Computer Vision - Interprétabilité

Cours inversé

Joey Skaf Emma Mendizabal

Centrale Méditerranée

Saliency maps for interpretability:
CAM, gradCAM, ScoreCAM, Opti-CAM, TIBAV

Plan

- Modèles de Deep Learning : pose des problèmes dans compréhension des décisions prises par l'IA.

- Modèles de Deep Learning : pose des problèmes dans compréhension des décisions prises par l'IA.
- Nécessité parfois de compréhension des motivations des décisions prédites !

- Modèles de Deep Learning : pose des problèmes dans compréhension des décisions prises par l'IA.
- Nécessité parfois de compréhension des motivations des décisions prédites !

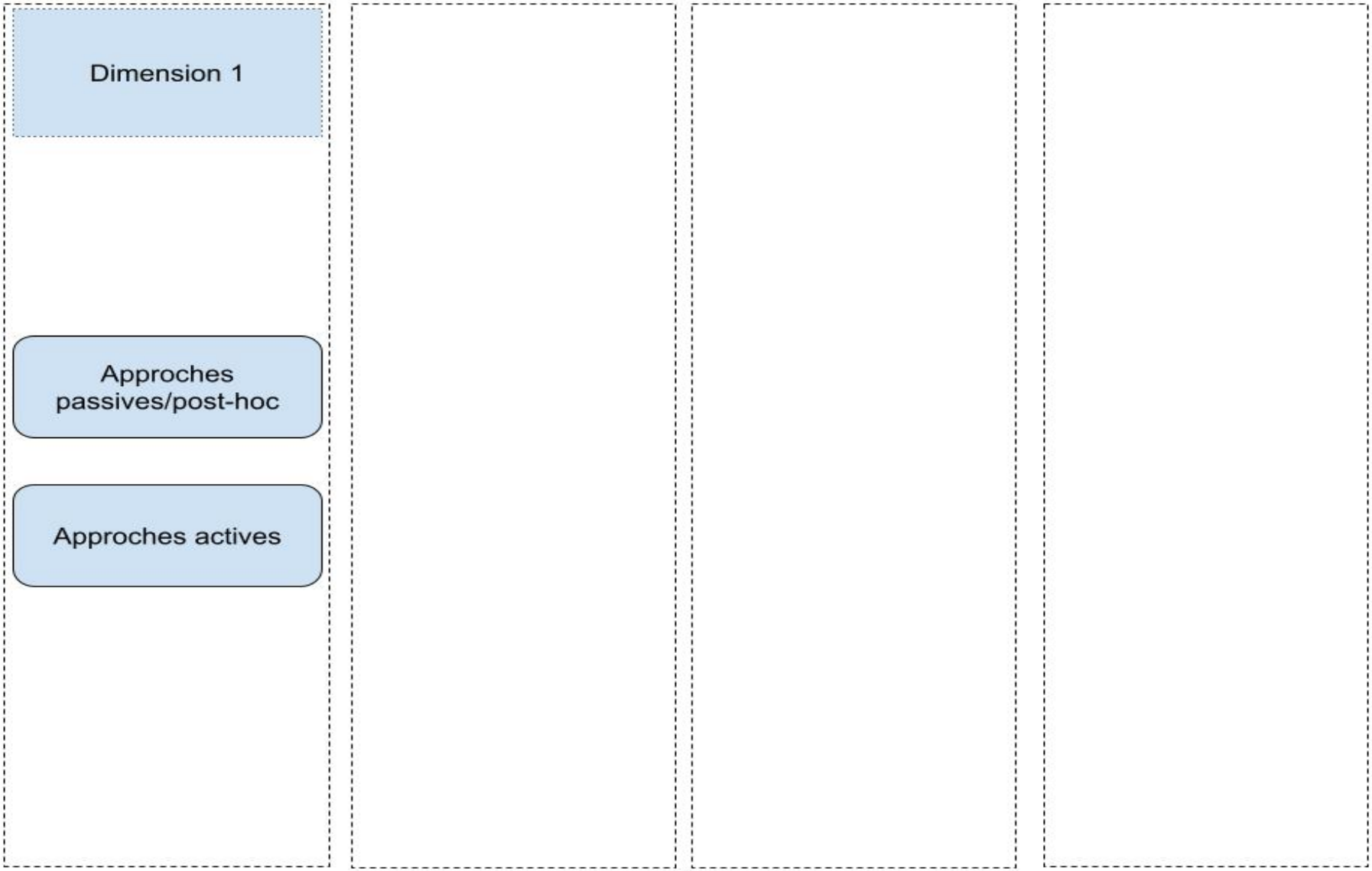
Example



Figure: Exemples de domaines où l'interprétabilité est un gros enjeu

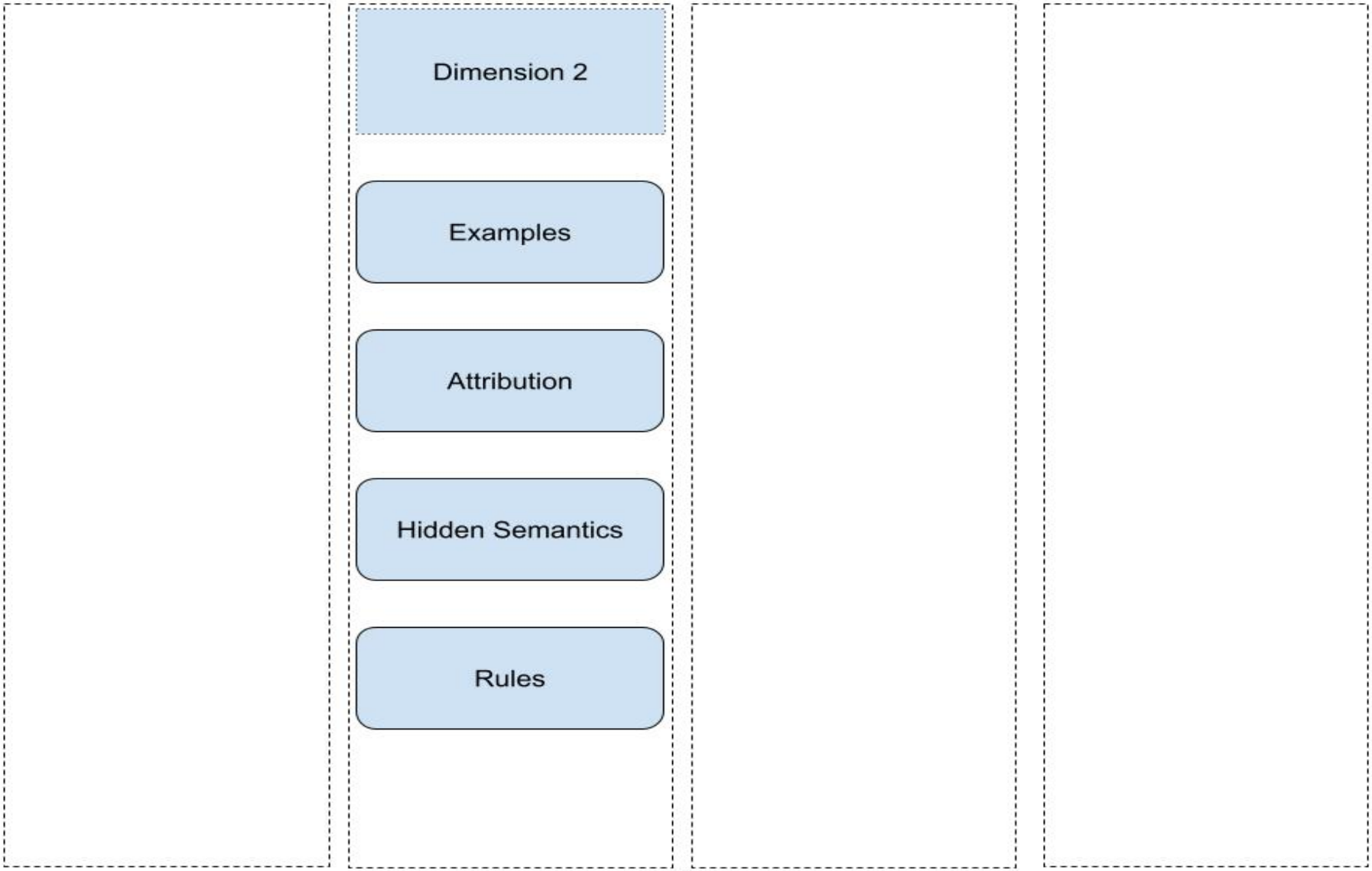
Arborescence du domaine

Modèles transparents vs. post-hoc



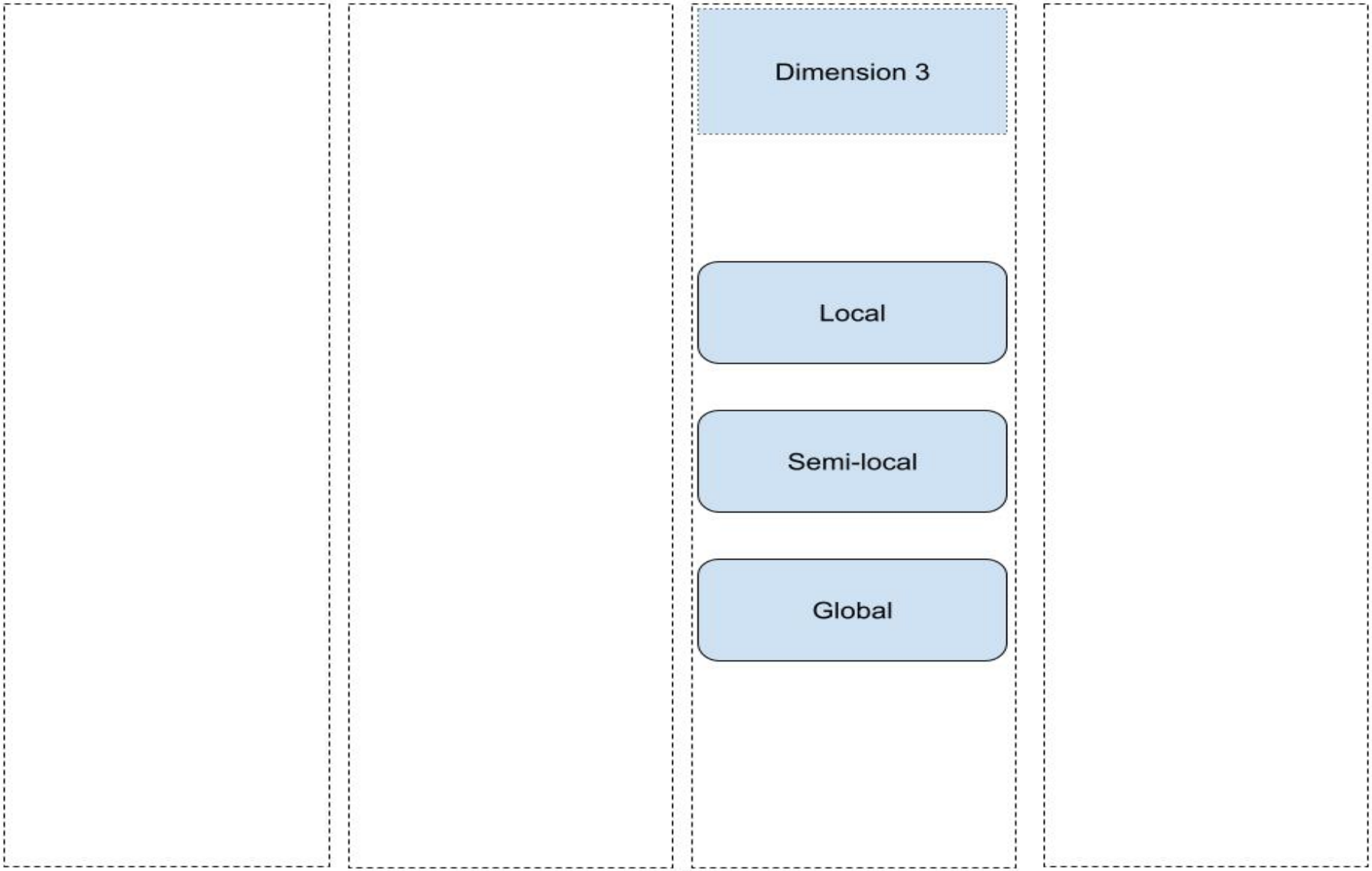
Arborescence du domaine

Type d’explications recherchés



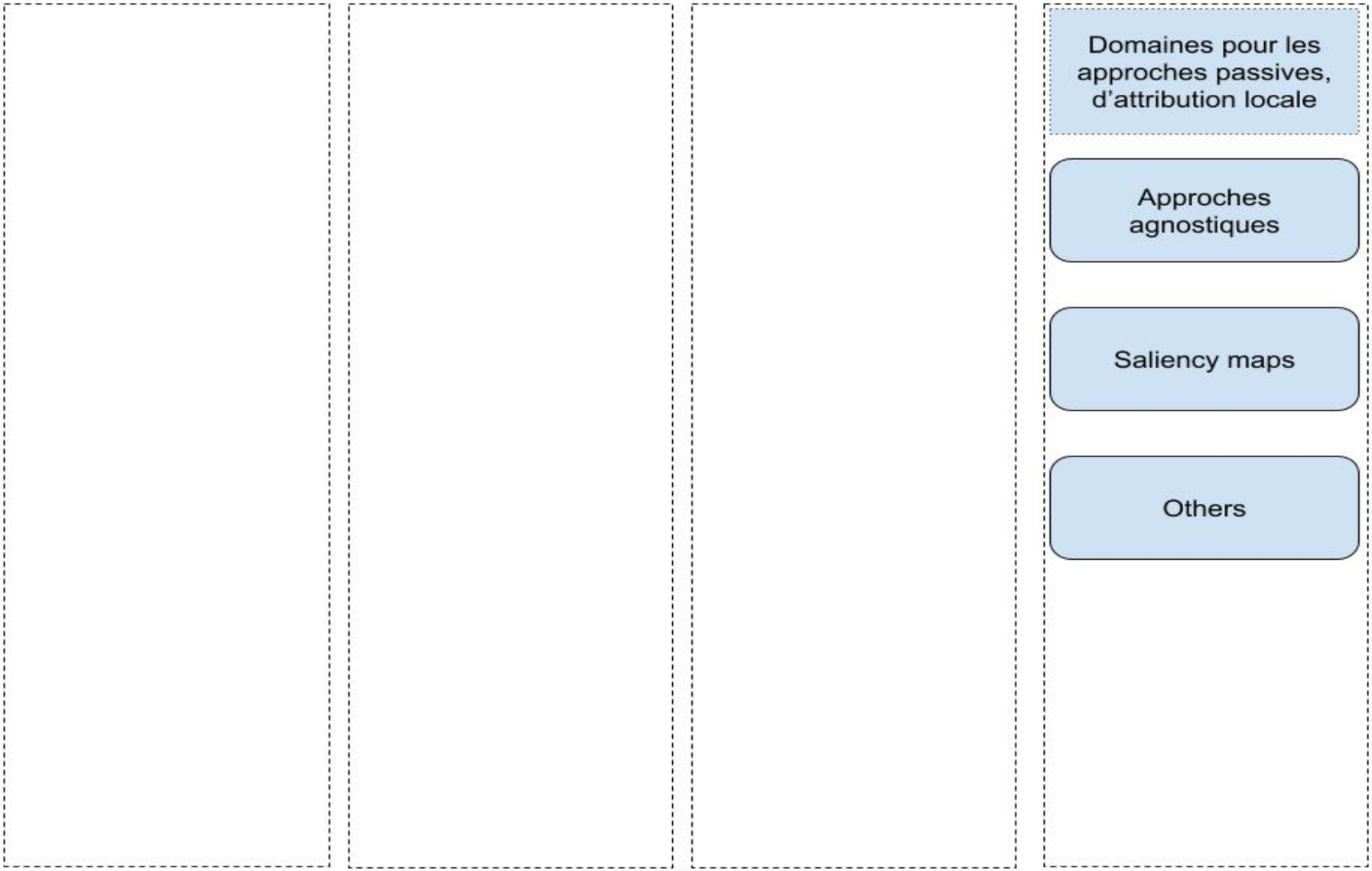
Arborescence du domaine

Interprétation globale vs. locale



Arborescence du domaine

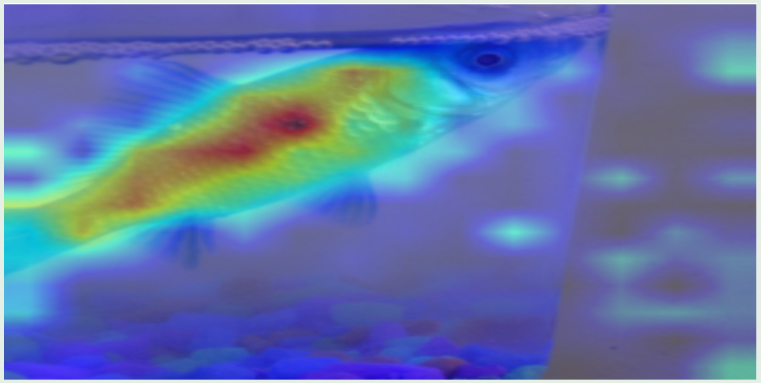
Sujet d'intérêt : Saliency maps



Definition (Saliency map)

