

Labo 5 - ARN - ImageNet

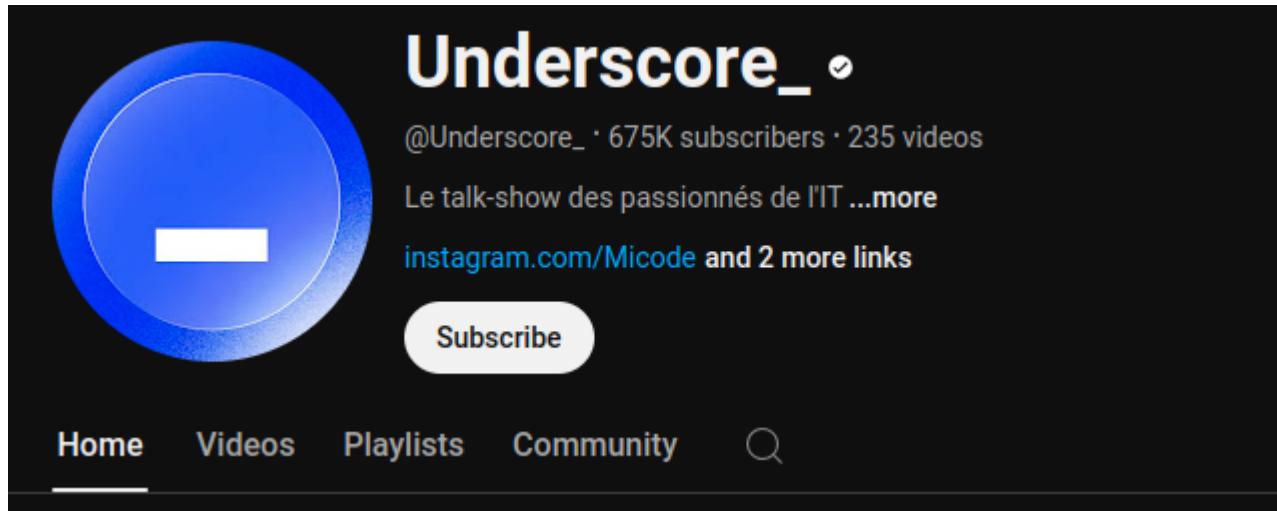
Auteurs: Felix Breval et Samuel Roland

Introduction

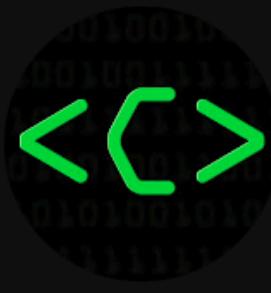
Notre modèle a pour but de classifier la chaîne YouTube à partir d'une vignette d'une de ses vidéos. Pour l'entraînement, nous allons nous baser sur le modèle existant MobileNetV2 et ses poids synaptiques (transfer learning). Les vignettes ne contenant pas fondamentalement de nouvelle caractéristique par rapport aux images de ImageNet (elles contiennent différents objets physiques, visages, personnes, textes, ...), nous estimons qu'il n'y a pas de nouvelle caractéristique à extraire et donc nous n'allons pas rajouter des couches de convolution. Nous allons juste rajouter nos couches de MLP à la fin pour qu'ils apprennent à classifier nos chaînes YouTube. Nous allons entraîner notre modèle grâce aux vignettes directement récupérée sur YouTube, c'est le plus simple, et après l'entraînement nous évaluons la performance du modèle, ce qui reste confus pour lui et ce qu'il arrive correctement classifier.

Le problème

Nous avons sélectionné 3 chaînes YouTube liées à l'informatique: **Underscore_**, chaîne de vulgarisation informatique en français, sous forme de talk show toutes les 2 semaines, créée par le Youtuber Micode



Computerphile: chaîne anglophone qui parle de beaucoup de sujets geeks ou techniques dans la sécurité ou des curiosités mathématiques derrière des algorithmes



Computerphile •

@Computerphile · 2.41M subscribers · 820 videos

Videos all about computers and computer stuff. Sister channel of Numberphile. ...[more](#)

facebook.com/computerphile and 1 more link

[Subscribe](#)

Home Videos Playlists Community 

Linus Tech Tips: chaîne anglophone qui fait des review de matériels qui contient de la technologie tout genre



Linus Tech Tips •

@LinusTechTips · 15.7M subscribers · 6.8K videos

Linus Tech Tips is a passionate team of "professionally curious" experts in consumer tech ...[more](#)

[Lttstore.com](https://lttstore.com) and 4 more links

[Subscribe](#) [Join](#)

