship.

My thanks also go to the members of the SAFE team for their welcome, and to the laboratory for granting me access to the computing resources without which none of the experiments reported here could have been run.

My deepest thanks go to my family. To my parents, above all. Everything I have been able to do this year rests on what they gave me long before it, and on the confidence they never withdrew, including in the months when I had little to show for the work. They carried the cost of this degree in ways that never appear in a report like this one. To my brothers and sisters, and to the whole family, for being present at every stage, at a distance and without ever making it feel like one.

I owe them everything. Nothing here would exist without them.

Contents

Acknowledgements	i
1 Context and framework	1
1.1 Introduction	1
1.1.1 Context	1
1.1.2 The gap this work addresses	1
1.1.3 Objective and research question	2
1.1.4 Approach and contributions	2
1.1.5 Structure of this report	3
1.2 Host organisation and internship framework	3
1.2.1 The GREYC laboratory	3
1.2.2 The SAFE team	4
1.2.3 Research context: the COMPROMIS project	5
1.2.4 Position and assignment	5
1.2.5 Missions	6
1.2.6 Working environment	6
1.3 Problem statement and specifications	7
1.3.1 Two non-negotiable constraints	7
1.3.2 What has to be measured	7
1.3.3 Acceptance criteria	7
1.3.4 A staged validation strategy	8
1.3.5 Summary of the chapter	8
2 State of the art	9
2.1 A structural weakness of deep architectures	9
2.2 Threat model	10
2.3 Per-image attacks	10
2.4 Universal adversarial perturbations	11
2.4.1 Vector cancellation and Fast-UAP	11
2.4.2 Input transformations	11
2.5 The data-free setting	12
2.5.1 Maximising internal activations	12
2.5.2 PSP-UAP, the reference method of this work	12

2.6	Measuring effectiveness and invisibility	13
2.7	Benchmarks and available implementations	14
2.8	Synthesis	14
2.9	Earlier work within the team	15
2.10	Where the literature stops	16
3	Approach and proposed method	17
3.1	Project management	17
3.1.1	An iterative approach, and why not a V-model	17
3.1.2	Three working rules	18
3.1.3	The evaluation protocol	18
3.1.4	Deliverables and schedule	18
3.1.5	Constraints	19
3.1.6	Risk management	20
3.1.7	Summary	21
3.2	The reference method and its reproduction	21
3.2.1	The problem the method solves	22
3.2.2	The pseudo-semantic prior	23
3.2.3	Input transformations	24
3.2.4	Sample reweighting	24
3.2.5	The objective function	24
3.2.6	The algorithm in one pass	25
3.2.7	Reproduction	26
3.2.8	What the baseline revealed	26
3.3	Proposed method	27
3.3.1	A staged pipeline	28
3.3.2	Contribution 1: an ensemble of surrogate models	28
3.3.3	Contribution 2: a diffusion-generated prior	30
3.3.4	Contribution 3: a perceptual amplitude budget	32
3.3.5	Consolidated results	35
4	Results, discussion and conclusion	37
4.1	Transfer to a deepfake detector	37
4.1.1	Target and protocol	37
4.1.2	Reusing the perturbation, and two necessary adaptations	37
4.1.3	A structural limitation	38
4.1.4	Results on images	39
4.2	Extension to video	39
4.3	Countermeasures	41
4.3.1	Input filtering	41
4.3.2	Adversarial training	42
4.3.3	Detecting the perturbation	43
4.3.4	Comparison	44

4.4	Discussion	44
4.4.1	Directions that did not work	44
4.4.2	Limitations	45
4.5	Conclusion	46
4.5.1	Technical conclusion	46
4.5.2	Perspectives	47
4.5.3	Personal assessment	47
	Bibliography	51

List of Figures

3.1	Actual schedule of the internship. Writing was deliberately started before the experimental work ended, so that figures and tables were produced as the measurements came in.	19
3.2	The limitation of earlier data-free methods, and the observation PSP-UAP builds on. Left: pure noise gives the optimisation nothing semantic to work with, and the perturbation overfits the surrogate. Right: regions cropped from a trained UAP are classified under diverse labels, which shows that the perturbation itself carries exploitable structure.	22
3.3	Generation of the semantic samples. Random noise is added to the current perturbation to form the pseudo-semantic prior. The sampler crops regions at random positions and resizes them to full scale, producing N semantic samples x_n . The perturbation is then added to each sample before it enters the surrogate model.	23
3.4	Computation of the sample weights. Each semantic sample follows a clean branch and an attacked branch through the same transformation and the same surrogate. The Kullback-Leibler divergence between the two output distributions measures how much the perturbation affected that sample, and the weight is its inverse.	25
3.5	The complete objective. Feature maps are extracted from the convolutional layers of the surrogate and their norms are summed, weighted by the coefficients obtained from the Kullback-Leibler branch.	25
3.6	Visual effect of the reproduced perturbation on three validation images, before and after. The mathematical patterns of the attack remain plainly visible, for a structural similarity of about 0.74.	27
3.7	The three contributions are applied sequentially. Each stage warm-starts from the perturbation produced by the previous one and refines it over a short run.	28
3.8	Ensemble of surrogates. The same semantic samples pass through three architectures. The three contributions are rebalanced before aggregation, so that the deepest network does not dictate the gradient, and a single shared perturbation is updated.	30

3.9	The three priors compared. The Gaussian prior carries no structure. The jigsaw prior adds coloured blocks but no scene. The third was produced by an unconditional diffusion model from pure noise; it depicts no real scene and no real object, but it has texture, contours and depth.	31
3.10	Replacing the random prior. The diffusion model runs once, offline, from pure noise and without conditioning. The rest of the pipeline, including the crop-and-resize sampling, is untouched.	31
3.11	Perturbation patterns, with values normalised for display. From left to right: ensemble baseline, diffusion prior, and two perceptual-budget configurations.	33
3.12	Perceptual amplitude budget. A mask derived from local contrast modulates the universal perturbation before it is applied. The perturbation itself stays unique and image-independent; only its local amplitude varies. Because the mask is active during training, the perturbation learns to concentrate its energy where the mask lets it pass.	34
3.13	Fooling rate against structural similarity for the perceptual budget sweep. Each curve holds the smooth-region amplitude constant while $\epsilon_{\max}$ varies. The two red markers are the ensemble baseline and the diffusion-prior perturbation, both under a uniform L_{∞} budget of 10/255.	35
3.14	Visual effect of the attack, with VGG16 predictions. Left, the original image. Centre, the uniform L_{∞} budget at 10/255. Right, the perceptual budget. The perturbation is markedly less visible in flat regions while the model is still fooled.	36
4.1	Deepfakes classified as authentic after the attack. Each row shows the original face, correctly detected, then the attacked face, classified as real, then the perturbation that was added, with its contrast amplified so that it can be seen at all. The mask is visible in that third image: the perturbation is almost flat over the plain background and regains its full pattern on the hair, the jacket and the edges of the face.	40

Chapter 1

Context and framework

1.1 Introduction

1.1.1 Context

Face manipulation has left the laboratory. In less than a decade, swapping one face for another in a video went from a research demonstration to something anyone can do in a browser. Generative models now produce results that a human observer cannot reliably question.

The damage is already measurable. In 2024, a fraudulent video call built around a synthetic company executive led to a transfer of roughly twenty-four million euros, and the number of deepfakes detected worldwide grew fourfold between 2023 and 2024. Fraud is only part of the problem. Identity theft, disinformation and the erosion of trust in video evidence follow from the same technology.

Automatic detection is the answer that has been deployed. Deep convolutional architectures are trained to spot what generation algorithms leave behind, such as lighting inconsistencies, compression artefacts or local deformations around the face. These systems work, and content platforms and investigation services now rely on them.

Here is the problem this report addresses. A detector is itself a neural network, and neural networks can be fooled on purpose. Add a carefully computed pattern to an image and the model changes its answer, while a human sees no difference at all. A detector that performs well on untouched deepfakes can therefore be silenced by an attacker who knows how to compute that pattern. From a forensic point of view this is worse than having no detector, because the system does not fail loudly. It keeps answering, with confidence, and it is wrong.

1.1.2 The gap this work addresses

Attacks of this kind have existed since 2014, and the field has produced a long line of increasingly refined methods. Chapter 2 follows that line in detail. Two of its conclusions matter enough to state here, because they define the problem I was given.

The first concerns cost. Most attacks compute a dedicated pattern for each image. That is fine for a single photograph and useless for a video, where thirty frames arrive every second. Attacks that compute one universal pattern, valid for all images, remove that cost entirely. They are the realistic threat against a detection pipeline, and they are the family studied here.

The second concerns visibility. A pattern that has to fool a model on every image cannot be as subtle as one tailored to a single image. It has to be stronger, and stronger means more visible. Published universal attacks reach high success rates, but the images they produce carry visible traces. An attack that can be seen is an attack that has failed, since a human reviewer will simply reject the file.

There is a third difficulty, and it is the one that makes the subject current. A real attacker does not own the dataset that trained the detector. Methods that need those images describe a scenario that does not happen. The methods that do not need them are the least mature, and improving one of them is where this internship began.

1.1.3 Objective and research question

The objective I was given was to design and evaluate universal adversarial perturbations able to fool deepfake detection models while remaining visually imperceptible.

Reduced to a single question:

Can a universal, data-free perturbation be made more effective and less visible at the same time, instead of trading one property against the other?

The words *at the same time* carry the whole difficulty. Turning the amplitude down improves invisibility and degrades effectiveness, always, in fixed proportion. Reporting that is reporting arithmetic. A contribution requires both measurements to move in the right direction at once, under a protocol that does not change between configurations.

I leave the question open here. Section 4.5 answers it, once the measurements that decide it have been presented.

1.1.4 Approach and contributions

I worked in four stages. I reviewed the literature to decide what to build on. I implemented and reproduced the chosen method, so that any later gain could be attributed to my own changes rather than to an implementation difference. I then developed two extensions, following the two directions my supervisors gave me. Finally I measured every configuration under one fixed protocol on eight target architectures, five of which were never used during training.

The contributions of this work are the following.

- A reproduction of a recent data-free universal attack, used as a reference point throughout.

- A new way of initialising the attack, using a diffusion model to produce structured starting images from pure noise, which improves the success rate without breaking the data-free constraint.
- A perceptual amplitude budget applied during training rather than after it, which improves visual quality and success rate together instead of exchanging one for the other.
- A documented set of failed directions, with the measurements that closed them.

Because the method had to reach a stable operating point before being moved to a new data domain, the experiments were run on the benchmark used by the reference method. Section 1.3.4 explains that decision and what follows from it.

1.1.5 Structure of this report

The present chapter continues with the laboratory that hosted me and the way the work was supervised (Section 1.2), then turns the objective above into a measurable specification (Section 1.3). Chapter 2 reviews the state of the art, from the first attacks to data-free universal methods, and examines how visual imperceptibility is measured. Chapter 3 describes how the project was run, reproduces the method I started from, and presents the three contributions with their measurements. Chapter 4 takes the method to a real deep-fake detector and to video, examines three families of countermeasure, and closes with the discussion, the failed directions, the conclusion and a personal assessment.

The next section describes where all of this took place.

1.2 Host organisation and internship framework

The previous section stated what the work is about. This one states where it was done, by whom it was supervised, and why this particular laboratory proposed this particular subject. That last question turns out to have a precise answer, and it explains several of the instructions I received.

1.2.1 The GREYC laboratory

I carried out my final-year internship at the GREYC, the *Groupe de Recherche en Informatique, Image et Instrumentation de Caen*. It is a digital science laboratory based in Caen, Normandy, and it has been a CNRS Joint Research Unit under the reference UMR 6072 since 1995. Three institutions supervise it jointly: the Université de Caen Normandie, the engineering school ENSICAEN, and the CNRS. At the time of writing, Jean-Luc Lamotte directs the laboratory on an interim basis, with Gaël Dias and Olivier Lézoray as deputy directors.

The laboratory brings together more than 230 members, around a hundred of them permanent researchers or faculty. Its work covers most of the disciplines of information

science and technology, organised around four orientations: data science from sensors to interpretation, digital security, mathematical computer science, and artificial intelligence.

What I noticed during my stay is that theory and application are not separated here. The same team that publishes on the security of biometric systems also maintains the software platform used to evaluate them. Several start-ups were founded by members of the laboratory, and industrial partnerships with companies such as Orange, NXP and Thales are routine rather than exceptional.

Research is divided into six teams, listed in Table 1.1.

Team	Research focus
AMACC	Mathematical computer science and models of computation
CODAG	Data mining, constraints and graphs
Électronique	Sensors and instrumentation, functional oxides, characterisation
Image	Signal, image and video processing and analysis
MAD	Autonomous agents and multi-agent systems
SAFE	Computer security, architecture, forensics and biometrics

Table 1.1: The six research teams of the GREYC laboratory. My internship took place within the SAFE team.

1.2.2 The SAFE team

SAFE stands for *Sécurité, Architecture, Forensique, biomÉtriE*. The team is led by Patrick Lacharme and works along three lines that are deliberately kept close to one another, so that theoretical results feed applications and applications raise new questions.

- **Biometrics.** Design and evaluation of biometric systems, and protection of biometric data. Christophe Rosenberger coordinates this line. It includes work on presentation attacks, that is, attempts to deceive a sensor with a fake fingerprint or a fake face.
- **Security architectures and models.** Network security for IoT, SDN and 5G, applied cryptography, randomness and information protection.
- **Digital investigation.** Natural language processing, forensic platforms and privacy protection. Emmanuel Giguet, one of my two supervisors, coordinates this line.

My subject belongs to the third line. It borrows heavily from the first. Asking whether an automatic detector can be deceived by a crafted signal is the same question as asking whether a biometric sensor can be deceived by a presentation attack, only the sensor has been replaced by a network. That proximity explains why the subject was proposed by this team rather than by an image processing team.

1.2.3 Research context: the COMPROMIS project

My internship is not an isolated exercise. It sits inside COMPROMIS, a project of the French PEPR Cybersecurity programme, in which the GREYC is a partner alongside CRISAL, LIRMM, EURECOM, INRIA Rennes, the CEA and MILES. The premise of that project is also the premise of my work. Multimedia data protection is probably the branch of cybersecurity that gained the most from deep learning. In adopting these tools, however, the field largely forgot to check how secure the tools themselves are. Artificial intelligence has become one of the weak links of the very systems it was brought in to strengthen. The project lists deep learning, forensic analysis, watermarking, deepfakes and steganography among its keywords.