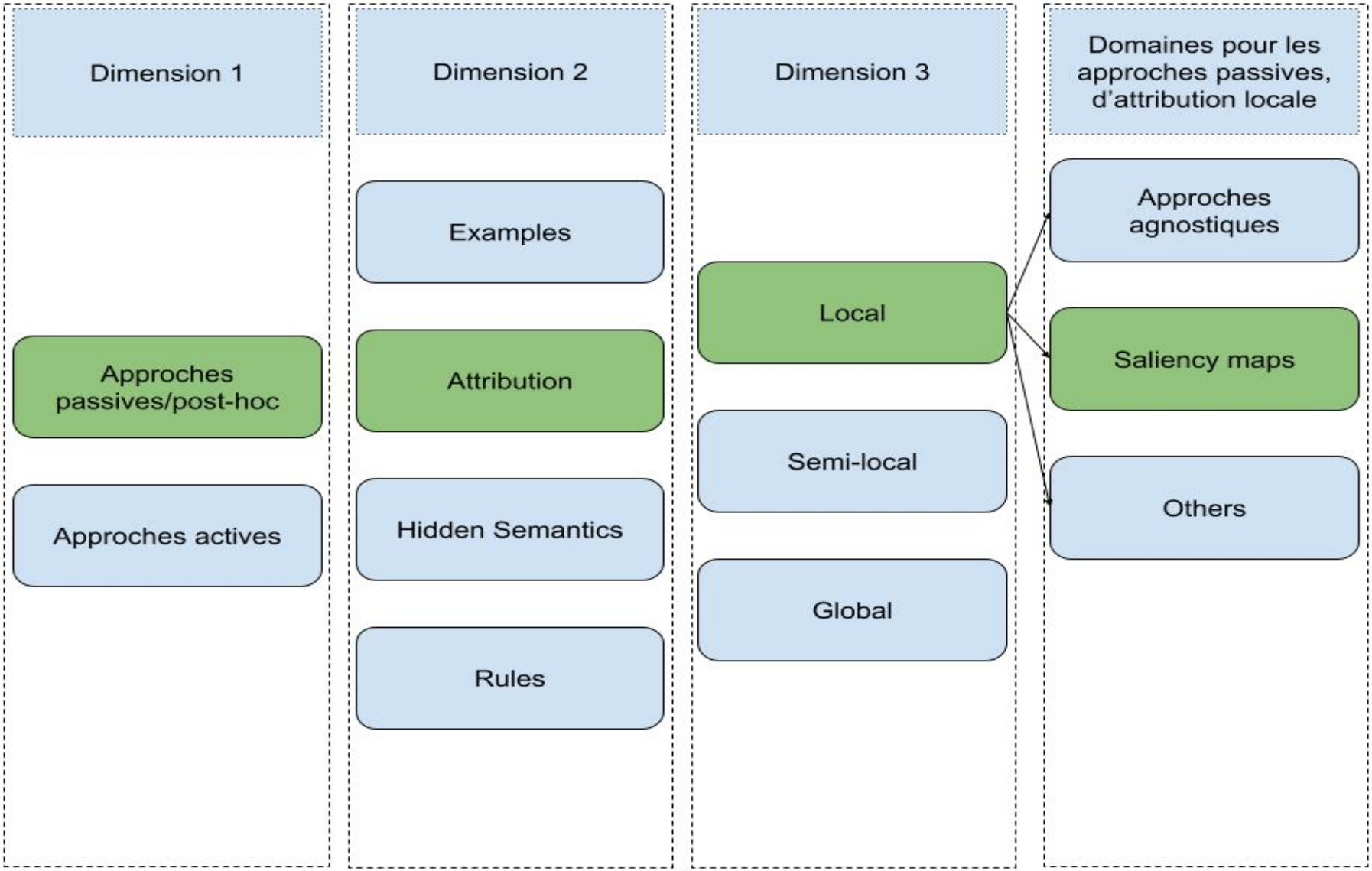
Carte de saillance (*Saliency map*) : **visualisation** qui met en évidence les **caractéristiques influentes ou cruciales** dans le processus de **prise de décision d'un modèle**, i.e. identifier quelles parties des données d'entrée contribuent le plus à la sortie du modèle.

Example



Arborescence du domaine

Une vue globale



Présentation :

- Utilisation des 2 dernières couches (feature map puis avg pooling) d'un CNN.
- CAM = pondération des features maps par les poids associés à la classe d'intérêt.

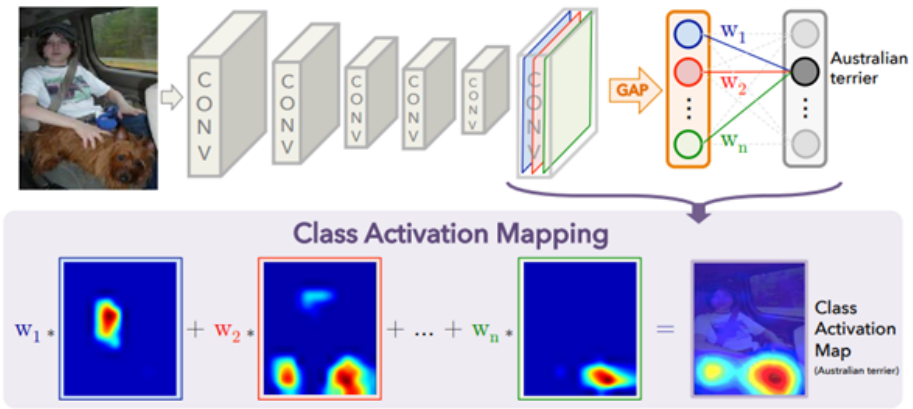


Figure: Schema CAM (2016)

Formule :

$$S_l^c(x) = \sum_k w_k^c A_k^l$$

Où :

- $S_l^c(x)$: la saillance de la classe c depuis la couche l pour l'image x.
- w_k^c : le poids de la classe c pour le filtre k.
- A_k^l : feature map k sur la couche l.

Limitations :

- Ne marche que sur des CNN avec dernière couche = Global Average Pooling
- Permet seulement de visualiser les feature maps en couche finale

$\rightarrow$ développement de gradCAM

GradCAM

Gradient-Weighted Class Activation Mapping : présentation

Présentation :

- Utilisation d'un CNN quelconque pour une tâche donnée (ici : classification)
- Rétropropagation sur la dernière couche convolutive (feature maps)
- Pondération par les gradients sur ces feature maps, ReLU et transformation en heatmap

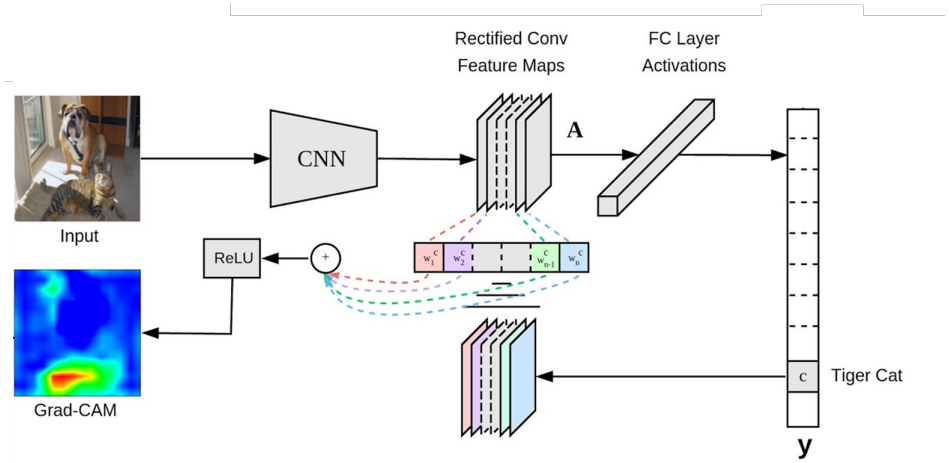


Figure: Schema gradCAM (2017)

Formule :

$$S_l^c(x) = ReLU\left(\sum_k w_k^c A_l^k\right)$$

Avec

$$w_k^c = GAM(\frac{\partial y^c}{\partial A_l^k})$$

Où :

- $S_l^c(x)$: Saliency map avec Grad-CAM pour la classe c.
- w_k^c : Les poids attribués à chaque carte d'activation A^k pour la classe c.
- A_l^k : La carte d'activation de la caractéristique k sur la couche l.
- y^c : Le score de logit pour la classe c.

Limitations :

- Grad CAM dépend de la dernière couche convolutive

Améliorations possibles :

- Grad-CAM++ = seuls les gradients positifs sont considérés
- Guided Grad CAM = Grad CAM jusqu'à l'image d'input (ne s'arrête pas à une couche de feature map), souvent combiné avec le Grad CAM classique.

Présentation

- Phase 1 : transformation des feature maps en heatmaps.
- Phase 2 : transformation des heatmaps pixel par pixel depuis l'image d'origine, concaténation, CNN, couche dense.
- Pondération des heatmaps de la phase 1 par les poids de la phase 2

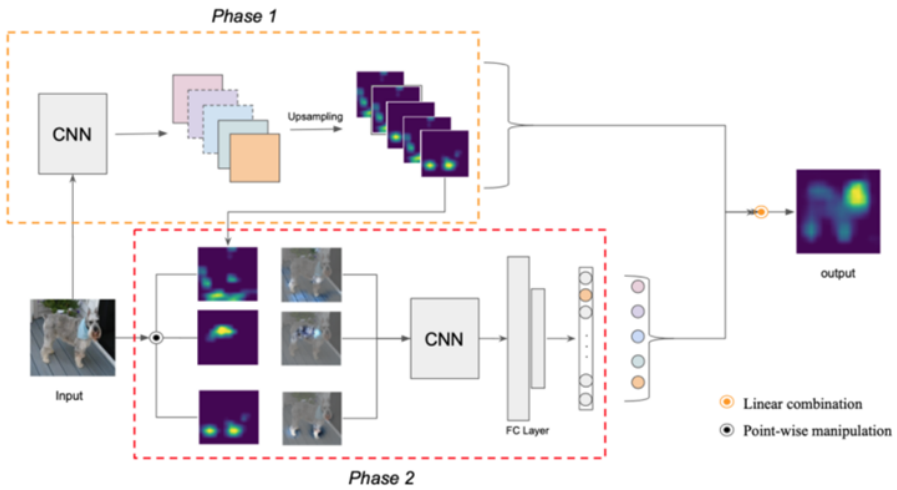


Figure: Schema ScoreCAM (2020)

Formule :

$$S_l^c(x) = ReLU\left(\sum_k w_k^c A_l^k\right)$$

Avec

$$w_k^c = softmax(u^c)_k$$

$$u^c = f(x \odot n(up(A_l^k)))_c - f(x)_c$$

Où :

- $S_l^c(x)$: Saliency map avec Score-CAM pour la classe c.
- u^c : L'augmentation de confiance pour la classe c de l'image d'entrée x masquée par la carte de saillance.
- up : upsampling, n : normalisation.