Home Videos Shorts Live Podcasts Playlists Community 

Comme on peut le voir en partie sur les vignettes au-dessus, elles contiennent

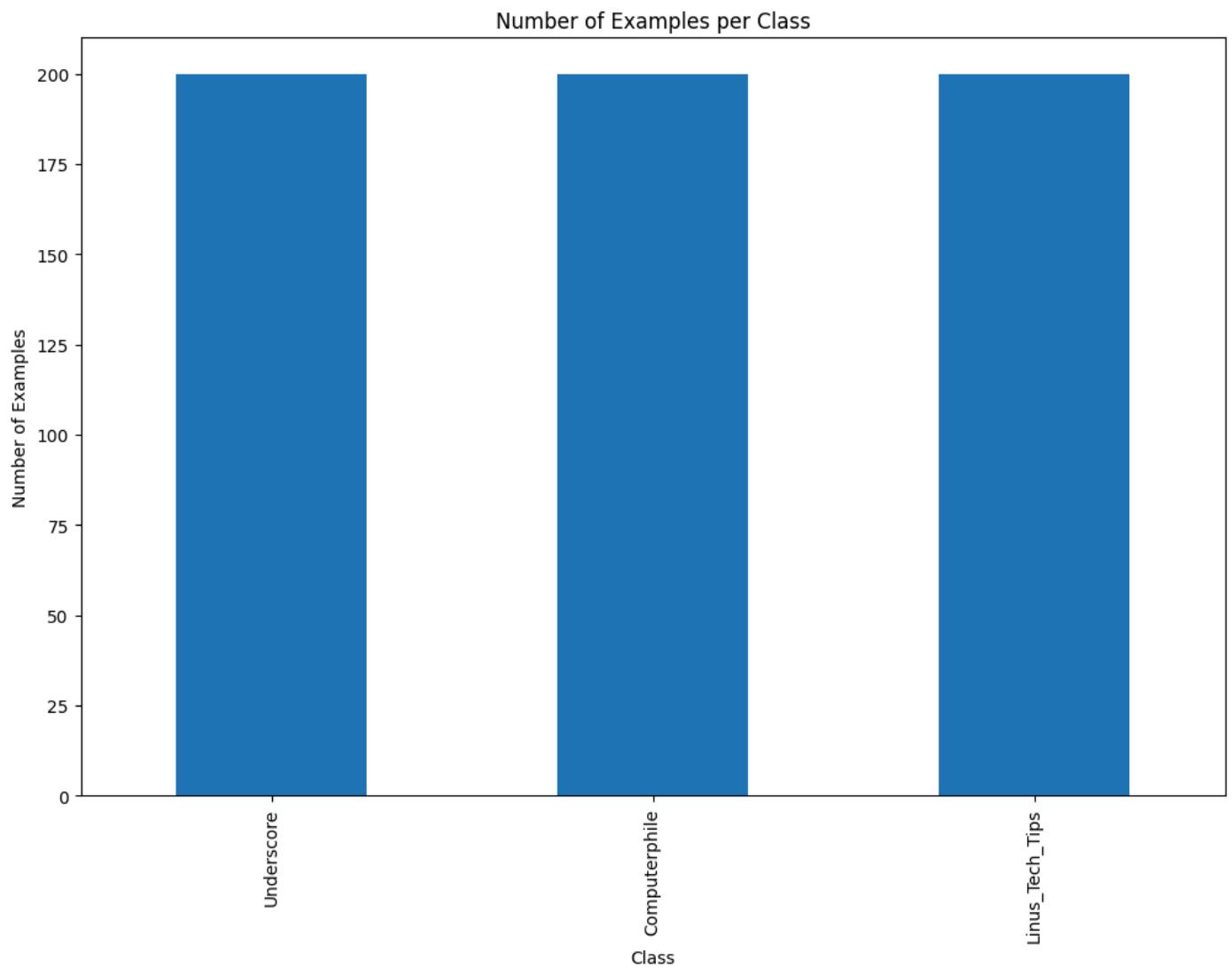


1. **Underscore_**: surtout souvent un bout de titre blanc, régulièrement un fond en teinte bleu foncé, régulièrement le visage de Micode (l'auteur) ou celui de 2-3 cohosts.
2. **Computerphile**: toujours un titre en lettre verte dans une police bien particulière, avec souvent un visage mais jamais le même
3. **Linus Tech Tips**: très souvent la tête de Linus avec un visage d'étonnement la bouche ouverte ou alors tout son corps, souvent accompagné d'un objet

Nous avons imaginé que certaines de ces caractéristiques qui reviennent constamment seraient utilisées par le modèle, nous verrons plus tard avec les résultats si c'est le cas ou non. Concernant la variété à l'interne d'une même classe, on voit que c'est bien diversifié: par ex. les vignettes d'Underscore_ ont différents formats (cela a évolué au fil du temps), n'ont pas toujours de visage, ont parfois plusieurs visages, ont parfois Micode, parfois d'autres personnes, parfois il y a un fond bleu très clair, d'autres fois c'est plus léger ou inexistant. En termes de difficulté lié aux similarités entre les classes, cela ne nous paraît pas trop compliqué à part qu'il y a souvent des visages expressifs (comme les Youtubers aiment

bien faire), il y a souvent des logos de géants de la tech et des titres parfois dans les mêmes couleurs (rouge, blanc).

