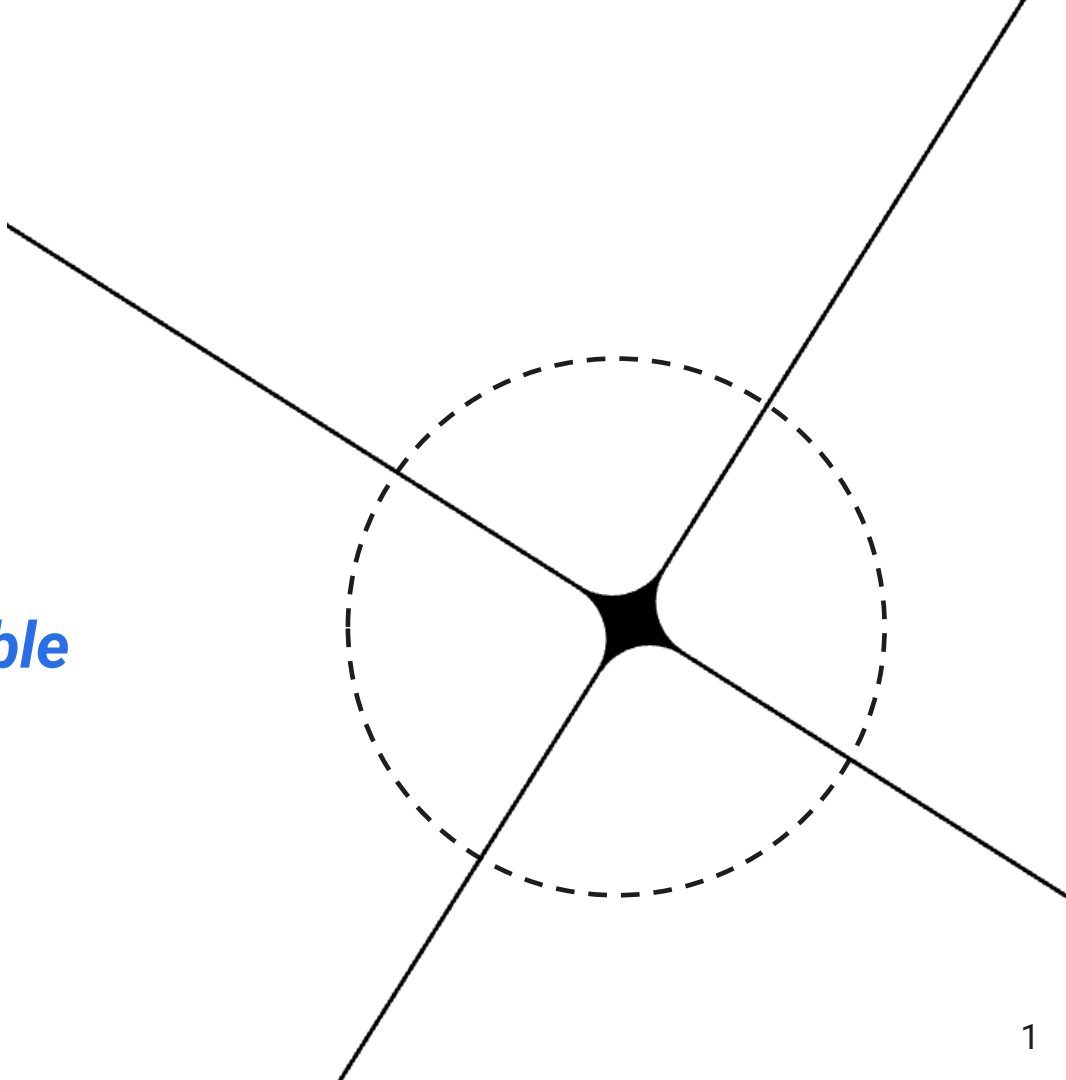
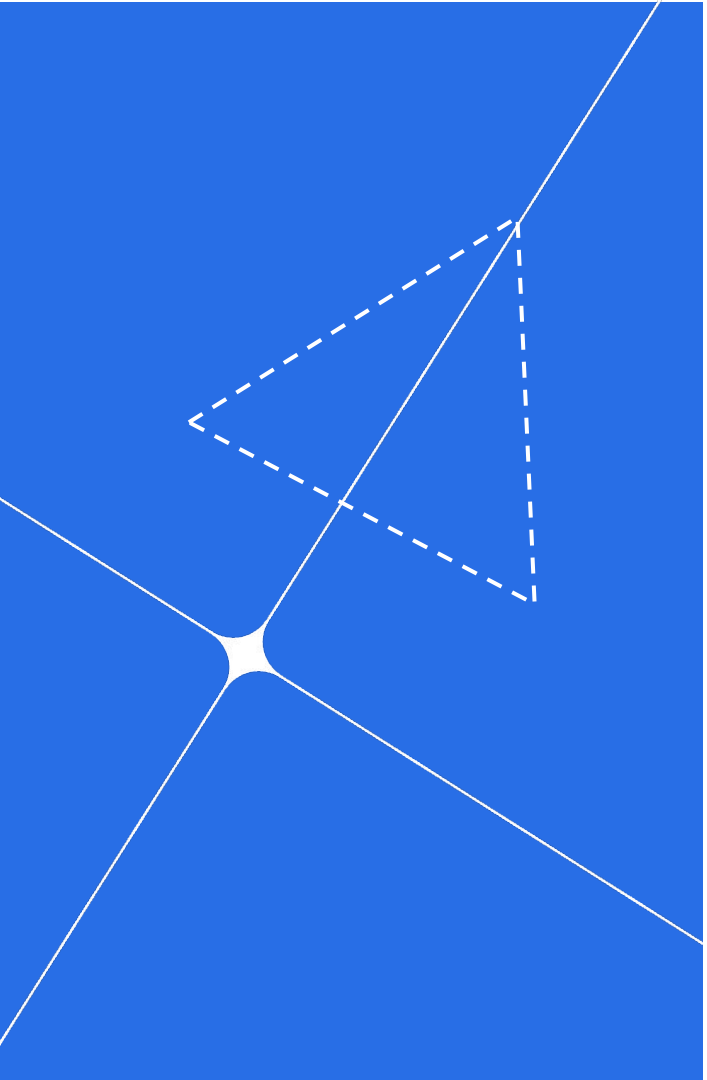
INSTITUT POLYTECHNIQUE DE PARIS

TELECOM SudParis

IA interprétable et explicable

Julien Romero

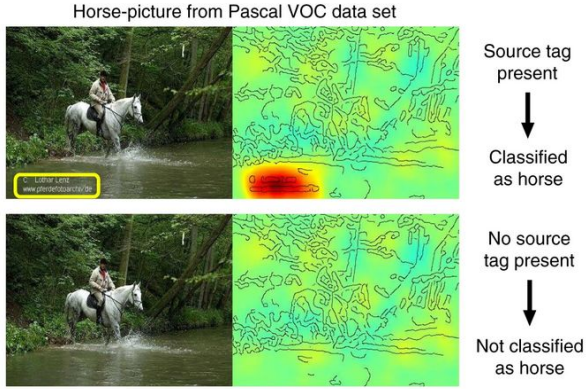




Le Mythe de la Boîte Noire

Motivation : Au-delà de l'Accuracy en Deep Learning

- Dans les projets académiques, on optimise des métriques globales : *Loss*, *F1-score*, *AUC*.
- En production (ex: aide au diagnostic médical), une *accuracy* de 99% ne suffit pas à garantir la sécurité.
- **Le problème du "Clever Hans" (Biais de confusion)** :
  - Un modèle CNN détecte parfaitement les pneumonies sur des radiographies...
  - *Réalité* : Le modèle a appris à détecter le type de scanner (marque ou règle en métal sur l'image) utilisé dans le service des cas graves, et non la pathologie.
- Sans explicabilité, un modèle hautement performant peut cacher une logique de décision absurde ou dangereuse.
- **Conséquence MLOps** : Le déploiement d'un tel modèle "boîte noire" en milieu hospitalier génère un rejet immédiat par le corps médical (manque de confiance).



Le Cadre Légal : L'XAI comme Obligation Réglementaire

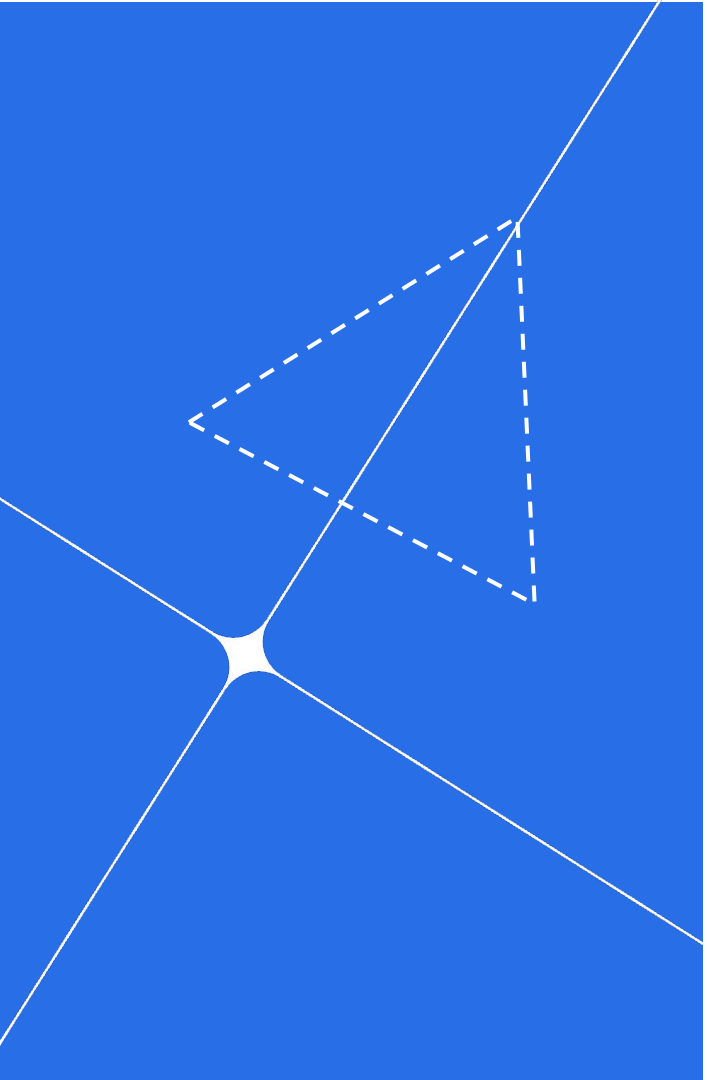
- XAI = eXplainable AI
- L'industrie ne peut plus déployer de boîtes noires sans garde-fous juridiques.
- **RGPD (Règlement Général sur la Protection des Données)** :
  - *Article 22* : Limite la prise de décision automatisée produisant des effets juridiques.
  - Instaure un "droit à l'explication" (*Right to explanation*) pour les utilisateurs finaux (ex: refus de prêt bancaire, ou refus de prise en charge mutuelle).
- **L'EU AI Act (en vigueur progressivement depuis 2024)** :
  - Classe les systèmes d'IA par niveau de risque. Le diagnostic médical est classé **"Haut Risque" (High-Risk)**.
  - Exige une transparence technique garantissant une surveillance humaine (*Human-in-the-loop*).
  - Les ingénieurs IA doivent documenter techniquement comment les sorties du système peuvent être interprétées par l'utilisateur.