OptiCAM

Optimisation Class Activation Mapping : présentation

Présentation :

- Pondération des feature maps par les poids u, à apprendre
- Produit d'Hadamard entre la carte de saillance et l'image d'origine, calcul d'une loss
- Entraînement de u à partir de la fonction objectif (mesure de la confiance du modèle dans chaque classe)

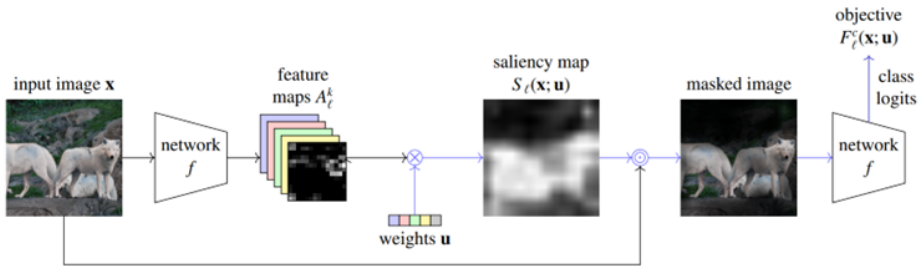


Figure: Schema OptiCAM (2024)

OptiCAM

Optimisation Class Activation Mapping : formule

Formule :

$$S_l^c(x) = S_l(x, u*) = ReLU\left(\sum_k w_k^c A_l^k\right)$$

Avec

$$w_k^c = softmax(u^c)_k$$

$$u* = \underset{u}{\text{argmax}}\, f(x \odot n(up(S_l(x, u))))$$

Où :

- $S_l^c(x)$: Saliency map finale via Opti-CAM pour la classe c.
- $S_l(x, u)$: Saliency map selon le paramètre u.
- $u*$: Confiance optimale pour c de l'image d'entrée x masquée.
- up : upsampling, n : normalisation.

Definition (*Relevance Score*)

On appelera *Relevance Score*, un score sur la pertinence que l'on attribue à un pixel d'une image dans sa classification par une fonction f. Pour une image $\mathbf{x} = \{x_p\}$, on notera $R_p(x)$, la pertinence du pixel p dans la classification de x par f.

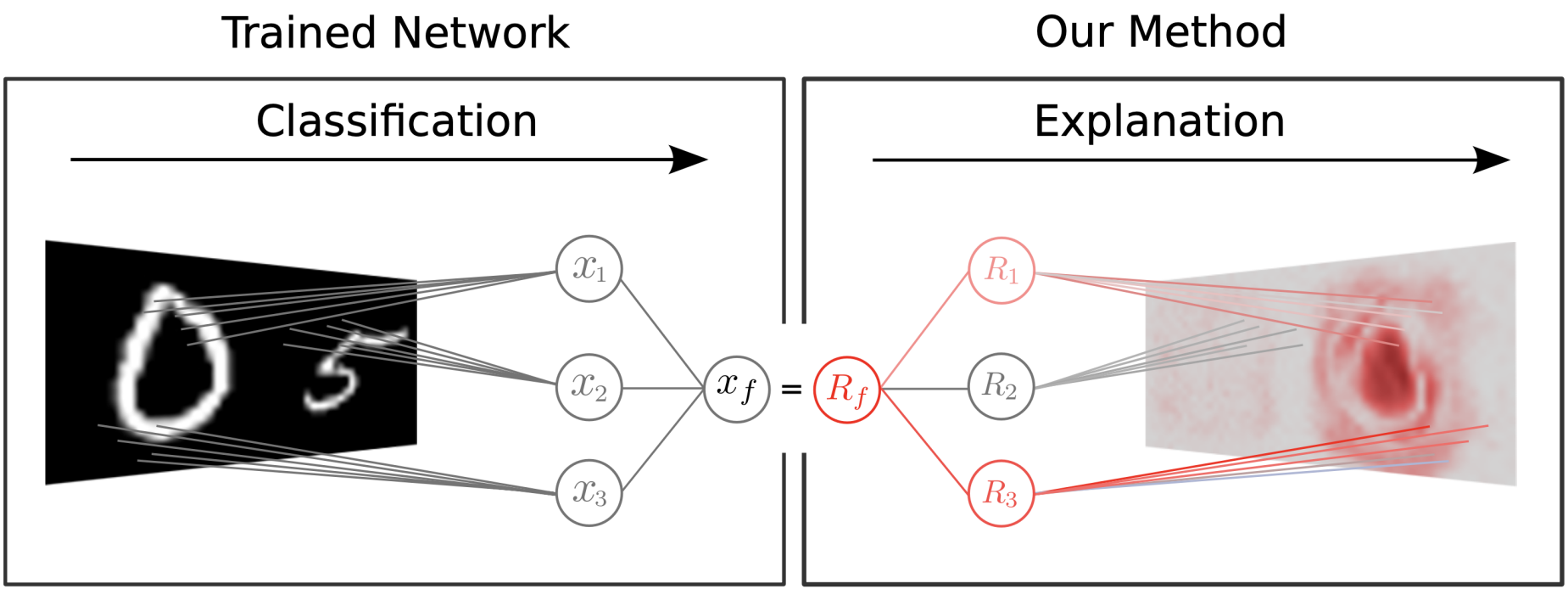
R(x) correspond alors à une carte de chaleur de la pertinence des pixels de x dans sa classification par f.

Definition (Propriété)

- On dira que de **R(x)** est conservatif si

$$\forall \mathbf{x} : f(\mathbf{x}) = \sum_{p} R_p(\mathbf{x})$$

Example


Trained Network
Our Method
Classification
Explanation
x_1
x_2
x_3
x_f
=
R_f
R_1
R_2
R_3

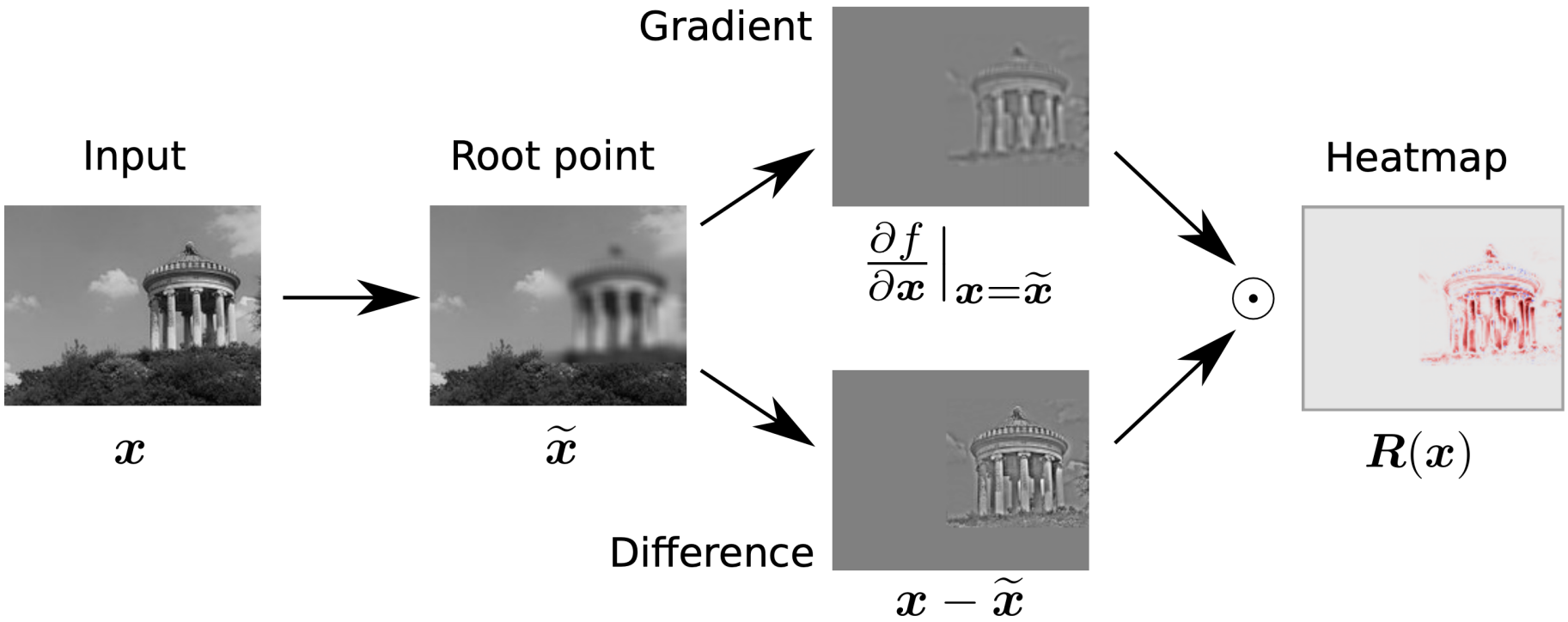
Relevancy Propagation

Exemples - Décomposition de Taylor

$$
\begin{aligned}
f(\boldsymbol{x}) &= f(\widetilde{\boldsymbol{x}}) + \left(\frac{\partial f}{\partial \boldsymbol{x}} \Big|_{\boldsymbol{x}=\widetilde{\boldsymbol{x}}} \right)^{\top} \cdot (\boldsymbol{x} - \widetilde{\boldsymbol{x}}) + \varepsilon \\
&= 0 + \sum_{p} \underbrace{\frac{\partial f}{\partial x_p} \Big|_{\boldsymbol{x}=\widetilde{\boldsymbol{x}}} \cdot (x_p - \widetilde{x}_p)}_{R_p(\boldsymbol{x})} + \varepsilon,
\end{aligned}
$$

