UMONS

Certificat d'Université en

Intelligence Artificielle

18 - 19 MAI 2024 (Hôtel Utopia)

6ÈME ÉDITION DU WORKSHOP RÉSIDENTIEL

Certificat IA UMONS
HackIA'24

Système *Edge AI*
Pour Villes Intelligentes

Ingénieurs **Informaticiens** **Doctorants**

COACHS TECHNIQUES

Pr. Sidi Ahmed MAHMOUDI	UMONS
Ir. Jean-Sébastien LERAT	UMONS
Ir. Mohamed BENKEDADRA	UMONS
Ir. Maxime Gloesener	UMONS

MEMBRES DE JURY

Pr. Thierry DUTOIT	UMONS
Pr. Pierre MANNEBACK	UMONS
Pr. Stéphane DUPONT	UMONS
Ir. Driss LAHEM	MATERIANOVA
Pr. Olivier VERSCHEURE	EPFL Suisse

ORGANISATION

Ir. Aurélie COOLS	UMONS

Hôtel Utopia
Chaussée Brunehault 392,
7050 Masnuy-Sait-Jean

065 84 87 85

Nombre de Participants 14
Nombre de Groupes 03

hackia.eu


Atelier d'Intelligence Artificielle (I-ISIA-202)

Système IA embarqué pour maisons et villes intelligentes

Edge AI System for Homes and Smart Cities


Table des matières


Atelier d'Intelligence Artificielle (I-TCTS-202)

Système IA embarqué pour maisons et villes intelligentes


Objectif : développer un système d'intelligence artificielle embarqué sur des ressources *Edge AI* (matériel proche des capteurs de collecte de données). Le système s'appuiera sur les modèles Deep Learning développés durant les défis du certificat IA. Ces modèles seront combinés pour fournir un module « *Edge AI* » appliqué aux vidéos capturées en temps réel. Le résultat serait représenté par un module **« *Edge AI* »** au service des villes intelligentes.

1. Développement d'un module « *Edge AI* » pour villes intelligentes :

Développer un support pour villes intelligentes avec une application utilisant des modèles Deep Learning pour détecter et localiser différents objets à partir d'images capturées via la caméra. Ce module devra utiliser 3 modèles au moins :

1.1. Modèle « Face »

Le premier modèle servira comme système d'authentification faciale grâce à la reconnaissance de visages. Il permettra d'autoriser l'utilisateur à employer les autres applications (modèles IA pour ville intelligente) si la personne est à la fois reconnue et autorisée (via son visage).

1.2. Modèle « Fire : Détection »

Le deuxième modèle (de **classification**) consiste à détecter les feux ou fumées dans des forêts proches de la ville intelligente à partir d'images capturées via la caméra. Vous pouvez adapter vos modèles du défi 1 du certificat IA. Des prétraitements d'images peuvent être appliqués pour améliorer la qualité de détection et généraliser. Vous **avez le libre choix de détecter des feux de forêts (villes intelligentes) et des feux pouvant être déclenché au sein des maisons (maisons intelligentes)**.

1.3. Modèle « Fire : Localisation »

Le 3ème modèle consiste à **localiser** les feux de manière précise (carrés englobants) si le deuxième modèle « Fire Detection » a détecté la présence de feu dans la scène filmée par la caméra.

Note 1 : chaque groupe devra développer/intégrer **trois modèles** au moins. Si un groupe le souhaite, il peut augmenter la solution par d'autres modèles mais cela est **optionnel**. Chaque groupe peut aussi intégrer d'autres idées innovantes pour améliorer la solution tout en gardant l'objectif principal du projet choisi.