Interprétabilité vs Explicabilité : Nuances Conceptuelles

- Bien que souvent utilisés comme synonymes, ils désignent deux paradigmes distincts dans la littérature.
- **Interprétabilité (Intrinsèque / *Glass-box*)** :
  - Le mécanisme interne du modèle est compréhensible par design.
  - La structure mathématique *est* l'explication.
  - *Exemples* : Régression logistique (poids des *features*), Arbres de décision peu profonds.
- **Explicabilité (Post-hoc / *Black-box*)** :
  - Le modèle est trop complexe (ex: ResNet, Transformers avec des milliards de paramètres).
  - On crée un algorithme séparé pour extraire une explication après coup.
  - *Objectif* : Donner du sens à la relation entre l'input X et l'output Y, sans forcément comprendre le cheminement des L couches cachées.

La Taxonomie de l'XAI

- Comment classifier les méthodes de notre arsenal d'ingénieur ?
- **Portée de l'explication (Scope)** :
  - *Globale* : Comprendre le comportement général du modèle (quelles *features* impactent le plus la prédiction de diabète en moyenne ?).
  - *Locale* : Expliquer une prédiction spécifique (pourquoi *ce* patient précis a-t-il été diagnostiqué positif aujourd'hui ?).
- **Dépendance à l'architecture** :
  - *Model-Specific* : Exploite les rouages internes de l'architecture (ex: utilisation des gradients ou des poids d'attention).
  - *Model-Agnostic* : Ne traite le modèle que par ses entrées/sorties (API). Fonctionne aussi bien sur un Random Forest que sur un LLM.
- En Deep Learning, le défi majeur est de fournir des explications **locales** fiables pour des prédictions à haut risque.



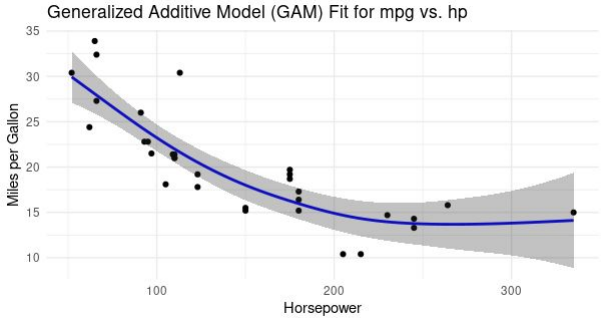
Modèles Intrinsèquement Interprétables (Glass-box)

Motivation : Le Faux Dilemme Performance vs Interprétabilité

- Dans l'industrie, on assume souvent qu'il faut sacrifier la performance (*accuracy*) pour obtenir de la transparence mathématique.
- Pour le diagnostic médical, les prédictions erronées des boîtes noires, même rares, ont un coût humain et légal inacceptable.
- **Le changement de paradigme** : La chercheuse Cynthia Rudin (Duke University) soutient qu'il ne faut pas essayer d'expliquer les boîtes noires, mais plutôt concevoir des modèles transparents par design.
- **L'objectif intrinsèque (*Glass-box*)** : Construire un algorithme dont l'architecture force une prise de décision directement lisible par un expert métier, sans sacrifier le pouvoir prédictif.
- Le défi d'ingénierie : Obtenir les performances des modèles d'état de l'art tout en imposant une structure de décision stricte.

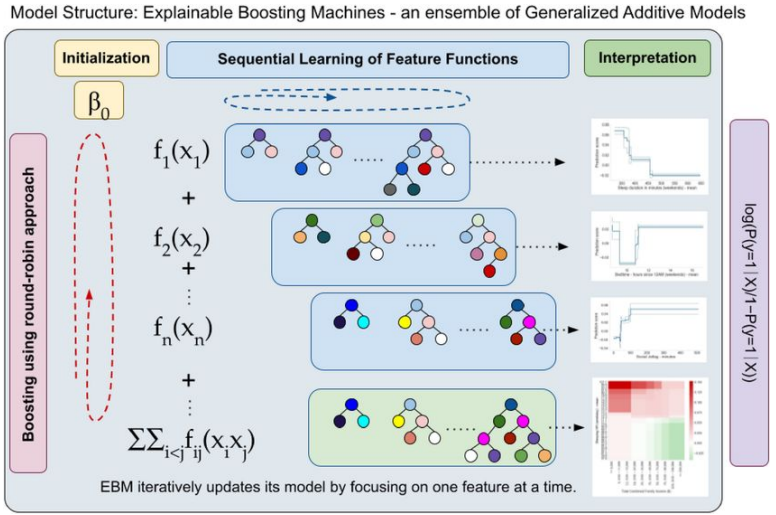
Plongée dans les Generalized Additive Models (GAM)

- Une régression logistique est parfaitement interprétable grâce à ses poids β_i, mais elle échoue à capturer les relations complexes.
- **Le principe du GAM** : Généraliser le modèle linéaire en appliquant des fonctions de forme (*shape functions*) non linéaires à chaque *feature* X_i de manière indépendante.
- **Équation fondamentale** : $g(E[Y])=\beta_0+\sum_{i=1...p} f_i(X_i)$
  - Où g est la fonction de lien (ex: logit pour la classification, identité).
  - Où f_i peut être n'importe quelle fonction (historiquement des *splines*).
- L'interprétabilité est préservée car l'effet de chaque variable sur la prédiction est purement additif et isolé.
- *Application clinique* : En prédisant le risque de mortalité, on isole l'impact de l'âge. Le modèle peut montrer que le risque bondit à 65 ans, stagne, puis rebondit à 80 ans, sans interférence avec d'autres variables.



L'État de l'Art : Explainable Boosting Machines (EBM)

- Les GAM classiques utilisant des *splines* sont difficiles à optimiser sur de grands jeux de données.
- **L'approche EBM (Microsoft InterpretML)** : Remplacer les *splines* par des forêts d'arbres de décision très peu profonds (souvent profondeur 1 ou 2) entraînés via *Gradient Boosting*.
    - Le *Gradient Boosting* est une méthode d'ensemble qui construit un modèle fort en ajoutant séquentiellement des modèles faibles (ici, des arbres très peu profonds) entraînés pour corriger les erreurs résiduelles de l'itération précédente.
- L'astuce technique : L'entraînement se fait *feature* par *feature* dans un mode *round-robin* (tour de rôle) à un taux d'apprentissage très faible.
- Le résultat : Chaque $f_i(X_i)$ est la somme des petits arbres entraînés **uniquement** sur la variable i.
- La prédiction devient un simple *lookup* (recherche dans une table) : on lit le score de l'âge, on y additionne le score de la pression artérielle, etc.

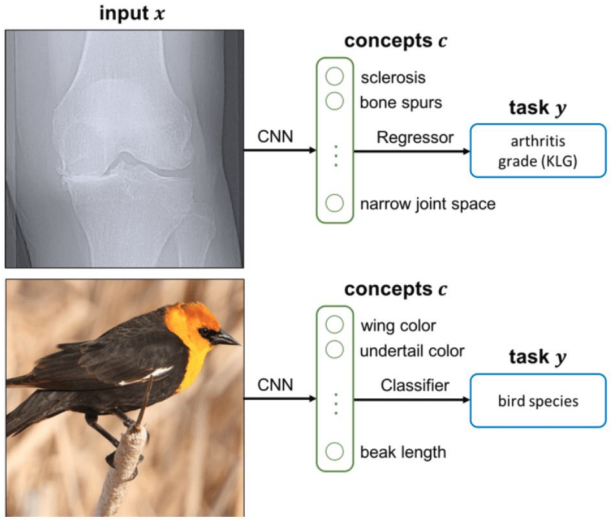


Au-delà de l'additif : Modéliser les Interactions (GA2M)

- **Limitation des GAM standard** : La médecine clinique repose souvent sur la synergie de symptômes (ex: Âge élevé ET Indice de Masse Corporelle (IMC) élevé).
- Un modèle purement additif rate l'effet multiplicateur de cette interaction.
- **GA2M (Generalized Additive Models with Interactions)** : Ajoute des termes d'interaction bidimensionnels à l'équation.
- **Nouvelle équation** : $g(E[Y])=\beta_0+\sum f_i(X_i)+\sum f_{ij}(X_i, X_j)$
- Pour préserver l'interprétabilité, le modèle n'ajoute que les *K* interactions les plus importantes (trouvées via un algorithme rapide de sélection de paires, ex: algorithme *FAST*).