Relevancy Propagation

Exemples - Décomposition de Taylor



Relevancy Propagation

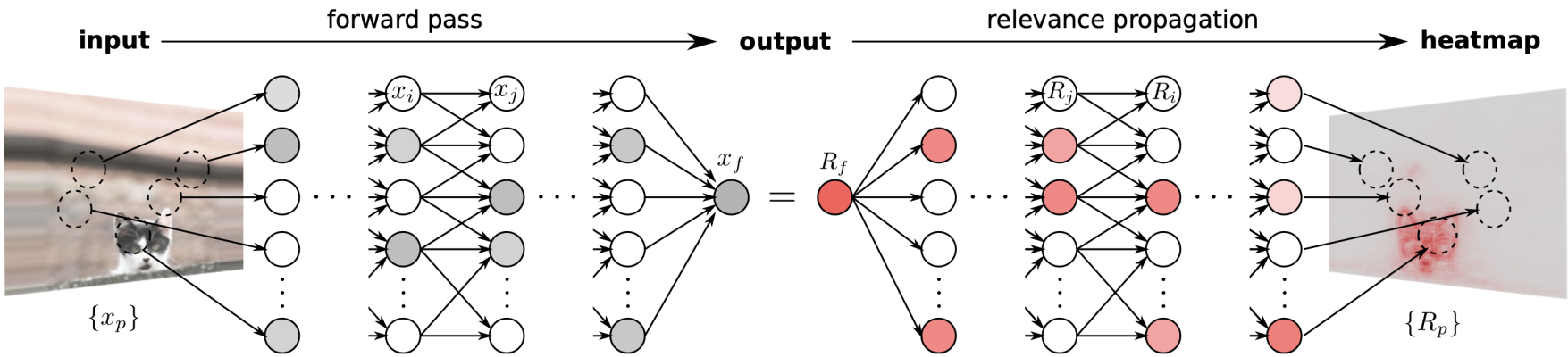
Deep Taylor Decomposition

- On se place dans le contexte des DNN, CNN, Transformers etc.
- Réseaux profonds, que l'on peut éclater en un treillis (*Directed Acyclic Graph*)
- Pour chaque neuron, on lui associe une pertinence dans la classification d'une image **x**

Idée : calculer la pertinence des neurones d'une couche *i* grâce à celle postérieure *j*

Relevancy Propagation

Deep Taylor Decomposition - Exemple



Relevancy Propagation

Deep Taylor Decomposition - le calcul

$$\sum_j R_j = \left(\frac{\partial \left(\sum_j R_j \right)}{\partial \{x_i\}} \Big|_{\{\widetilde{x}_i\}} \right)^\top \cdot (\{x_i\} - \{\widetilde{x}_i\}) + \varepsilon$$

$$= \sum_i \underbrace{\sum_j \frac{\partial R_j}{\partial x_i} \Big|_{\{\widetilde{x}_i\}}}_{R_i} \cdot (x_i - \widetilde{x}_i) + \varepsilon,$$

Theorem (Conservation de la pertinence à travers les couches)

Si tous les scores de pertinence sont conservatifs, alors la pertinence de classification d'une image se conserve dans chaque couche du réseau profond, i.e. :

$$R_f = ... = \sum_j R_j = \sum_i R_i = ... = \sum_p R_p$$

Relevancy Propagation

Remarques

- Tous les opérateurs ne permettaient pas, de manière directe, le calcul d'une pertinence conservative
- Le calcul pouvait se faire pour des couches et activations simples
- Peu à peu, introduction des calculs pour des couches de normalisations, connection résiduelle etc.
- TIBAV : introduit calcul pour multiplication matricielle et fonction d'activation *softmax*

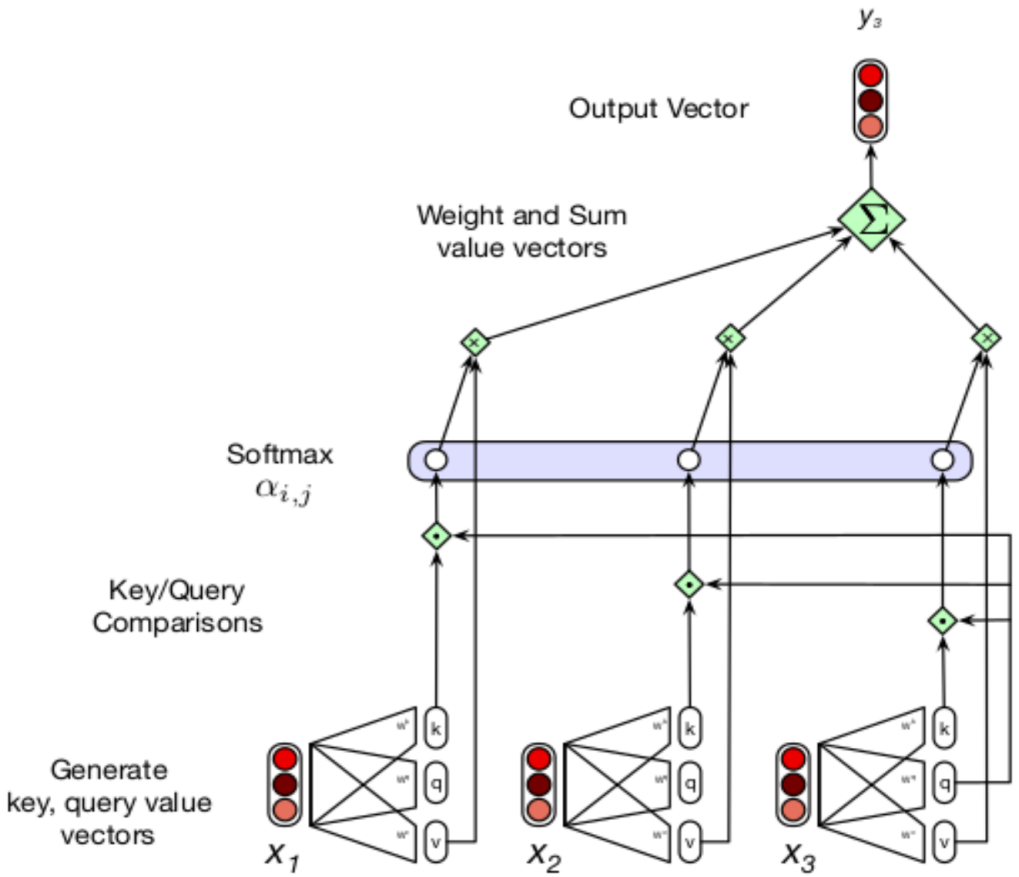


Figure: Caption

- The previous equations are applied to a single position i
- We can generate all outputs simultaneously using matrices
  - $\mathbf{X}$ is a $n \times d$ matrix of embeddings (n input words)
  - $\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}} \quad \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}} \quad \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}} \in \mathbb{R}^{n \times d}$
- $\mathbf{AB}$ = dot product of all lines of $\mathbf{A}$ with all columns of $\mathbf{B}$

$$\text{SelfAttn}(\mathbf{X}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{d}}\right)\mathbf{V}$$

