Détection de Deepfakes

Combinaison de EfficientNet et de Vision Transformers pour la détection de deepfakes.

MODÉLISATION SYSTÈMES VISION

CAMILLE KURTZ

Taeyeon Kim, M2 Vision et Machine Intelligente

20 Juin, 2024

A G E N D A

- Dataset
- Architectures de modèles
- Entraînement
- Resultats
- Références

DATASET

Real sample

1. Flickr-Faces-HQ Dataset (FFHQ)

- FFHQ a été créé par NVIDIA.
- **70 000 images** de visages humains.
  - Les images sont tirées de **Flickr**
  - Automatiquement aligné et recadré en utilisant **dlib**
  - **haute résolution (1024x1024)**

2. CelebA Dataset

- CelebA a été créé par une équipe l'Université de Hong Kong.
- Plus de **200 000** images de visages de célébrités.
  - Une large gamme de poses, d'expressions faciales et de conditions d'éclairage.
  - Une large gamme de célébrités de **diverses industries**.
  - Obtenues à partir de **sources publiques sur Internet.**

DATASET

Fake sample (StyleGAN)

FFHQ Dataset

StyleGAN

StyleGAN FFHQ Dataset

CelebA Dataset

StyleGAN

StyleGAN CelebA Dataset

ARCHITECTURES DE MODÈLES

EfficientNet

1. Concept

- *Scaling-Up methods*
  - *Accroître* la profondeur du réseau *(depth)*.
  - Amplifier le nombre des canaux *(channel width).*
  - *Augmenter la résolution de l'image.*

- ***Compound Scaling-Up***

- ***Mobile Inverted Bottleneck Blocks (MBConvBlock)***
  - **Convolution en Profondeur** *(Depthwise Separable convolution).*
  - **Compression et Excitation** *(Squeeze and Excitation).*

2. Architecture Globale

- *Input Image (224, 224, 3)*

- ***Stem Layer***
  - *Input (224, 224, 3)*
  - *Output (112, 112, 32)*

- ***MBConvBlock[1] 7]***
  - *Input (112, 112, 32)*
  - *FinalOutput (7, 7, 1280)*