Deep Learning Intrinsèque : Concept Bottleneck Models (CBM)

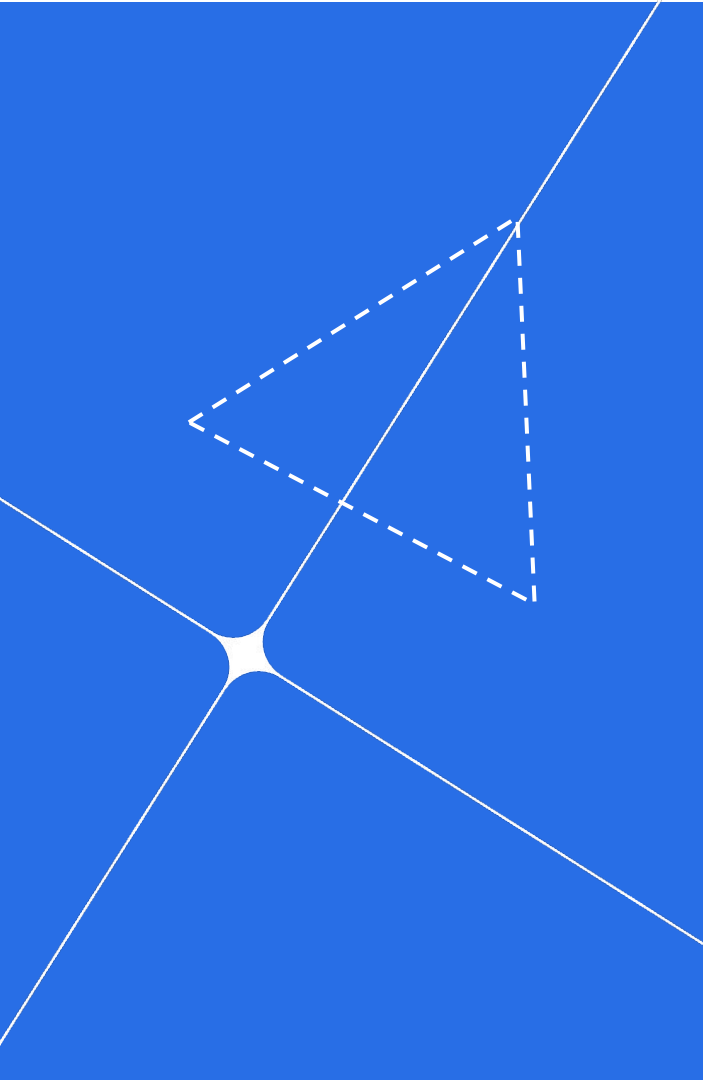
- Comment appliquer cette philosophie intrinsèque à des données non-tabulaires (images, texte) analysées par Deep Learning ?
- **Le principe du goulot d'étranglement (*Bottleneck*)** : Forcer le réseau de neurones à prédire des "concepts humains" de haut niveau avant de donner sa prédiction finale.
- **Architecture en deux étapes successives** :
  1. Un CNN (ex: ResNet) mappe l'image X vers un vecteur de concepts C (ex: présence de nodules, épanchement pleural).
  2. Un modèle linéaire transparent (comme une simple régression) mappe ces concepts C vers le diagnostic final Y.
- Si le modèle prédit une pathologie, le médecin lit la pondération exacte des concepts visuels qui ont déclenché cette décision.
- *Human-in-the-loop* : Si l'étape 1 échoue et hallucine un nodule, le médecin peut forcer la valeur du concept à 0, et l'étape 2 ajustera mathématiquement et automatiquement le diagnostic final.



Implémentation Conceptuelle d'un CBM (PyTorch)

- En pratique, l'architecture sépare l'extraction des concepts (complexe) de la classification (transparente).
- Lors de l'entraînement, la *loss function* doit souvent pénaliser à la fois l'erreur sur les concepts C et l'erreur sur la cible finale Y.

```
class ConceptBottleneckModel(nn.Module):
    def __init__(self, cnn_backbone, num_concepts, num_classes):
        super().__init__()
        # Étape 1 : X -> Concepts (La partie "boîte noire" d'extraction)
        self.feature_extractor = cnn_backbone
        self.fc_concepts = nn.Linear(cnn_backbone.output_dim, num_concepts)

        # Étape 2 : Concepts -> Y (La partie "glass-box" de décision)
        self.classifier = nn.Linear(num_concepts, num_classes)

    def forward(self, x):
        features = self.feature_extractor(x)
        # Prédiction des concepts humains (ex: probabilité de 0 à 1 pour la fièvre)
        concepts = torch.sigmoid(self.fc_concepts(features))
        # Diagnostic final calculé STRICTEMENT à partir des concepts
        logits = self.classifier(concepts)
        return concepts, logits
```



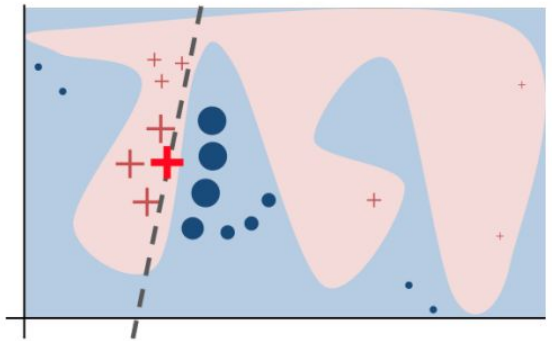
Explicabilité Post-Hoc Agnostique (LIME et SHAP)

Motivation : Expliquer l'Inexplicable (L'Approche Post-Hoc)

- En entreprise, on hérite souvent de modèles "boîtes noires" déjà déployés (ex: un Random Forest massif ou un Transformer analysant des dossiers patients).
- Re-entraîner un modèle intrinsèque (comme un EBM) n'est pas toujours faisable (coût, accès aux données historiques, complexité de l'ingénierie).
- **Le paradigme Post-Hoc Agnostique** : Considérer le modèle complexe f comme un simple "Oracle" (une API).
- On n'a pas accès aux poids ni à l'architecture, seulement aux entrées x et aux prédictions f(x).
- **Objectif** : Faire de la rétro-ingénierie sur la surface de décision du modèle en observant comment il réagit à des perturbations contrôlées de l'entrée.

LIME : Local Interpretable Model-agnostic Explanations

- **L'intuition clé** : La surface de décision globale d'un modèle (ex: limite entre "Sain" et "Malade") est hautement non-linéaire.
- Cependant, à une échelle strictement **locale** (autour d'un patient spécifique x), cette surface peut être approximée par un hyperplan linéaire.
- **Le mécanisme de LIME (Génération d'un Surrogate Local)** :
  1. Prendre l'instance x à expliquer (ex: le dossier clinique de M. Dupont).
  2. Générer un jeu de données local en perturbant x (ajouter du bruit aux variables, ou masquer des mots/superpixels).
  3. Demander à l'Oracle f de prédire toutes ces perturbations.
  4. Pondérer ces échantillons selon leur proximité géographique (distance) avec x original.
  5. Entraîner un modèle linéaire régularisé (Lasso/Ridge) sur ce voisinage.



Limites Mathématiques et Pratiques de LIME

- Bien que très populaire par sa simplicité, LIME présente des vulnérabilités critiques pour des systèmes à haut risque.
- **L'instabilité (Variance d'échantillonnage)** : L'explication dépend des perturbations générées aléatoirement. Deux exécutions de LIME sur le même patient peuvent donner des poids différents.
- **Le fléau de la dimensionnalité** : Comment définir mathématiquement le "voisinage local" (la largeur du kernel de distance) ? C'est un hyperparamètre arbitraire très sensible.
- **Sensibilité aux attaques adversariennes** : Il est prouvé qu'un attaquant peut concevoir un modèle raciste/biaisé qui détecte quand il est "audité" par LIME et fournit des prédictions parfaitement éthiques uniquement sur les données perturbées.

SHAP : Les Fondations Axiomatiques (Théorie des Jeux)

- Pour pallier l'arbitraire de LIME, SHAP (*SHapley Additive exPlanations*) s'appuie sur la théorie des jeux coopératifs (Prix Nobel d'Économie 2012 pour Lloyd Shapley).
- **L'analogie du jeu coopératif** :
  - Le *Jeu* : Prédire le risque de diabète d'un patient.
  - Les *Joueurs* : Les *features* du patient (Âge, IMC, Tension).
  - Le *Gain (Payout)* : La différence entre la prédiction pour ce patient (ex: 85%) et la prédiction moyenne du dataset (ex: 20%).
- **Le problème de Shapley** : Comment répartir équitablement ce gain (+65%) entre les joueurs, sachant qu'ils interagissent (l'IMC et l'Âge ont une forte synergie) ?
- SHAP est la *seule* méthode garantissant la répartition parfaite selon 4 axiomes stricts (Efficacité, Symétrie, Joueur Factice, Additivité).

SHAP : Calcul des Contributions Marginales

- Pour trouver la valeur exacte de Shapley ϕ_i d'une *feature* (ex: l'Âge), il faut évaluer sa **contribution marginale**.
- **La formule conceptuelle** : On mesure l'impact de l'Âge en l'ajoutant à *toutes les combinaisons possibles* (coalitions) des autres variables.
  - *Coalition 1* : Modèle(IMC, Âge) - Modèle(IMC)
  - *Coalition 2* : Modèle(Tension, Âge) - Modèle(Tension)
  - *Coalition 3* : Modèle(IMC, Tension, Âge) - Modèle(IMC, Tension)
- On fait la moyenne pondérée de toutes ces contributions marginales.
- **Le mur de la complexité** : Pour M variables, il y a 2^M coalitions à évaluer par l'Oracle. C'est calculatoirement intraitable en $O(2^M)$.
- **Les solutions industrielles** :
  - *TreeSHAP* : Algorithme exact en temps polynomial spécifiquement conçu pour les arbres (XGBoost, Random Forest).
  - *KernelSHAP* : Une version modifiée de LIME qui retrouve les valeurs de Shapley par approximation.