2. Phases du Workshop

Afin de répondre à l'énoncé décrit ci-dessus et d'organiser un partage de tâches entre les membres de chaque groupe, nous vous proposons de suivre les étapes suivantes :

- **Partie 1 :** développement et entraînement des différents modèles ;
- **Partie 2 :** développement du programme de test (inférence + notification) à partir de séquences vidéo sur la ressource embarquée « Edge AI » : **Jetson Xavier** ;
- **Partie 3 :** optimisation/compression d'au moins un modèle ;
- **Partie 4 :** analyse d'explicabilité de la solution « **XAI** ».

2.1. Partie 1 : développement et entraînement des modèles

Les entraînements peuvent être réalisés sous Python avec le Framework **Pytorch** (pour le Deep Learning) et la librairie **OpenCV** (pour le traitement d'images). Ces derniers sont déjà installés sur la carte Jetson Xavier qui sera fourni durant le Workshop. En termes de ressources, vous pouvez utiliser Google Colab Pro, vast.ai ou paperspace. Chaque groupe disposera, si besoin, de 50h de calcul avec vast.ai ou de 2 abonnements (1 moins) avec Colab Pro ou paperspace. Les principaux modèles à développer sont :

2.1.1. Modèle « Face »

Ce modèle permettra d'identifier et reconnaître les visages des personnes présentes dans la scène. Pour cela, nous vous recommandons d'utiliser le programme Face recognition permettant de détecter et reconnaître des personnes (Nom de la personne) à partir de l'introduction d'une seule image (visage) par personne. Vous êtes invités à bien analyser le code et l'appliquer correctement dans votre projet pour permette l'accès à l'application uniquement aux membres faisant partie de votre groupe.

2.1.2. Modèle « Fire : Détection »

Il s'agit du modèle de classification développé lors du défi 1, vous pouvez utiliser, adapter et améliorer vos modèles du défi 1 sans oublier de les porter sous Pytorch (Framework à utiliser durant le Workshop). Vous pouvez vous appuyer sur ce notebook **« fire_detection_training_torch.ipynb »** qu'il faudra bien évidemment adapter avec votre architecture, données et choix optimaux de paramètres.

2.1.3. Modèle « Fire : Localisation »

Le troisième modèle à paour objectif de localiser les objets sur images. Plus particulièrement, vous travaillerez sur le développement d'un réseau de neurones profond pour la localisation des feux de forêt ou fumées d'images de la caméra. En effet, si le modèle de classification décrit ci-dessus détecte un feu de forêt ou de la fumée, ce deuxième modèle devra localiser la position exacte du feu ou de la fumée à l'aide de carrés

englobants (Bounding Box) (Fig. 1). Si l'image présente une ou plusieurs zones de feu (ou fumée), il faudra les entourer en fonction de leurs positions. Pour réaliser ce modèle, il faudra réaliser les trois étapes suivantes :

a. **Préparation et annotation de données :** là aussi, il faudra lancer les entraînements sur des données avec une différence importante lors de l'annotation des données en fournissant :
   - Les images présentant des feux de forêt, fumée ainsi que d'autre objets ;
   - Le ou les labels (nom de classes) d'objets/zones présents dans chaque image ;
   - Pour chaque objet/zone, il faudra fournir sa position ainsi que sa taille.

   Comme point de départ, nous vous fournissons une base de données déjà annotée. Bien évidemment vous pourrez l'augmenter avec plus de classes et images à annoter par vos soins. Pour annoter vos images, vous pouvez utiliser différents outils d'annotations tels que : **Roboflow, labelme, labelimg, supervisely, etc.**

b. **Entraînement des données :** une fois que votre base de données est annotée, vous pouvez commencer le développement de votre classifieur qui pourra s'appuyer sur des architectures de localisation d'objets pré-entrainées (Yolo V3, Yolo V4, Yolo V5, Yolo V7, Yolo V8, Faster R-CNN, SSD, etc.). La technique de Transfert Learning est fort conseillée en vue de réduire les temps de calcul et d'améliorer la précision des résultats.

c. **Test et évaluation du modèle :** une fois que votre entrainement de données est finalisé, vous pouvez tester le modèle résultant avec les données de test, partagées via Moodle, afin d'évaluer votre modèle. Là aussi, l'objectif est d'avoir le meilleur taux de précision en localisant au mieux les feux de forêt.

