**HANDS ON AI – Certificat IA de l’UMONS**

**Rapport Défi 1 :  IA et analyse d’images : détection et localisation sur images**

* **Etudiant 1 :** Gatta Nicolas
* **Etudiant 2 :** Danese Loris

1. **Description succincte du problème : choix d’options (classification et XAI + options facultatives : localisation de feu ou Vison Transformers ou pas d’option 😉)**

|  |
| --- |
| **Description du problème :**  L'objectif du défi est d'utiliser des réseaux de neurones profonds pour classifier des images, détecter et localiser des objets en l'occurrence des feux de forêt à partir d'images de caméras. |
| **Choix des options :**   Aucune option n’a été choisie pour ce défi. |
| **Qu’est-ce que la classification d’image :** La classification d’image est une tâche fondamentale du Deep Learning. Cela consiste à attribuer des étiquettes ou des catégories à des images en fonction de leur contenu de manière automatique ou bien manuelle. Cette tâche repose sur l’utilisation d’un modèle qui est l’une des étapes clés de la classification d’image de celui-ci permettant après entrainement d’identifier automatiquement le contenu des images et de pouvoir les catégoriser. |
| **Qu’est-ce que le XAI :** L'intelligence artificielle explicable ou Explainable AI dans la langue de Shakespeare (XAI) fait référence aux technologies et méthodes qui permettent à l’être humain d’obtenir des explications compréhensibles sur les décisions prisent par les modèles. Pour cela, une visualisation grâce à une carte de couleur est souvent nécessaire pour permettre la bonne compréhension et l’identification des pixels responsables de chaque décision prise par le modèle. |

1. **Description et évaluation des données (Small, BD1, BD2, BD3 et données personnelles)**

|  |
| --- |
| **Donner une analyse de chaque base de données avec évaluation de son influence sur le résultat de classification**  L’analyse des bases de données a été effectuée après avoir retiré tous les fichiers en double se trouvant de celle-ci. Les fichiers en double ont été identifiés grâce au petit mot clé « copie » se trouvant dans leurs noms. |
| |  | | --- | | **DB1 :**   * **Nombre total de fichiers** : 1379 * **Répartition des classes** :   + start\_fire: 7.25%   + fire: 56.49%   + no\_fire: 36.26% * **Influence potentielle** : Le déséquilibre est notable, avec une forte majorité de données dans la classe 'fire' (56.49%), ce qui pourrait conduire le modèle à être biaisé vers cette classe. | | **DB2 :** | | * **Nombre total de fichiers :** 74 * **Répartition des classes :**   + start\_fire: 18.92%   + fire: 40.54%   + no\_fire: 40.54% * **Influence potentielle :** La base de données DB2 présente une répartition relativement équilibrée entre les classes fire et no\_fire mais la classe start\_fire se voit délaisser. Cela pourrait avoir comme conséquence de limiter la capacité du modèle à généraliser correctement. | | **DB3 :** | | * **Nombre total de fichiers :** 1500 * **Répartition des classes :**   + start\_fire: 33.33%   + fire: 33.33%   + no\_fire: 33.33% * **Influence potentielle :** Une répartition équilibrée peut permettre au modèle d'apprendre de manière plus équitable entre les différentes classes. | | **Small :** | | * **Nombre total de fichiers :** 791 * **Répartition des classes :**   + start\_fire: 19.34%   + fire: 39.32%   + no\_fire: 41.34% * **Influence potentielle :** Bien que les classes soient relativement équilibrées, le nombre total de fichiers est trop petit, ce qui pourrait limiter la capacité du modèle à généraliser correctement. | |
|  |
| **Donner la méthode de collecte de votre base de données personnelle (vidéos, data augmentation, etc.)**  Cela n’étant pas obligatoire, nous ne l’avons pas fait. Cependant, si nous avions dû le faire, nous aurions utilisé le code fourni « 4. extract\_from\_frames.py » ou nous aurions utilisé les ressources fournies par le site Kaggle qui répertorie une grande quantité des dataset. Il nous aurait alors suffi d’en choisir un ou plusieurs correspondants à notre cas actuel pour créer notre base de données personnelles. |