Figure: Caption

Attention methods

Directed Acyclic Graph

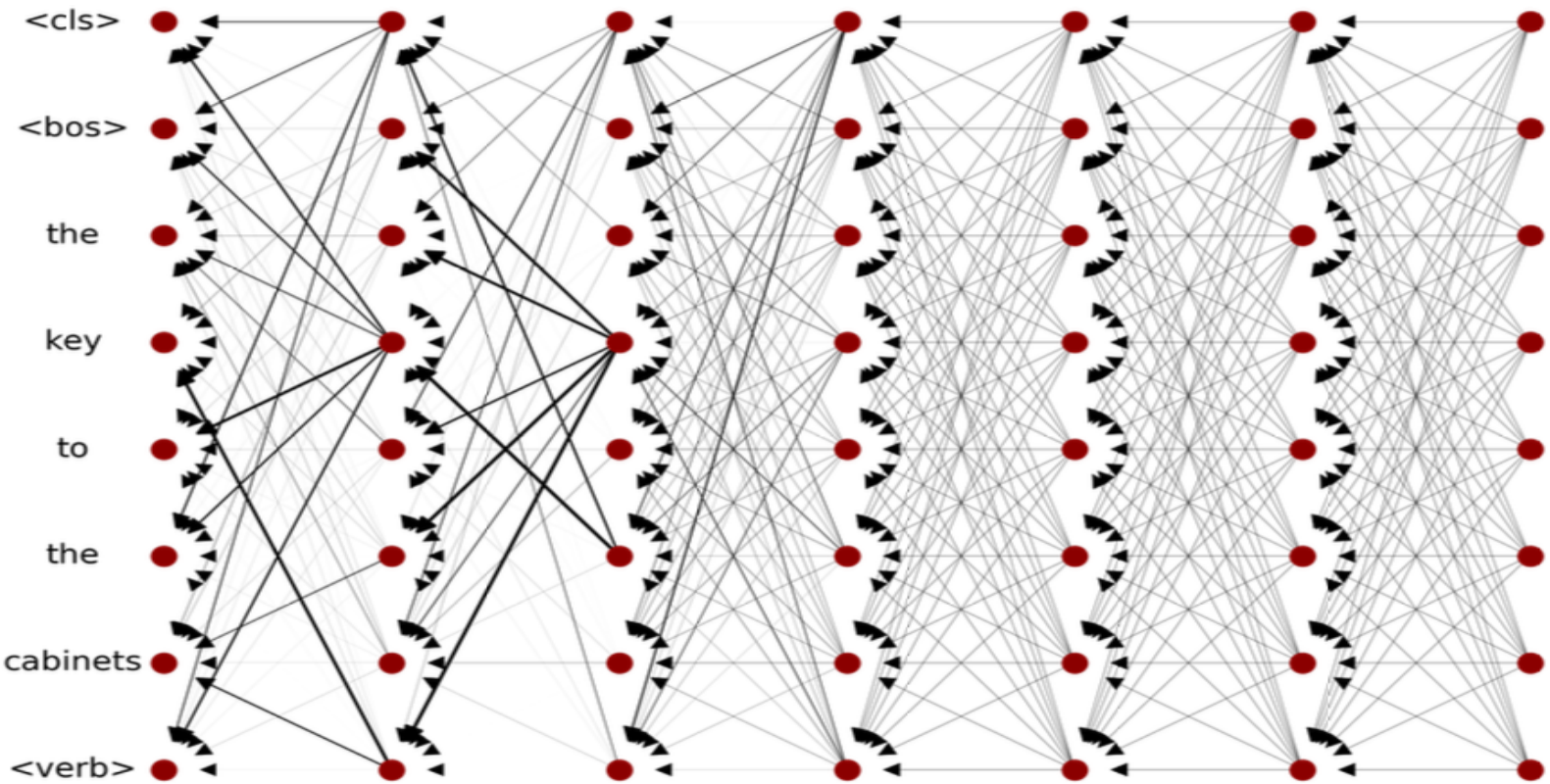


Figure: Caption

Attention methods

Raw Attention, Rollout

On se place dans le cas d'une activation linéaire, avec des connections résiduelles

- Première méthode pour des saliences : prendre les matrices d'attention brute pour avoir une interprétation de notre modèle, i.e. : On sait que $V_{l+1} = V_l + W_{att} V_l$ Donc

$$A = 0.5 W_{att} + 0.5 I$$

On se place dans le cas d'une activation linéaire, avec des connections résiduelles

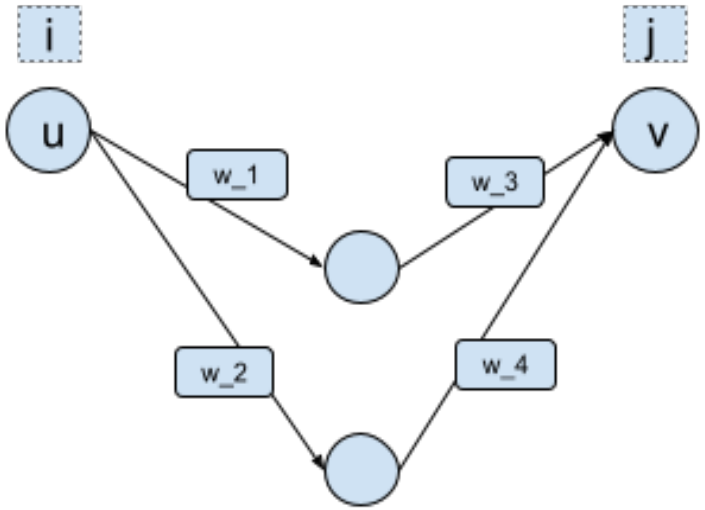
- Première méthode pour des saliences : prendre les matrices d'attention brute pour avoir une interprétation de notre modèle, i.e. : On sait que $V_{l+1} = V_l + W_{att} V_l$ Donc

$$A = 0.5 W_{att} + 0.5 I$$

- Deuxième méthode : modéliser le flux d'information entre un noeud u d'une couche i et v d'une couche j, avec $i < j$

Attention methods

Rollout - Une image pour expliquer le calcul



Information entre u et v :

- par le chemin 1 : $w_1 w_3$
- au total :

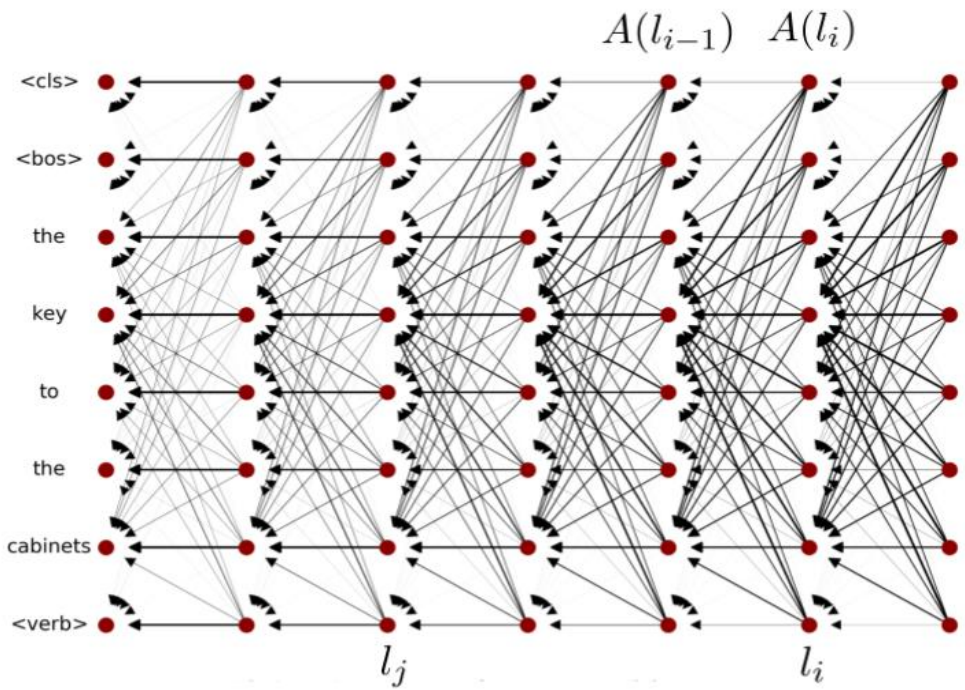
$$\widetilde{A_{u \to v}} = w_1 w_3 + w_2 w_4$$

- si on pose A_u, la matrice d'attention pour w_1 et w_2 et A_v pour w_3 et w_4, on a

$$\widetilde{A_{u \to v}} = A_u^T A_v$$

Attention methods

Rollout - Une image pour expliquer le calcul



$$\tilde{A}(l_i) = \begin{cases} A(l_i)\tilde{A}(l_{i-1}) & \text{if } i > j \\ A(l_i) & \text{if } i = j \end{cases} \qquad (1)$$

- Utilise la propagation de la relevance, en introduisant une méthode pour les opérateurs de multiplications matricielles (cruciales dans un transformer)
- Utilise le Rollout pour le calcul de la Saliency map pour chaque token