Two things follow for this report. Attacking a detector is not an end in itself; the point is to measure how fragile a detection pipeline becomes once it rests on a convolutional network, so that the fragility can be documented and eventually corrected. And the instruction I received to borrow from image watermarking was not an arbitrary idea. Watermarking and adversarial perturbation solve the same problem from opposite sides, since both inject a signal into an image that a human must not notice, and both appear in the scope of this project.

1.2.4 Position and assignment

I joined the SAFE team as a research intern, under an agreement between the Université de Poitiers and the Université de Caen Normandie. The internship ran from 28 April to 30 September 2026, full time, thirty-five hours a week, on the Campus 2 site in Caen. This report covers the work done up to the end of August. The remaining weeks were used to consolidate the results and prepare the defence.

Three people supervised me.

- **Prof. Christophe Charrier**, professor at IUT Grand Ouest Normandie, Université de Caen Normandie, was my host supervisor and signed the agreement. He set the scientific direction and validated every change of course.
- **Dr. Emmanuel Giguet**, CNRS research fellow and coordinator of the forensics line, co-supervised the work and brought the forensic reading of the subject.
- **Prof. David Helbert**, Université de Poitiers, was my academic supervisor. He received my activity reports and kept the work aligned with what the degree expects, in particular on the project management side.

I was the only person working on this subject. That is normal in a research laboratory, but it has a consequence I had to take seriously. With nobody checking my results day to day, the evaluation protocol had to do that job instead. Section 3.1.2 explains the rules I ended up imposing on myself for that reason.

Coordination happened through weekly meetings with my supervisors, rescheduled when their availability required it. The pattern was always the same. I presented the

measurements obtained since the last meeting, we discussed what they meant, and we agreed on one priority for the following week. Between meetings I worked on my own and reported by email whenever a result arrived early or, more often, whenever a promising direction turned out to lead nowhere.

I also sent periodic activity reports to my academic supervisor. Five were produced during the internship, covering the immersion phase, the implementation phase, the first evaluation campaign, the work on diffusion priors and perceptual budgeting, and the transfer to a deepfake detector. They are the main source for Section 3.1.

1.2.5 Missions

The internship agreement lists four assigned activities.

1. Review recent work on universal adversarial attacks and on deepfake detectors.
2. Develop prototypes in Python using PyTorch.
3. Measure performance along three axes: fooling rate, visual perceptibility and robustness across models.
4. Apply the resulting perturbations to face swapping video to observe their temporal effect.

All four were carried out, and the rest of this report documents them. The fourth came last, and on purpose. A pattern has to reach a stable operating point on still images before it is worth applying to a stream, since a weak pattern would fail on video for reasons that have nothing to do with video. Section 4.2 reports that stage: the perturbation is added frame by frame, with no optimisation in between, and every sequence the detector had correctly identified flips to *real*.

1.2.6 Working environment

The laboratory gave me access to a shared multi-GPU server for all training and evaluation runs. The dataset and the pre-trained networks used in the experiments were already available locally.

Working on shared and access-restricted infrastructure imposed a few practical constraints on how experiments were organised. They are described, together with the other constraints of the project, in Section 3.1.5.

The internship therefore took place in a laboratory whose current concern is exactly the one this subject raises, namely the security of the artificial intelligence now used to protect multimedia data. I worked alone, with weekly supervision, inside a national project that already lists deepfakes and watermarking among its keywords. What remains is to turn the general objective of Section 1.1 into something measurable, and that is the purpose of the next section.

1.3 Problem statement and specifications

The two previous sections said what the subject is about and where the work was done. This one turns the general objective into something that can be measured, states the constraints that could not be negotiated, and explains one decision that shaped the whole internship.

1.3.1 Two non-negotiable constraints

Everything that follows respects two rules. They come from the subject itself, not from convenience, and every experiment reported in this document was checked against them.

The perturbation must be universal. One single pattern, computed once, is added to every image without any recomputation. Nothing may be adapted to the image being attacked. This is what makes the threat realistic on a video stream, and it is also what makes the problem hard, since a pattern that must work everywhere cannot be tailored anywhere.

The perturbation must be built without data. No image of the target domain enters the optimisation. No label is used either. An attacker in a real situation does not own the corpus that trained the detector, so any method that requires it describes a scenario that does not happen.

1.3.2 What has to be measured

An attack is judged on two quantities that move in opposite directions. The **fooling rate** counts how often the model changes its answer. The **structural similarity** between the clean and the attacked image says whether the change can be seen. Both are defined in Section 2.6, together with the alternatives I considered and did not retain.

The difficulty is that the second is easy to improve at the expense of the first. Reducing the amplitude of the perturbation raises visual quality and lowers the fooling rate, in fixed proportion. Reporting that exchange is reporting arithmetic, not a contribution.

The specification I set for myself is therefore stricter: **both quantities must improve at the same time**, compared with the method I started from, measured under conditions that do not change between configurations.

1.3.3 Acceptance criteria

A result was considered worth keeping when it satisfied all of the following.

- The mean fooling rate over the target models exceeds that of the reference method.
- The structural similarity exceeds that of the reference method.

- The gain holds on models that were never used during training, and not only on those the perturbation was optimised against.
- The measurement was obtained under the frozen protocol of Section 3.1.3, with the same images and the same random seed as every other configuration.

The third criterion matters more than it looks. A perturbation can always be made stronger on the network it was trained on; what determines whether it is a real threat is whether it still works on a network the attacker does not own.

1.3.4 A staged validation strategy

The final target is a deepfake detector. Most of the experiments in Chapter 3 are nevertheless run on a general-purpose image benchmark, and that choice has to be justified before anything else.

There are two reasons for it. Comparing my contributions with the published state of the art requires measuring them under the conditions of the published state of the art; a gain obtained on a different corpus, against different architectures, would be comparable to nothing. And transferring an unstable method to a new domain confuses two questions: if the attack fails on faces, is the method weak or was the transfer badly done?

I therefore proceeded in two stages, in agreement with my supervisors. The method was developed and validated on the reference benchmark, which occupies Chapter 3, and only then adapted to a real deepfake detector, which opens Chapter 4. Both stages respect the two constraints stated above.

1.3.5 Summary of the chapter

The problem is now stated in measurable terms: raise the fooling rate and the visual fidelity of a universal, data-free perturbation at the same time, and verify that the gain survives on unseen models.

Before building anything, I needed to know what the community had already built and where it stopped. That is the subject of the next chapter.

Chapter 2

State of the art

The previous chapter turned a general objective into a specification. This one asks a simpler question. What has the community already built, and where does it stop?

I follow the field in the order in which it actually developed. It starts with a discovery about neural networks that nobody expected. It moves on to attacks tailored to one image, then to attacks valid for all images, then to attacks that need no images at all. Each step removes a limitation and creates a new one. The chapter closes on the metrics used to decide whether an attack is invisible, since that is the second half of the problem and the half the literature treats least carefully.

2.1 A structural weakness of deep architectures

Deepfake detection rests on deep convolutional networks. Architectures such as Xception-Net or EfficientNet are trained on reference corpora and learn to recognise what generation algorithms leave behind, down to artefacts no human would notice (Tolosana et al., 2020). They are effective, and that effectiveness is why they were deployed.

Szegedy et al. (2014) discovered that these same networks have a counter-intuitive property. Take an image the network classifies correctly. Add a specific pattern, small enough that the modified image looks identical to the original. The network changes its answer, often with high confidence. The authors called these inputs adversarial examples, and the phenomenon has since been reproduced on every architecture family tested.

The reason matters for what follows. The added pattern is not random noise. It is computed to exploit the directions in which the network’s decision function is most sensitive. It acts as a form of semantic camouflage: the convolutional filters respond to it as if it carried information, and the class boundary is crossed. For a detection pipeline, the consequence is direct. A manipulated video can be presented as authentic without altering anything a human reviewer would see.

2.2 Threat model

Before comparing algorithms, the capabilities assumed of the attacker have to be stated. Two questions define the setting: what does the attacker know about the target, and how much is the attacker allowed to modify.

In the **white box** setting the attacker has full access to the target: its architecture, its weights, and therefore its gradients. This is the most favourable setting and the least realistic one. In the **black box** setting the attacker sees only the output. No gradient is available, and the standard workaround relies on *transferability*, a property first analysed by Papernot et al. (2016): a perturbation optimised on a locally owned surrogate often remains effective against an unknown target. Deep architectures trained on similar tasks learn similar internal representations, so an attack that exploits one tends to exploit the others. The distinction is not academic. A deepfake detector deployed by a platform is a black box, so any attack that only works in white box describes a threat that does not exist in practice.

The second question is how much the attacker may modify. An attack that changes the model’s answer but leaves a visible mark has failed, so the literature caps the perturbation with a norm. The L_∞ **norm** caps the change applied to each individual pixel: with $\epsilon = 10/255$, no colour channel may move by more than that amount. It is the native constraint of most gradient-based attacks and the one used throughout this work. The alternative, the L_2 norm, caps the total energy instead and forces a smooth, diffuse perturbation.

Both share the same limitation, and it becomes important later. They treat every pixel identically. A budget of $10/255$ is applied with the same value to a uniform sky and to a patch of foliage, even though the human eye reacts to those two regions in completely different ways. Section 2.6 returns to this point, and Section 3.3.4 builds a contribution on it.

2.3 Per-image attacks

The first generation of methods computes a dedicated perturbation for each input image. Goodfellow et al. (2015) proposed the simplest formulation, the Fast Gradient Sign Method, on the argument that neural networks behave too linearly for a single well-chosen step not to suffice:

$$x_{\text{adv}} = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \quad (2.1)$$

FGSM is instantaneous and crude: one giant step along a linearised approximation frequently misses the maximum of the error surface. Madry et al. (2018) replaced it by many small ones with Projected Gradient Descent, recomputing the gradient at each iteration and projecting back into the allowed ball whenever the perturbation leaves it. PGD is the reference white box attack under an L_∞ constraint, substantially more expensive and substantially more effective.

Two other methods complete this generation. Moosavi-Dezfooli et al. (2016) approached the problem geometrically with DeepFool, following at each step the shortest orthogonal distance to the nearest decision boundary; the result is the minimal perturbation that flips the verdict, which makes it the most discreet of the family and the most specific to its image. Papernot et al. (2016) took a different route with JSMA, using a Jacobian-derived saliency map to modify only the few most influential pixels.

All of these share a limitation that rules them out here. The perturbation is valid for one image only, so attacking a video at thirty frames per second means recomputing it thirty times per second. The second limitation is transferability: a perturbation tailored to one model and one image degrades quickly when either changes.

2.4 Universal adversarial perturbations

Moosavi-Dezfooli et al. (2017) removed the first limitation. Their proposal is to compute, once and offline, a single perturbation δ that is image-agnostic. Applied by simple addition, it fools the model on a large fraction of unseen images. The cost at attack time falls to zero, the pattern can be distributed, and the threat becomes realistic on a video stream.

Two construction strategies coexist in the literature. The original one accumulates incrementally: the algorithm sweeps the dataset, and for each image not yet fooled by the current δ , computes the small DeepFool step needed to flip it, adds it to δ and projects the result back into the constraint ball. This produces fine perturbations but converges slowly. The second strategy formulates the problem as stochastic optimisation over batches, computing at each step an averaged gradient representative of a whole group of images. It converges faster and is statistically more robust.

2.4.1 Vector cancellation and Fast-UAP

A practical obstacle limits both strategies. Individual updates frequently oppose one another. If fooling image A requires darkening a group of pixels while fooling image B requires brightening the same group, the two updates cancel. The perturbation stalls and loses strength.

Dai and Shu (2020) addressed this by constraining orientation. During accumulation, each new contribution is filtered so that it aligns with the current global perturbation, by maximising their cosine similarity. Contributions then add constructively instead of destroying one another. The authors report comparable perturbations with roughly half the data and half the computation.

2.4.2 Input transformations

Even a perturbation with a high fooling rate on its own surrogate can overfit that surrogate’s specific filters and fail on a different architecture. Xie et al. (2019) proposed to

counter this by applying random geometric transformations to the input during optimisation, such as random resizing and padding. Unable to rely on the exact position of a pixel, the algorithm is forced to attack more global structures, and transferability improves.

Neekhara et al. (2020) carried this idea into the deepfake setting. Their contribution is the demonstration, on real detectors, that low-intensity adversarial perturbations defeat them easily, and that universal perturbations are the more dangerous variant because they can be applied in real time or shared between users to bypass automated moderation.

2.5 The data-free setting

Universal perturbations remove the per-image cost, but the methods above still need training images. In an industrial security context, that assumption fails. An attacker does not own FaceForensics++, and does not own the corpus used to calibrate a proprietary detector. This is what motivates data-free universal attacks, which build δ from the internal behaviour of the model alone.

2.5.1 Maximising internal activations

Mopuri et al. (2017) introduced the founding idea with Fast Feature Fool. Since no labels are available, classification loss cannot be used. Instead the objective becomes the maximisation of the activations of the convolutional layers: a perturbation that saturates the internal filters corrupts the representation the network builds, and the final decision follows. Mopuri et al. (2018a) generalised this objective and made it applicable across tasks.

Later work refined the input given to this objective rather than the objective itself. Mopuri et al. (2018b) synthesised class impressions to stand in for real images. Liu et al. (2019) exploited prior-driven uncertainty. Zhang et al. (2021) observed that the logit distribution induced by a UAP tends to dominate that of the input image, which suggests that a universal perturbation behaves like a single image carrying a strong dominant label. Li et al. (2022) and Liu et al. (2023) pushed white-box performance further, the latter through truncated ratio maximisation.

A common weakness runs through this line of work. The prior fed to the algorithm is random, typically Gaussian noise or an artificial jigsaw pattern designed to mimic the statistics of natural images. Random noise carries no semantic content. The optimisation therefore has little to hold on to, and the resulting perturbation tends to overfit the surrogate model, which shows up as poor black-box transferability.

2.5.2 PSP-UAP, the reference method of this work

Lee et al. (2025) attack exactly that weakness, and their method is the starting point of my own work.