- ***Head***
  - *Input (7, 7, 1280)*
  - *Output (7, 7, 1280)*

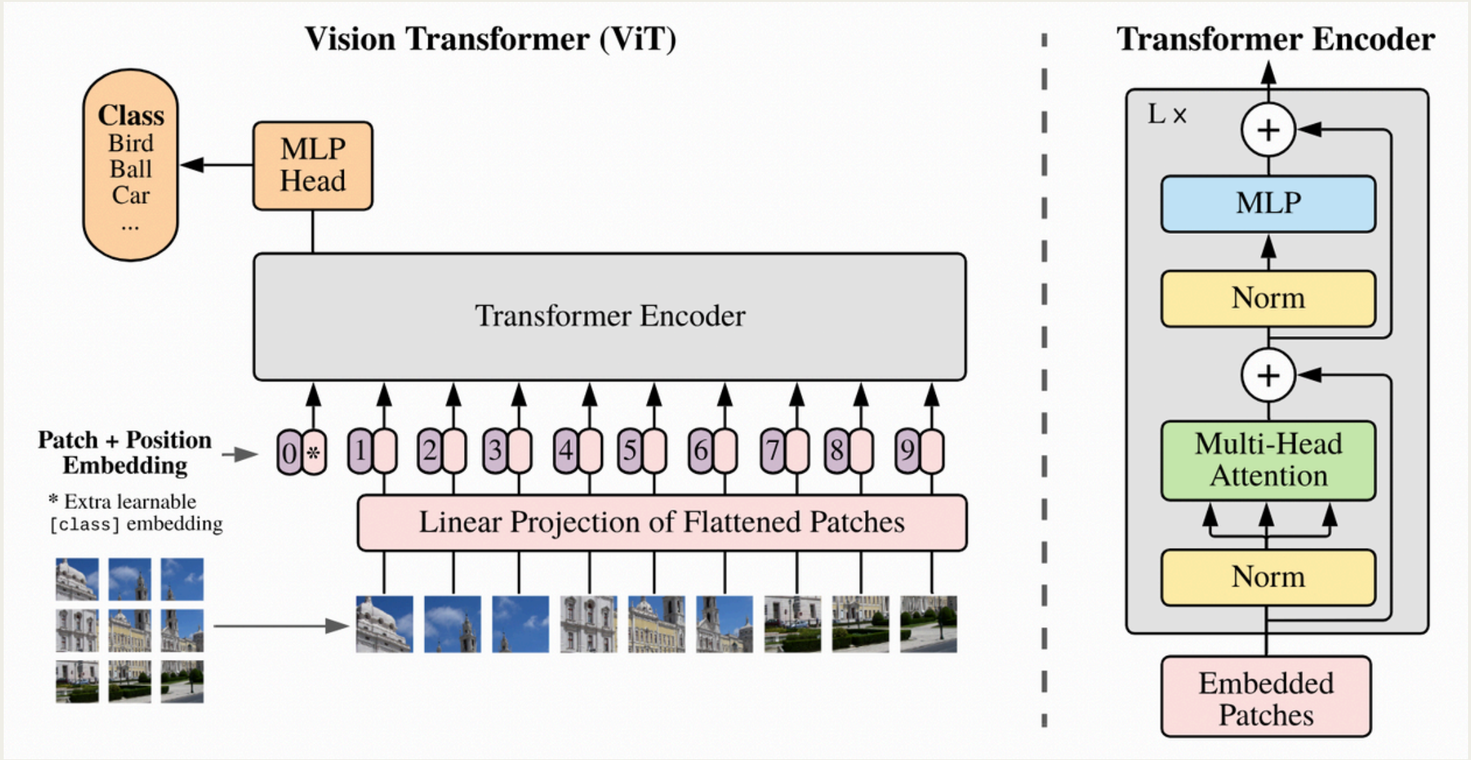
- *Fully-Connected Layer*
  - *Input (7, 7, 1280)*
  - *Output (7x7x1280, NumClass)*

ARCHITECTURES DE MODÈLES

1. Procédure

- Diviser une image en patches de taille fixe
- Effectuer des embeddings linéaires de chacun de patch.
- Ajouter des embeddings de position.
- Rajouter un token de classification apprenable à la séquence de vecteurs.
- Alimenter la séquence de vecteurs résultante dans **un encodeur *Transformer*** standard.
- Extraire le vecteur correspondant au token de classification.
- Passer le vecteur par la tête MLP *(MLP Head), puis effectuer la classification.*

2. Architecture Globale



ARCHITECTURES DE MODÈLES

Efficient ViT

Stem Layer

MBConvBlock1 * 1

MBConvBlock6 * 2

MBConvBlock6 * 2

MBConvBlock6 * 3

MBConvBlock6 * 3

MBConvBlock6 * 4

MBConvBlock6 * 1

Head

FC Layer

Output

Real / Fake

MLP Head

Transformer Encoder

Linear Projection

P1 P2 P3 P4 P5 P6 P7

Features de patches

L'utilisation des features extraits par le EfficientNet au lieu d'employer des patches de l'image d'origine.

ENTRAÎNEMENT

Détailles

1. Réalisation des entraînements

- Développement dans un environnement local.
- Entraînement du modèle sur Google Colab.
  - GPU NVIDIA Tesla T4
- Architectures de modèles testés.
  - EfficientNet-B0 (Baseline)
  - EfficientNet-B0 + Vision Transformer
  - EfficientNet-B1 + Vision Transformer

2. Répartition des images

Type	Dataset	Train	Validation	Inference
Real	FFHQ	10 000	999	9 000
Fake	StyleGAN FFHQ	9 999	1 000	8 997
Fake	StyleGAN CelebA	-	-	9 000
Total		**19 999**	**1 999**	**26 997**

ENTRAÎNEMENT

Liste des méthodes de l’ensemble des transformations utilisées

Modalité	Entraînement	Validation	Description
[ImageCompression]	O		réduit la qualité de l'image en appliquant une compression.
[GaussNoise]	O		ajoute du bruit gaussien à l'image.
[HorizontalFlip]	O		effectue un flip horizontal de l'image.
[IsotropicResize]	O	O	redimensionne l'image tout en maintenant le rapport d'aspect.
[PadIfNeeded]	O	O	ajoute du padding à l'image.
[RandomBrightnessContrast]	O		ajuste la luminosité et le contraste.
[FancyPCA]	O		applique une transformation PCA pour ajuster les couleurs.
[HueSaturationValue]	O		modifie la teinte, la saturation et la valeur.
[ToGray]	O		convertit l'image en niveaux de gris.
[ShiftScaleRotate]	O		applique des transformations de décalage, de mise à l'échelle et de rotation à l'image.