Note 2 : un code de démarrage « ***input_fire_localisation_torch.ipynb*** » est fourni pour le développement de ce modèle, il faudra bien l'analyser et le comprendre avant de le compléter et le tester avec les différentes bases de données et architectures de localisation (ou votre architecture). Il faudra veiller à l'analyse des hyper paramètres pour obtenir des résultats satisfaisants.

FIGURE 1: EXEMPLE DE DE DÉTECTION DE FEUX ET FUMÉES (ARCHITECTURES YOLO)

2.2. Partie 2 : portage de solution sur la ressource Edge : Jetson Xavier

Après validation des modèles, il faudra les porter vers la ressource Edge Nividia Jetson Xavier afin de développer une solution embraquée et appliquée aux vidéos capturées via la caméra connectée à la carte (via un port USB). L'idée serait de combiner les modèles pour fournir un module Edge AI pour villes intelligentes en fonction de votre choix. La figure 3 illustre des exemples de programmes à développer et compléter (Edge AI System for Smart Cities)

```
#        ******************        EDGA AI module For Smart Cities        ******************        #
# face : model de reconnaissance facile
# fire : model de détéction de feu, fumée, etc.
# suspect : modèle de détection ou localisation d'objets suspects
# movement : modèle de reconnaissance de mouvements/actions (facultatif)
cap = cv2.VideoCapture(0)    # Capture de la vidéo en temps réel
    while(cap.isOpened()):
        frame = cap.read(); # lecture de chaque image de la vidéo capturée
        faces = face_recognition.identify(frame)    # Modèle de reconnaissance faciale (à lancer pendant 10 secondes par exemple)
        if (face.name in liste)                     # liste : membres authorisés à utiliser le module
        FirePred1 = fire.predict(x)[0] ;
        SuspectPred1 = suspect.predict(x)[0] ;
        MvtPred1 = movement.predict(x)[0] ;         # Facultatif
        }
cap.release()
cv2.destroyAllWindows()
```

FIGURE 3: "EDGE AI" ALGORITHM FOR SMART CITIES

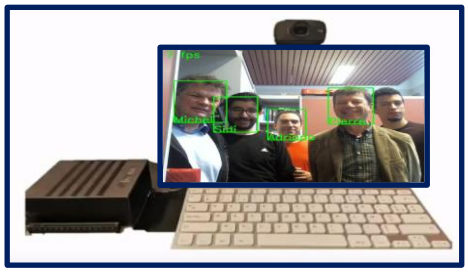


FIGURE 4 : SYSTÈME IA UTILISANT LA RESSOURCE EDGE "JETSON XAVIER"

Vous avez le libre choix de proposer une interface graphique ou pas et avec l'outil de votre choix (tkinter, PyQt, etc.). Notons que chaque groupe disposera d'une carte Jetson Xavier préconfigurée avec les différentes librairies : Pyorch, OpenCV, Sickit-learn, Matplotlib, etc. Les modalités d'accès aux environnements préconfigurés seront fournies durant le Workshop.

2.3. Partie 3 : optimisation/compression de modèles et/ou analyse d'explicabilité de la solution « XAI »

Afin d'avoir une solution optimale, nous vous proposons de l'optimiser en travaillant sur les points suivants :