Their observation is that a trained UAP is not semantically empty. Crop different regions of one and a classifier assigns them different labels, with substantial confidence and distinct Grad-CAM heatmaps. The perturbation already looks like an image with content. From this they build a **pseudo-semantic prior** extracted from the perturbation itself, recursively, during training: random noise is added to the current perturbation, regions of that sum are cropped at random positions and resized to full scale, and those crops become the data the attack trains on. No real image is ever involved, so the data-free constraint holds, but the algorithm now optimises against something structured rather than against noise. Two mechanisms complete the method. Random input transformations, which had been of little use on pure noise, become effective once the samples carry structure. And a reweighting scheme gives more weight to the samples the attack is currently failing on.

The authors report an average white-box fooling rate of 89.95% over five ImageNet architectures, ahead of TRM-UAP at 86.39% and AT-UAP at 87.95%, and they report surpassing several data-dependent methods. Section 3.2 describes the algorithm in full, with its equations and its diagrams, since I reproduced it before modifying it.

2.6 Measuring effectiveness and invisibility

An attack is judged on two axes that pull in opposite directions. The literature is precise on the first and, in my view, careless on the second. Table 2.2 summarises the measures available; four of them deserve a comment.

The **fooling rate** is the proportion of images for which the perturbation changes the predicted label. It is the standard measure of effectiveness and the one used here. Its weakness is that it says nothing about visibility, so a method can improve it simply by using a larger budget.

On the visual side, PSNR compares pixels one by one through the mean squared error and has no notion of image content: a single strongly modified pixel ruins the score even though no observer would see it. Wang et al. (2004) designed the **structural similarity index** to correct this, comparing local luminance, contrast and structure on the assumption that the human visual system extracts structural information from a scene. It returns a value between 0 and 1, and it is the fidelity metric used throughout this report.

The **just noticeable difference** (Watson, 2001) is a psychophysical threshold: the smallest change a human observer can detect. It is measured on observers through detection experiments of the kind formalised by Green and Swets (1966), and it characterises the human visual system as a whole rather than any single organ. What makes it useful here is that the threshold is not uniform across an image. The visual system is highly sensitive to a change in a smooth region such as bare skin or a clear sky, and comparatively blind to the same change inside hair or foliage. This is a masking effect, and it is exactly the property that image watermarking exploits when it hides a signal where it will not be seen. Section 3.3.4 builds on it directly. Saliency modelling (Dalal et al., 2020) addresses the related question of where the eye looks rather than what it can detect.

No objective metric replaces an observer. The reference protocol is ITU-T Recommendation P.910 (International Telecommunication Union, 1999), in which a panel rates the modified content from 1 to 5, with a mean opinion score above 4.5 the usual threshold for claiming invisibility. This protocol was not run during the internship, for reasons of time; it is listed among the perspectives in Section 4.5.

2.7 Benchmarks and available implementations

Three corpora dominate deepfake detection research. FaceForensics++ (Tolosana et al., 2020) is the reference, with a thousand original video sequences manipulated by four different algorithms at several compression levels; access is subject to signing an institutional academic charter. Celeb-DF offers second-generation manipulations without the coarse visual artefacts of earlier datasets. DFDC provides the widest diversity of ethnicities, lighting conditions and backgrounds, and simulates the heterogeneity of real social media content.

For the attacks themselves, the ImageNet ILSVRC2012 validation set (Russakovsky et al., 2015) is the common benchmark of the universal perturbation literature, including every data-free method cited in Section 2.5. Reference implementations are widely available, either in libraries such as Torchattacks and IBM’s Adversarial Robustness Toolbox, or as author code released with the paper. PSP-UAP falls in the second category, which is what made the reproduction described in Section 3.2 possible.

2.8 Synthesis

Tables 2.1 and 2.2 summarise the chapter.

Method	Type	Setting	Norm	Core idea
FGSM (2015)	Per-image	White box	L_∞	One large step along the sign of the gradient.
PGD (2018)	Per-image	White box	L_∞	Small iterative steps with projection back into the ball.
DeepFool (2016)	Per-image	White box	L_2	Shortest path to the nearest decision boundary.
UAP (2017)	Universal	Data-dep.	L_∞	One image-agnostic pattern accumulated over a dataset.
Fast-UAP (2020)	Universal	Data-dep.	L_∞	Aligns contributions to stop them cancelling out.
Input diversity (2019)	Enhancement	Black box	L_∞	Random input transformations to prevent overfitting.
GD-UAP (2018a)	Universal	Data-free	L_∞	Maximises activations of the convolutional layers.
TRM-UAP (2023)	Universal	Data-free	L_∞	Truncated ratio maximisation over layer responses.
PSP-UAP (2025)	Universal	Data-free	L_∞	Extracts a semantic prior from the perturbation itself.

Table 2.1: Attack methods discussed in this chapter.

Metric	What it evaluates	Principle
Fooling rate	The model	Proportion of images whose predicted label changes.
PSNR	Raw pixel error	Energy difference between clean and attacked image, in dB.
SSIM (2004)	Structure	Local luminance, contrast and structure similarity.
JND (2001)	Human visual system	Detection threshold; not uniform, higher in textured regions.
Saliency (2020)	Visual attention	Models where the observer looks rather than what can be seen.
MOS (1999)	Human judgement	Panel rating from 1 to 5 under ITU-T P.910.

Table 2.2: Metrics used to evaluate universal adversarial perturbations.

2.9 Earlier work within the team

Before my arrival, Hartmann and Montier (2025) had carried out a semester project in the SAFE team on the same broad question, supervised by the same researchers. Their report was given to me at the start of the internship, and reading it was my entry point into the subject.

They implemented FGSM, PGD, DeepFool and UAP variants, and applied them to an XceptionNet-based detector fine-tuned on FaceForensics++. Their strongest result is a UAP built on PGD reaching a fooling rate of 97.24% with a mean SSIM of 0.9644, evaluated on 1632 images. They also verified transferability by attacking a public online

detector, where the PGD-based perturbation succeeded and the DeepFool-based one did not.

Their experimental setting is not the one used in this report, and the two sets of figures answer different questions. Theirs is a single binary detector, attacked in white box, with the perturbation computed from real images of the target domain. Mine is a thousand-class problem, evaluated across eight architectures including five the perturbation never saw, and built without a single real image. What their work established, and what I took from it, is that the detectors are genuinely vulnerable and that the universal formulation transfers better than the per-image one.

2.10 Where the literature stops

Three observations close this chapter, and together they define what I set out to do.

Data-free universal attacks have converged on activation maximisation as an objective, and their remaining weakness is the input they are given. PSP-UAP established that enriching the prior improves transferability. It extracts that enrichment from the perturbation itself, which is elegant but bounded: the algorithm can only recover the structure it has already produced.

Visual quality is, in this literature, an afterthought. It is enforced through a uniform L_∞ budget and reported, when it is reported at all, as a single number. Yet the perceptual literature has known for decades that the detection threshold varies across an image. No data-free universal method I found exploits that variation during training.

Finally, the two objectives are treated as a trade-off. Lower the budget and the attack becomes less visible and less effective, in fixed proportion. Whether both can improve together is a question the literature leaves open.

The next chapter describes the method I built to answer it.

Chapter 3

Approach and proposed method

3.1 Project management

This section describes how the work was organised. It is short on purpose, but it is not a formality: the evaluation protocol described in Section 3.1.3 is what makes every number in the following pages comparable, and the working rules of Section 3.1.2 are what allowed me to attribute each gain to the right cause.

3.1.1 An iterative approach, and why not a V-model

I worked in weekly cycles, aligned with the supervision meetings described in Section 1.2.4.

That choice follows from the nature of the subject. A V-model assumes that the expected result can be specified upfront, then built, then validated. In research that assumption does not hold, because nobody knows in advance which direction will work. What can be planned is not a result but a question, and the point of iterating is to answer it quickly enough to change course if the answer is no.

Each cycle followed the same four steps.

1. **Objective.** I state a precise hypothesis and the figure that will decide whether it is true or false.
2. **Implementation.** I build the variant, with a quick sanity check before any long training run.
3. **Measurement.** I evaluate with one protocol, always the same.
4. **Decision.** I keep, abandon or redirect, and I write down why.

The weekly meeting served as the synchronisation point. I presented the measurements of the cycle just ended, and we agreed on the priority for the next one. Directions that were not retained went into a list and could be picked up later; one of them, the frequency-domain budget, was reopened twice on that basis.

3.1.2 Three working rules

I was the only person on this subject, so no colleague was going to notice a mistake in my measurements. I compensated with three rules, applied without exception.

One change at a time. I never test two modifications together. If two are introduced at once and the result improves, there is no way of knowing which one is responsible, or whether one of them is in fact harmful and simply hidden by the other. This rule is what allows me to state separately how much the diffusion prior contributes and how much the perceptual budget contributes.

A working version is never modified. Every variant is saved as a new file, and a result is kept only if the measurement improves on the previous one. Nothing that already works is edited in place, so a failed experiment can never destroy an earlier result.

A single evaluation protocol. All configurations are measured in exactly the same conditions. A figure obtained in any other way is discarded rather than compared, because comparing two numbers produced differently means comparing the protocols instead of the methods.

3.1.3 The evaluation protocol

Every figure reported in this document, unless stated otherwise, comes from the following protocol.

- **Data.** 1000 images of the ImageNet ILSVRC2012 validation set for the main pipeline; 1998 faces extracted from FaceForensics++ for the deepfake application.
- **Seed.** Fixed at 126 for every run, so that the sampling of the validation subset and the random draws of the training loop are reproducible.
- **Targets.** Eight architectures for the main pipeline. Three of them are the surrogates and five were never seen during training.
- **Metrics.** Fooling rate as the proportion of images whose predicted label changes, and structural similarity between the clean and attacked image.

These settings were fixed once and for all in early July, after I discovered that two of my earlier evaluations had used different image samples and were therefore not comparable. Every measurement taken before that date was recomputed with the settings above, so that all the figures in this report come from the same conditions.

3.1.4 Deliverables and schedule

Figure 3.1 gives the actual schedule of the five phases.

Deliverable	Content	Status
Ensemble training module	Perturbation trained on three surrogates	Delivered
Diffusion prior module	Offline prior generation and refinement	Delivered
Perceptual budget module	Training under the JND mask	Delivered
Deepfake attack pipeline	Adaptation to the binary detector	Delivered
Video attack pipeline	Application to sequences, majority verdict	Delivered
Countermeasure modules	Filtering, adversarial training, perturbation detector	Delivered
Evaluation protocol	Eight target models, visual quality	Delivered
Result set	All configurations, identical protocol (CSV)	Delivered
Report figures	Analysis figures	Delivered
Internship report	This document	31 August

Table 3.1: Deliverables produced during the internship.

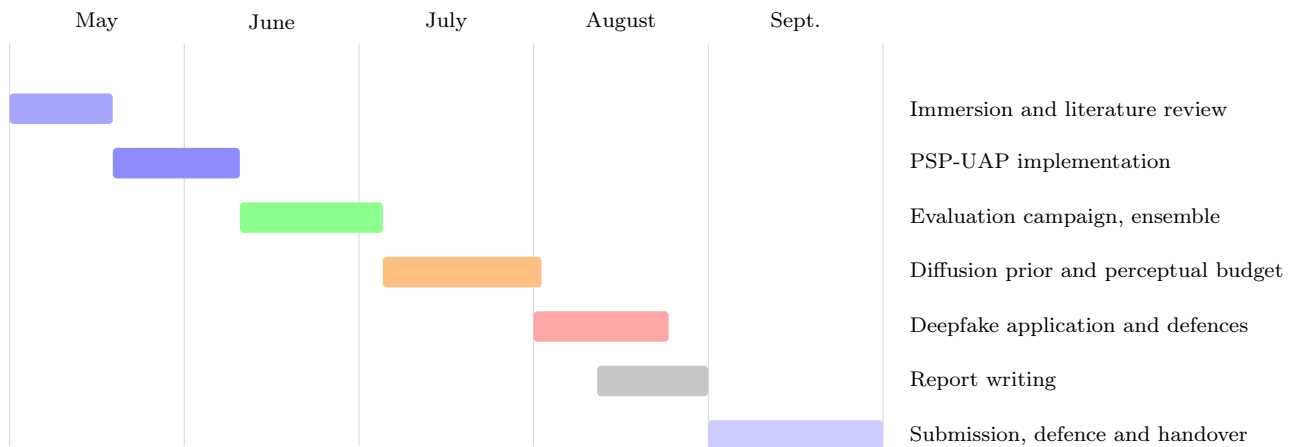


Figure 3.1: Actual schedule of the internship. Writing was deliberately started before the experimental work ended, so that figures and tables were produced as the measurements came in.

3.1.5 Constraints

Three constraints shaped the way experiments were run.

Shared and offline computing. The GPU server is shared with other members of the laboratory and has no internet access. Nothing can be installed while a job runs, so every dependency, network checkpoint and generative model had to be transferred beforehand and loaded from local files. A missing file is not a two-minute problem, it is a half-day problem, which is why an environment check preceded every launch. Long runs had to survive disconnections and leave memory available for other users.

Access to reference data. FaceForensics++ is not distributed openly. The download script is obtained by filling in a form on the authors’ repository and accepting their terms of use. The request was submitted as soon as the deepfake phase was scheduled, and other work continued while it was processed, so no time was lost.

Cost of a measurement. A full training run takes several hours and a complete evaluation across eight models takes about half an hour. Exploring a two-parameter space exhaustively was therefore not affordable. Pilot runs on 200 images, and parallel execution of independent configurations on different cards, were the two levers used to keep the sweep of Section 3.3.4 within the available time.

3.1.6 Risk management

Risks were listed in the first weeks and reviewed at every cycle. I kept in the register only events that could genuinely have happened and would have had a real consequence on the work.

Each entry is rated on two three-point scales, and the two ratings are read together. **Probability** is 1 if the event was unlikely, 2 if plausible, 3 if likely given how the work was organised. **Impact** is 1 if the consequence would have been a lost afternoon, 2 a lost week, 3 a lost result or a conclusion I could no longer defend. **Criticality** is simply the product of the two, so it runs from 1 to 9. Anything reaching 6 received a written countermeasure before the corresponding work started; below that, the risk was monitored but not acted upon.

Two entries of this register turned into actual incidents. Each one is described below, together with what it cost and how it was resolved.

Outage of the compute server. The server was unavailable for four days in early August. Because every configuration was already saved as a separate file and long runs wrote intermediate checkpoints, no result was lost. The jobs in progress were moved to another compute node and resumed from their last checkpoint.