Nous avons téléchargé les dernières **200** vignettes de chaque chaîne pour l'entraînement et pris les **30** suivantes comme ensemble de test. Nous avons développé un petit script Python `'thumbnail_dl/script.py'` permettant de facilement les récupérer, il suffit de le lancer depuis le dossier `'thumbnail_dl'` pour qu'il télécharge toutes les images nécessaires. Durant le téléchargement, il y a parfois quelques images qui génèrent des erreurs et ne sont pas téléchargées, mais cela ne concerne que 1 image pour Linus Tech Tips et Computerphile, et 4 pour Underscore_. Nous n'avons pas cherché à utiliser plus d'images car la chaîne Underscore_ étant récente, elle n'a pas plus de 230 vidéos publiées.



Notre dataset est donc équilibré, empêchant d'avoir une classe avec moins d'opportunités d'apprentissage que les autres.

Préparation des données

Nous n'avons pas eu besoin de labelliser ni trier nos données, parce qu'elles sont toutes déjà adaptées. En termes de préparation, nos images sont mises au même format 224x224 que celles d'ImageNet avec `'pad_to_aspect_ratio=True'` qui permet d'au lieu de recadrer vers le rectangle des vignettes en prenant le

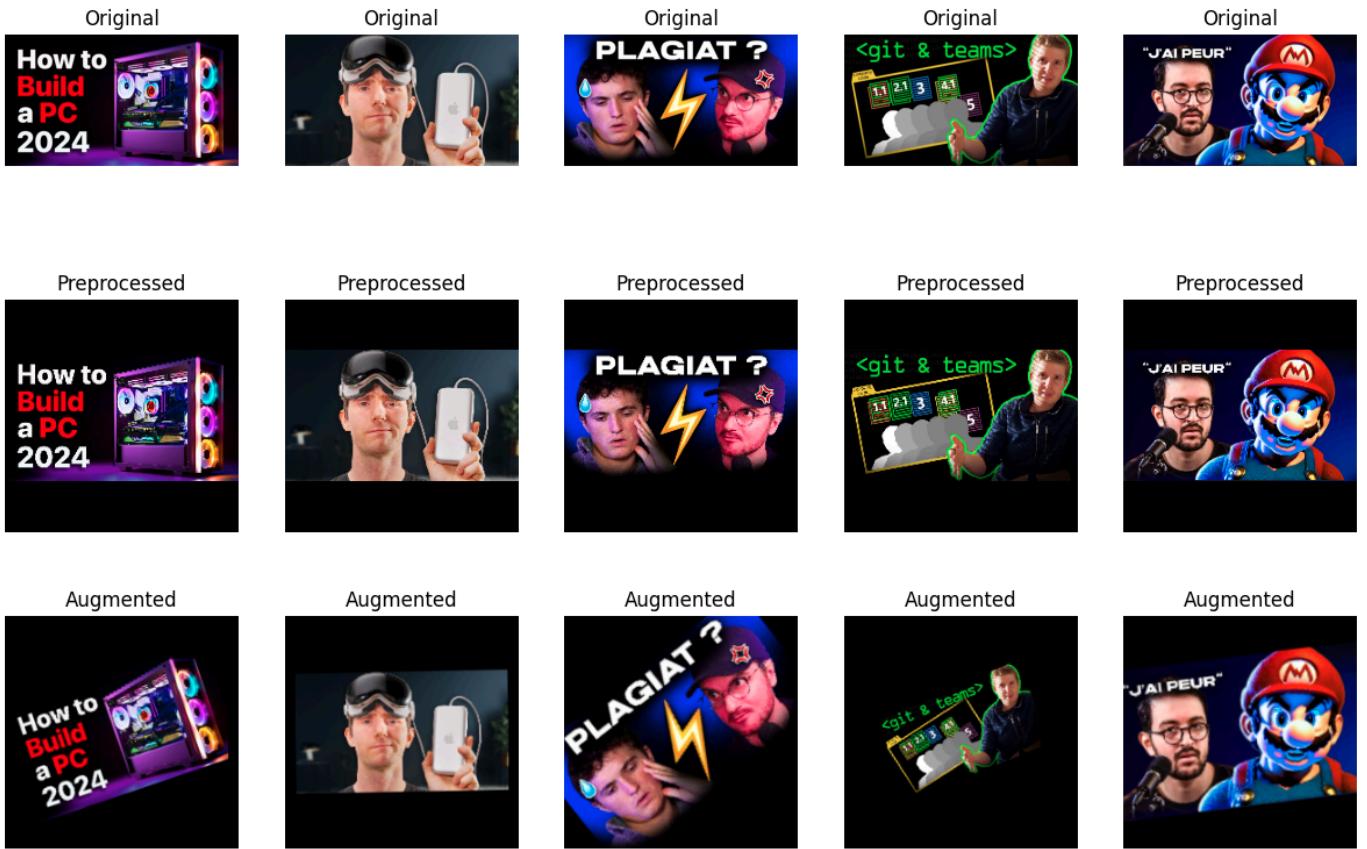
carré au milieu, il dézoom dans l'image jusqu'à que la largeur fasse 224 pixels. La zone en haut et bas est remplie de noir. Nous avons changé du mode par défaut pour ne pas perdre les bords droits et gauches des images qui contiennent des informations utiles (comme des visages sur les bords qui seraient coupés en deux autrement). Nous appliquons également une normalisation de chaque canal de chaque pixel à des valeurs entre 0 et 1.

```
# Sizes used in MobileNetV2
IMG_HEIGHT = 224
IMG_WIDTH = 224
image_preprocesses = Sequential([
    Resizing(IMG_HEIGHT, IMG_WIDTH, pad_to_aspect_ratio=True),
    Rescaling(1. / 255) # normalize values between 0 and 1
])
```