2.3.1. Analyser les performances : des modèles en termes de précision, temps de calcul (**FPS**) et espace mémoire ;

2.3.2. Optimiser et compresser les modèles : à l'aide des techniques d'élagage « pruning » de réseaux de neurones, quantification, de distillation de connaissances et de convolutions séparables :

a. Elagage (pruning) : les réseaux de neurones profonds sont caractérisés par un nombre de paramètres (hyperparamètres) très élevé dont certains peuvent présenter une redondance inutile. Des techniques d'élagage (pruning) [1] et de Dropout peuvent être utilisées pour identifier et garder uniquement les neurones

et connexion les plus significatives afin de générer des modèles à taille réduite préservant une précision suffisante. Il faudra veiller à garder un bon compromis entre taille mémoire, temps de calcul et précision des modèles. Les réseaux compressés sont conçus pour avoir moins de paramètres et utiliser moins de mémoire. Pour faire le pruning, il faudra savoir réfléchir à quatre éléments principaux :

- Comment élaguer « **How to prune** » ? : pour définir la stratégie d'élagage par couche [10];
- Ou « **Where to prune** » ? : pour définir le taux d'élagage par couche ;
- Quoi « **What to prune** » ? : en fonction de la magnitude, gradient, etc.
- Quand « **When to prune** » ? : durant ou après l'entrainement.

La figure 5 illustre des modes d'élagage permettant de répondre à la dernière question « When to prune ?».

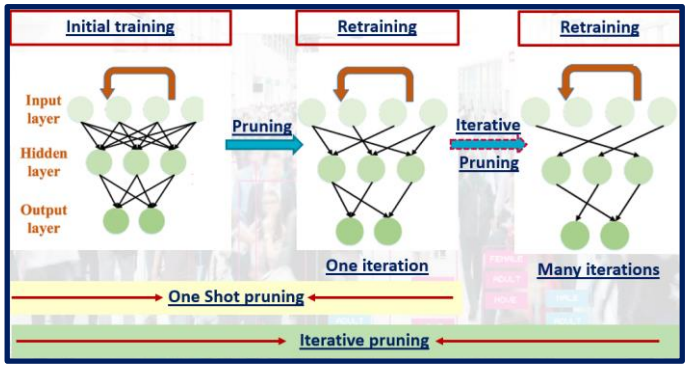


Figure 5 : illustration des modes d'élagage (pruning) de réseaux de neurones profonds [1]

b. La quantification : consiste à appliquer un processus d'approximation sur les réseaux de neurones afin de représenter les poids avec des nombres à faible nombre de bits sachant que les poids sont initialement représentés par des nombres en virgules flottantes [2]. Ce processus permet de réduire considérablement la taille mémoire et le temps de calcul des réseaux de neurones profonds. La figure 6 illustre un exemple de quantification de poids d'un réseau de neurones. Le nombre de bits doit être fixé en veillant à ce que la précision initiale reste maintenue car une réduction très grande de la précision des valeurs de poids risquera d'affecter négativement la précision du modèle.

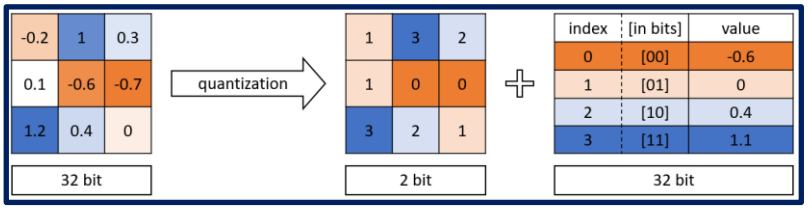


Figure 6 : illustration du principe de quantification

c. Distillation de connaissances « Knowledge distillation » : consiste à développer un petit réseau de neurones « **Student** » qui pourra apprendre durant l'entrainement à partir d'un grand réseau de neurones « **Teacher** ». L'objectif est d'avoir un réseau de neurones simplifié mais qui prendra des décisions semblables aux décisions d'un réseau de neurones grand et complexe. La figure 7 illustre le processus de cette approche.