Transfer to a binary decision. The whole pipeline had been calibrated on a thousand-class problem, and nothing guaranteed that it would work on a true-or-false decision. It did not, at first. Activation maximisation alone plateaued around 50 %, because saturating the internal representation pushes every image to the same side without ever aiming at the right one. Adding an explicit fooling term to the objective resolved it; Section 4.1.2 describes that adaptation and the parameter it introduces.

Two others deserve a word because they were anticipated rather than suffered. The laboratory closes in August, so remote access to the compute server was configured and tested in July, before it was needed. And access to FaceForensics++ is granted on request, so the request was submitted as soon as the deepfake phase appeared in the schedule, and the work on the benchmark continued while it was being processed.

Risk	Prob. (1–3)	Imp. (1–3)	Crit. $= P \times I$	Mitigation applied	Status
Delay in obtaining access to FaceForensics++	2	2	4	Access requested as soon as the deepfake phase was scheduled	Controlled
No access to the compute server during the August laboratory closure	3	3	9	Remote access configured and tested before the closure	Controlled
Outage of the compute server	1	2	2	Jobs moved to another compute node	Occurred
The reference method does not reproduce	2	3	6	Reproduction carried out first and validated against the published values before any modification	Controlled
A long training run is interrupted	3	1	3	Detached execution and intermediate checkpoints	Controlled
The deepfake application is not reached within the time available	2	3	6	A stopping criterion on the ImageNet phase agreed in advance with the supervisors	Controlled
The report is not validated in time	2	2	4	Chapters sent for review as they were written, rather than one complete document at the end	Controlled
The attack does not transfer from a thousand-class problem to a binary detector	3	2	6	Objective adapted with an explicit fooling term	Occurred

Table 3.2: Risk register kept during the internship. Probability and impact are rated from 1, low, to 3, high; criticality is their product.

3.1.7 Summary

The organisation described here is not separate from the scientific work; it produced it. The frozen protocol is what makes the four rows of Table 3.6 comparable. The one-variable rule is what allows the diffusion prior and the perceptual budget to be credited separately. And the numerical abandonment criterion is what limited the cost of the failed directions reported in Section 4.4.1.

The next section presents the method built under this organisation.

3.2 The reference method and its reproduction

Everything in this report is built on one existing method, PSP-UAP (Lee et al., 2025). Before changing anything, I had to understand it precisely and reproduce its results on my own installation. This section does both. It explains how the algorithm works, then

reports what I measured when I ran it, and ends with the three limitations that motivated my own contributions.

The order matters. A gain measured against a baseline I had not reproduced myself would prove nothing, since it could come from a difference in setup rather than from my changes.

3.2.1 The problem the method solves

Data-free universal attacks need something to optimise against. Since no real image is allowed, earlier methods feed the network random noise, either Gaussian or an artificial jigsaw pattern meant to imitate the statistics of natural images. The perturbation is added to that noise, the mixture is pushed through the surrogate network, and the internal activations are maximised.

The weakness is visible in the left half of Figure 3.2. Pure noise carries no semantic content. The convolutional filters have nothing meaningful to respond to, so the optimisation ends up fitting the particular quirks of the surrogate rather than anything a different architecture would share. The result is a perturbation that works well on the model it was trained on and much less well anywhere else.

The insight of Lee et al. (2025) is on the right half of the same figure. A trained UAP is not empty. Crop different regions of it, and a classifier assigns them different labels with substantial confidence: one region reads as *brain coral*, another as *puck*, another as *slot*. The perturbation already contains textures and shapes. So instead of looking for semantic content outside the algorithm, the method extracts it from the perturbation itself.

PSP-UAP Algorithm

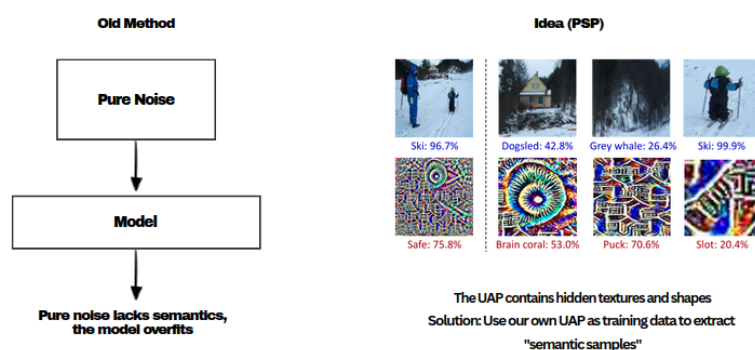


Figure 3.2: The limitation of earlier data-free methods, and the observation PSP-UAP builds on. Left: pure noise gives the optimisation nothing semantic to work with, and the perturbation overfits the surrogate. Right: regions cropped from a trained UAP are classified under diverse labels, which shows that the perturbation itself carries exploitable structure.

3.2.2 The pseudo-semantic prior

The construction is recursive, and Figure 3.3 shows one iteration of it.

At iteration t , a random noise tensor z is drawn. It is added to the current perturbation δ_t , and the sum forms what the authors call the pseudo-semantic prior:

$$p_{\mathbf{x}} = \{z + \delta_t \mid z \sim p_z\} \quad (3.1)$$

A sampler $\mathcal{C}$ then draws N *semantic samples* from this prior. Each sample is obtained by cropping a region of the prior at a random position and resizing it back to the full input resolution:

$$\{x_1, x_2, \dots, x_N\} = \mathcal{C}(\mathbf{x}; N) \quad (3.2)$$

These N samples play the role that real training images play in a data-dependent method. They are the data the attack learns on, and none of them ever came from a photograph.

One detail on the figure deserves attention. The arrow feeding δ_t into the prior is marked *Sg*, for stop gradient. The current perturbation is detached when the prior is built, so no gradient flows back through the construction of the samples. Only the copy of δ_t that is added to the samples afterwards receives gradient. Without that detachment the perturbation would be optimising against a target that moves with it, and the training would not be stable.

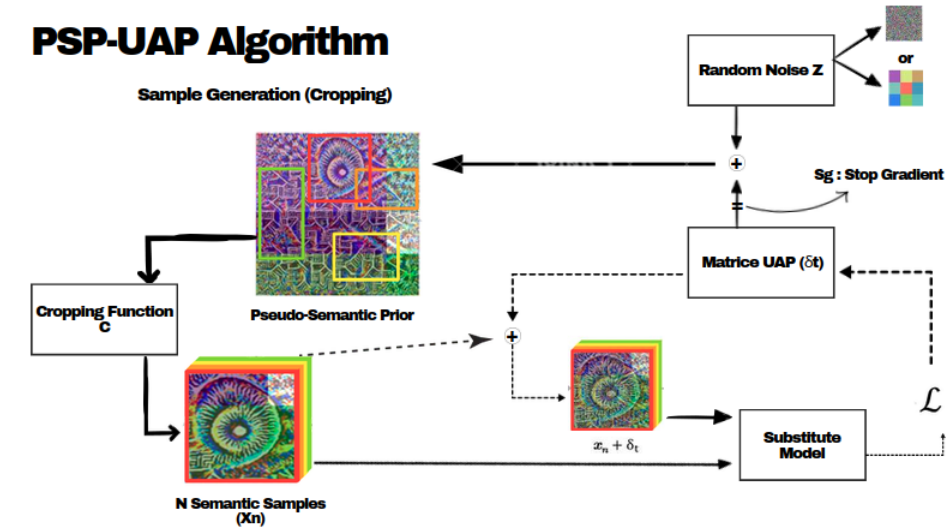


Figure 3.3: Generation of the semantic samples. Random noise is added to the current perturbation to form the pseudo-semantic prior. The sampler crops regions at random positions and resizes them to full scale, producing N semantic samples x_n . The perturbation is then added to each sample before it enters the surrogate model.

3.2.3 Input transformations

Random geometric transformations are a standard way of improving transferability (Xie et al., 2019). They had been of little use in the data-free setting, for the same reason as before: transforming pure noise produces more pure noise.

Once the samples carry structure, the technique becomes useful again. Following Zhu et al. (2024), one transformation is drawn at random for each sample among rotation by a small angle, scaling by a random factor, and shuffling, which rearranges the image into $m \times m$ blocks and permutes them. The transformation T is applied to the sample before it reaches the network, which prevents the perturbation from relying on the exact position of any structure.

3.2.4 Sample reweighting

Random cropping and random transformation produce samples of very uneven difficulty. Some are flipped by the current perturbation without effort; others resist. Treating them equally wastes most of the update on samples that are already handled.

Figure 3.4 shows how the method measures that difficulty. Each semantic sample is sent through the surrogate twice: once clean, giving the output distribution $P(x_n)$, and once with the perturbation added, giving $Q(x_n)$. Both branches go through the same input transformation, so the comparison is fair. The weight assigned to the sample is the inverse of the Kullback-Leibler divergence between the two distributions:

$$P(x_n) = f(T(x_n)), \quad Q(x_n) = f(T(x_n + \delta_t)), \quad w_n = D_{\text{KL}}(P(x_n) \| Q(x_n))^{-1} \quad (3.3)$$

The logic reads directly off the formula. A large divergence means the perturbation already changed that sample’s output considerably, so the sample is easy and its weight is small. A small divergence means the attack barely moved it, so the sample is hard and its weight is large. The optimisation is pushed towards the samples it is currently failing on.

3.2.5 The objective function

The three components come together in a single loss, shown in Figure 3.5. As in every data-free method since Mopuri et al. (2017), there are no labels, so classification loss is unavailable. What is maximised instead is the magnitude of the internal activations. A perturbation that saturates the convolutional filters corrupts the representation the network builds, and the decision follows.

$$\mathcal{L} = -\mathbb{E}_{\mathbf{x} \sim p_{\mathbf{x}}} \sum_{n=1}^N \sum_{i=1}^l \log \left(w_n \left\| \mathcal{A}_i^f(T(x_n + \delta_t)) \right\|_2 \right) \quad (3.4)$$

Here $\mathcal{A}_i^f$ denotes the activation of the i -th convolutional layer of the surrogate f , l the

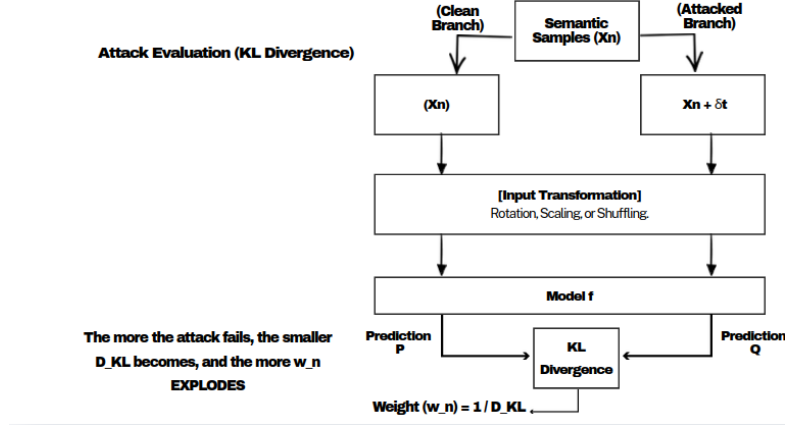


Figure 3.4: Computation of the sample weights. Each semantic sample follows a clean branch and an attacked branch through the same transformation and the same surrogate. The Kullback-Leibler divergence between the two output distributions measures how much the perturbation affected that sample, and the weight is its inverse.

number of layers summed over, x_n the n -th semantic sample, T the transformation drawn for it, and w_n its weight from Equation 3.3. The perturbation is updated by gradient ascent on this quantity and clipped after each step so that $\|\delta\|_\infty \leq \epsilon$.

PSP-UAP Algorithm

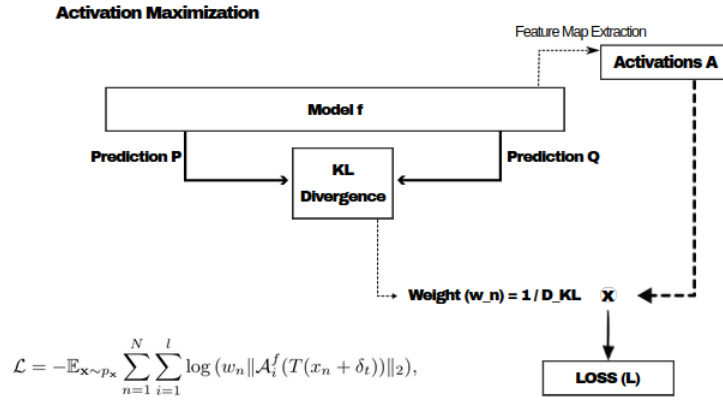


Figure 3.5: The complete objective. Feature maps are extracted from the convolutional layers of the surrogate and their norms are summed, weighted by the coefficients obtained from the Kullback-Leibler branch.

3.2.6 The algorithm in one pass

Putting the pieces in order, one iteration proceeds as follows.

1. Draw random noise z and form the pseudo-semantic prior $z + \delta_t$, with δ_t detached.
2. Sample N semantic samples by random crop and resize.

3. Draw one input transformation per sample among rotation, scaling and shuffling.
4. Compute the weight of each sample from the Kullback-Leibler divergence between its clean and attacked outputs.
5. Compute the weighted sum of activation norms and its gradient.
6. Update δ by one ascent step and clip it to the budget ϵ .

The perturbation is initialised uniformly in $[-\epsilon, \epsilon]$. Every H iterations the current fooling rate is measured on a validation subset, and training stops when it has not improved for a fixed number of such checks.

3.2.7 Reproduction

I reproduced the method from the authors' public implementation, adapted so that all networks and datasets load from local files. The evaluation used the ImageNet ILSVRC2012 validation set (Russakovsky et al., 2015).

Table 3.3 gives the white-box fooling rates I obtained with a single surrogate, that is, the case where the attacked model is the model the perturbation was trained on.

Surrogate = target	Fooling rate
VGG16	95.00 %
ResNet50	75.00 %
ResNet152	77.61 %
DenseNet121	79.00 %

Table 3.3: White-box fooling rate of the reproduced PSP-UAP, one surrogate at a time.

Transferability is the more informative measurement, since it corresponds to the realistic black-box scenario. Table 3.4 reports it.

Trained on	VGG16	ResNet50	DenseNet121
VGG16	<i>95.00</i>	73.00	63.00
ResNet50	77.00	<i>75.00</i>	66.00
DenseNet121	82.00	52.00	<i>79.00</i>

Table 3.4: Transferability of the reproduced perturbation, in fooling rate (%). Italics mark the white-box cells.