ENTRAÎNEMENT

Liste des hyper-paramètres

Modalités	Hyper-paramètres	Valeurs	Description
Training	Num Epochs	30	-
	Learning Rate	1e−3	-
	Weight Decay	1e−7	-
	Batch Size	32 ou 16 ou 8	Varier en fonction des modèles
	Scheduler	'steplr'	Ajuster le taux d'apprentissage en fonction du nombre d'époques
	Gamma (scheduler)	0.1	Détermine de combien le taux d'apprentissage est diminué chaque fois que la fonction StepLR est invoquée. ***new_learning_rate = old_learning_rate × gamma***
	Step Size (scheduler)	15	Définit à quelle fréquence (epoch) le taux d'apprentissage est réduit
Model	Input Image Size	224	Largeur ou hauteur de l'image d'entrée
	Input Image Channel	3	Nombre de canaux de l'image d'entrée
	Patch Size	7	Largeur ou hauteur du patch
	Num Classes	1	Nombre de classes
	Dimension	1024	Dimension du modèle (Transformer)
	Depth	6	L'encodeur est composé d'une pile de N = 6 couches identiques
	Heads	8	Nombre de têtes dans Multi Self-Attention
	Dimension Head	64	Dimension pour chaque tête dans Multi Self-Attention
	MLP Dimension	2048	Dimension pour le MLP dans le Transformer
	Embedding Dimension	32	Dimension pour chaque embedding de patch
	Dropout	0.15	Taux de dropout pour le MLP dans le Transformer
	Embedding Dropout	0.15	Taux de dropout pour des embeddings de patch

RESULTATS

Courbes de la perte selon modèle

EfficientNet-B0 (Baseline)

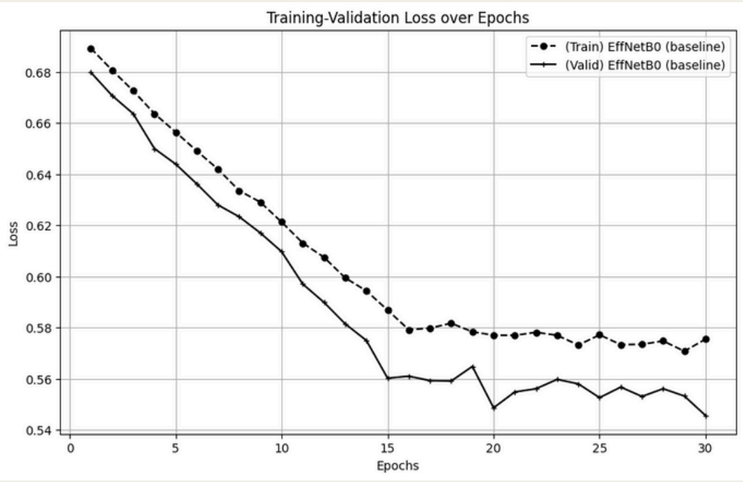
- Paramètres : 4 008 829
- Temps d’entraînement : 238 sec/ep