Ensuite pour augmenter nos images, nous utilisons 1 couche pour appliquer de la rotation aléatoire de plus ou moins 45 degrés en remplaçant les trous par du noir. Même chose pour un zoom aléatoire, on zoomé entre 20% vers l'avant et 50% vers l'arrière. Le but de ces transformations est à terme de mieux supporter l'utilisation en condition réelle. Quand on prend une photo avec un téléphone d'une vignette affichée sur un écran devant nous, on n'aura jamais le même angle que la photo originale, et on sera probablement plus loin (comme si on était dézoomé) c'est pour ça que nous appliquons un dézoom plus fort.

```
image_augmentations = Sequential([
    RandomRotation(factor=1/8, fill_mode='constant'), # A little bit of rotation (45degrees, fi
    RandomZoom(height_factor=(-0.2, 0.5), fill_mode='constant') # A little bit of zoom out (20%
])
```

Le preprocessing et l'augmentation peut se visualiser ci-dessous:



Conception du modèle

Notre modèle finale est défini par

- entrainement sur 30 epochs
- une taille de batch de 32
- optimizer: RMSProp
- 2 folds
- loss function: SparseCategoricalCrossentropy

Notre architecture consiste en toutes les couches non denses de ImageNet + les couches suivantes:

- Global average pooling (puis c'est l'entrée du MLP il nous faut revenir en 1 dimension)
- Dropout de 30%
- Couche dense de 100 neurones utilisant la Relu
- Couche dense de sortie de 3 neurones utilisant la softmax

Cette nouvelle partie compte **128,403** paramètres entraînables. Les couches existantes du MobileNetV2 contient **2,257,984** paramètres, mais ceux-ci sont gelés donc non entraînables.

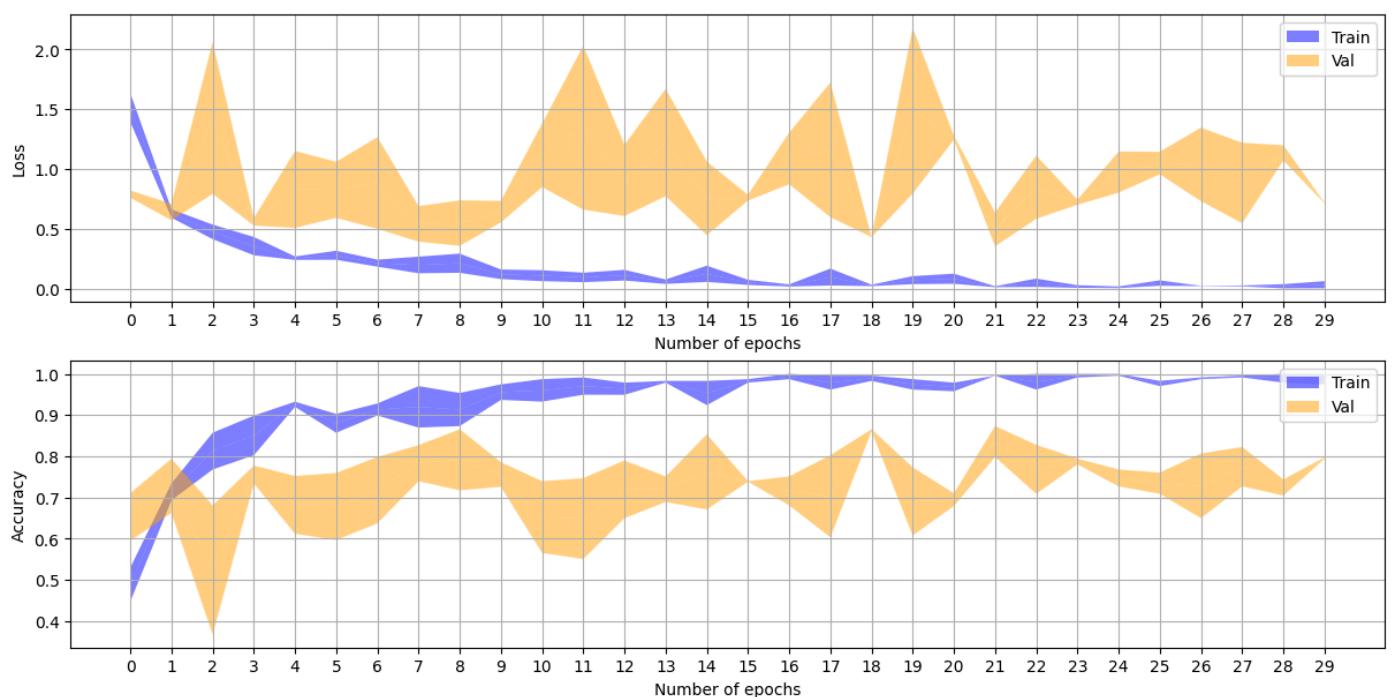
Pourquoi le transfer learning ? Cela nous permet de pouvoir entraîner un modèle avec moins d'images comme on bénéficie déjà de "l'intelligence" du modèle existant à déjà reconnaître une grande quantité d'objets, les couches de convolution fonctionnent déjà bien pour extraire toutes les caractéristiques nécessaires à notre problème. En plus de reprendre l'architecture, en reprenant les poids synaptiques,