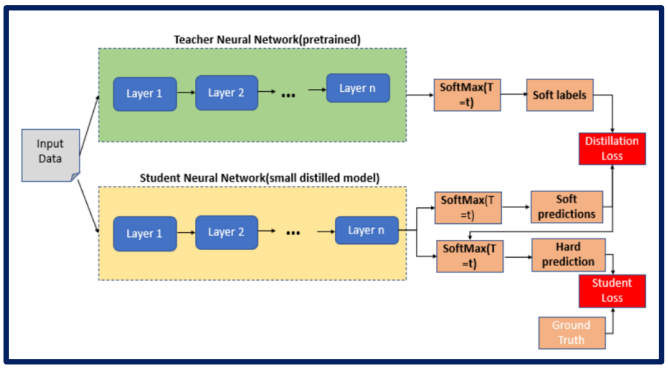


FIGURE 7 : PROCESSUS DE LA MÉTHODE DE DISTILLATION DE CONNAISSANCES

d. **Convolutions séparables :** dans le cas d'application des réseaux de neurones profonds sur des images, nous serons fort probablement amenés à utiliser des couches convolutionelles pour l'extraction de caractéristiques d'images au sein du réseau. Cependant, ces opérations sont très coûteuses en calcul et en mémoire. Dans ce contexte, Google a proposé en 2017 une nouvelle technique de convolution appelée « Depthwise Seperate Convolution » qui consiste à appliquer les filtres de convolutions séparément à chaque canal de l'image contrairement à la convolution classique qui applique un filtre sur l'ensemble des canaux [3]. Les 3 images résultant de cette opération (Depthwise Separate Convolution) seront ensuite combinées pour fournir une image caractéristique (Figure 8). Cette approche permet de réduire de manière significative les temps de calcul et espace mémoire tout en ayant un faible impact sur la précision. Cette technique est utilisée dans l'architecture « MobileNet » connue d'ailleurs par un faible temps de calcul et un espace mémoire réduit.

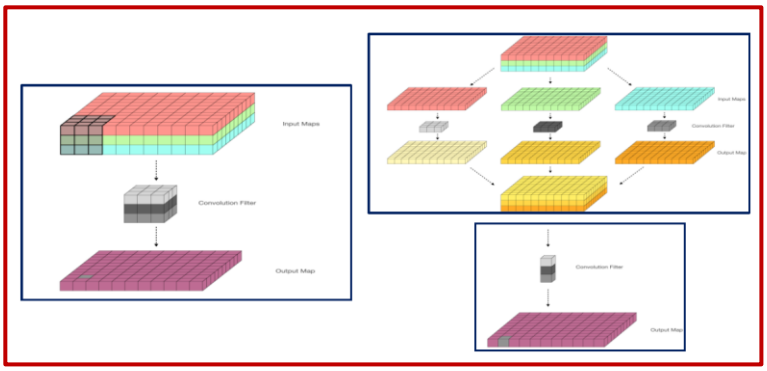


Figure 8 : convolution classique vs. Convolutions séparables

Nous partageons avec vous quelques liens vers des tutoriels pouvant vous aider pour la compression :

- **Pruning :** lien → mode d'application du pruning sur un DNN développé avec Pytorch.
- **Knowledge Distillation :** lien → exemple d'application de la KD sur un DNN développé sous Pytorch.
- **Quantization :** lien → mode d'application de la quantization sous Pytorch.

Note : Vous avez le libre choix d'identifier et utiliser la méthode de compression qui réponds aux mieux à vos besoins de déploiement (temps de calcul, mémoire, espace de stockage, etc.)

2.3.3. Expliquer et interpréter votre modèle de classification de feu « Model 1 » : à l'aide d'une approche d'« ***Explainable AI, XAI*** » et techniques de visualisation de caractéristiques extraites à partir de réseaux de neurones. L'idée est de calculer, quantifier et visualiser la contribution/ influence de chaque caractéristique et le faire transparaitre dans le résultat du modèle. Récemment, un certain nombre de méthodes d'interprétation de réseaux de neurones profonds ont été proposées où l'interprétation est principalement représentée par la visualisation des parties de l'image ayant une forte ou faible contribution pour la décision [4]. Nous pouvons identifier trois principales catégories d'approches d'explicabilité de réseaux de neurones profonds :