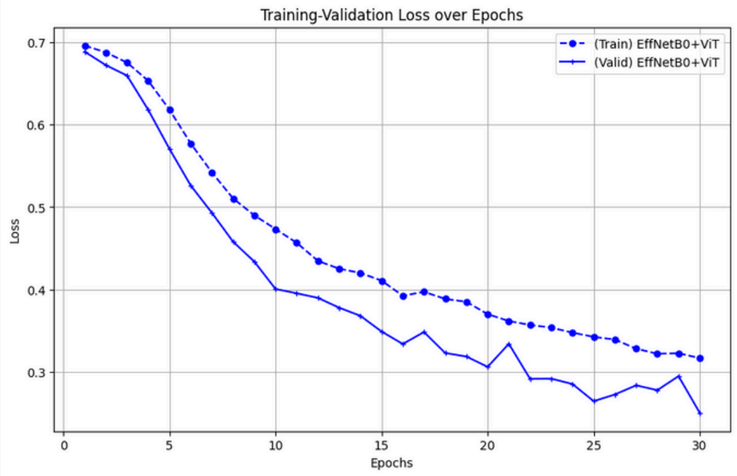
EfficientNet-B0 + ViT

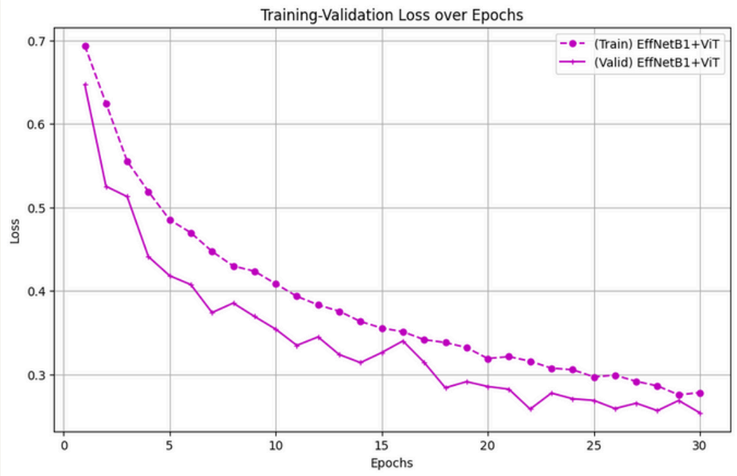
- Paramètres : 109 447 781
- Temps d’entraînement : 255 sec/ep

EfficientNet-B1 + ViT

- Paramètres : 111 953 417
- Temps d’entraînement : 420 sec/ep







Train - Validation Loss

RESULTATS

Validation

EfficientNet-B0 (Baseline)

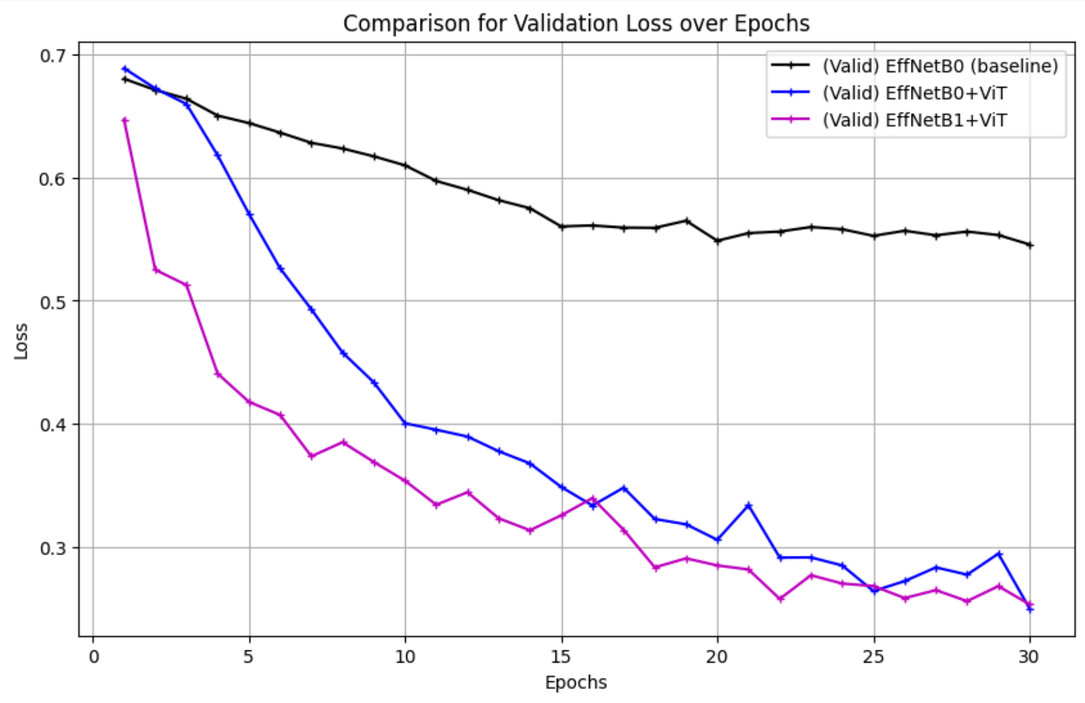
- Paramètres : 4 008 829
- F1-score (30 epoch) : 0.7505