3.2.8 What the baseline revealed

Three observations came out of this reproduction, and each one led to a contribution.

The choice of surrogate dominates the result. A perturbation trained on VGG16 reaches 95 % on VGG16 and loses more than thirty points on DenseNet121. One trained on DenseNet121 behaves in a completely different way: 82 % on VGG16, 79 % on its own architecture, but only 52 % on ResNet50. Each surrogate has its own profile of strengths and weaknesses, and the spread across a single row of Table 3.4 is wider than the difference between two published methods. Committing to a single architecture therefore means committing to its particular biases. Section 3.3.2 addresses this by training on several surrogates at once.

The prior is bounded by what the algorithm has already produced. The pseudo-semantic prior is elegant, but it is recursive: the structure it supplies is the structure the perturbation has already generated. It enriches the optimisation without ever introducing anything the algorithm did not already contain. Section 3.3.3 replaces it with structure produced from outside the loop, while keeping the data-free constraint intact.

The attack is visible. This is the limitation the published work says the least about. Across my runs the structural similarity between clean and attacked images settled around 0.74. Figure 3.6 shows what that number means in practice: repetitive patterns and colour shifts that any reviewer would notice. An attack that can be seen is an attack that fails, whatever its fooling rate. Section 3.3.4 deals with this.



Figure 3.6: Visual effect of the reproduced perturbation on three validation images, before and after. The mathematical patterns of the attack remain plainly visible, for a structural similarity of about 0.74.

These three limitations map one-to-one onto the three contributions of the next section, and each one is measured separately under the protocol of Section 3.1.3.

3.3 Proposed method

Section 3.2 ended on three limitations of the reference method. The surrogate model dominates the outcome, the prior can only return structure the algorithm has already produced, and the perturbation is plainly visible. This section presents what I built to address each of them, and what each one changed.

3.3.1 A staged pipeline

One point has to be made before anything else, because it conditions every number in this section and it is the kind of detail that is usually left out of a report.

The three contributions were not combined into a single training run. They form a chain. The ensemble is trained from scratch, from a perturbation initialised uniformly in $[-\epsilon, \epsilon]$, and runs to convergence. The resulting perturbation is then loaded as the starting point of the diffusion stage, which refines it for a small number of iterations. That output in turn becomes the starting point of the perceptual stage. Figure 3.7 shows the chain.

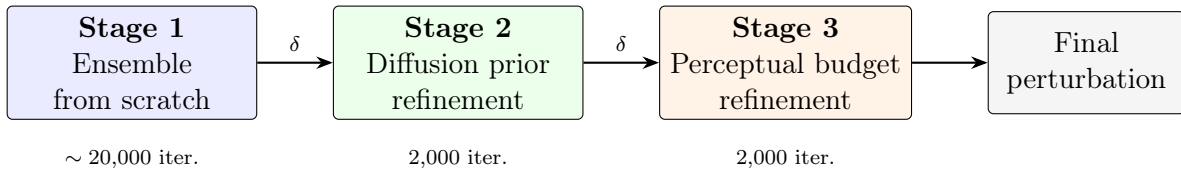


Figure 3.7: The three contributions are applied sequentially. Each stage warm-starts from the perturbation produced by the previous one and refines it over a short run.

This ordering was not chosen for convenience. Warm-starting a refinement from a weak perturbation stalls: the short run does not have enough iterations to build structure from nothing, so the seed has to be a perturbation that already works. The consequence is worth stating plainly. Someone who merged the three mechanisms into one script and trained once would not reproduce the numbers reported here. They would get less. The gain comes partly from the mechanisms and partly from the order in which they are applied.

All the figures in this section come from the protocol of Section 3.1.3: 1000 images of the ImageNet validation set, a fixed seed, and eight target models. Three of them, ResNet152, ResNet50 and DenseNet121, are the surrogates. The other five, AlexNet, VGG16, VGG19, GoogLeNet and EfficientNet-B0, were never seen during training and constitute the black-box measurement.

3.3.2 Contribution 1: an ensemble of surrogate models

Why one surrogate is not enough

The transferability matrix of Table 3.4 makes the problem concrete. A perturbation trained on VGG16 reaches its own target almost perfectly and loses more than thirty points on a densely connected architecture. One trained on DenseNet121 transfers well to VGG16 and collapses on ResNet50. No single surrogate produces a perturbation that behaves consistently across the panel.

The cause is in the objective. It maximises the activations of *one* network, so the perturbation converges towards the shapes and frequencies that saturate the filters of that particular architecture. A ResNet, with its residual connections, and a DenseNet, with its dense concatenations, do not react to the same patterns. Optimising against one means specialising against it.

I therefore require the perturbation to saturate three architectures at once: ResNet152, ResNet50 and DenseNet121. A pattern that excites all three cannot depend on a peculiarity of any of them.

Balancing the contributions

Adding the three losses does not work, and the reason cost me a first round of experiments. The networks do not contribute comparably. Each model uses only a fraction p_m of its convolutional layers in the activation sum, and the resulting layer counts are far apart: DenseNet121 contributes about a hundred terms against roughly forty for ResNet50. A raw sum lets the deepest network dictate the gradient, and the result is close to training on DenseNet121 alone.

The aggregation is therefore normalised by the number of layers each model actually uses:

$$\mathcal{L}_{\text{ens}} = \frac{1}{M} \sum_{m=1}^M \frac{\mathcal{L}_m}{L_m}, \quad L_m = \lfloor p_m \cdot n_m \rfloor \quad (3.5)$$

where n_m is the number of convolutional layers of model m and p_m its layer ratio. The values used are $p = 0.65$ for ResNet152, 0.7 for ResNet50 and 0.9 for DenseNet121, with softmax temperatures of 3, 3 and 10 respectively, following the hyperparameter study of Lee et al. (2025). After normalisation each model contributes roughly fifteen units to the objective, so none dominates.

In the two refinement stages the same balancing is achieved differently, by normalising each model’s gradient before accumulation rather than its loss:

$$\nabla_{\delta} \mathcal{L}_{\text{ens}} = \sum_{m=1}^M \frac{\nabla_{\delta} \mathcal{L}_m}{\|\nabla_{\delta} \mathcal{L}_m\| + \varepsilon} \quad (3.6)$$

Both formulations serve the same purpose. Equation 3.5 is cheaper, since it needs one backward pass, and it is used for the long run from scratch. Equation 3.6 equalises the contributions exactly rather than approximately, at the cost of one backward pass per model, which is affordable over two thousand iterations.

Result

Trained with the jigsaw prior of the reference implementation, a grid of randomly coloured blocks redrawn at each iteration, under a uniform budget of $\epsilon = 10/255$, the ensemble perturbation reaches a mean fooling rate of 77.14 % over the eight evaluation models, for a structural similarity of 0.7303. Its detailed scores appear in the first row of Table 3.6. This configuration is the reference against which the next two contributions are measured.

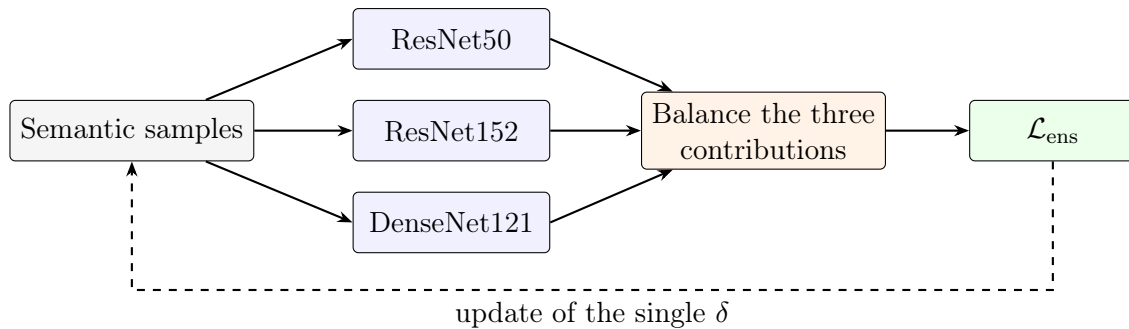


Figure 3.8: Ensemble of surrogates. The same semantic samples pass through three architectures. The three contributions are rebalanced before aggregation, so that the deepest network does not dictate the gradient, and a single shared perturbation is updated.

3.3.3 Contribution 2: a diffusion-generated prior

The limit of a recursive prior

The pseudo-semantic prior draws its structure from the perturbation itself. That is what makes it data-free, and it is also its ceiling. At the first iteration the perturbation is uniform noise, so the prior is noise. Structure only appears as the perturbation develops it, and the prior can never contain anything the algorithm has not already produced.

The alternatives offered in the published implementation confirm the difficulty. Beside Gaussian noise there is a jigsaw prior, an image cut into rectangles at random positions, each filled with a random colour and then smoothed by a median filter. Both are attempts to manufacture the statistics of a natural image without owning one, and Figure 3.9 shows how far both remain from what an image looks like.

Generating structure from pure noise

A diffusion model turns noise into an image. Run unconditionally, with an empty prompt and no reference image, it produces an output that has texture, contours and depth and that depicts nothing in particular.

I use Stable Diffusion v1.4 in exactly this way. A latent tensor is drawn from a standard normal distribution, the denoising loop is run for fifteen steps with an empty text embedding, the latent is decoded by the variational autoencoder and the result is resized to the network input resolution. A pool of sixty-four such images is generated once, cached, and sampled from at each training iteration. These images replace the Gaussian and jigsaw priors. Nothing else in the pipeline changes.

The data-free constraint holds, and it is worth being precise about why, since this is the first question a reader will ask. No image of the target domain is used, no image of the evaluation set is used, and no label is used. The diffusion model is a generator of plausible visual structure, not a source of data.

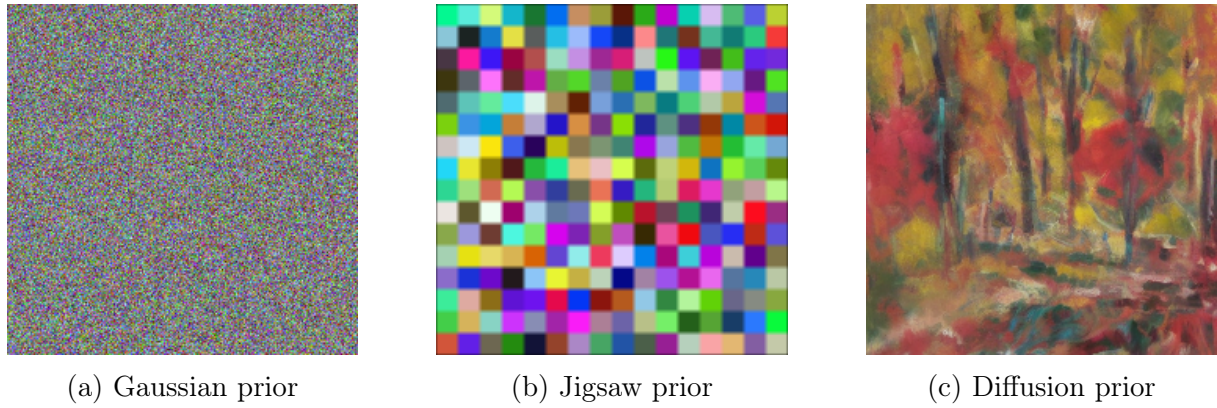


Figure 3.9: The three priors compared. The Gaussian prior carries no structure. The jigsaw prior adds coloured blocks but no scene. The third was produced by an unconditional diffusion model from pure noise; it depicts no real scene and no real object, but it has texture, contours and depth.

What changes for the algorithm

The construction of the prior is unchanged in form. The code still computes $z + \delta_t$, crops a region at random and resizes it. What changes is the role of each term, and the distinction matters.

With a Gaussian prior, z contributes nothing. All the semantic content of the samples has to be produced by δ itself, which is what makes the original method recursive. With a diffusion prior, z supplies the structure and δ returns to being what it should be, the adversarial component embedded in a plausible scene. The crops then yield samples that resemble fragments of real images. This is precisely the condition under which the input transformations and the sample reweighting of the reference method were designed to work.

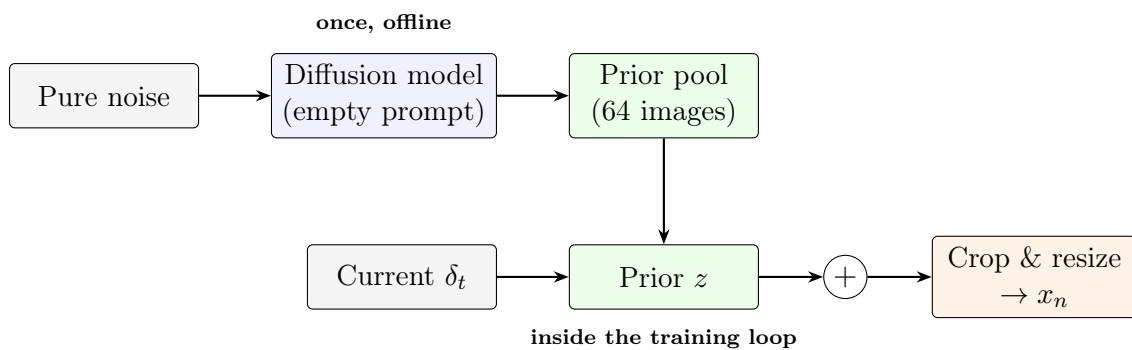


Figure 3.10: Replacing the random prior. The diffusion model runs once, offline, from pure noise and without conditioning. The rest of the pipeline, including the crop-and-resize sampling, is untouched.

Result, and what it does not do

The diffusion prior raises the mean fooling rate from 77.14 % to 78.42 % and the structural similarity from 0.7303 to 0.7443. The second row of Table 3.6 gives the detail, and that detail matters, because the two headline averages describe the contribution badly.

Seven of the eight models gain or hold. The table below gives the change model by model.

	AN	V16	V19	R50	R152	D121	GN	EN
Ensemble	57.4	87.5	86.9	78.9	78.8	79.6	76.8	71.2
+ diffusion	53.2	89.5	87.0	87.6	82.1	79.8	76.8	71.4
Change	−4.2	+2.0	+0.1	+ 8.7	+3.3	+0.2	0.0	+0.2

Table 3.5: Effect of the diffusion prior, model by model, in points of fooling rate. Seven of the eight models gain or hold; AlexNet is the only one that loses ground.