cela nous évite de devoir réentraîner ces 2 millions de paramètres, ils sont déjà à "valeurs utiles" pour notre problème. On prend ainsi l'architecture + les poids et on retire les couches de fin (le MLP de MobileNetV2) pour pouvoir ajouter notre propre MLP spécifiques à notre problème (qui a 3 classes et non 1000).

Résultats

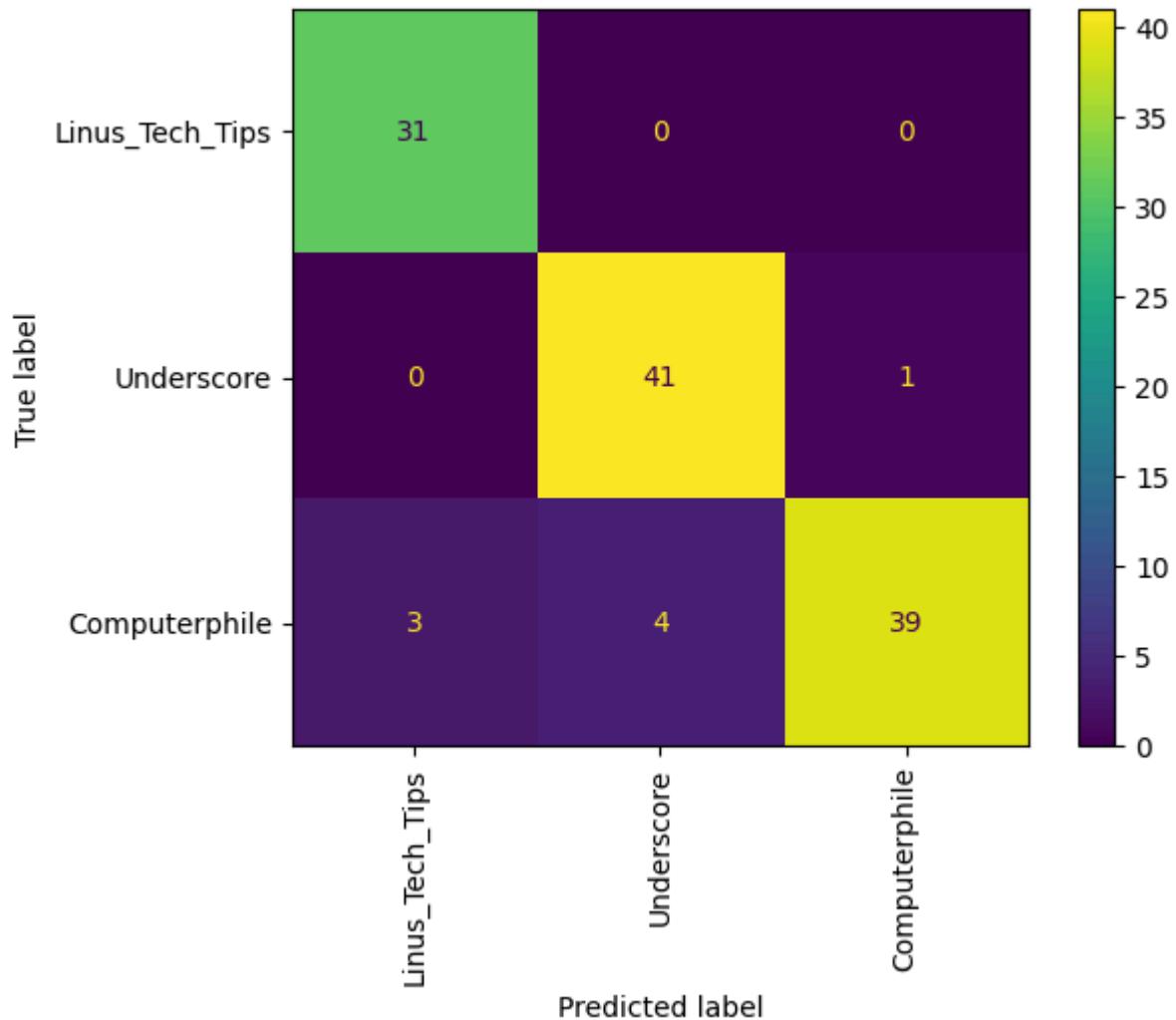
Nous n'avons pas eu le temps de tester notre modèle sur application mobile, nous avons seulement pu analyser les résultats grâce au set d'entraînement. Nous observons que les graphes d'accuracy et de loss sont difficiles à évaluer à l'oeil. Il y a beaucoup de vagues et l'épaisseur du brin représentant le minimum et maximum à travers les différents folds étant parfois large, n'aide pas à savoir si le changement d'un hyperparamètre améliore ou non notre modèle. Nous avons pris comme métrique l'accuracy maximum de validation parmi tous les folds comme valeur "finale" nous permettant de savoir si nous allions dans la bonne direction. Cette valeur étant encore moins stable que les labos précédents, le tâtonnement a été encore plus compliqué.