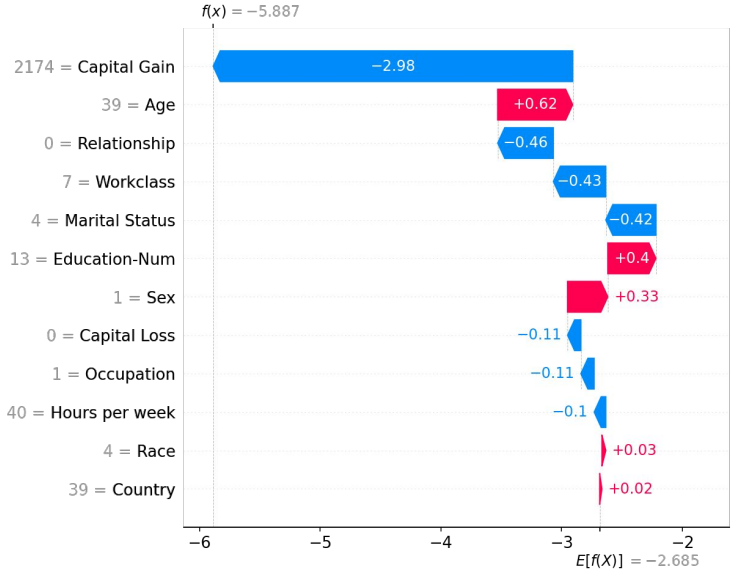
L'Écosystème SHAP en Pratique (Code & Visualisation)

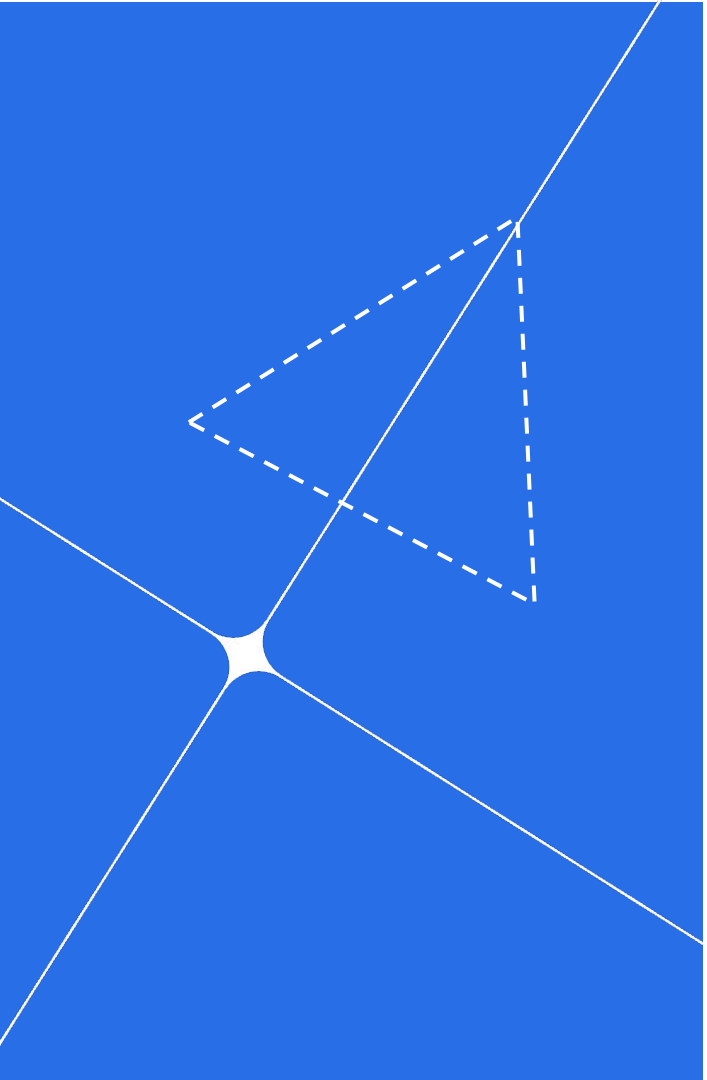
- La force de SHAP est de fournir à la fois une explication locale stricte et de l'agréger pour une vue globale.
- **Le *Waterfall Plot* (Local)** : Visualise la décomposition mathématique de la prédiction pour *un* patient donné.
  - *Point de départ (E[f(X)])* : La prédiction moyenne du modèle sur le dataset de référence (la *base value*, ex: 20% de risque).
  - *Contributions (ϕi)* : Chaque variable est une barre qui ajoute (rouge/droite) ou soustrait (bleu/gauche) sa valeur de Shapley exacte.
  - *Point d'arrivée (f(x))* : La somme de la *base value* et de toutes les contributions correspond *strictement* à la prédiction finale du modèle pour ce patient (ex: 60%).
- **Le *Summary Plot* (Global)** : Agrège les valeurs SHAP de tous les patients. Révèle l'importance globale des variables et la direction de leur impact.

```
import shap

# 1. Initialiser l'explainer sur le modèle "boîte noire" (API agnostique)
# L'explainer a besoin d'un background dataset pour calculer les valeurs moyennes
explainer = shap.Explainer(black_box_model.predict, background_data)

# 2. Calculer les valeurs de Shapley pour les nouveaux patients
shap_values = explainer(new_patients_data)

# 3. Générer le graphique local (pour le patient 0)
shap.plots.waterfall(shap_values[0])
```

L'Écosystème SHAP en Pratique (Code & Visualisation)

- La force de SHAP est de fournir à la fois une explication locale stricte et de l'agréger pour une vue globale.
- **Le *Waterfall Plot* (Local)** : Visualise la décomposition mathématique de la prédiction pour *un* patient donné.
  - *Point de départ (E[f(X)])* : La prédiction moyenne du modèle sur le dataset de référence (la *base value*, ex: 20% de risque).
  - *Contributions (ϕ_i)* : Chaque variable est une barre qui ajoute (rouge/droite) ou soustrait (bleu/gauche) sa valeur de Shapley exacte.
  - *Point d'arrivée (f(x))* : La somme de la *base value* et de toutes les contributions correspond *strictement* à la prédiction finale du modèle pour ce patient (ex: 60%).
- **Le *Summary Plot* (Global)** : Agrège les valeurs SHAP de tous les patients. Révèle l'importance globale des variables et la direction de leur impact.





Explicabilité Spécifique au Deep Learning (Vision & Gradients)

Motivation : Les Limites de l'Agnostique en Vision par Ordinateur

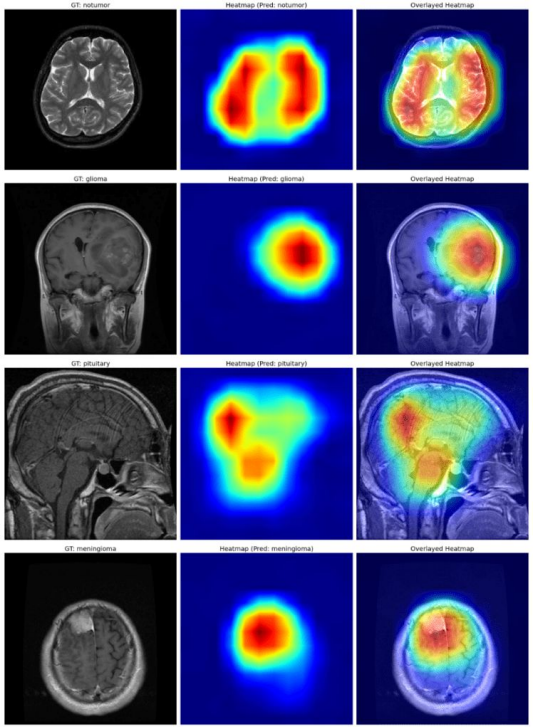
- Les méthodes comme SHAP ou LIME traitent le modèle comme une "boîte noire" (requêtes API répétées).
- Sur une IRM médicale haute résolution (ex: 1024×1024 pixels), évaluer des milliers de perturbations locales est extrêmement coûteux en calcul ($O(2^M)$).
- Pire, ces méthodes agnostiques ignorent l'a priori spatial (topologie) appris par les convolutions du CNN.
- **Le changement de paradigme (*White-box*)** :
    - Puisque nous avons accès aux poids du réseau, pourquoi ne pas simplement utiliser la *backpropagation* pour faire circuler l'information de la décision finale Y jusqu'à l'image d'entrée X ?

Les Gradients Simples (Saliency Maps) et le Piège de la Saturation

- **L'intuition de base (Simonyan et al., 2013)** : La carte de saillance est simplement la dérivée partielle de la probabilité de la classe cible Y_c par rapport à chaque pixel x_i de l'image d'entrée : $S_c = |\partial Y_c / \partial x_i|$.
- Elle répond à la question : "Si je modifie ce pixel d'une valeur infinitésimale, la probabilité de tumeur va-t-elle changer ?"
- **Le défaut fatal (Le problème de Saturation)** : Dans un réseau bien entraîné, les fonctions d'activation (Sigmoid, ReLU) finissent par plafonner.
- Si un motif d'opacité pulmonaire est déjà parfaitement détecté, l'intensifier ne changera plus la probabilité finale (la dérivée devient nulle : $\partial Y_c / \partial x_i = 0$).
- *Conséquence désastreuse* : Les régions de l'image les *plus importantes* pour le diagnostic peuvent recevoir un gradient de zéro et disparaître de l'explication !