1. **Gestion des données (train/validation/test)**

|  |
| --- |
| **Décrire l’influence du taux division des données (train/validation/test) sur le résultat de classification**  Le taux de division des données est généralement lié à la création de trois ensembles, respectivement entraînement, validation et test. Celui-ci si mal choisi peut avoir des impacts significatifs qui sont les suivants :   * **Surajustement (overfitting) :**   Si une trop grande partie est allouée à l’ensemble d’entrainement et une trop petite portion à l’ensemble de validation/test, on pourrait avoir un modèle qui surajuste les données et ne généralise pas. Le modèle pourrait se retrouver avec des performances plus que médiocres.   * **Sous-ajustement (underfitting) :**   Si une trop petite partie est allouée à l’ensemble d’entrainement et une trop grande partie à l’ensemble de validation/test, le modèle risque de ne pas être en mesure d’apprendre correctement les caractéristiques de données entrainant alors des problèmes de généralisation ainsi que des de performances médiocres.   * **Sensibilité aux données :**   Certains jeux de données peuvent être assez sensibles à la manière dont les données sont divisées. Par conséquent différentes répartitions peuvent donner des résultats différents. Pour minimiser cela, il est souvent recommandé d’utiliser la validation croisée (Cross-Validation).   * **Optimisation des hyperparamètres :**   Si le taux de division est mal choisi pour les données de validations, cela pourrait poser des problèmes lors de l’optimisation des hyperparamètres. En effet, on utilise les données de validations pour s’assurer que les paramètres sont corrects ou si ceux-ci ont besoin d’être ajuster. Si l’ensemble est trop petit, cela pourrait conduire à des ajustements sous-optimaux et si l’ensemble est trop grand, cela pourrait conduire à des ajustements suroptimaux.  De manière générale, la convention est d’utiliser une répartition de 70 % pour l’entrainement, 15% pour la validation et 15% pour le test. |

|  |
| --- |
| **Présenter votre meilleure courbe d’entrainement et matrice de confusion (avec les données de test)**  Voici la courbe d’entrainement ainsi que la matrice de confusion sur les données test du meilleur modèle présent dans « *V. Tableau comparatif entre vos 5 meilleurs modèles au maximum (03 modèles au minimum)* »    Une image contenant texte, capture d’écran, carré, Rectangle  Description générée automatiquement |
| Interpréter brièvement votre courbe d’entrainement  **Perte d'entraînement et perte de validation :**   * **Perte d'entraînement** (plus basse est meilleure) : Elle commence relativement élevée et diminue au fil des époques, indiquant que le modèle apprend correctement à partir des données, arrive à généraliser sans pour autant overfit. * **Perte de validation** (plus basse est meilleure) : Elle suit une tendance similaire, diminuant au fil des époques. Cela suggère que le modèle généralise bien et qu’il arrive à classer correctement des données non vues.   **Précision d'entraînement et précision de validation** :   * **Précision d'entraînement** (plus élevée est meilleure) : Elle commence à une valeur plus basse et augmente au fil des épochs, montrant que le modèle améliore sa performance sur les données d'entraînement. * **Précision de validation** (plus élevée est meilleure) : Elle suit une tendance similaire, augmentant au fil des épochs. Cela indique que le modèle se comporte également bien sur l'ensemble de validation et dont qu’il généralise bien.   **Comparaison des métriques d'entraînement et de validation :** Si les métriques d'entraînement s'améliorent tandis que celles de validation stagnent ou se dégradent ce qui indique un surajustement. Cependant, ce n’est pas le cas car on peut observer une amélioration de manière constante ce qui suggère une bonne performance du modèle.  **Convergence :** Les métriques d'entraînement et de validation semblent converger ce qui est un signe positif. La convergence indique que le modèle se stabilise et a appris correctement.  **Surajustement potentiel :** Il pourrait y avoir une légère indication de surajustement car la précision d'entraînement est supérieure à la précision de validation dans les épochs ultérieures. Cependant, la différence n’étant pas grande et les précisions d'entraînement ainsi que de validation étant élevées indiquent que le modèle est sur la bonne voie. |
| Interpréter brièvement votre matrice de confusion :  **Classe 1 :**   * *Vraies positifs* (correctement prédits en classe 1) : 39 * *Faux négatifs* (prédits comme non classe 1, mais sont en classe 1) : 26 + 35 = 61   **Classe 2 :**   * *Vraies positifs* (correctement prédits en classe 2) : 36 * *Faux négatifs* (prédits comme non classe 2, mais sont en classe 2) : 32 + 32 = 64   **Classe 3 :**   * *Vraies positifs* (correctement prédits en classe 3) : 37 * *Faux négatifs* (prédits comme non classe 3, mais sont en classe 3) : 36 + 27 = 63 |