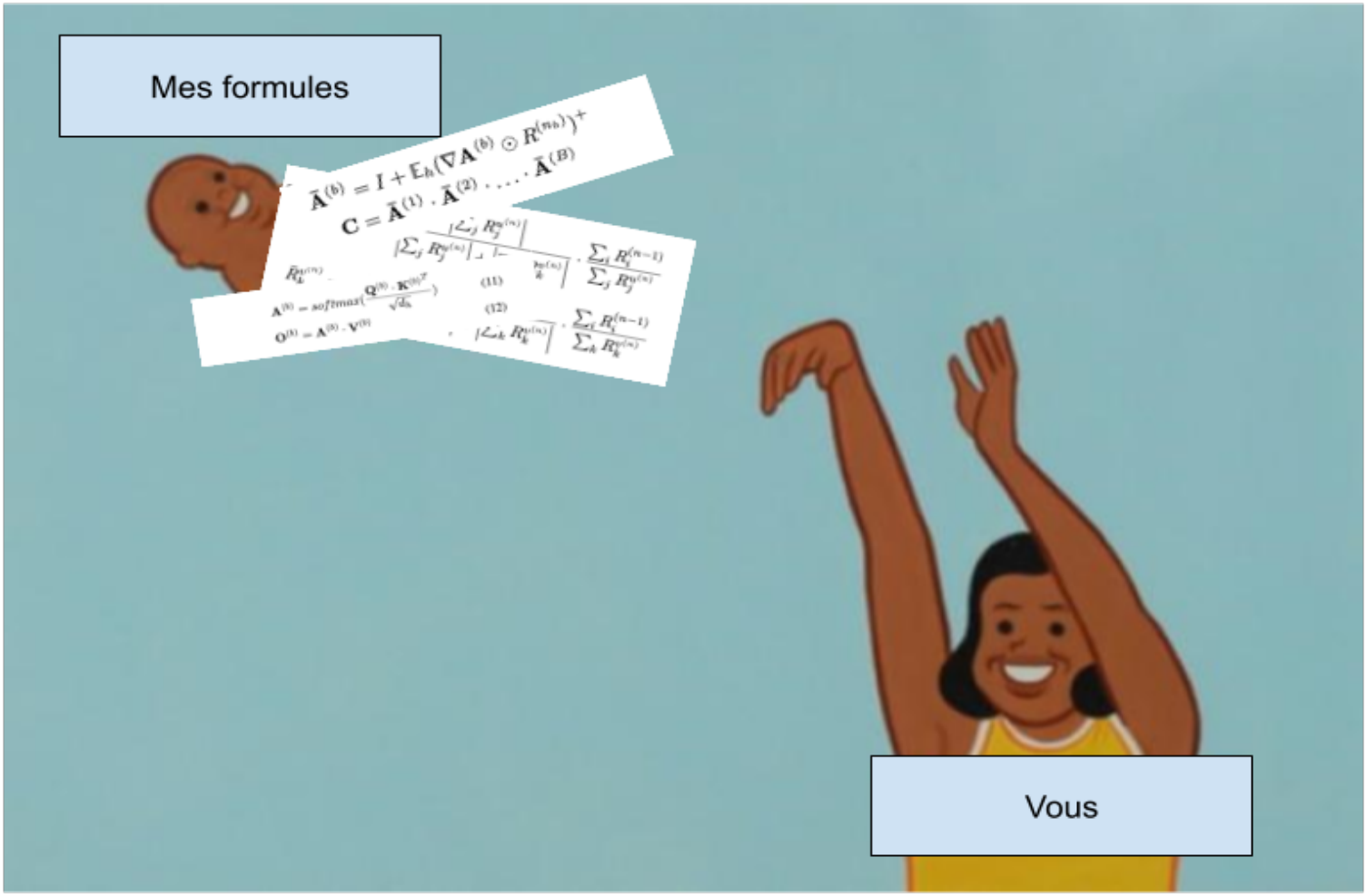
$$\bar{\mathbf{A}}^{(b)} = I + \mathbb{E}_h(\nabla \mathbf{A}^{(b)} \odot R^{(n_b)})^+$$
$$\mathbf{C} = \bar{\mathbf{A}}^{(1)} \cdot \bar{\mathbf{A}}^{(2)} \cdot \ldots \cdot \bar{\mathbf{A}}^{(B)}$$

Figure: Formule TIBAV

$$\hat{\mathbf{A}}^{(b)} = I + \mathbb{E}_h \mathbf{A}^{(b)}$$
$$\text{rollout} = \hat{\mathbf{A}}^{(1)} \cdot \hat{\mathbf{A}}^{(2)} \cdot \ldots \cdot \hat{\mathbf{A}}^{(B)}$$

Figure: Formule Rollout

TIBAV



- S'appuie sur des méthodes (Pertinence/Relevance et Rollout), à la base de beaucoup d'autres travaux.
- À l'état de l'art pour les transformers
- Le papier présente des variantes : utiliser A au lieu de son gradient, la dernière couche au lieu de calculer le rollout etc.

Métriques

Présentations

Plusieurs métriques, très dépendants de ce que l'on cherche à faire

- Métriques de performances de **classifications** (Average Drop, Average Gain, Average Increase)
- Métriques sur **détection dans des images** (Official Metric (OM), Localization Error (LE), Pixel-wise F1 score (F1), Box Accuracy (BA), Standard Pointing game (SP), Energy Pointing game (EP)).
- Métriques de performances de **segmentations** (pixel-accuracy, mAP, and mIoU)
- Métrique **qualitatives** (affichage des saliency maps et évaluation à l'oeil)
- Métriques sur des **tests de perturbations** (AUC sur tests de perturbations positives et négatives)

Évaluations

Différents résultats

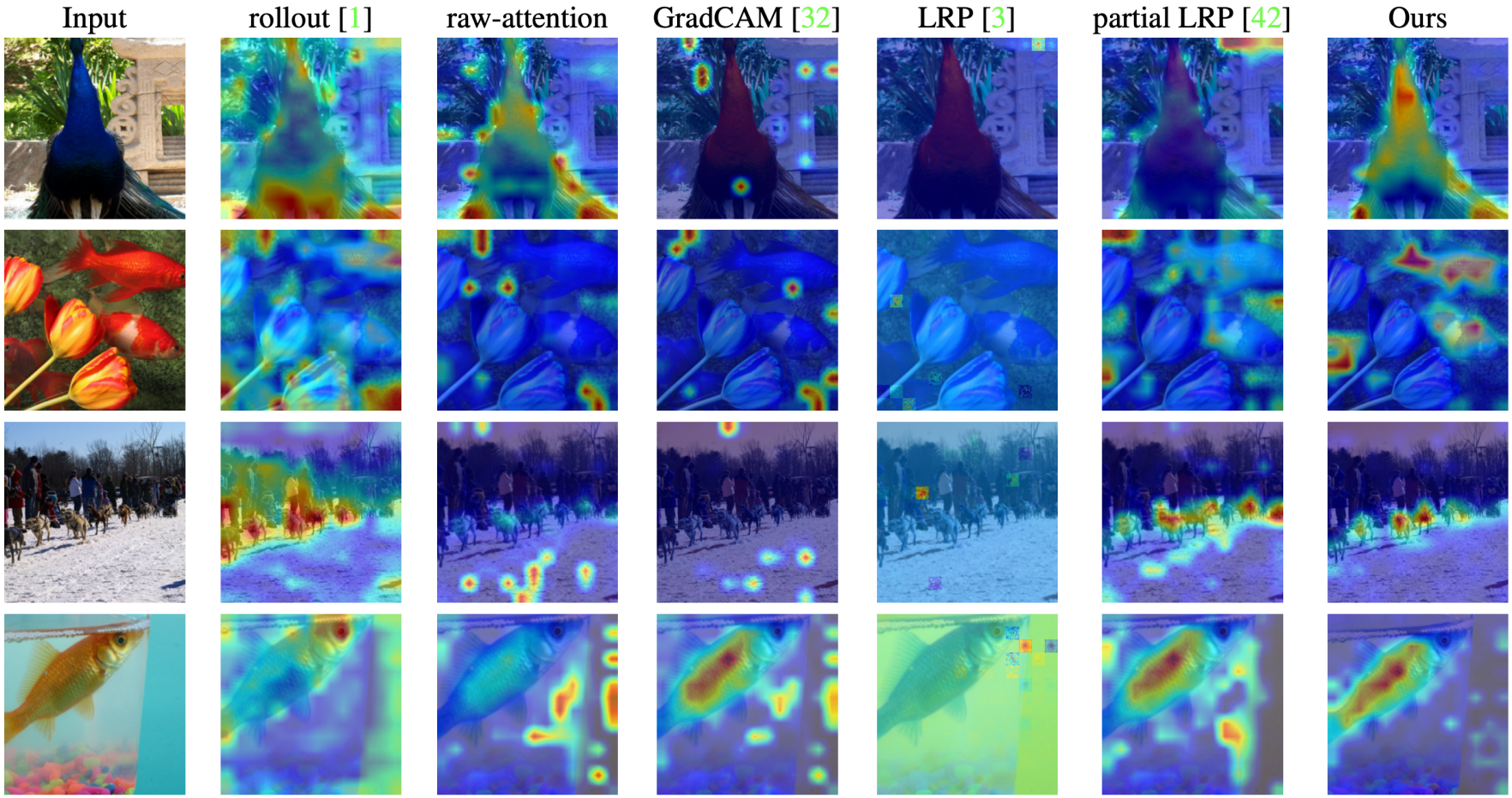


Figure 2: Sample results. As can be seen, our method produces more accurate visualizations.