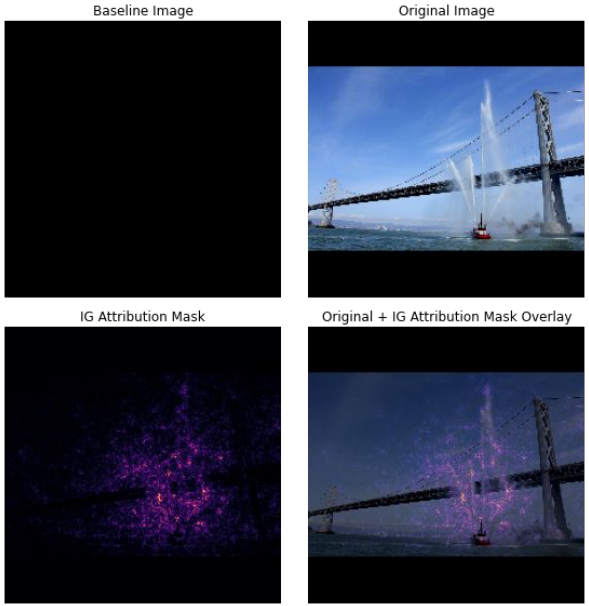
Grad-CAM : L'Exploitation Sémantique des Couches Convolutives

- Pour éviter le bruit des gradients au niveau des pixels, Grad-CAM (*Gradient-weighted Class Activation Mapping*) s'arrête à la dernière couche convolutive (qui possède la sémantique spatiale la plus riche).
- **Le mécanisme interne** :
  1. On calcule les gradients de la classe cible par rapport aux K *feature maps* A_k de cette dernière couche.
  2. On applique un *Global Average Pooling* sur ces gradients pour obtenir un poids d'importance α_k^c pour chaque filtre.
  3. On réalise une combinaison linéaire pondérée : $\sum_k \alpha_k^c A_k$.
  4. On applique une activation ReLU : $L_{Grad-CAM}^c = ReLU(\sum_k \alpha_k^c A_k)$.
- **L'astuce du ReLU** : Il supprime l'influence des pixels qui poussent vers *d'autres* classes, ne conservant que les zones ayant une influence **positive** sur le diagnostic actuel.
- *En clinique* : La *heatmap* générée est interpolée à la taille de l'image originale pour s'assurer que le modèle regarde bien le lobe pulmonaire, et non les annotations de l'appareil de radiographie.



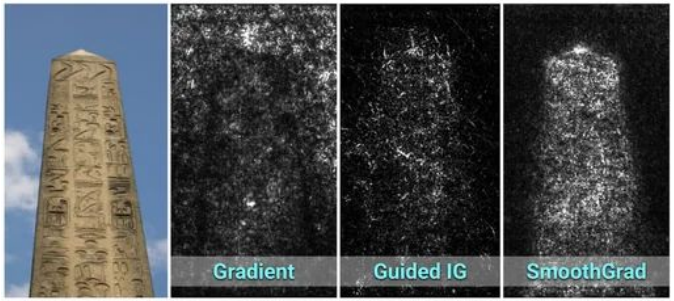
Integrated Gradients (IG) : La Solution Axiomatique par l'Intégration

- Comment résoudre mathématiquement le problème de saturation des gradients bruts tout en garantissant une attribution exacte (comme le ferait SHAP) ?
- **L'idée (Sundararajan et al., 2017)** : Ne pas évaluer le gradient en un seul point, mais l'intégrer le long d'un chemin rectiligne entre une image de référence neutre (la *baseline*, ex: une image totalement noire) et l'image médicale étudiée x.
- **La Formule Mathématique Intégrale** :
  - $IG_i(x) = (x_i - x_i') \times \int_{\alpha=0..1} \partial F(x' + \alpha(x - x'))/\partial xi \, d\alpha$
  - i = feature i (ex, le pixel i) et F = notre réseau de neurones
- *Axiome de Complétude* : La somme de tous les IG des pixels est strictement égale à la différence entre la prédiction du modèle sur l'image x et sur la baseline x'.
- *En pratique* : On approxime l'intégrale continue par une somme de Riemann discrète (en générant 50 à 300 images interpolées entre le noir et la radio).



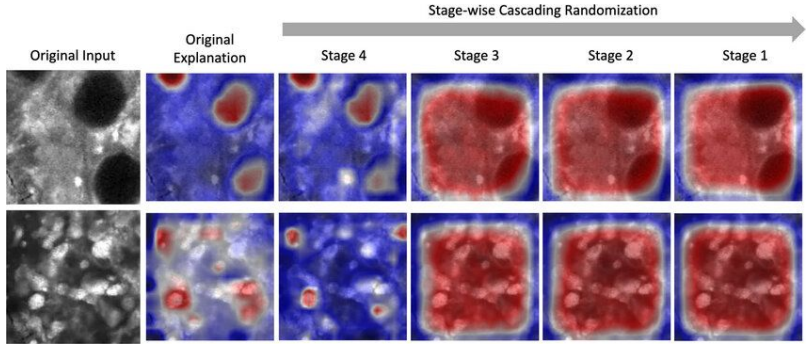
SmoothGrad : Lutter contre le Bruit Visuel des Gradients

- Les cartes de *saliency* brutes ou même les *Integrated Gradients* peuvent être visuellement très bruitées (des pixels isolés s'allument partout sur la radiographie).
- **Cause mathématique** : La fonction de prédiction d'un réseau profond fluctue fortement localement. Les dérivées partielles sont donc instables.
- **La solution SmoothGrad (Smilkov et al., 2017)** : Ajouter du bruit gaussien à l'image d'entrée de multiples fois, et moyenner les gradients résultants.
- **Formule** : $M_c(x) = 1/N * \sum_{i=1..N} \partial Y_c(x + N(0,\sigma^2))/\partial x$
- L'idée est d'évaluer la dérivée dans un voisinage stochastique plutôt qu'en un point unique, ce qui agit comme un lissage géométrique (*smoothing*).
- *Avantage d'ingénierie* : C'est une méthode "wrapper". On peut l'appliquer par-dessus n'importe quelle méthode basée sur les gradients (ex: on peut faire du *Smooth Integrated Gradients*).



Le Test Acide : Les "Sanity Checks" des Cartes de Saliency

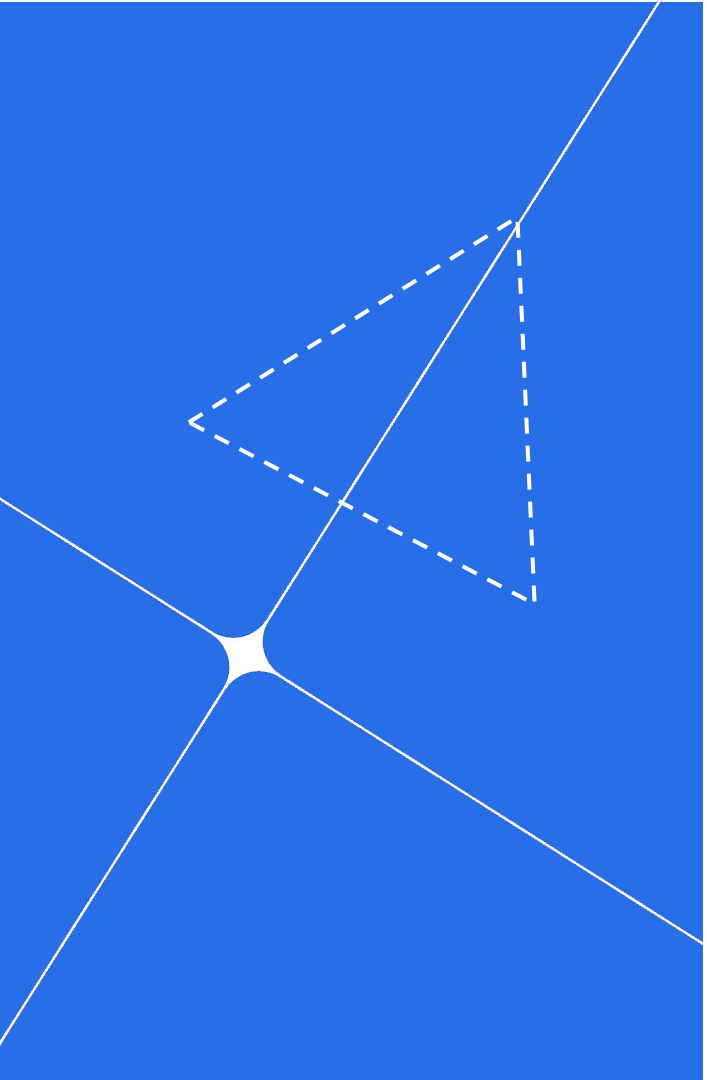
- **L'illusion du biais de confirmation** : Une *heatmap* qui met en évidence les poumons nous semble "bonne" car elle correspond à notre a priori humain, mais explique-t-elle *vraiment* le modèle ?
- **L'expérience de Adebayo et al. (2018)** : Comment prouver qu'une méthode XAI n'est pas juste un détecteur de contours d'images déguisé ?
- **Le test de randomisation en cascade (Cascading Randomization)** :
    1. Prendre le modèle CNN entraîné et générer l'explication (ex: une pneumonie).
    2. Réinitialiser aléatoirement les poids de la toute dernière couche (le modèle devient alors stupide).
    3. Régénérer l'explication. Recommencer en remontant couche par couche jusqu'à l'entrée.
- **Le diagnostic MLOps** :
    - Si l'explication visuelle *ne change pas* ou peu malgré un modèle aux poids aléatoires, la méthode XAI est invalide (elle ignore ce que le modèle a appris).
    - *Résultat* : Des méthodes comme *Guided Backpropagation* échouent à ce test. Grad-CAM et Integrated Gradients le passent avec succès (l'image devient du bruit).