EfficientNet-B0 + ViT

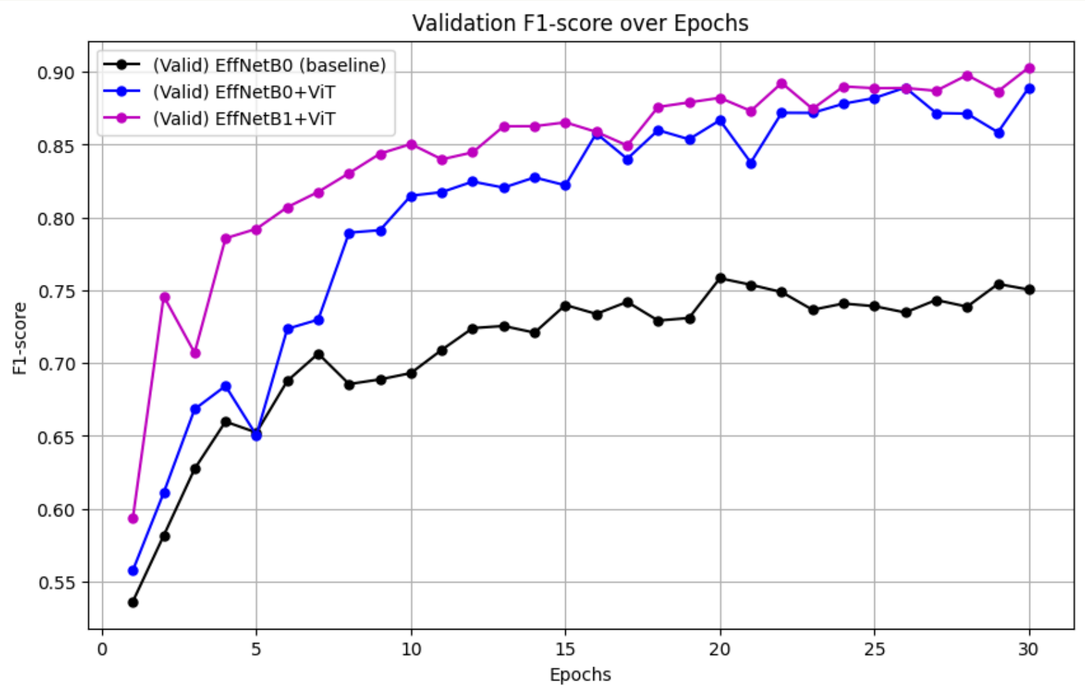
- Paramètres : 109 447 781
- F1-score (30 epoch) : 0.8892