Évaluations

Différents résultats

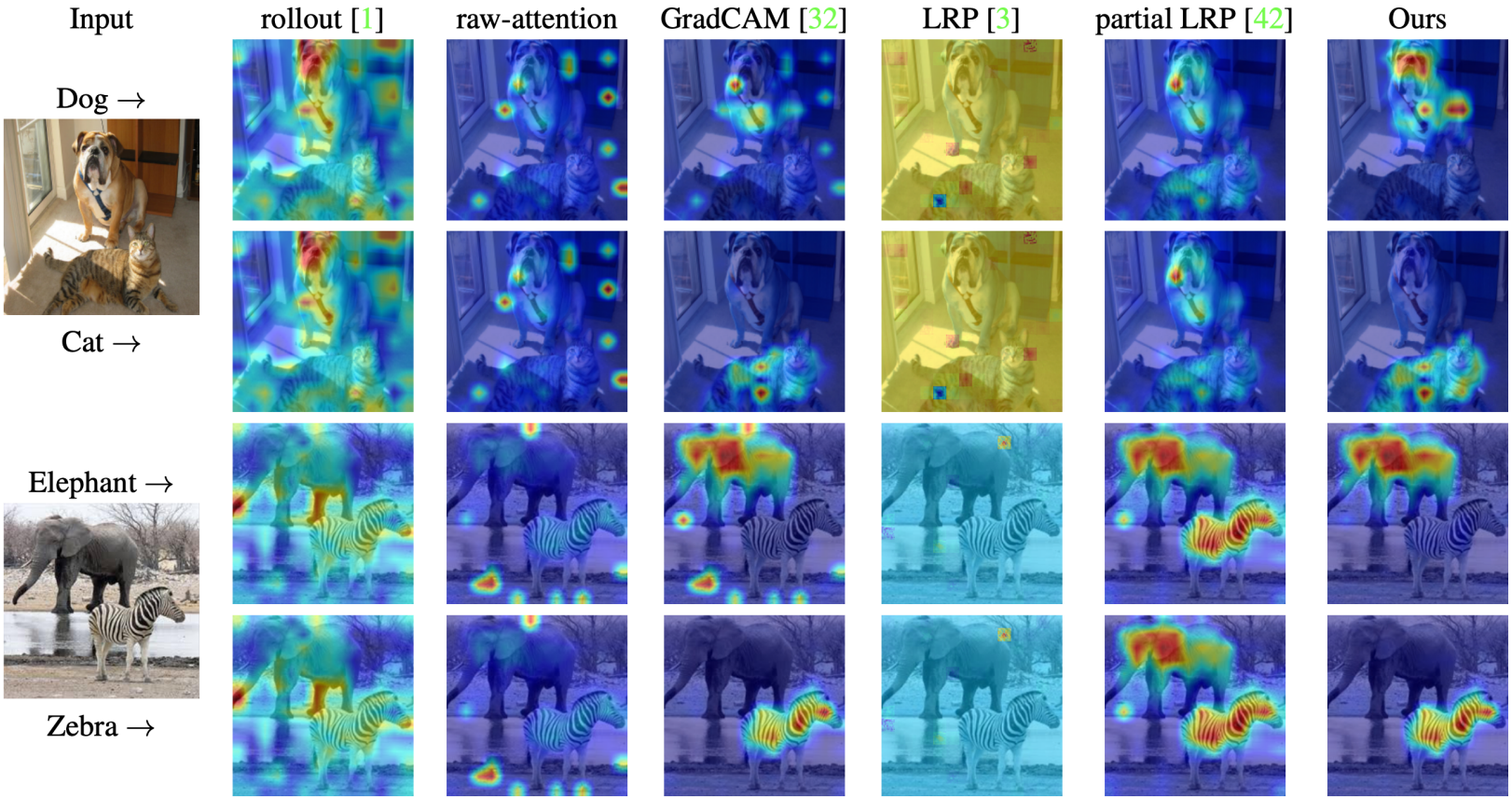
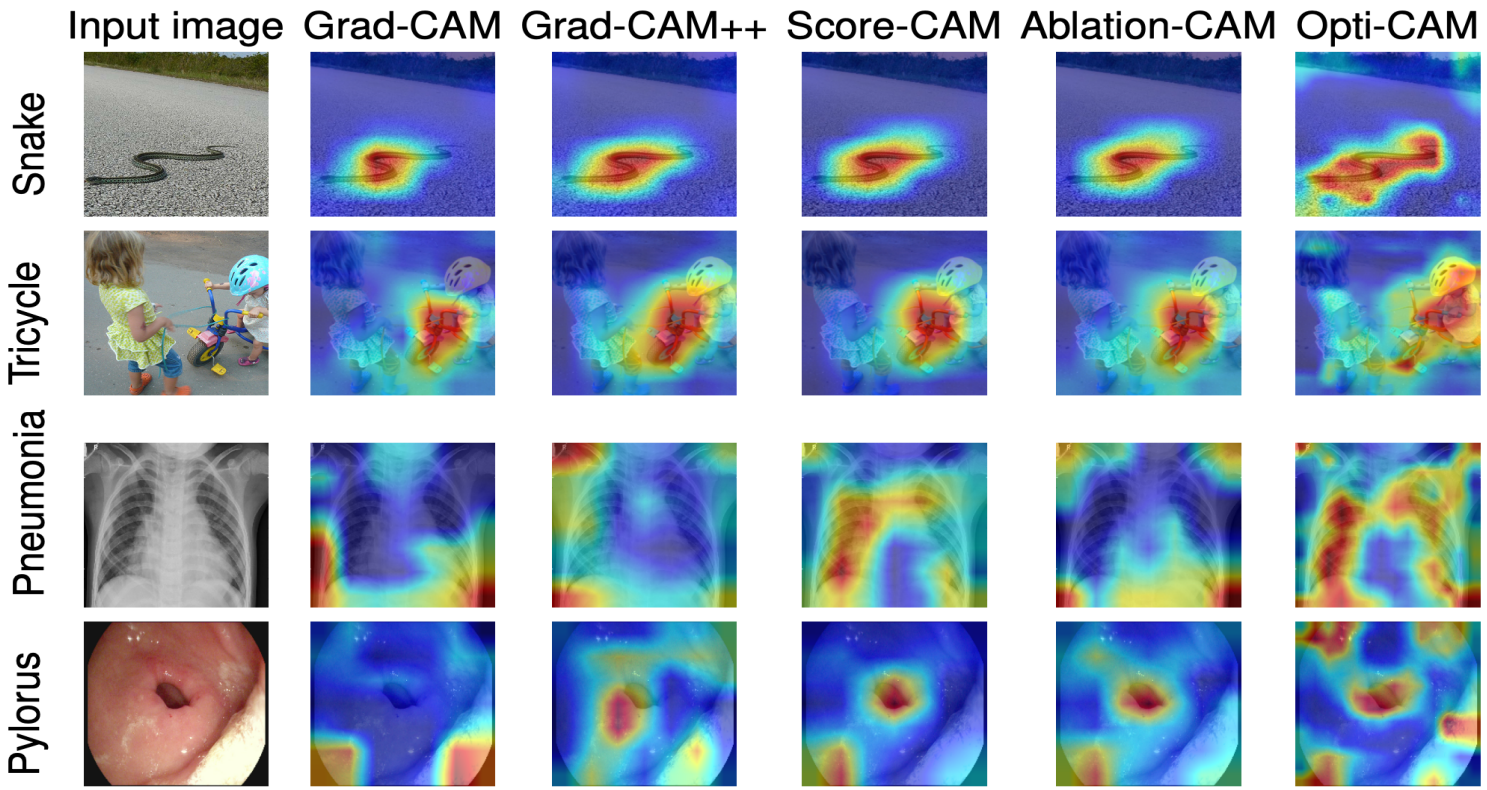


Figure 3: Class-specific visualizations. For each image we present results for two different classes. GradCam is the only method to generate different maps. However, its results are not convincing.

Évaluations

Différents résultats

Visualizations



Évaluations

Limitations

Si chaque métrique semble intéressante dans l'évaluation,

- Les papiers semblent utiliser les métriques qu'ils souhaitent pour mettre en avant leur modèle
- Beaucoup trop de métriques, comparaisons des modèles difficiles, facilement détournable pour montrer que son modèle est bon
- Chaque modèle fait plus ou moins bien selon les métriques utilisées, mais aussi les tâches initiales (classifications, détection, segmentations etc.), ou encore le dataset utilisé...

Conclusion

- Les saliency maps sont des méthodes passives, d'attribution et locale d'interprétabilité pour les images, qui reposent sur l'achitecture des modèles (notamment leur sorties).
- Famille CAM pour CNN, Relevancy Propagation & cie, Rollout et TIBAV pour Transformers
- Beaucoup de métriques d'évaluation, savoir rester critique lorsqu'on montre des résultats
- Domaine en plein essor, beaucoup de thèses sur ce sujet

Merci pour votre attention

Des Questions?

Bibliography I

- B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," 2015.
- S. Abnar and W. Zuidema, "Quantifying attention flow in transformers," 2020.
- H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," 2020.
- H. Zhang, F. Torres, R. Sicre, Y. Avrithis, and S. Ayache, "Opti-cam: Optimizing saliency maps for interpretability," 2024.
- A. Binder, G. Montavon, S. Bach, K.-R. Müller, and W. Samek, "Layer-wise relevance propagation for neural networks with local renormalization layers," 2016.

Bibliography II

- G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, "Explaining nonlinear classification decisions with deep taylor decomposition," *Pattern Recognition*, vol. 65, p. 211–222, May 2017. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2016.11.008
- R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct. 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7
- H. Chefer, S. Gur, and L. Wolf, "Transformer interpretability beyond attention visualization," 2021.
- M. Q. t. Ronan Sicre LIS, "Visual interpretability: saliency maps and interpretable classification," 2023.

Bibliography III

[1, 2, 3, 4, 5, 6, 7, 8, 9]