L'Écosystème MLOps : Implémentation avec Captum (PyTorch)

- Implémenter ces intégrales ou ces opérations sur les gradients manuellement est risqué. L'industrie utilise Captum (la bibliothèque officielle Meta pour PyTorch).
- Captum gère automatiquement les *forward/backward hooks* de PyTorch pour extraire les tenseurs internes sans modifier l'architecture source.

```
from captum.attr import IntegratedGradients

# 1. Instancier l'algorithme IG sur notre modèle PyTorch pré-entraîné
ig = IntegratedGradients(resnet_medical_model)

# 2. Définir l'image cible (X) et la baseline (X')
# Pour de l'imagerie, la baseline est souvent un tenseur de zéros (image noire)
input_image = image_tensor.requires_grad_()
baseline = torch.zeros_like(input_image)

# 3. Calculer les attributions (L'intégrale de Riemann avec 50 étapes)
# target=1 correspond à la classe diagnostiquée (ex: Tumeur maligne)
attributions, delta = ig.attribute(
    input_image,
    baselines=baseline,
    target=1,
    n_steps=50,
    return_convergence_delta=True
)
```



La Frontière Actuelle : Interprétabilité des LLMs (Transformers)

Motivation : Le Défi de l'Auto-Régression Clinique

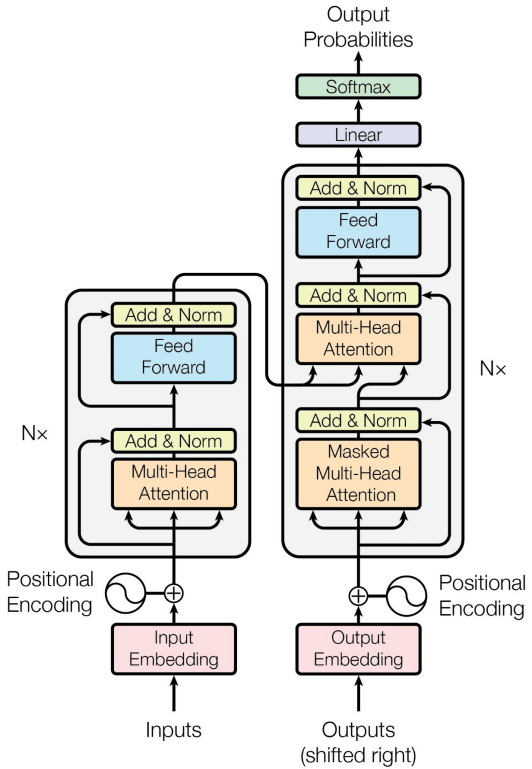
- Les méthodes post-hoc (SHAP, IG) ont été pensées pour des sorties continues ou des espaces de classification fermés.
- Un LLM médical génère du texte de manière auto-régressive : la prédiction du token t_i conditionne celle du token t_{i+1}. L'espace d'état est immense ($|V|^{\text{sequence\_length}}$).
- **Le problème de l'hallucination** : Si le modèle génère "Traitement : [Chimiothérapie]", s'appuie-t-il sur les antécédents de la radiographie mentionnés dans le *prompt*, ou sur un biais mémorisé dans ses poids ?
- L'objectif n'est plus d'expliquer une "probabilité", mais de faire du *Reverse-Engineering* sur les algorithmes appris par les matrices de poids (décomposer le réseau en "circuits" logiques).

L'Illusion de l'Attention : Une Perspective Algébrique

- L'intuition commune consiste à regarder les poids d'attention : $A=\text{Softmax}(QK^T/\text{sqrt}(d_k))$. Si le token cible a un fort poids sur le token "Tumeur", on conclut qu'il est la cause.
- **Le mélange des tokens (*Token Mixing*)** : L'équation complète est $\text{Attention}(X)=A(XW_V)$.
- La matrice A dicte seulement *quelle proportion* du vecteur de valeur V est déplacée, mais ce vecteur V a déjà été transformé et mélangé avec d'autres concepts par les couches précédentes.
- **Le théorème de Jain & Wallace** : Les poids de la matrice A ne sont pas causalement liés à l'importance des *features*. Un poids d'attention élevé peut simplement signifier que le modèle déplace un token neutre (comme la ponctuation) pour des raisons de syntaxe.

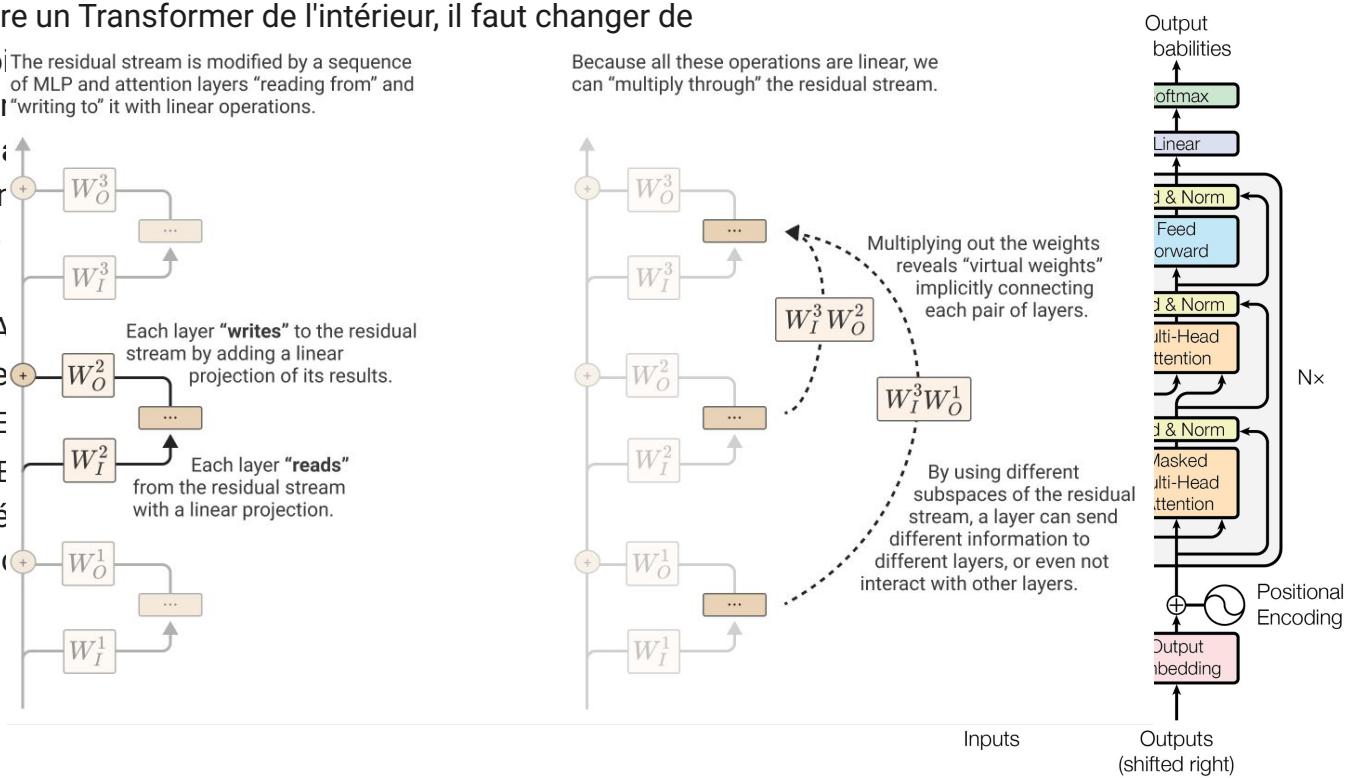
Le Paradigme du Courant Résiduel (Residual Stream)

- Pour comprendre un Transformer de l'intérieur, il faut changer de perspective : voir le réseau comme un canal de communication central de dimension d_{model}.
- **L'équation fondamentale de l'état caché** : $x_l = x_{l-1} + \text{Attention}(x_{l-1}) + \text{MLP}(x_{l-1})$
- Le vecteur x_l (le courant résiduel) est la mémoire vive (RAM) du modèle.
- Les couches d'Attention et les couches MLP ne sont que des opérations de lecture/écriture :
  - *Lecture* : Elles projettent le flux résiduel dans un sous-espace.
  - *Écriture* : Elles additionnent leurs résultats au flux résiduel.
- L'interprétabilité consiste donc à décoder géométriquement ce qui est "écrit" dans ce vecteur x_l à chaque étape de la génération du diagnostic.



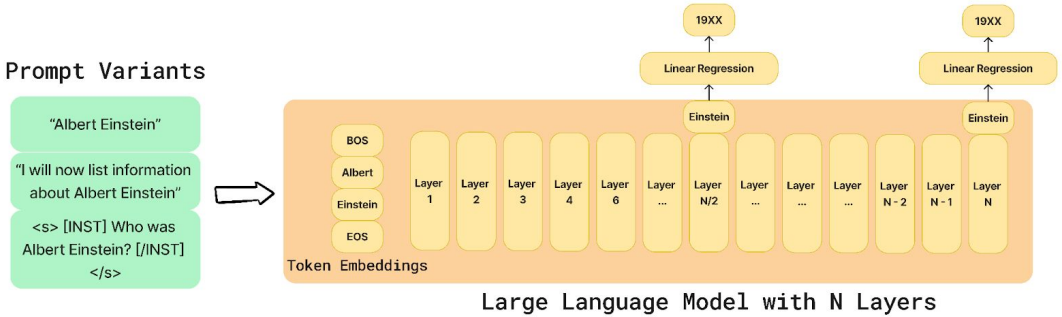
Le Paradigme du Courant Résiduel (Residual Stream)

- Pour comprendre un Transformer de l'intérieur, il faut changer de perspective : voi
central de dimer
- **L'équation fond**
$x_l = x_{l-1} + \text{Attentior}$
- Le vecteur x_l (le
modèle.
- Les couches d'A
opérations de le
  - *Lecture* : E
  - *Écriture* : E
- L'interprétabilité
est "écrit" dans
diagnostic.



Probing Linéaire : Sonder le Courant Résiduel

- Si une information (ex: le concept "Malignité") est utilisée par le modèle, elle doit exister sous forme de direction vectorielle dans le flux résiduel x_l.
- **La méthode du *Linear Probing*** : Entraîner un classifieur linéaire extrêmement simple (pour ne pas rajouter de complexité) sur les activations internes.
- $P(\text{Concept} \mid x_l) = \sigma(W_{probe}^{T} x_l + b)$
- Si ce petit classifieur atteint une haute précision sur un jeu de données médical annoté, on prouve que la couche l a encodé linéairement ce concept abstrait.
- **L'hypothèse des représentations linéaires** : Les LLMs structurent les concepts de haut niveau non pas comme des points complexes, mais comme des directions géométriques simples dans l'espace latent.