1. **Modèles proposés (modèle personnel et/ou existant avec Transfert Learning)**

|  |
| --- |
| **Décrire les modèles proposés avec une analyse comparative en termes d’architectures, profondeurs et courbes d’entraînement (vous pouvez inclure une photo d’architectures de vos modèles)** Pour cette étape, nous allons d’abord présenter les modèles en termes de profondeur d’architecture et de courbe d’entrainement pour ensuite les comparer dans un tableau.  **VGG16 :**   * *Profondeur* : Comme indiqué dans le nom, la profondeur du VGG16 est de 16 couches qui sont décomposées en 13 couches convolutives et 3 couches entièrement connectée. * *Architecture* :   + **Couche d’entrée** : reçoit une image 224×224 en entrée   + **Couches convolutives** : 13 premières couches du modèle qui sont chacune suivies d’une couche de normalisation ainsi que d’une fonction d’activation ReLu (Rectified Linear Unit). Il faut aussi y inclure les filtres de convolutions qui commence en 3x3 mais qui augmente en taille au plus on s’enfonce dans le réseau.   + **Couches entièrement connectées** : 3 couches entièrement connectées dont les 2 premières couches comportent chacune 4096 canaux et la troisième couche comporte 1000 canaux.   Structure VGG 16   * *Courbe d’entrainement* :     **Densenet201**   * *Profondeur* : Comme indiqué dans le nom, la profondeur du Desnet201 est de 201 couches qui sont décomposées en couches de convolution, les couches de normalisation, les couches d'activation, les couches entièrement connectées, bloc dense, etc. * *Architecture* :   + **Couche d’entrée : Image RGB en entrée 224x224 pixels**   + **Couche Convolutionnelle Initiale : Convolution avec des opérations telles que convolution, normalisation, et activation.**   + **Bloc Dense : Composé d'unités "bottleneck layers" dans laquelle chaque unité utilise les caractéristiques de toutes les couches précédentes.**   + **Couche de Transition :** **Elle réduit la taille des caractéristiques spatiales, utilise la convolution et le sous-échantillonnage, tout en ajustant le nombre de canaux pour contrôler la complexité du modèle.**   + **Global Average Pooling : Opération de pooling globale qui calcule la moyenne de chaque canal, réduisant les dimensions spatiales à une seule valeur par canal.**   + **Couche entièrement connectée : Elle prend les caractéristiques des couches précédentes, les fusionne pour produire les sorties finales du modèle. On retrouve souvent cela suivi d'une fonction softmax pour la classification.**      * *Courbe d’entrainement* :     **InceptionV3**   * *Profondeur* : La profondeur du modèle inceptionV3 est de 48 couches décomposées en stem network, modules Inception, bloc de réduction, module Inception, Global Average pooling, couche entièrement connectée * *Architecture* :   + **Couche d’entrée :** L'image en entrée est généralement de taille 299x299 pixels.   + **Stem :** InceptionV3 commence par une séquence de couches de convolutions et de pooling pour extraire des caractéristiques de bas niveau de l'image.   + **Modules Inception :** Chaque module Inception est constitué de plusieurs branches parallèles, chacune utilisant différents types de filtres (convolutions 1x1, 3x3, 5x5, pooling) pour capturer des informations à différentes échelles.   + **Bloc de réduction :** Entre les modules Inception, on retrouve des blocs de réduction de dimension pour réduire la résolution spatiale et le nombre de canaux pour réduire la charge de calcul.   + **Global Average Pooling :** Plutôt que d'utiliser des couches entièrement connectées traditionnelles, InceptionV3 utilise une couche de pooling globale moyenne pour réduire les dimensions spatiales à une seule valeur par canal.   + **Couche de sortie :** La couche de sortie est généralement une couche dense avec des unités correspondant au nombre de classes dans la tâche de classification.   + **Softmax :** La dernière couche est généralement activée par une fonction softmax pour obtenir des probabilités de classe.   Guide avancé d'Inception v3 | Cloud TPU | Google Cloud   * *Courbe d’entrainement* :     **Resnet152**   * *Profondeur* : Comme indiqué dans le nom, la profondeur du Resnet152 est de 152 couches décomposées en les couches de convolution, les couches de normalisation, les couches d'activation, les couches entièrement connectées, etc. * *Architecture* :   + **Couche d'entrée :** L'image en entrée est généralement de taille 224x224 pixels.   + **Convolution initiale :** Une couche de convolution initiale suivie d'une couche de normalisation et d'une activation ReLU.   + **Blocs résiduels :**     - **ResNet-152 est composé de plusieurs blocs résiduels. Chaque bloc est constitué de plusieurs couches de convolution empilées.**     - **La particularité réside dans la présence de connexions résiduelles qui permettent le saut des informations directement d'une couche à une autre, en court-circuitant certaines couches. Cela facilite le flux d'informations à travers le réseau, contribuant à prévenir le problème de disparition du gradient.**   + **Blocs de transition :** Entre les blocs résiduels, il peut y avoir des blocs de transition qui effectuent une réduction de la résolution spatiale tout en augmentant le nombre de canaux.   + **Couche de pooling :** ResNet-152 utilise une couche de pooling moyen pour réduire la dimension spatiale avant d'atteindre les couches entièrement connectées.   + **Couches entièrement connectées :** La sortie de la couche de pooling est aplatie et alimentée dans une ou plusieurs couches entièrement connectées pour la tâche de classification.   + **Couche de sortie :** La couche de sortie est généralement une couche dense avec des unités correspondant au nombre de classes dans la tâche de classification.      * *Courbe d’entrainement* :     **Comparaison des différents modèles :** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Modèle | Architecture | Profondeur | Courbe de perte | Courbe de précision | | VGG16 | Conv(3x3) - Pooling - Conv(3x3) - Pooling - ... - FC | 16 | On peut remarquer que les deux droites se croisent après 5 épochs et qu’elles commencent toutes deux à converger vers 25 épochs | On peut remarquer que les deux droites se croisent après 7 épochs et qu’elles commencent toutes deux à converger vers 25 épochs | | ResNet152 | Residual Blocks | 152 | On peut remarquer que les deux droites se croisent après 9 épochs et qu’elles commencent toutes deux à converger vers 10 épochs | On peut remarquer que les deux droites se croisent après 6 épochs et qu’elles commencent toutes deux à se stabiliser vers 8 épochs | | InceptionV3 | Inception Modules | 48 | On peut remarquer que les deux droites se croisent après 7 épochs et qu’elles commencent toutes deux à converger vers 20 épochs | On peut remarquer que les deux droites se croisent après 5 épochs et qu’elles commencent toutes deux à se stabiliser vers 20 épochs | | DenseNet201 | Dense Blocks | 201 | On peut remarquer que les deux droites se croisent après 2 épochs et qu’elles commencent toutes deux à converger vers 8 épochs | On peut remarquer que les deux droites se croisent après 2 épochs et qu’elles commencent toutes deux à se stabiliser vers 8 épochs | |