Graphes obtenus:



On observe qu'après 10 époques, on atteint déjà un peu le maximum possible et que cela s'améliore plus beaucoup, à part au niveau de tous les folds (le trait s'amincit). Les 2 courbes de validation par contre c'est beaucoup plus mitigé, les traits sont très épais et l'amélioration n'est pas très clair. On obtient quand même pas mal d'overfitting malgré notre dropout.

Sur le test set, notre matrice de confusion montre que Linus Tech Tips est très bien classifié, de même pour Underscore_ et par contre Computherphile a 7 images mal classifiées.



Résultats chiffrés:

- Meilleure accuracy de validation: 0.8734177350997925 at fold 1
- F1 Score sur le test set
- F1 Score per class: Computerphille: 0.95384615, Linus Tech Tips: 0.94252874, Underscore_: 0.90697674
- F1 Score global: 0.9344505445547947

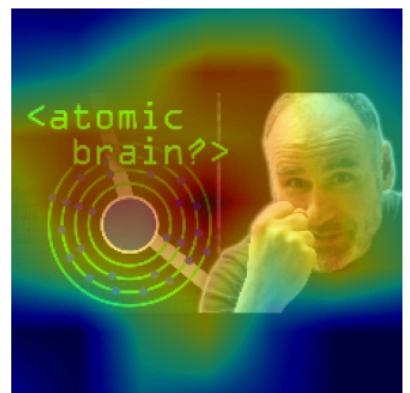
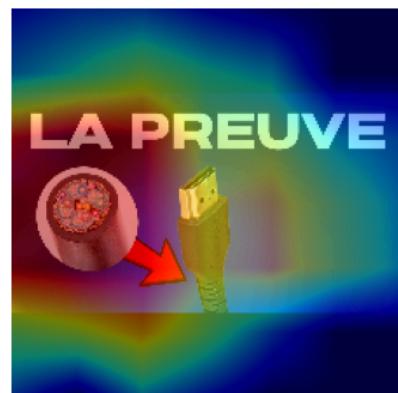
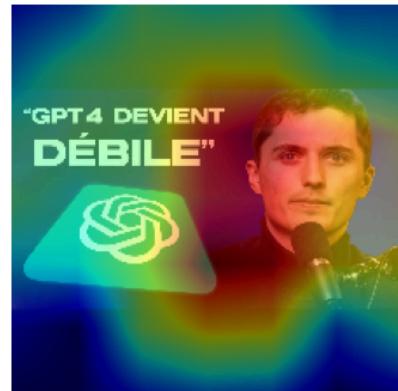
Notre test set n'a pas une performance très proche (-10%) par rapport à notre performance de validation.

On peut maintenant analyser sur quoi se concentre notre système

Linus_Tech_Tips

Underscore

Computerphile



Observons quelques images mal classifiées, on voit que le modèle se concentre bizarrement sur des parties qui n'ont pas vraiment d'intérêts comme certaines bordures, certains fonds non communs et peu souvent sur le visage de Linus ou le texte vert caractéristique de Computerphile.

True: Underscore, Predicted: Linus_Tech_Tips



True: Underscore, Predicted: Linus_Tech_Tips



True: Underscore, Predicted: Linus_Tech_Tips



True: Underscore, Predicted: Computerphile



True: Underscore, Predicted: Computerphile



True: Linus_Tech_Tips, Predicted: Underscore



True: Underscore, Predicted: Linus_Tech_Tips



True: Underscore, Predicted: Computerphile

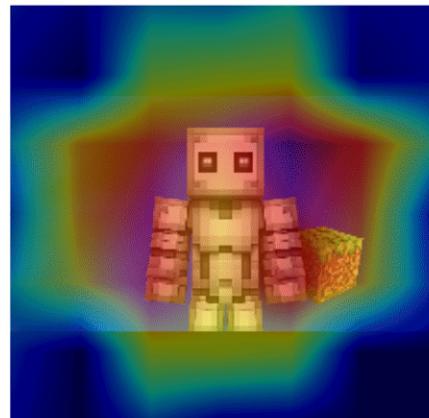


Nous pourrions améliorer notre dataset en ajoutant plus d'images des chaines qui ont plus de vidéos.