Mechanistic Interpretability : Décomposer l'Attention (OV / QK)

- Le framework d'Elhage et al. (Anthropic) décompose chaque tête d'attention en deux circuits mathématiques indépendants.
  1. **Le Circuit QK (Query-Key)** : $W_Q W_K^T$. Il calcule *l'attention*. Il lit le flux résiduel pour décider "À quel token source dois-je prêter attention ?".
  2. **Le Circuit OV (Output-Value)** : $W_V W_O$. Il détermine *l'information à déplacer*. Il lit le token source et écrit l'information pertinente dans le flux résiduel du token de destination.
- *Analyse clinique* : Si le modèle doit copier l'âge du patient ("Patient de 45 ans ... Âge : 45"), le Circuit QK s'active sur les chiffres proches du mot "Patient", et le Circuit OV extrait la valeur numérique brute pour l'écrire dans le flux.

Mechanistic Interpretability : Décomposer l'Attention (OV / QK)

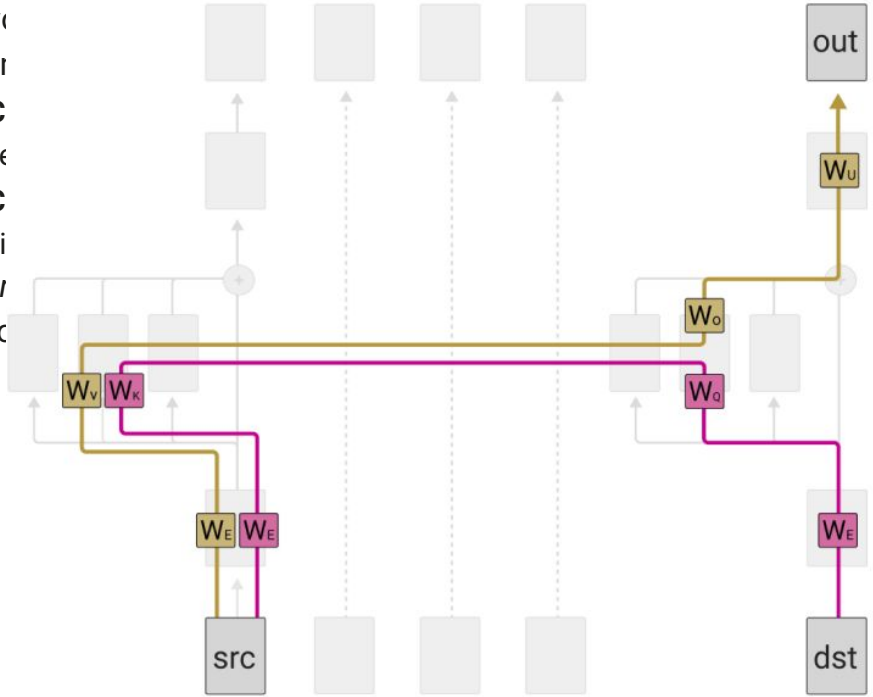
- Le framew
  indépendar
  1. **Le C**
     prête
  2. **Le C**
     perti
- *Analyse cli*
  chiffres pr

atiques

ource dois-je

nformation

t s'active sur les
flux.



The **OV ("output-value") circuit** determines how attending to a given token affects the logits.

$$W_U W_O W_V W_E$$

The **QK ("query-key") circuit** controls which tokens the head prefers to attend to.

$$W_E^T W_Q^T W_K W_E$$

Le Mur de la Superposition et de la Polysémantique

- Historiquement, on cherchait des "neurones" individuels responsables de concepts (ex: le neurone 4092 s'active pour "Cancer").
- **Le phénomène de Superposition** : Un LLM possède moins de dimensions d_{model} (ex: 4096) qu'il n'y a de concepts médicaux dans le monde.
- Pour contourner cette limite géographique, le modèle compresse l'information : un concept n'est pas un neurone, mais une combinaison linéaire de neurones.
- **Polysémantisme** : Par conséquent, un seul neurone MLP peut s'activer à la fois pour "Diabète", "Syntaxe HTML", et "Noms de villes russes", selon le contexte. Analyser des neurones isolés est donc une impasse technique.

Sparse Autoencoders (SAEs) : L'État de l'Art (2023+)

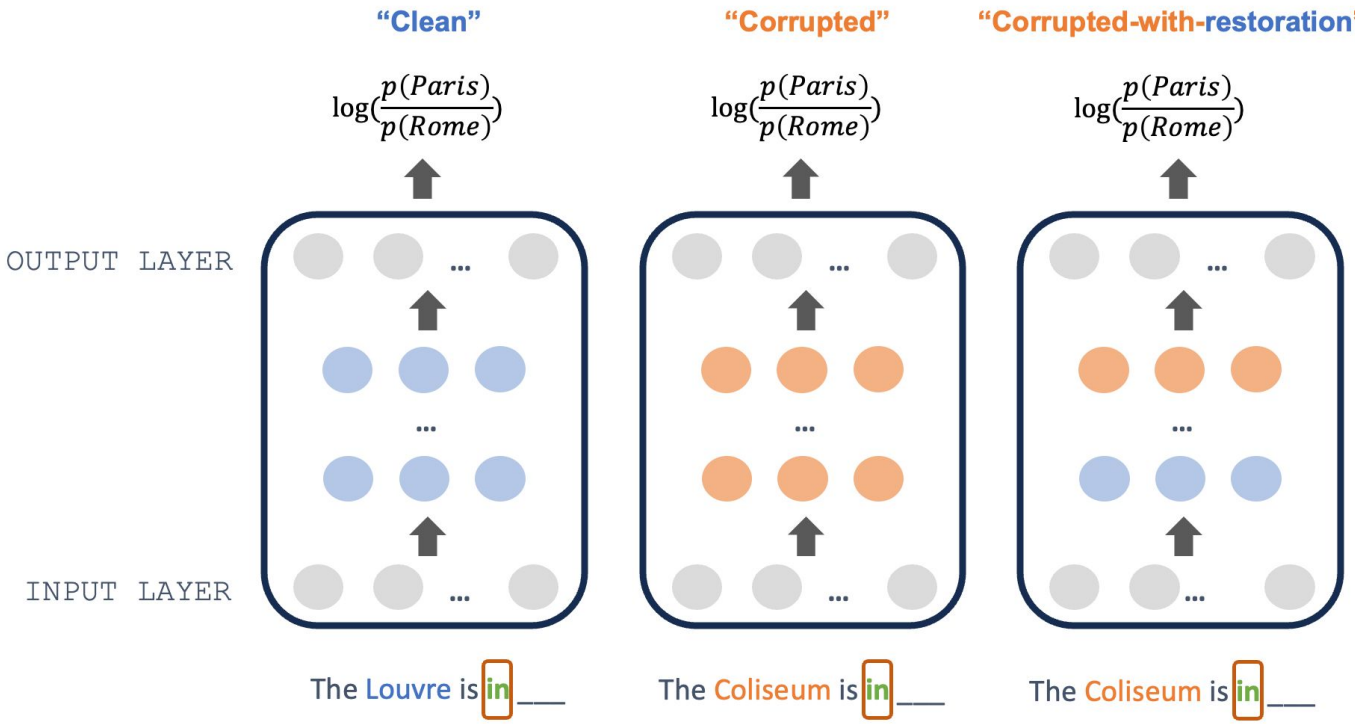
- Comment démêler mathématiquement ces concepts superposés ? La solution actuelle repose sur l'apprentissage non supervisé d'un dictionnaire.
- **Le mécanisme des SAEs** : On entraîne un auto-encodeur sur des millions d'activations internes x_l d'un LLM gelé.
- $f(x)=W_{dec}ReLU(W_{enc}x+b_{enc})+b_{dec}$
- **La contrainte de parcimonie (Sparsity)** : On ajoute une pénalité L1 intense sur les activations cachées de l'auto-encodeur.
- *Résultat* : Le SAE force l'espace dense et polysémantique x_l à se projeter dans un espace beaucoup plus grand (ex: 100 000 dimensions), mais où seules 5 ou 10 directions s'activent en même temps.
- Ces nouvelles directions isolées correspondent à des concepts humains purs, désintriqués et directement interprétables (ex: le vecteur "Symptômes respiratoires").

Activation Patching (Causal Tracing)

- Le probing ou les SAEs ne montrent que des corrélations. L'*Activation Patching* permet de prouver causalement le rôle d'un circuit ou d'une direction vectorielle.
- **Le protocole d'intervention en 3 étapes** :
  - *Run propre (Cache)* : Passer un prompt ("Patient allergique à la pénicilline") et sauvegarder les tenseurs d'activation de toutes les couches.
  - *Run corrompu* : Passer un prompt modifié ("Patient allergique au paracétamol").
  - *Intervention (Patching)* : Lors du passage du prompt propre, on intercepte l'activation à la couche L et on la remplace (partiellement, sur une tête en particulier par exemple) par celle du prompt corrompu.
- Si la prédiction finale bascule, on a localisé *exactement* où et quand le modèle prend sa décision clinique.