1. **Tableau comparatif entre vos 5 meilleurs modèles au maximum (03 modèles au minimum)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Caractéristiques** | **Modèle 1** | **Modèle 2** | **Modèle 3** | **Modèle 4** |
| **Entrainement** | | | | |
| **Classifier** | VGG16 | DenseNet201 | InceptionV3 | Resnet152 |
| **Epochs** | 30 (stop = 17) | 20 (stop = 16) | 20 | 10 |
| **Batch** | 16 | 32 | 16 | 16 |
| **Input Dim** | 224 | 224 | 229 | 224 |
| **Dataset** | DB3 | DB3 | DB3 | DB3 |
| **Nombre de couches réentraîner** | Toutes | Toutes | Toutes | Toutes |
| **Learning Rate** | 0.0001 | 0.001 | 0.001 | 0.01 |
| **Taille du modèle en MB** | 123 | 175 | 204 | 473 |
| **Training\_Loss** | 0.2304 | 0.0492 | 0.3111 | 0.0125 |
| **Training\_Accuracy** | 0.9132 | 0.986 | 0.9034 | 0.9958 |
| **Validation\_Loss** | 0.2234 | 0.0884 | 0.3389 | 0.1649 |
| **Validation\_Accuracy** | 0.9079 | 0.9652 | 0.8903 | 0.9700 |
| **Temps (optionnel)** | 351 secondes | 353 secondes | 393 secondes | 504 secondes |
| **Test 1 en Local (**[**https://nextcloud.ig.umons.ac.be/s/RsZpqtYfDgYzKxr**](https://nextcloud.ig.umons.ac.be/s/RsZpqtYfDgYzKxr)**)** | | | | |
| **Test\_Loss** | 0.2998548150062561 | 0.2631300091743469 | 0.2919806241989136 | 0.24706260859966278 |
| **Test\_Accuracy** | 0.903333306312561 | 0.8966666460037231 | 0.8966666460037231 | 0.9266666769981384 |
| **Test 2 en ligne (**<https://challenges.hackia.eu/login/>) | | | | |
| **Classement challenge en ligne** | 1 | 5 | 5 | 2 |
| **Score en ligne** | 99.16666746139526 | 95.83333134651184 | 94.9999988079071 | 98.33333492279053 |

1. **Test des modèles sur vidéo (lien vers la vidéo annotée)**

**Analyser et interpréter vos résultats lors de l’application du modèle sur une vidéo de votre choix**

Durant l’analyse de la première partie de la vidéo par le modèle, on peut remarquer que celui qui confond à plusieurs reprises l’état « fire » et l’état « start\_fire ». On peut expliquer cela par 2 choses, l’une le modèle détecte la fumée à l’arrière qui se dégage de l’incendie ou l’autre qui est que la fumée dégagée par le feu est venue accentuer l’arrière presque blanc de la vidéo ce qui aurait causé la confusion.