The gain is concentrated on the three surrogates. ResNet50 moves from 78.9 % to 87.6 % and ResNet152 from 78.8 % to 82.1 %, so the surrogate mean rises by four points, from 79.10 % to 83.17 %. On the five models the perturbation never saw, the effect is small: VGG16 gains two points and the other four move by less than half a point in either direction.

AlexNet is the exception, and it loses four points. Since the panel average weights all eight models equally, that single drop is enough to hold the black-box mean still, 75.96 % against 75.58 %. The reason is architectural. AlexNet is the only network of the panel with neither residual connections nor dense concatenations, and by far the shallowest. It responds to spatial frequencies the three surrogates do not use, so a prior that sharpens the attack on modern architectures has no particular reason to help there.

What this stage produces, then, is margin on the surrogates rather than reach on new architectures. That margin is not wasted. Section 3.3.4 spends it: the perceptual stage starts from this perturbation, and the last row of Table 3.6 shows the black-box mean rising from 75.58 % to 80.40 % once it has been spent.

Figure 3.11 shows a second effect, this one visual. Both patterns carry colour, but not at the same scale. The baseline mixes its colours over a few pixels, so the eye averages them into grey. The diffusion prior produces broad coloured areas that stay visible. Values are normalised for display, so the figure exaggerates both.

3.3.4 Contribution 3: a perceptual amplitude budget

The flaw is in the constraint, not in the optimisation

Everything above raises the fooling rate and leaves the structural similarity at 0.74. The attack works and it is visible, which as argued in Section 1.3 means it does not work.

The cause is the constraint itself. An L_∞ budget applies the same ceiling to every pixel. A change of 10/255 is authorised in a uniform sky and in a patch of foliage alike.

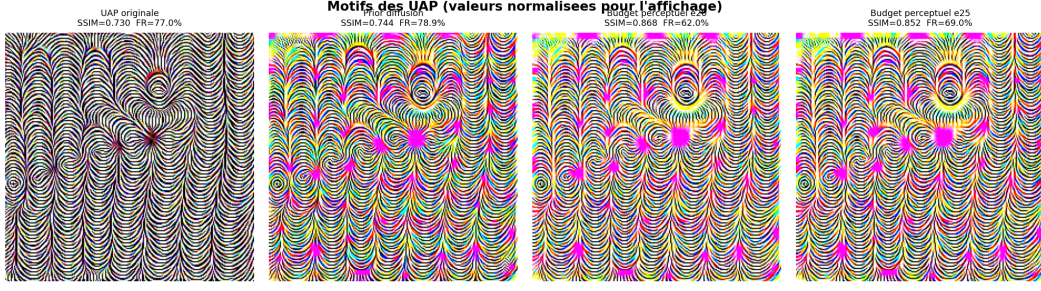


Figure 3.11: Perturbation patterns, with values normalised for display. From left to right: ensemble baseline, diffusion prior, and two perceptual-budget configurations.

The human visual system does not treat those regions alike, as Section 2.6 recalled: the detection threshold is low in smooth areas and much higher inside textured ones, through a masking effect.

A uniform budget is therefore wrong twice. It is too generous where the eye is sensitive, which is what makes the attack visible. And it is too restrictive where the eye is blind, which wastes power available for free.

The mask

This is the problem image watermarking solves, and the direction my supervisors pointed me to. A watermark is hidden where it will not be seen rather than weakened everywhere.

I compute, for each image at application time, a mask that measures local contrast. The image is converted to greyscale, the local mean and variance are estimated over a 7×7 window, and the local standard deviation is normalised by its maximum over the image:

$$\hat{\sigma}(x) = \frac{\sigma_{7 \times 7}(x)}{\max \sigma_{7 \times 7}(x)}, \quad M(x) = s_{\min} + (1 - s_{\min}) \hat{\sigma}(x) \quad (3.7)$$

The mask takes values in $[s_{\min}, 1]$: close to $s_{\min}$ in a flat region, close to 1 in a textured one. The attacked image is

$$x_{\text{adv}} = \text{clamp}(x + M(x) \odot \delta, 0, 1) \quad (3.8)$$

Universality is preserved, and this point matters. There is still one single δ , shared by every image and learned once. The mask is a deterministic function of the image computed at application time, with no learned parameter, so nothing image-specific is stored in the attack.

Two decisions, and only the second one worked

The mask must also act during training. The mask is a function of the image, so it is computed every time the perturbation is applied: during training, during evaluation and at attack time alike. Without the image there is no way to know which regions are smooth and which are textured. The question is therefore not whether the mask is used

when the perturbation is applied, but whether it is *also* active while the perturbation is being learnt.

My first attempt used it only at application, on a perturbation trained under a uniform budget. The result was a pure exchange: exactly as much fooling rate lost as visual quality gained. Three settings gave 31.1 %, 45.0 % and 56.7 % for structural similarities of 0.947, 0.906 and 0.861. The explanation is clear after the fact. The perturbation had invested its energy in smooth regions, and attenuating it there afterwards left no opportunity to compensate elsewhere.

Putting the mask inside the training loop changes the problem. The perturbation is masked when it is added to the semantic samples, so from the first iteration it is optimised under the same attenuation it will face at attack time, and it learns to place its energy where the mask lets it through.

The ceiling must be raised, not lowered. With the mask in the loop, ϵ no longer describes what reaches the image. I therefore raised it well above the conventional 10/255. At $\epsilon_{\max} = 40/255$ with $s_{\min} = 0.13$, what lands on the image is about 5.2/255 in a smooth region, which is half of what the baseline applied there, and up to 40/255 in a textured one, where the eye will not notice it. The perturbation is not reduced. It is displaced.

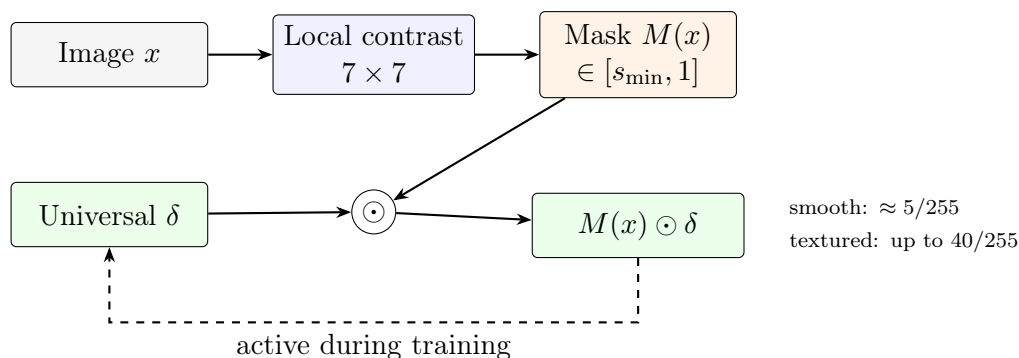


Figure 3.12: Perceptual amplitude budget. A mask derived from local contrast modulates the universal perturbation before it is applied. The perturbation itself stays unique and image-independent; only its local amplitude varies. Because the mask is active during training, the perturbation learns to concentrate its energy where the mask lets it pass.

Choosing an operating point

The two parameters $\epsilon_{\max}$ and $s_{\min}$ are not independent: what matters perceptually is their product, the amplitude that actually reaches a flat region. I therefore swept both jointly. Figure 3.13 plots every configuration in the plane of visual quality against fooling rate, each curve corresponding to a fixed smooth amplitude.

Two things are visible. Every point of the cloud lies to the right of the two red markers, so the whole family improves visual quality over the uniform budget. And the smooth amplitude behaves as a cliff rather than a slider: below roughly 4.8/255 the fooling rate

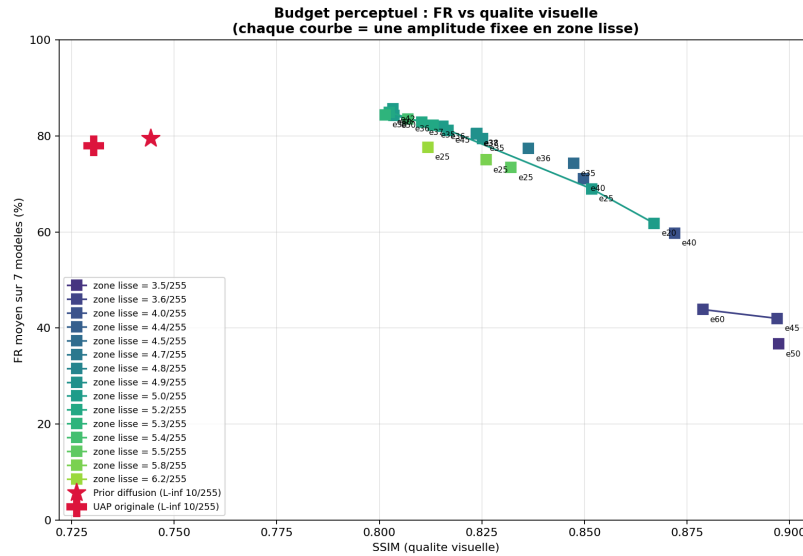


Figure 3.13: Fooling rate against structural similarity for the perceptual budget sweep. Each curve holds the smooth-region amplitude constant while $\epsilon_{\max}$ varies. The two red markers are the ensemble baseline and the diffusion-prior perturbation, both under a uniform L_{∞} budget of 10/255.

collapses, while above it the family stays in a narrow band. The retained configuration sits just above that cliff.

Result

At $\epsilon_{\max} = 40/255$ and $s_{\min} = 0.13$, the perturbation reaches a mean fooling rate of 84.38 % for a structural similarity of 0.8025. Compared with the ensemble baseline at 77.14 % and 0.7303, both quantities improve at once. This is the point the whole section was aimed at, because it means the result is not a trade-off.

The gain is also, this time, spread across the unseen models: their mean rises from 75.96 % to 80.40 %. Figure 3.14 shows what the difference looks like on three validation images.

3.3.5 Consolidated results

Table 3.6 gathers the three stages under the single evaluation protocol.

Read from top to bottom, the table tells a simple story. The ensemble sets a reference at 77.14 % and 0.7303. The diffusion prior adds a little on both axes, essentially by strengthening the attack on the surrogates. The perceptual budget converts that surrogate margin into a gain that holds across every model, while raising visual quality by seven points of structural similarity.

The last two rows differ only by the ceiling, and I kept the higher one because of the acceptance criterion of Section 1.3: the gain has to hold on models that were never used during training. At $\epsilon_{\max} = 38/255$ the mean over those five models is 75.72 %, below the 75.96 % of the ensemble baseline. At 40/255 it reaches 80.40 %.

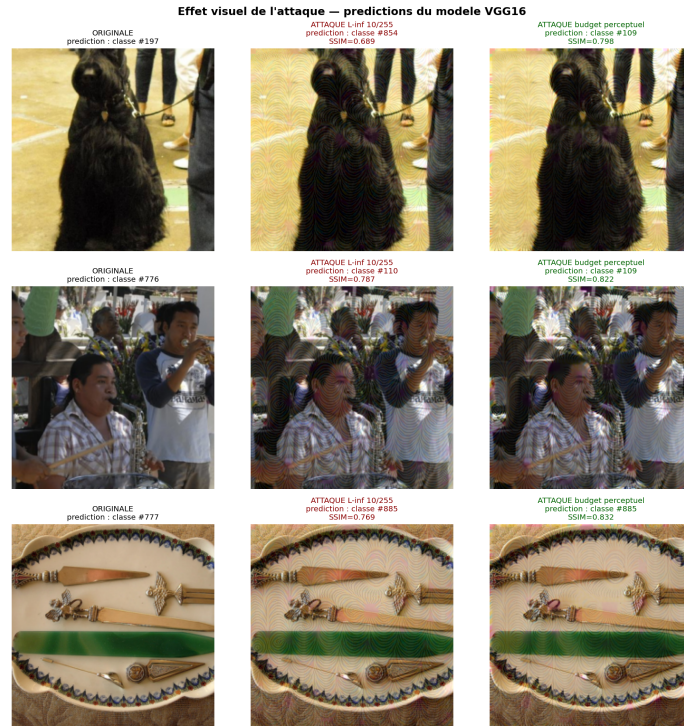


Figure 3.14: Visual effect of the attack, with VGG16 predictions. Left, the original image. Centre, the uniform L_∞ budget at 10/255. Right, the perceptual budget. The perturbation is markedly less visible in flat regions while the model is still fooled.

Configuration	SSIM	AN	V16	V19	R50	R152	D121	GN	EN	BB	All
Ensemble ($\epsilon = 10$)	0.7303	57.4	87.5	86.9	78.9	78.8	79.6	76.8	71.2	75.96	77.14
+ diffusion prior	0.7443	53.2	89.5	87.0	87.6	82.1	79.8	76.8	71.4	75.58	78.42
+ perceptual (38, 0.13)	0.8238	50.0	92.4	90.3	92.7	84.6	81.2	72.7	73.2	75.72	79.64
+ perceptual (40, 0.13)	0.8025	53.1	95.8	94.0	95.4	91.2	86.4	78.6	80.5	80.40	84.38

Table 3.6: Fooling rate (%) per target model and structural similarity, on 1000 images of the ImageNet validation set. R50, R152 and D121 are the surrogates. BB is the mean over the five unseen models, All the mean over the eight. The last row is the retained configuration.

Not everything worked, however, and some of the failures were expensive. The next chapter reports them, together with the evaluation of the method against a deepfake detector.

Chapter 4

Results, discussion and conclusion

The previous chapter validated the method on a public benchmark. This one takes it to the domain the subject was about. The attack is transferred to a real deepfake detector, then extended to video, and the question is finally reversed: how could such an attack be defeated?

4.1 Transfer to a deepfake detector

4.1.1 Target and protocol

The attack is evaluated against the reference detector of the field, the official FaceForensics++ Xception model published by Tolosana et al. (2020). No training was done on my side: both the model and the data come from the authors, which keeps the protocol directly comparable to the literature. The checkpoint is the one trained on cropped faces at compression level c23, with two classes and 299×299 inputs.

The evaluation corpus consists of 50 manipulated and 50 original videos, from which 20 frames each were extracted, giving 999 fake and 999 real faces. Faces are located with a Haar cascade and cropped with a 30 % margin, reproducing the framing used by the authors for training. On these 1998 faces and without any attack, the detector recognises 92.79 % of the deepfakes and 97.00 % of the real images, for an overall accuracy of 94.89 %. This baseline validates the protocol: any degradation observed afterwards is attributable to the attack.