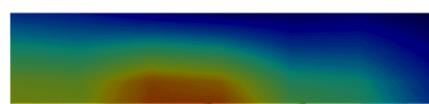
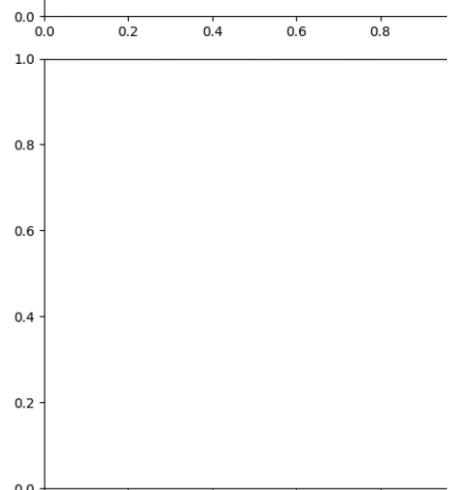
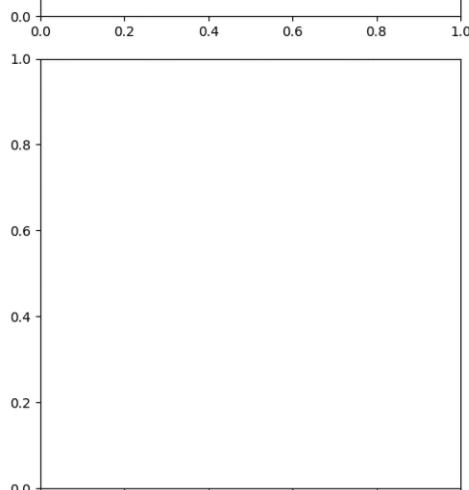
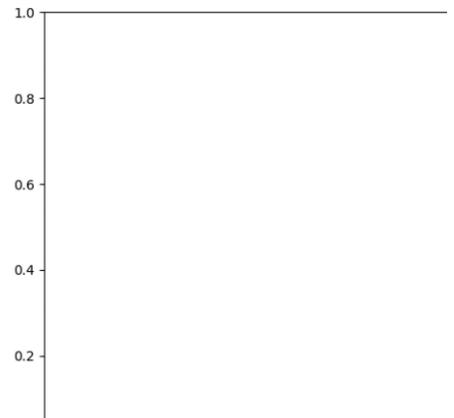
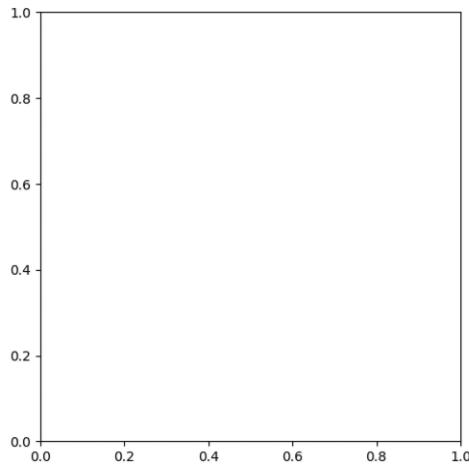
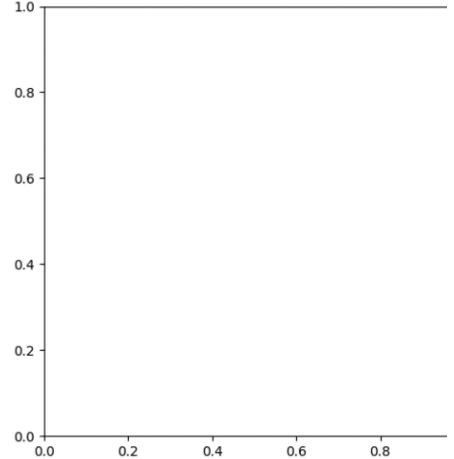
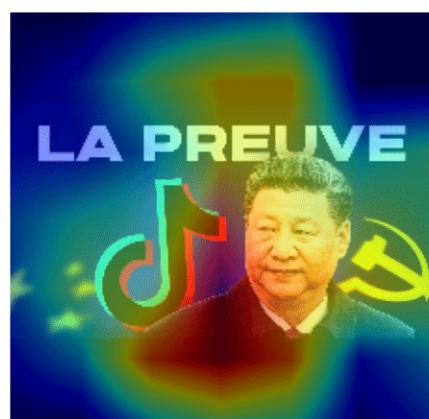
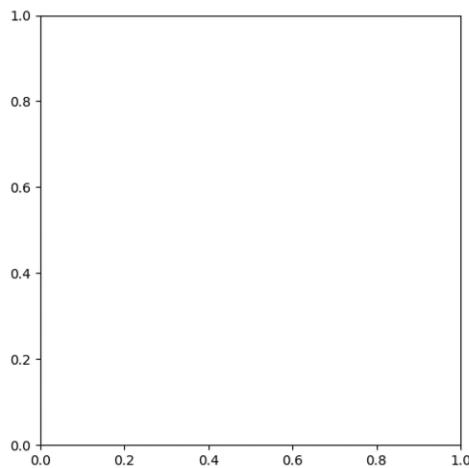
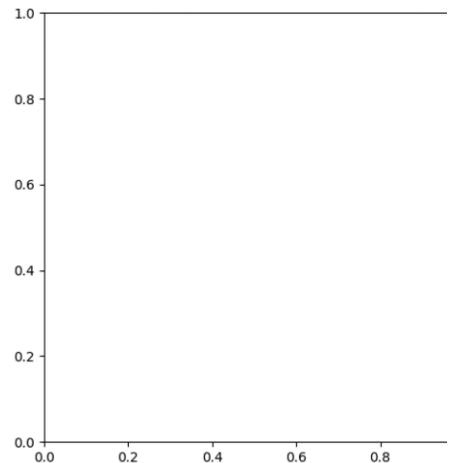
Linus_Tech_Tips