Activation Patching (Causal Tracing)

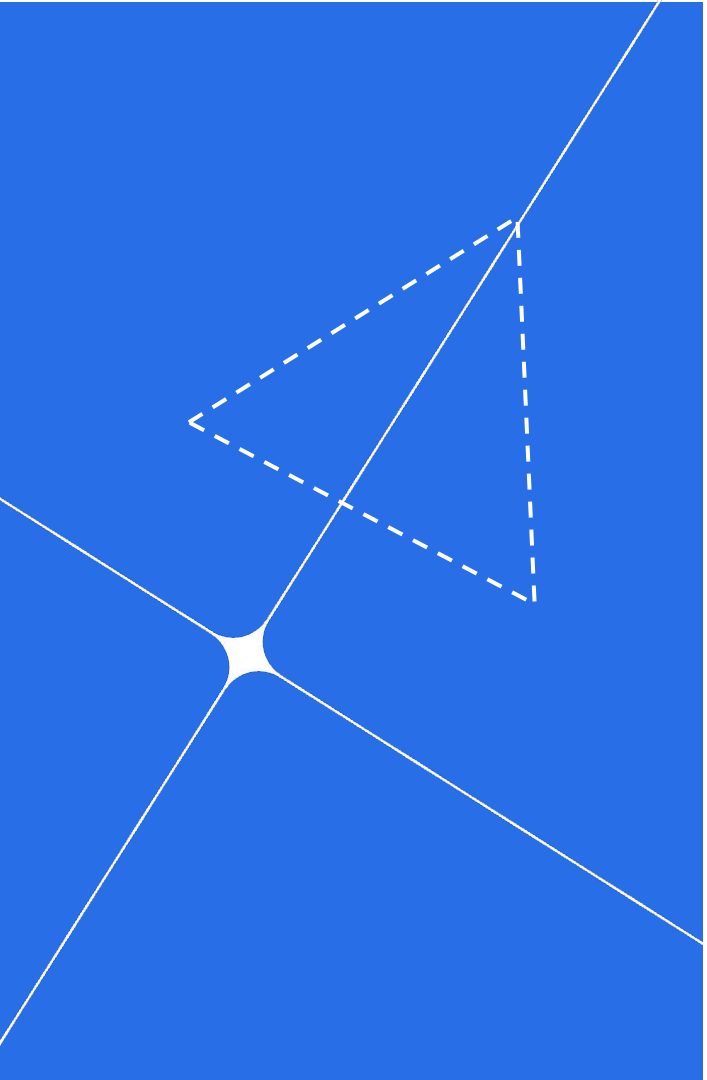
- Le prob … ement le rôle d'un circ…
- **Le proto…**
  - *F… c…*
  - *F… …eurs*
  - *I… r…*
- Si la pré… … et on la



Implémentation : Interception de Tenseurs (PyTorch)

En ingénierie, cette technique nécessite de maîtriser les *forward hooks* pour manipuler le graphe de calcul dynamique sans recompiler le modèle.

```python
import torch

# Dictionnaire pour stocker (cacher) les activations du "Run Corrompu"
cache = {}

def save_activation_hook(module, input, output, layer_name):
    cache[layer_name] = output.detach()
    return output

def patch_activation_hook(module, input, output, layer_name):
    # On écrase le tenseur actuel par le tenseur corrompu caché
    # On peut cibler précisément la dimension temporelle (token) ou spatiale
    output[:, target_token_idx, :] = cache[layer_name][:, target_token_idx, :]
    return output

# 1. Run Corrompu : Sauvegarde des tenseurs de la couche 12
handle = model.layers[12].register_forward_hook(...)
model(corrupted_prompt)
handle.remove()

# 2. Run Propre avec Intervention : Remplacement à la volée
handle = model.layers[12].register_forward_hook(...)
predictions = model(clean_prompt)
handle.remove()
```



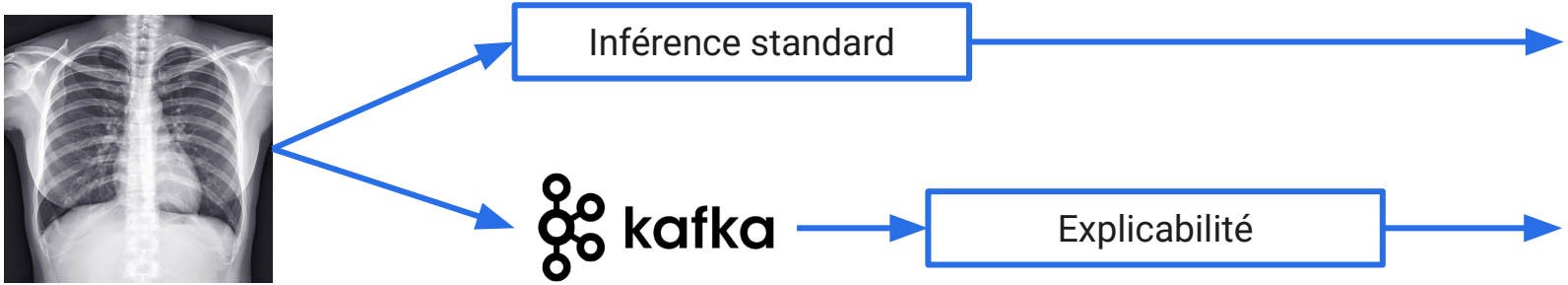
Outils MLOps & TP

Motivation : L'XAI comme Composant MLOps Critique

- L'explicabilité n'est pas qu'un outil de débogage pour la phase d'entraînement, c'est un livrable de production.
- **Le coût de l'inférence** : Générer une explication SHAP ou un *Integrated Gradient* est souvent 10 à 100 fois plus lourd en calcul que la prédiction initiale.
- **L'exigence légale (AI Act)** : Chaque prédiction à haut risque (ex: refus de traitement médical) doit être archivée avec son explication locale pour un éventuel audit.
- *Défi d'ingénierie* : Comment servir des prédictions en temps réel (latence < 100ms) tout en fournissant des explications mathématiquement lourdes au médecin ?

Architecture Logicielle : L'XAI en Production

- La solution standard est de découpler la prédiction de l'explication via une architecture asynchrone (microservices).
- **Le flux synchrone (Critique)** :
  - Le médecin envoie la radiographie.
  - Le service d'inférence (TensorRT/ONNX) renvoie immédiatement le diagnostic (ex: "Pneumonie 98%").
- **Le flux asynchrone (Heavy Compute)** :
  - Un message est envoyé dans une queue (ex: Kafka/RabbitMQ) avec l'ID du patient.
  - Un *worker* GPU dédié à l'XAI intercepte le message, charge les tenseurs via Captum, et calcule l'explication (Grad-CAM/IG).
  - L'explication finale (la *heatmap*) est sauvegardée en base de données et poussée sur l'interface web du médecin quelques secondes plus tard.

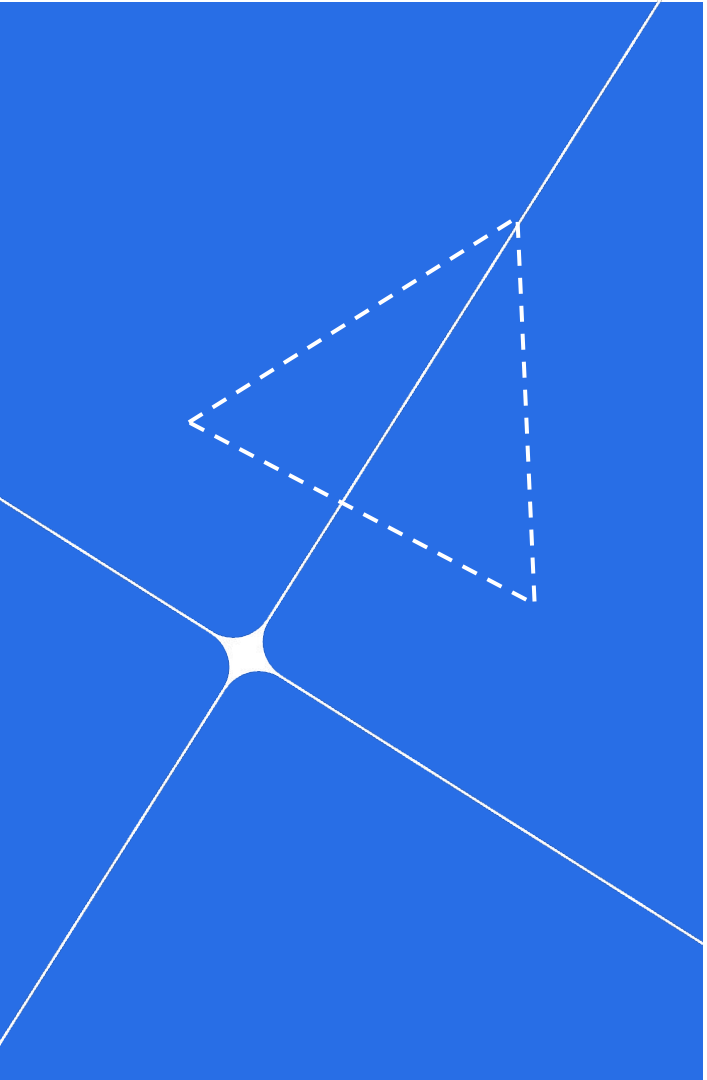


Synthèse : Quelle Méthode Choisir et Quand ?

- Le choix de la méthode dépend strictement de l'accès au modèle (*White/Black box*) et de la modalité des données.
- **Données Tabulaires (Dossiers cliniques, SQL)** :
  - *Accès total au design* : Créer un EBM (*Glass-box*).
  - *Modèle d'arbre pré-existant (XGBoost, Random Forest)* : Utiliser **TreeSHAP** (exact et ultra-rapide).
  - *Boîte noire stricte (API distante)* : Utiliser KernelSHAP ou LIME (avec prudence).
- **Vision par Ordinateur (Radiographies, IRM)** :
  - *Modèle PyTorch/TensorFlow local* : Utiliser **Integrated Gradients** (si besoin de précision au pixel) ou **Grad-CAM** (si besoin de la sémantique de l'objet).
- **NLP & Grands Modèles de Langage (LLMs)** :
  - *Analyse de surface* : Utiliser LIME sur les mots (masquage).
  - *Audit de sécurité profond* : Utiliser le *Probing* ou l'*Activation Patching*.

Outils pour le TP : SHAP et Captum

- Dans la session pratique qui suit, nous allons manipuler les deux bibliothèques standards de l'industrie.
- **1. La librairie shap (Agnostique & Tabulaire)** :
  - Nous utiliserons un jeu de données de dossiers patients réels (risques cardiovasculaires).
  - Entraînement d'un modèle d'ensemble et génération de *Summary Plots* et *Waterfall Plots* pour justifier une décision clinique.
- **2. La librairie captum (Spécifique Deep Learning & PyTorch)** :
  - Créée par Meta, elle s'intègre nativement aux graphes de calcul PyTorch.
  - Nous chargerons un ResNet pré-entraîné sur des radiographies pulmonaires.
  - Objectif : Implémenter *Integrated Gradients*, extraire les attributions de chaque pixel, et superposer la *heatmap* sur la radio originale pour déceler les biais (effet *Clever Hans*).



En route vers le TP