4.1.2 Reusing the perturbation, and two necessary adaptations

Nothing is trained from scratch here. The perturbation obtained at the end of Section 3.3, on the image benchmark, is loaded as the starting point and refined against the detector for a short run. This is the same warm-start logic as the three stages of Section 3.3.1: a refinement needs a pattern that already works to build on. The detector itself is the published one, used as-is; only the perturbation is refined.

The algorithm of Section 3.3 is kept in full, including the diffusion-generated prior, the semantic sampling, the input transformations, the KL reweighting and the perceptual

mask. The data-free setting is preserved: no face ever enters the optimisation loop, and FaceForensics++ is used only to measure. Two changes were nevertheless required.

The perturbation is optimised **directly at** 299×299 . A pattern trained at 224 and then resized loses 29.5 % of its amplitude, which weakened the attack before it was even evaluated.

More importantly, a **fooling term had to be added**. The original objective only maximises the internal activations of the network. On ImageNet that is enough, because disturbing the representation strongly enough will move the prediction to one of a thousand classes. Here there are only two classes, and the objective saturates immediately: it pushes every image to the same side without ever aiming at the right one.

I therefore added a second term to the objective. It compares the detector’s output on an attacked sample with the class the attack is meant to reach, and it rewards the perturbation when the output moves that way. The perturbation is still what changes the verdict. The new term only tells the optimisation which of the two classes to aim at, which the activation term on its own never does.

The objective therefore has two terms, and λ is nothing more than the weight of the second:

$$\mathcal{L}_{\text{deepfake}} = \mathcal{L}_{\text{act}} + \lambda \mathcal{L}_{\text{fool}} \quad (4.1)$$

Here $\mathcal{L}_{\text{act}}$ is the activation objective of Section 3.2.5, unchanged, and $\mathcal{L}_{\text{fool}}$ is the new term. At $\lambda = 0$ the attack behaves exactly as it does on ImageNet. The larger λ is, the more each update is decided by the class being aimed at rather than by the amount of internal disturbance. The two gradients are normalised before they are combined, so λ compares two quantities of the same size.

4.1.3 A structural limitation

I trained the perturbation at three values of λ , expecting to find the one that makes the detector wrong most often. There is no such value.

λ	fake $\rightarrow$ real	real $\rightarrow$ fake	overall fooling rate
1	7.14 %	96.92 %	50.83 %
3	87.66 %	11.99 %	50.83 %
5	92.86 %	4.45 %	50.83 %

Table 4.1: Effect of the weight λ given to the fooling term. The first column counts manipulated faces that pass for authentic after the attack, the second authentic faces flagged as manipulated. Whatever the weight, the total stays put: λ moves the cursor between the two directions and nothing else.

Read the last column first. Whatever weight the fooling term receives, the proportion of faces the detector gets wrong stays at 50.83 %, to the second decimal. λ does not change *how many* faces are misclassified. It only chooses *which ones*. At $\lambda = 1$ the attack

turns authentic faces into fakes; at $\lambda = 5$ it does the reverse; in between it trades one direction for the other, point for point.

The explanation is structural. A fixed pattern adds approximately the same offset to the logits of every image. On a binary problem the decision reduces to the sign of a scalar, so a constant offset pushes every image to the same side of the boundary. **Fooling both directions at once is out of reach for a universal perturbation on a binary task.** This is not a tuning issue but a consequence of the universality constraint itself.

It is not a practical limitation either: the relevant attack scenario is to make deepfakes pass for authentic, not to flag real images as manipulated. I therefore kept $\lambda = 5$.

4.1.4 Results on images

I used $\epsilon_{\max} = 38/255$ here rather than the $40/255$ of Chapter 3. On a binary detector the attack saturates well below that ceiling, so the extra amplitude buys no evasion and only costs visual quality.

With $\lambda = 5$, $\epsilon_{\max} = 38/255$ and $s_{\min} = 0.13$, the 1998 faces were passed to the detector with real and fake images mixed.

	before attack	after attack
deepfakes recognised as fake	92.79 %	0.70 %
deepfakes taken for real	7.21 %	99.30 %
real images recognised as real	97.00 %	98.80 %
overall accuracy	94.89 %	49.75 %
SSIM	—	0.864

Table 4.2: Effect of the attack on the official FaceForensics++ detector, over 1998 faces.

The attack is **selective**. It neutralises deepfake detection while leaving the classification of authentic images intact, and even marginally improved. A detector under attack therefore produces no false alarm that would betray its compromise, which is precisely what makes the scenario dangerous.

One comparison puts this figure in context. A perturbation trained under direct white-box supervision, that is with access to the FF++ images themselves, reaches only 79.55 % evasion. The **data-free version therefore outperforms the supervised one**, which was not expected.

4.2 Extension to video

Since the pattern is fixed, applying it to a video reduces to adding it to each frame. No optimisation is redone between frames: the per-frame cost is one addition and the computation of the perceptual mask. This is what makes the attack compatible with a live stream, unlike per-image attacks which would require an optimisation per frame.

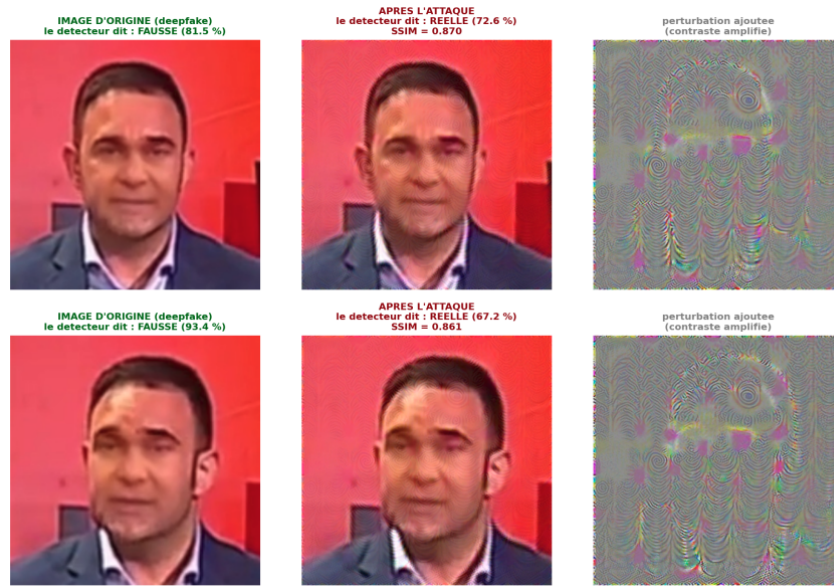


Figure 4.1: Deepfakes classified as authentic after the attack. Each row shows the original face, correctly detected, then the attacked face, classified as real, then the perturbation that was added, with its contrast amplified so that it can be seen at all. The mask is visible in that third image: the perturbation is almost flat over the plain background and regains its full pattern on the hair, the jacket and the edges of the face.

For each video, the face is located on every frame, the detector predicts before and after the perturbation is applied to the cropped face, and a **sequence verdict** is obtained by majority vote. That last point matters: a video detector does not emit a verdict per frame but aggregates over the sequence, so it is the flip of this global verdict that measures real success.

Video	fake frames before	after	verdict before	after	SSIM
035_036	100.0 %	0.00 %	fake	real	0.844
147_055	100.0 %	0.00 %	fake	real	0.875
159_175	100.0 %	0.00 %	fake	real	0.877
097_033	99.33 %	0.00 %	fake	real	0.883
044_945	97.33 %	0.67 %	fake	real	0.854

Table 4.3: Five of the seven sequences correctly identified by the detector before the attack. All seven flip to *real*.

Over the ten sequences tested, the proportion of frames judged fake falls from 75.9 % to 5.8 % on average, at a mean structural similarity of 0.861.

All seven sequences that the detector identified correctly before the attack flip to *real* afterwards. The proportion of frames judged fake falls from 99.4 % to 0.1 % on average across those seven, at a mean structural similarity of 0.863.

4.3 Countermeasures

Having established the threat, the last part of the internship asked the symmetric question. Three families of countermeasure were explored. All measurements below use a 70/30 split of the face corpus with seed 126; the 600 measurement faces never take part in any training.

4.3.1 Input filtering

The first idea is to place a filter before the detector, to erase the perturbation without damaging the image. Five families were tested at several strengths: JPEG compression, Gaussian blur, median filtering, bit-depth reduction, and downsampling followed by upsampling.

Table 4.4 has three columns because a filter has to be judged on three different inputs. The first is a manipulated face carrying the perturbation: this is the attack, and the filter is supposed to repair it. The second is a manipulated face with no perturbation at all, and the third an authentic face. Those last two are ordinary traffic, the images the detector sees all day when nobody is attacking it. A filter is only useful if it raises the first column without collapsing either of the other two.

Filter	Deepfake + UAP	Deepfake, no UAP	Authentic face
<i>what the detector is given</i>	<i>% called fake</i>	<i>% called fake</i>	<i>% called real</i>
none	0.70 %	92.79 %	97.00 %
JPEG quality 50	0.90 %	71.67 %	97.50 %
Gaussian blur $\sigma = 1.0$	25.23 %	92.69 %	88.39 %
median 5×5	45.15 %	96.60 %	71.07 %
downsampling to 25 %	53.55 %	92.09 %	56.06 %
Gaussian blur $\sigma = 1.5$	81.68 %	93.79 %	55.76 %
reduction to 3 bits	83.78 %	38.24 %	82.78 %

Table 4.4: Input filtering. Every cell is the percentage of images the detector classifies *correctly* once the filter has been applied, so higher is better everywhere. Seven of the fourteen configurations tested are shown; the first row is the reference, with no filter at all.

No configuration restores detection without breaking something else, and the two best rows illustrate two different ways of failing.

Blur at $\sigma = 1.5$ recovers 81.68 % of detection under attack and keeps clean deepfakes at 93.79 %, but it drops recognition of authentic images from 97 % to 55.76 %. A detector that flags one authentic image in two is unusable.

Reduction to 3 bits looks better at first glance, since it recovers 83.78 % under attack while keeping 82.78 % of authentic images. The second column is what rules it out: detection of *unattacked* deepfakes collapses from 92.79 % to 38.24 %. The filter repairs the attacked images by destroying the very traces the detector relies on, so almost two manipulated videos in three would pass unnoticed on an ordinary stream.

The underlying reason is the same in both cases. A filter aggressive enough to reach the perturbation is also aggressive enough to erase the manipulation artefacts. **Filtering is not a viable defence.**

4.3.2 Adversarial training

The second approach retrains the detector on a mixture of clean and attacked images, five epochs at a learning rate of 10^{-5} .

A control is indispensable here, and it is the part of this experiment I am most attached to. Retraining the official detector on our own faces improves it anyway, regardless of any robustness. To separate the two effects, exactly the same retraining was repeated on clean images only. That gives three detectors: the published one, one retrained on clean faces, and one retrained on a mixture.

Detector	Deepfake, no UAP	Deepfake + UAP
<i>what it is given</i>	% called <i>fake</i>	% called <i>fake</i>
published, not retrained	92.86 %	0.65 %
retrained on clean faces (control)	99.35 %	9.74 %
retrained on clean + attacked faces	99.03 %	97.73 %

Table 4.5: Adversarial training and its control. Both columns count manipulated faces the detector correctly calls fake, so higher is better. The two columns use the same faces: without the perturbation on the left, with it on the right.

Read Table 4.5 one column at a time. In the first column, both retrained detectors gain about six points over the published one, and the control gains slightly more. That improvement has nothing to do with robustness. It comes from retraining on our own faces, and it would have happened with or without a single attacked image.

The second column tells a different story. The control never saw a perturbed image during its retraining, and it still collapses to 9.74 %. The detector retrained on the mixture holds at 97.73 %. The gap between those two numbers, close to ninety points, is what adversarial training actually contributes. Without the control, that gap could not have been separated from the ordinary benefit of retraining.

The interesting question is what happens against an attacker who adapts. Three patterns were built with the same pipeline, the same budget and the same mask: the initial UAP, an *adapted* UAP built against the retrained detector with access to its weights, and a *transferred* UAP built on the published detector from a different random draw.

The defence has two very different behaviours depending on what the attacker holds.

If the attacker **does not have the weights** of the retrained detector, they attack the public model. Detection only falls to 81.82 %, and the important point is that this holds against a pattern the defence had never seen. It blocks the whole family of patterns produced by this pipeline, not merely the one it was shown.

If the attacker **has the weights**, they rebuild a pattern against the hardened detector and recover complete evasion, at equivalent perceptual quality. The defence collapses

Detector	Condition	deepfakes detected	real recognised	SSIM
published	clean images	92.86 %	96.92 %	—
published	initial UAP	0.65 %	97.26 %	0.864
published	adapted UAP	25.32 %	81.16 %	0.874
published	transferred UAP	0.97 %	98.63 %	0.876
retrained	clean images	99.03 %	98.29 %	—
retrained	initial UAP	97.73 %	97.95 %	0.864
retrained	adapted UAP	0.00 %	100.00 %	0.874
retrained	transferred UAP	81.82 %	96.92 %	0.876

Table 4.6: Cross-evaluation of two detectors against three perturbations. Measured on the 600 held-out faces.

entirely. It therefore rests on the confidentiality of the deployed model, which is a fragile assumption and a known one in security.

4.3.3 Detecting the perturbation

The third approach abandons the idea of making the detector robust and places a module upstream whose only job is to say whether an image has been altered.

The starting point is an observation about my own mask. It places the perturbation in textured regions where the eye does not see it, but it never cancels it completely: in a smooth region it still lets through 13 %, the value of $s_{\min}$. That is precisely where natural image noise is weakest. The perturbation therefore raises the noise floor in quiet areas and makes its distribution abnormally regular across the image.

In other words, **the mask that makes the attack invisible to the eye is what creates its statistical signature**. This symmetry is, to me, the most interesting finding of the whole internship.

The module works on a residual. The image is blurred and subtracted from itself, leaving only fine detail and no face. Nine statistics summarise that residual, and a logistic regression, nine coefficients and no neural network, is trained on 1398 clean images and the same 1398 images carrying the initial UAP. The decision threshold is placed just above the highest score observed on the clean training images and then frozen.