a. Explicabilité basée sur les perturbations : par le remplacement, permutation ou occlusion des caractéristiques ou groupe de caractéristiques afin de calculer la différence (prédiction). Une faible différence signifie que les caractéristiques remplacées (permutées/occlues) sont moins importantes et vice versa [12]. Dans ce contexte, une nouvelle méthode « RISE, Randomized Input Sampling for Explantation of Black-Box Models » [13]. Cette dernière calcule des masques basés sur un suréchantillonnage de masques binaires remplis de manière aléatoire et crée une carte thermique basée sur la pertinence de chaque masque pour la prédiction. Ces méthodes sont plus lourdes en temps de calcul que la plupart des méthodes post-hoc, elles nécessitent de remplacer les valeurs d'entrée et de calculer le résultat des modèles à chaque itération.

b. Explicabilité basée sur l'analyse de rétropropagation : cette approche ne s'appuie pas sur les résultats de modèles mais plutôt sur la structure interne des modèles. Le principe est d'identifier la saillance des caractéristiques d'entrée et couches intermédiaires en analysant les gradients transmis de la sortie vers l'entrée pendant l'apprentissage du réseau de neurones. Il existe dans la littérature plusieurs variantes de méthodes utilisant cette approche tels que : Gradient*Input, Integrated Gradient, Guided Backpropagation, Deconvolutional Network, Layer-wise Relevance Propagation (LRP) [5], DeepLIFT, DeepTaylorDecomposition, Class Activation Mapping (CAM), Grad-CAM [6], etc.

c. Explicabilité basée sur les méthodes CAM : les méthodes « CAM » sont appliquées aux réseaux de neurones convolutifs et ont la particularité d'exploiter les couches d'activation (généralement les dernières couches convolutives) pour produire des cartes de saillance. La mtéhode Grad-CAM [14] est la plus largement utilisée et pondère les cartes d'activation par les gradients obtenus par une rétropropagation à partir du neurone de sortie d'une classe choisie à la couche convolutive. Il existe de nombreuses variantes qui combinent des activations de différentes couches (Layer-CAM [15], Poly-CAM [16]), agréger les résultats pour l'entrée image à différentes échelles (CAMERAS [17]), ou utiliser les activations comme masques pour prédire leur importance (Score-CAM [18]).

d. Explicabilité basée sur les modèles Proxy : permettent de réduire la complexité de modèles Machine et Deep Learning afin d'avoir des modèles ayant des comportements et résultats similaires aux modèles initiaux mais utilisant des architectures simples dont le fonctionnement est plus facile à expliquer. Les principales méthodes utilisant cette approche sont : LIME [7] (modèle linéaire), arbre de décision et DeepRed [8], etc.

3. Conclusion :

Nous tenons à rappeler que l'objectif de ce Workshop est de mettre en œuvre des modèles de classification d'images/vidéos et de détection d'objets avant de les embarquer dans un module « ***Edge AI*** » pour villes intelligentes. Nous vous invitons à commencer par le développement des modèles (3) avant de les porter vers la carte Jetson Xavier en offrant un traitement temps réel (ou proche) à partir d'images capturées via la webcam. A l'issue de cela, vous pourrez intégrer les tâches de compression et explicabilité et mettre en œuvre l'envoi d'alertes et notifications. Vous avez aussi la possibilité d'intégrer d'autres options imaginées par vos soins. Nous vous proposons de quantifier les résultats de vos modèles choisis et développés en complétant le tableau 1.