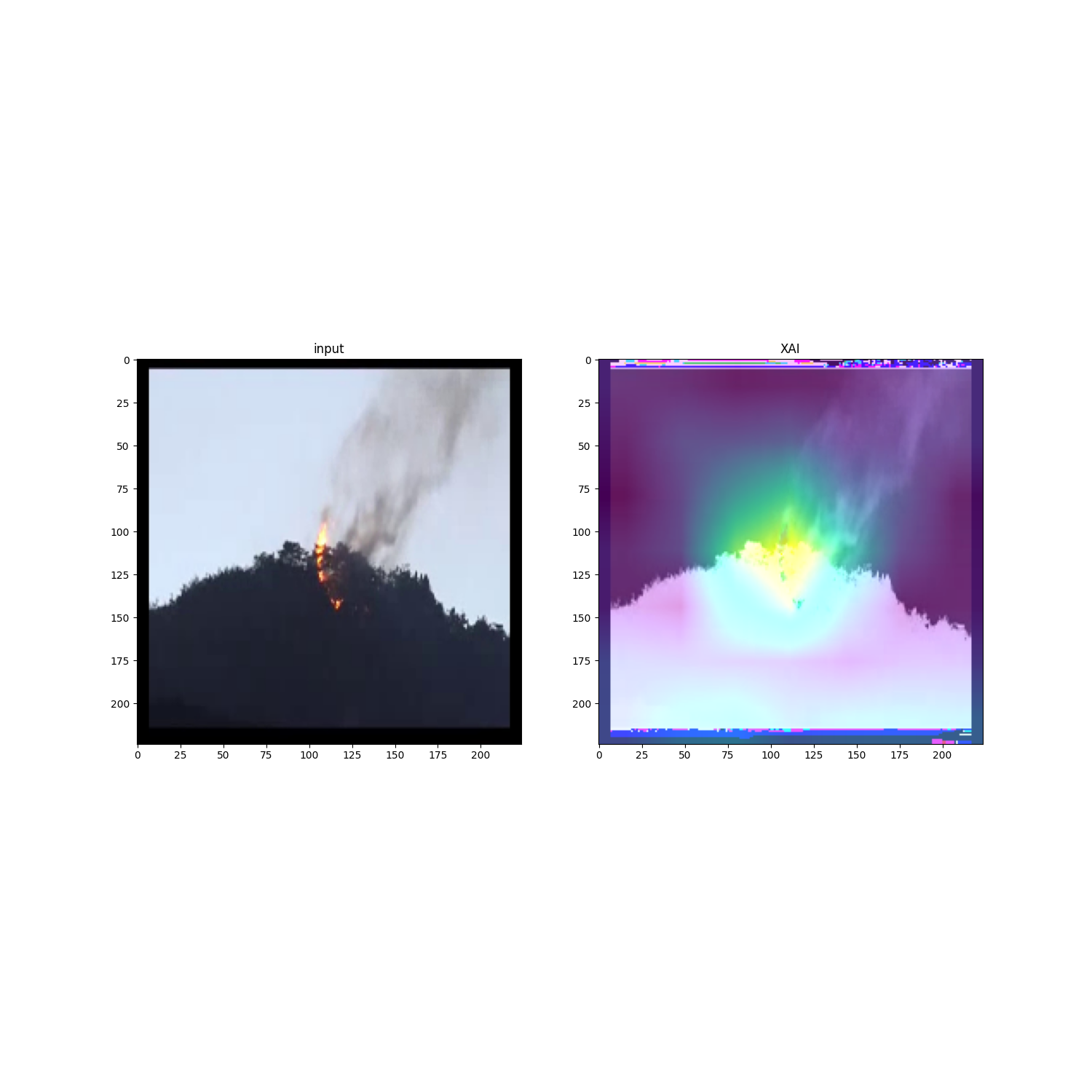
Durant l’analyse de la deuxième partie de la vidéo par le modèle, on peut remarquer que le pourcentage « no\_fire » ne descend pas en dessous de 60% montrant bien que le modèle est assez sûr de lui sur sa prédiction et qu’il arrive à bien détecter qu’il n’y a pas de feu. Cependant, on peut remarquer que le pourcentage oscille entre 60 % – 70 %, ce qui peut s’expliquer par la présence de nuage pouvant être confondu avec de la fumée provenant d’un début d’incendie.

Durant l’analyse de la troisième partie de la vidéo par le modèle, on peut remarquer un pourcentage plus qu’élevé à plus de 90 % durant toute la durée de cette partie de la vidéo. On peut très facilement expliquer cela par la présence en grande quantité de fumée sur cette partie de vidéo qui est signe d’un départ de feu (d’après la base de donnes utilisée pour l’entrainement)  
  
**Lien vers vidéo annotée** : <https://youtu.be/rvtvnyStn6I> (la vidéo a été créée en utilisant 3 fichiers vidéo combinés)