The module generalises for a specific reason: it never sees the pixels of the pattern, only nine numbers, so it cannot memorise a particular pattern and can only learn a general rule about what the noise looks like. The largest coefficient, -2.404 , is that of spatial irregularity, and its negative sign means the quantity drops when the image is altered. The noise becomes more regular. That is a consequence of applying any universal perturbation through a mask, not a property of one specific pattern.

The module was also run on video, on the same processing chain as the attack. On two sequences of 150 analysed faces, it flags 0.00 % of the unattacked faces and 100.00 % of the attacked ones, at 21 to 28 faces per second, which is compatible with a live stream.

The module is not unconditionally reliable, and Section 4.4.2 states the limits I could

Detector	Image given	% called <i>fake</i>	% reported altered
published	clean images	92.86 %	0.33 %
published	initial UAP	0.65 %	100.00 %
retrained	clean images	99.03 %	0.33 %
retrained	adapted UAP	0.00 %	100.00 %

Table 4.7: Perturbation detection module. The third column is the detector’s own verdict, the fourth the module’s. On clean images the fourth column is the false alarm rate, so there it should be as low as possible; on attacked images it should be as high as possible. The adapted UAP was never shown to the module during training.

measure or anticipate.

4.3.4 Comparison

Countermeasure	Known pattern	Unknown pattern
input filtering	fails	fails
adversarial training	97.73 %	81.82 % without the weights, 0 % with them
perturbation detection	100 %	100 %

Table 4.8: The three countermeasures compared.

The first two act on the detector; the third acts on the image, upstream, and it is the only one that generalises to a pattern it never met.

The general finding that emerges is this. A universal perturbation is hard to neutralise, but it is not discreet. Its universality is its strength as an attack, since one pattern applies to everything including a live video stream, and it is also what leaves a regular statistical trace across every image it touches. That trace is invisible to a classifier and to the eye. It becomes visible to whoever knows what to measure, which is a very different claim from saying that the attack is easy to spot: the module had to be built for it, and it was built by reasoning about the mask rather than by looking at the images.

The next section draws the conclusions of this work and outlines what remains to be done.

4.4 Discussion

4.4.1 Directions that did not work

Four directions were tried and abandoned. Each one closes a line of enquiry that looks reasonable on paper, and each was closed by a measurement rather than by an impression.

The first row is the one that mattered most for the schedule. The test took an afternoon and closed a direction that would have consumed weeks. This is also why the mask of

Direction	Measured outcome	Why it fails
Optimising the perturbation inside the diffusion autoencoder	93 % of the pattern lost on a round trip; fooling rate $76 \rightarrow 34$ %	The autoencoder keeps what the eye sees and discards what it does not, which is exactly what the perturbation is built to be.
Adding a visual quality penalty to the loss	Fooling rate plateaus at 42 %	The penalty is minimal when the perturbation is zero, so it pulls the attack towards zero at every iteration and always wins.
Redistributing the perturbation across frequencies	Slight gain in quality, collapse in fooling rate	Visual quality depends on how strong the noise is, not on what kind it is. Only varying the strength by location helps.
Input filtering as a defence	No configuration among fourteen is viable	A filter aggressive enough to reach the perturbation also erases the manipulation traces the detector relies on.

Table 4.9: Directions abandoned during the internship, with the measurement that closed each one.

Section 3.3.4 is a constraint applied to the perturbation rather than a term added to the loss: the second row had already shown what happens otherwise.

4.4.2 Limitations

Several limitations should be kept in mind when reading the figures of this chapter.

Visual quality is measured, not judged. Structural similarity is an objective metric, and no subjective evaluation was carried out. The reference protocol for that, ITU-T Recommendation P.910 (International Telecommunication Union, 1999), requires a panel of human observers rating the content from 1 to 5. A structural similarity of 0.86 does not mechanically translate into an imperceptible alteration, and only a panel could settle that.

Structural similarity says little about colour. SSIM compares luminance, contrast and structure. A change that shifts colour without changing brightness barely moves it. The diffusion prior does exactly that, since it spreads the pattern’s colour over larger areas. The reported gain in structural similarity is therefore real but incomplete, and a colour-aware metric would be the natural complement.

The attack is one-directional. As shown in Section 4.1, a universal perturbation cannot fool both directions of a binary decision at once. This is a consequence of universality

itself, not a tuning problem. It is not a practical limitation for the scenario studied, but it does bound what this class of attack can do.

The detection module depends on the image domain. Applied to whole frames resized from high definition, without cropping the face first, it flags everything. Resizing alters the fine noise it measures. Cropping the face is the natural position of such a module in an anti-deepfake system, but the dependence is real and would be lifted by multi-domain training.

An informed attacker could evade the detection module. It relies on the noise becoming abnormally regular. Adding a constraint that preserves the natural irregularity of the noise would plausibly defeat it. This is untested.

The countermeasures were tested against one family of patterns. The three perturbations used in Section 4.3 come from the same pipeline and share a common structure, and the detection module was learnt from a single one of them. A pattern produced by a different method is not guaranteed to be caught.

4.5 Conclusion

4.5.1 Technical conclusion

Section 1.1.3 asked whether a universal, data-free perturbation could be made more effective and less visible at the same time, instead of trading one property against the other. The answer is yes, and it holds under one condition: the perceptual constraint has to be active *during* training, and not only at the moment the perturbation is applied. Used only at application, the mask produces a pure exchange, as much effectiveness lost as quality gained. Active inside the loop, it lets the perturbation learn where it is allowed to spend its energy, and both quantities improve together.

Starting from PSP-UAP, which I reproduced before modifying, I built three contributions. Training against an ensemble of three surrogate architectures, with their contributions rebalanced, removes the dependence on a single network. Replacing the random prior with images generated offline by an unconditional diffusion model gives the optimisation real structure to work on, without introducing a single real photograph. And the perceptual amplitude budget moves the perturbation into the regions where the eye does not see it instead of weakening it everywhere.

Together these raise the mean fooling rate from 77.14 % to 84.38 % and the structural similarity from 0.7303 to 0.8025, across eight target architectures, five of which were never seen during training. Both quantities improve at once, which was the specification set in Section 1.3.

Transferred to the official FaceForensics++ detector, the attack brings detection from 92.79 % to 0.70 % at a structural similarity of 0.864, while leaving authentic images cor-

rectly classified. The attack is silent: a compromised detector raises no alarm. On video, the seven sequences the detector identified correctly all flip to *real*, with no recomputation between frames, which is what makes the threat realistic on a live stream.

On the defensive side, input filtering was shown not to be viable. Adversarial training works against a pattern the attacker cannot adapt, but collapses entirely against an attacker who obtains the retrained weights, so it rests on the confidentiality of the deployed model. The only countermeasure that generalised is the one that looks at the image rather than at the decision. And it works for a reason worth stating on its own: the mask that makes the attack invisible to the eye is exactly what gives it a statistical signature. It never removes the perturbation from smooth regions, and smooth regions are where natural image noise is weakest, so the noise floor rises and becomes abnormally regular. Nine coefficients are enough to see it, including on a perturbation the module was never shown.

A universal perturbation is therefore hard to neutralise, but it is not discreet to whoever knows what to measure.

4.5.2 Perspectives

Four directions follow naturally from this work.

The first is subjective validation. Running an ITU-T P.910 panel on the attacked images and videos would replace the structural similarity by the judgement it is meant to approximate, and would settle whether the current operating point is genuinely imperceptible.

The second is an adaptive attacker against the detection module. The module relies on the noise becoming abnormally regular. An attacker aware of this could add a constraint that preserves the natural irregularity of the noise. Whether the module survives that is the question that determines its real value.

The third is multi-domain training for the module, over several resolutions and compression levels, which would lift the domain dependence identified above.

The fourth concerns architectures. The attacks studied here exploit convolutional filters. Vision Transformers process the image as a set of patches rather than through local convolutions, and preliminary observations suggest they resist this family of attack markedly better. Extending the method to them would require attacking global structure rather than local texture.

4.5.3 Personal assessment

I arrived at the GREYC with a background in deep learning but none in adversarial robustness, and with no experience of research work as such. Five months later, three things stand out.

The first is methodological. I learned that in research, the discipline of measurement matters as much as the ideas. The single most useful decision of the whole internship was not an algorithmic one; it was freezing the evaluation protocol in July and recomputing

everything under it. Before that, I had figures that looked good and could not be compared with each other, which is to say I had nothing.

The second concerns failure. Roughly half of what I tried did not work, and at first I experienced that as time lost. It is not. The measurement that showed 93 % of the perturbation was destroyed by the diffusion autoencoder took an afternoon and closed a direction that would have consumed weeks. Deciding in advance on the figure that would end an experiment is what makes a failure cheap.

The third is about working alone. Being the only person on a subject means nobody catches your mistakes, and it forces a rigour I would not have imposed on myself in a team. It also means the weekly meeting becomes the moment that structures the work, and I learned to prepare it as a decision point rather than as a progress report.

I would like to thank my supervisors for their availability and for the freedom they left me throughout these five months.

Bibliography

- Jiazhu Dai and Le Shu. Fast-uap: An algorithm for speeding up universal adversarial perturbation generation with orientation of perturbation vectors. *arXiv preprint arXiv:2009.02472*, 2020.
- Sagar Dalal, Ashish Mittal, and Shanmuganathan Raman. Perceptual image quality metrics based on visual saliency and structural information. *Signal Processing: Image Communication*, 84:115828, 2020.
- Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.
- David M. Green and John A. Swets. *Signal Detection Theory and Psychophysics*. Wiley, New York, 1966.
- Cindy Hartmann and Antoine Montier. Attaques adversariales sur les systèmes de détection de deepfakes. Master’s semester project report, GREYC, Université de Caen Normandie, 2025. Unpublished internal report.
- International Telecommunication Union. Subjective video quality assessment methods for multimedia applications. Technical report, ITU-T Recommendation P.910, 1999.
- Chanhui Lee, Yeonghwan Song, and Jeany Son. Data-free universal adversarial perturbation with pseudo-semantic prior. *arXiv preprint arXiv:2502.21048*, 2025.
- Maosen Li, Yanhua Yang, Kun Wei, Xu Yang, and Heng Huang. Learning universal adversarial perturbation by adversarial example. In *AAAI Conference on Artificial Intelligence*, 2022.
- Hong Liu, Rongrong Ji, Jie Li, Baochang Zhang, Yue Gao, Yongjian Wu, and Feiyue Huang. Universal adversarial perturbation via prior driven uncertainty approximation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.
- Yiran Liu, Xin Feng, Yunlong Wang, Wu Yang, and Di Ming. Trm-uap: Enhancing the transferability of data-free universal adversarial perturbation via truncated ratio maximization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- Konda Reddy Mopuri, Utsav Garg, and R. Venkatesh Babu. Fast feature fool: A data independent approach to universal adversarial perturbations. In *British Machine Vision Conference (BMVC)*, 2017.
- Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable data-free objective for crafting universal adversarial perturbations. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(10), 2018a.
- Konda Reddy Mopuri, Phani Krishna Uppala, and R. Venkatesh Babu. Ask, acquire, and attack: Data-free uap generation using class impressions. In *European Conference on Computer Vision (ECCV)*, 2018b.
- Paarth Neekhara, Brian Dolhansky, Joanna Bitton, and Cristian Canton Ferrer. Adversarial threats to deepfake detection: A practical perspective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020.
- Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy*, 2016.
- Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115, 2015.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.
- Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion*, 64:131–148, 2020.

- Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.
- Andrew B. Watson. Digital images and human vision: The just noticeable difference model. In *Society for Information Display (SID) Symposium Digest of Technical Papers*, pages 419–422, 2001.
- Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- Chaoning Zhang, Philipp Benz, Adil Karjauv, and In So Kweon. Data-free universal adversarial perturbation and black-box attack. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.
- Rongyi Zhu, Zeliang Zhang, Susan Liang, Zhuo Liu, and Chenliang Xu. Learning to transform dynamically for better adversarial transferability. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Abstract

Deepfake detectors are neural networks. That is their strength, and it is also their weakness: a small pattern added to an image can flip their verdict while a human sees nothing unusual. Universal perturbations make this weakness practical, because one fixed pattern, computed once, can be added to any image without recomputation. On a video stream that changes everything.

This report presents five months of work in the SAFE team of the GREYC laboratory. My goal was to build such a pattern without trading effectiveness for invisibility. Two rules could not be broken: one single pattern for every image, and no access to the training data of the target model. Starting from PSP-UAP, a data-free method published in 2025, I added a surrogate ensemble, a prior generated by an unconditional diffusion model, and a perceptual amplitude budget applied during training rather than after it. Together they raise the mean fooling rate from 77.14 % to 84.38 % and the structural similarity from 0.7303 to 0.8025, over eight target architectures. Applied to the official FaceForensics++ detector, the attack brings detection from 92.79 % to 0.70 %.

Keywords: adversarial attack, universal adversarial perturbation, deepfake detection, data-free attack, diffusion model, visual perception, image watermarking.

Résumé

Les détecteurs de deepfakes sont des réseaux de neurones. C'est leur force, et c'est aussi leur faiblesse : un motif discret ajouté à une image suffit à inverser leur verdict, sans qu'un œil humain remarque quoi que ce soit. Les perturbations universelles rendent cette faiblesse exploitable, car un motif fixe, calculé une seule fois, s'ajoute à n'importe quelle image sans recalcul. Sur un flux vidéo, cela change tout.

Ce rapport présente cinq mois de travail au sein de l'équipe SAFE du laboratoire GREYC. Mon objectif était de construire un tel motif sans échanger l'efficacité contre l'invisibilité. Deux règles ne pouvaient pas être enfreintes : un seul motif pour toutes les images, et aucun accès aux données d'entraînement du modèle visé. Parti de PSP-UAP, une méthode sans données publiée en 2025, j'ai ajouté un ensemble de modèles substitués, un prior généré par un modèle de diffusion non conditionné, et un budget d'amplitude perceptuel appliqué pendant l'entraînement plutôt qu'après. L'ensemble fait passer le taux de tromperie moyen de 77,14 % à 84,38 % et la similarité structurelle de 0,7303 à 0,8025, sur huit architectures cibles. Appliquée au détecteur officiel FaceForensics++, l'attaque fait tomber la détection de 92,79 % à 0,70 %.

Mots-clés : attaque adversariale, perturbation adversariale universelle, détection de deepfake, attaque sans données, modèle de diffusion, perception visuelle, tatouage d'image.