Note 3 : à partir de **dimanche 19/05 à 11h**, vous pouvez soumettre vos deux modèles (classification et localisation de feu) sur un challenge en ligne pour évaluer vos modèles avec une BD de test en termes précision, temps de calcul et espace mémoire. Le classement sera fait sur base du meilleur compromis entre ces trois paramètres. L

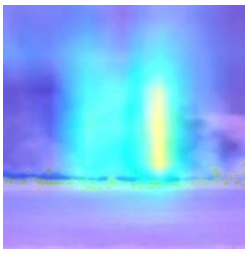
Note 4 : L'explicabilité doit être appliqué au modèle de classification seulement.

	Précision (Acc)	Taille du modèle .pt (MB)	FPS	Classement LeaderBoard
Modèle « Fire Classification »				-------
Modèle « Fire Detection »				

TABLEAU 1 : EVALUATION DES MODÈLES EN TERMES DE PRÉCISION, TEMPS DE CALCUL ET ESPACE MÉMOIRE

4. Quelques exemples :

Localisation de feu



Détection explicable de feu

5. Quelques liens intéressants :

- Pruning : https://github.com/VainF/Torch-Pruning
- Quantization : https://github.com/NVIDIA-AI-IOT/torch2trt

Bibliographie

[1] Han, S., Pool, J., Narang, S., Mao, H., Gong, E., Tang, S., ... & Dally, W. J. (2016). Dsd: Dense-sparse-dense training for deep neural networks. arXiv preprint arXiv:1607.04381.

[2] Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[3] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[4] Vilone, G., & Longo, L. (2020). Explainable artificial intelligence: a systematic review. arXiv preprint arXiv:2006.00093.

[5] G. Montavon, A. Binder, S. Lapuschkin, W. Samek, and K.-R. Müller, "Layer-wise relevance propagation: an overview," in Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. Springer, 2019, pp. 193-209.

[6] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

[7] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Model-agnostic interpretability of machine learning". arXiv preprint arXiv:1606.05386.

[8] Zilke, J. R., Mencía, E. L., & Janssen, F. (2016, October). Deepred–rule extraction from deep neural networks. In International Conference on Discovery Science (pp. 457-473). Springer, Cham.

[9] Jianping Gou, Baosheng Yu, Stephen J. Maybank Dacheng Tao, "Konowledge Distillation: A survey" . Vol 129, pp 1789-1819, 2021.

[10] Sajid Anwar, Kyuyeon Hwang, Wonyong Sung, "Structured Pruning of Deep Convolutional Neural Networks", *ACM Journal on Emerging Technologies in Computing Systems,* Vol. 13, Issue 3, July 2017 Article No.: 32pp 1–18. https://doi.org/10.1145/3005348

[11] J. -H. Luo, H. Zhang, H. -Y. Zhou, C. -W. Xie, J. Wu and W. Lin, "ThiNet: Pruning CNN Filters for a Thinner Net," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525-2538, 1 Oct. 2019, doi: 10.1109/TPAMI.2018.2858232.

[12] Zeiler, M.D.; Fergus, R. "Visualizing and understanding convolutional networks". *In Proceedings of the European conference on computer vision. Springer,* 2014, pp. 818–833.

[13] Petsiuk, V.; Das, A.; Saenko, K. "RISE: Randomized Input Sampling for Explanation of Black-box Models". *In Proceedings of the BMC*, 2018.

[14] Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. "Grad-cam: Visual explanations from deep networks via gradient-based localization". *In Proceedings of the ICCV*, 2017.

[15] Jiang, P.T.; Zhang, C.B.; Hou, Q.; Cheng, M.M.; Wei, Y. "LayerCAM: Exploring hierarchical class activation maps for localization". *IEEE Trans. Image Process. 2021*, 30, 5875–5888.

[16] Englebert, A.; Cornu, O.; Vleeschouwer, C.D. "Poly-CAM: High resolution class activation map for convolutional neural networks". *arXiv:2204.13359v2* 2022.

[17] Jalwana, M.A.; Akhtar, N.; Bennamoun, M.; Mian, A. "CAMERAS: Enhanced resolution and sanity preserving class activation mapping for image saliency". *In Proceedings of the CVPR*, 2021.