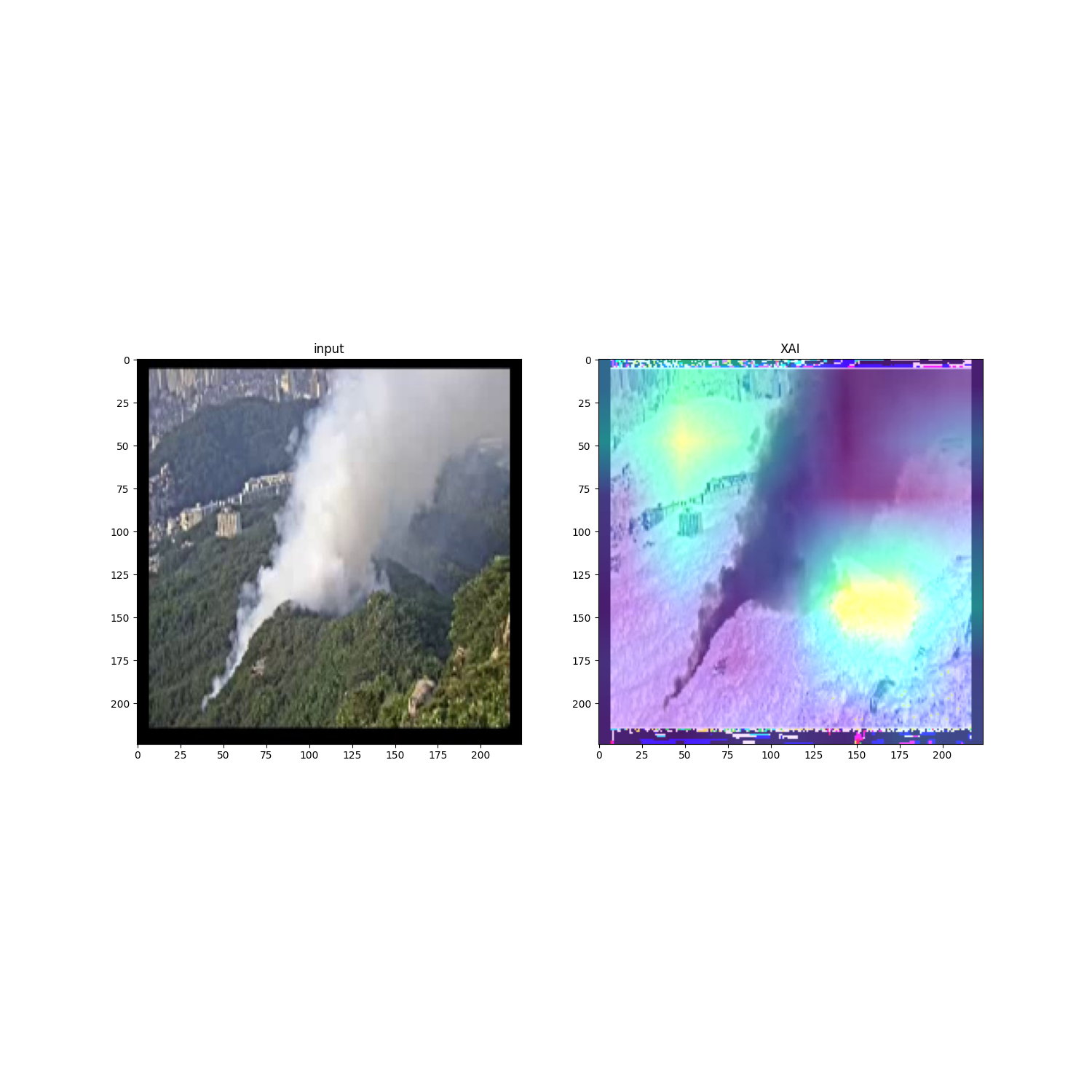
1. **Analyse des résultats et conclusions**

On peut expliquer les résultats de ce modèle en utilisant le XAI. Comme indiquer au début, le XAI va nous permettre de comprendre plus « facilement » pourquoi le modèle a effectué telle prédiction au lieu d’une autre. Nous allons pour cette question montrer quelques exemples, mais le reste de ceux-ci peuvent être trouvés dans le dossier « images » du dossier « 3 - Explicabilité et interprétabilité »