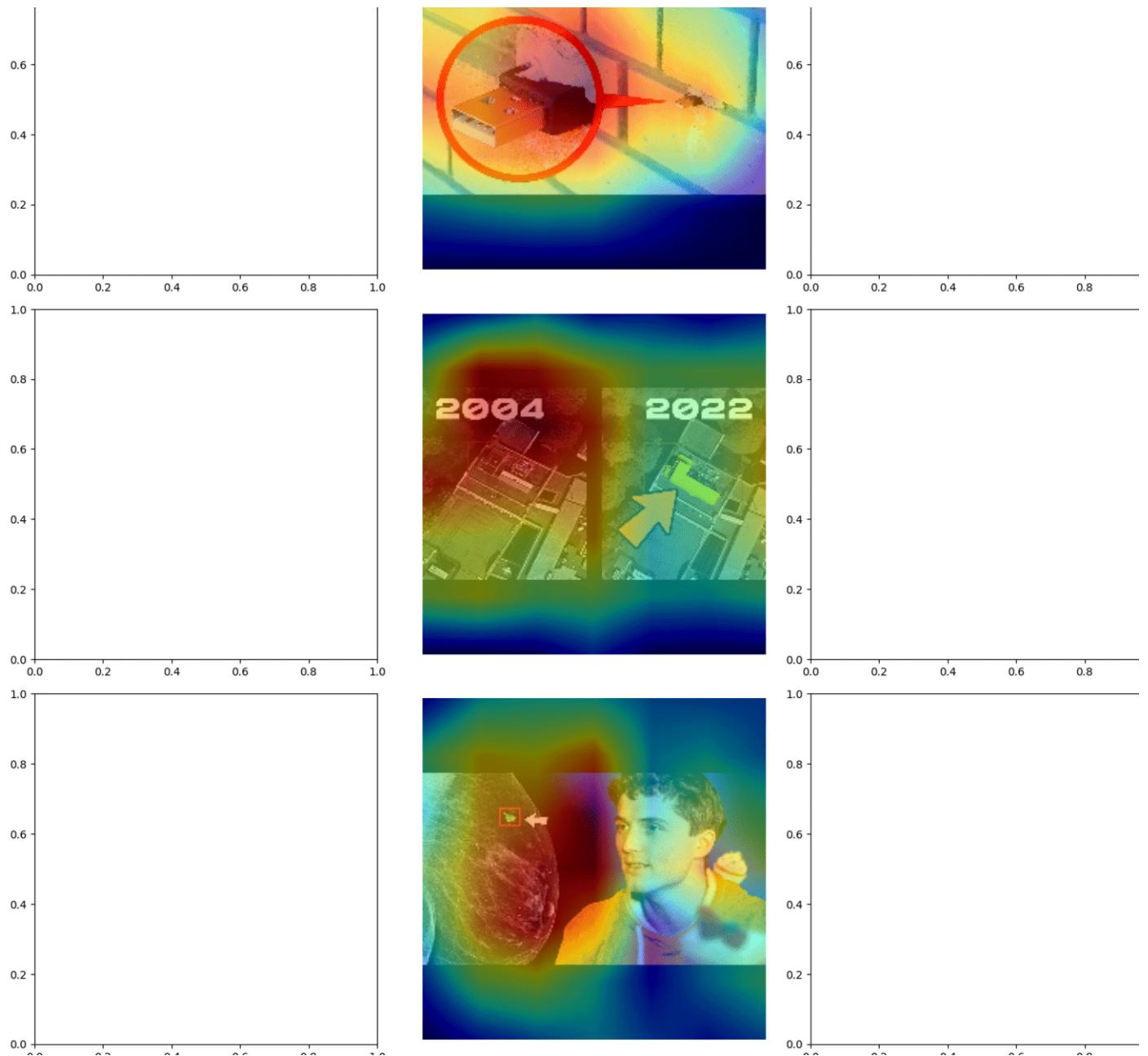


Underscore



Computerphile





Conclusion

En conclusion, notre F1 score global est de **0.934**. Le fait d'avoir pris que des chaines uniquement liée à l'informatique rend l'entraînement plus compliqué.