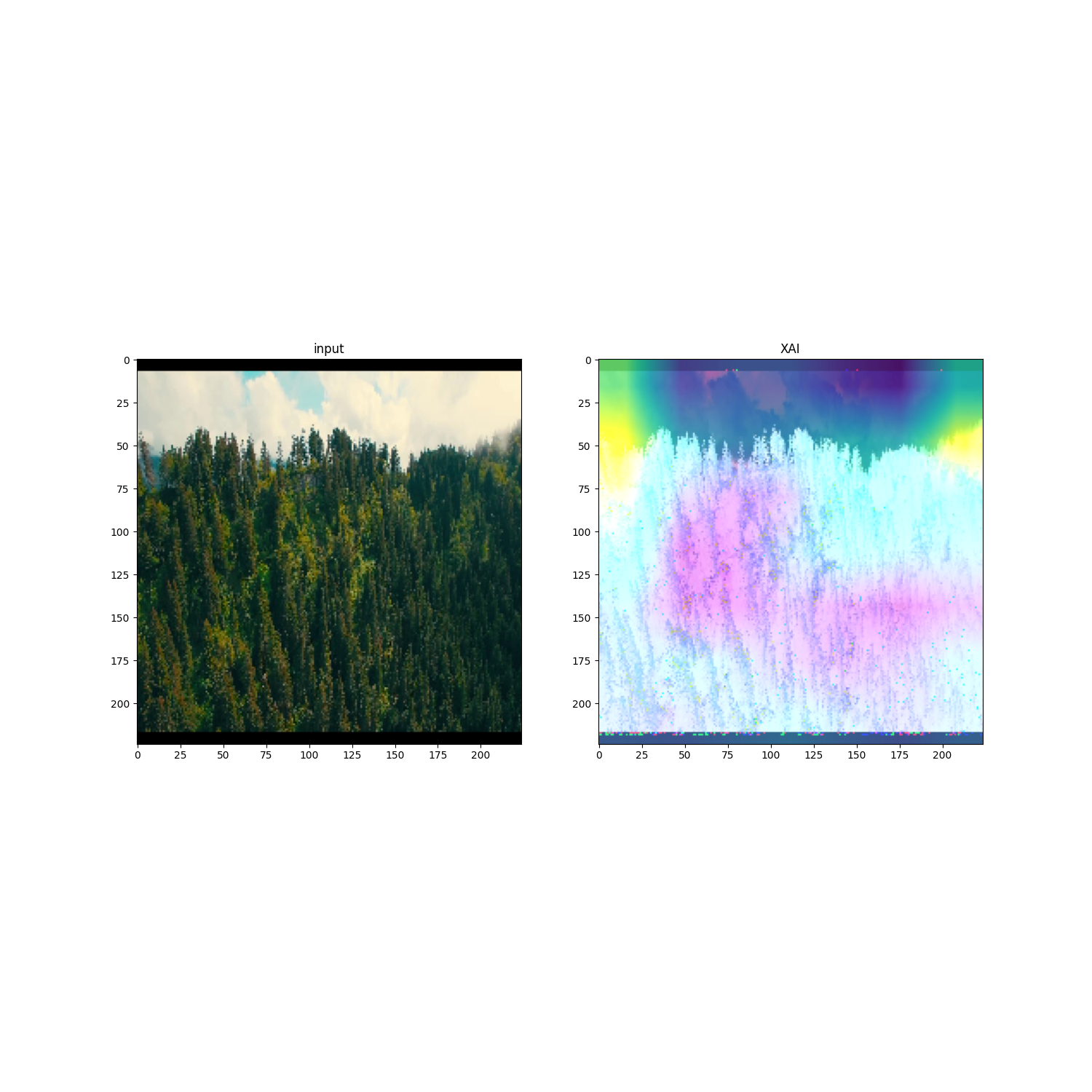
Commençons par le XAI pour la prédiction « fire ». On peut voir en examinant l’image produite par le XAI que celui-ci a quelques difficultés à classer l’image dans la bonne catégorie à cause à la présence d’une grande majorité de vert et jaune ainsi que violet. Ce qui est lié au point suivant nous indique bien que le modèle à du mal à différencier un « fire » d’une « start\_fire » dans certains cas.



Ensuite, regardons le XAI pour la prédiction « start\_fire ». On peut voir en examinant l’image produite par le XAI que celui-ci parvient à classer l’image dans la bonne catégorie grâce à la présence d’une grande majorité de couleurs sombres ainsi que de couleurs vertes devant surement référencer le cas d’un « start\_fire »



Enfin, examinons le XAI de la partie « no\_fire ». On peut remarquer en examinant l’image produite par le XAI que celui-ci parvient à classer l’image dans la bonne catégorie grâce à la présence en grande majorité de la couleur violette et d’une couleur blanche bleutée devant donc surement référencer le cas d’un « no\_fire »



En conclusion, grâce au XAI nous avons pu comprendre comment le modèle atteint telle ou telle prédiction. De plus, nous avons pu remarquer pourquoi le modèle se retrouve hésitant de temps à autre. On peut donc conclure que le modèle est un modèle assez bon même s’il lui arrive de confondre « fire » et « start\_fire » qui sont des états assez proches.