EfficientNet-B1 + ViT

- Paramètres : 111 953 417
- F1-score (30 epoch) : **0.9030**



Comparaison des pertes de validation



Comparaison des F1-score de validation

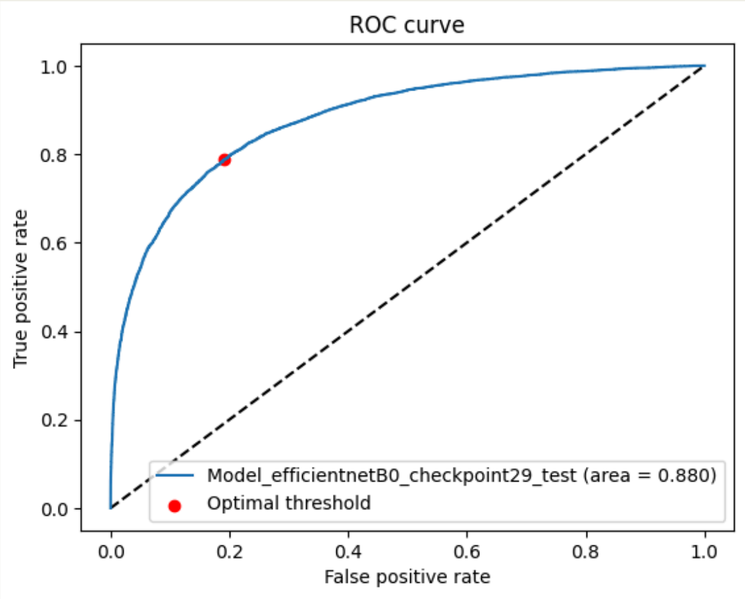
RESULTATS

Inference

Nombre d'images : 17 997
Nombre de classes : {Real : 9 000, Fake : 8 997}

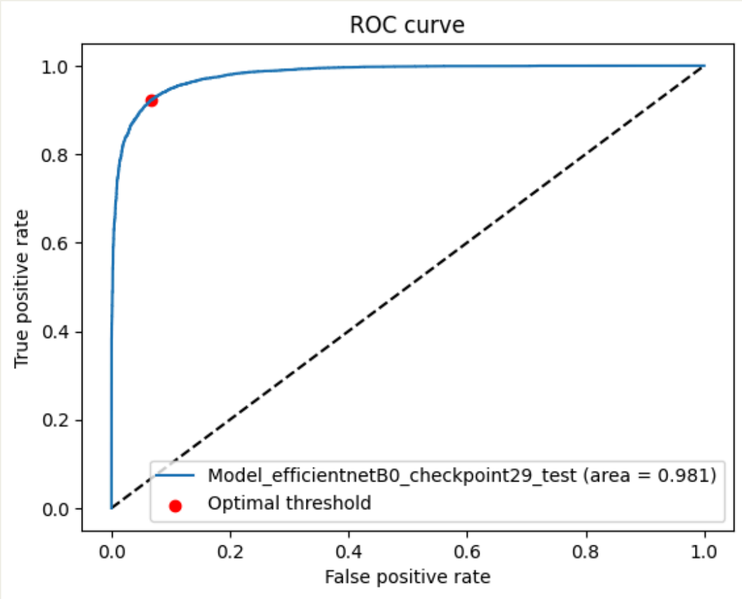
EfficientNet-B0 (Baseline)

- Temps d'inference : 108 sec
- AUC : 0.880



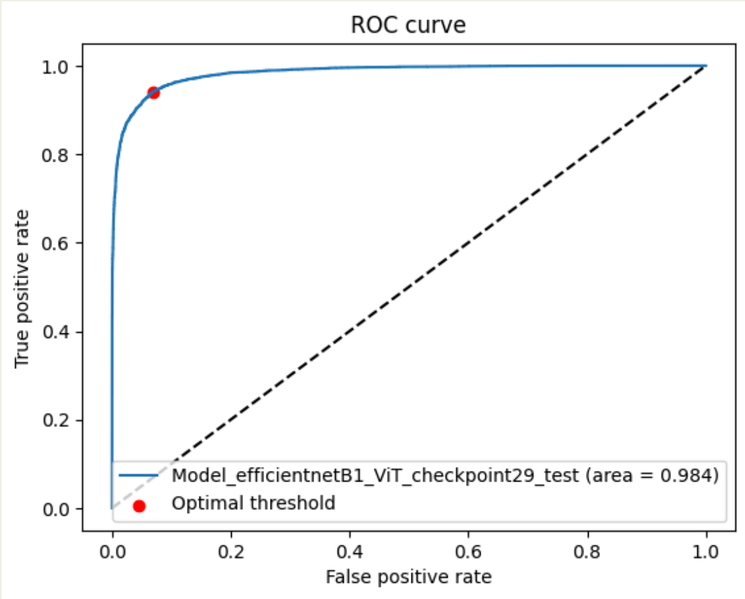
EfficientNet-B0 + ViT

- Temps d'inference : 116 sec
- AUC : 0.981



EfficientNet-B1 + ViT

- Temps d'inference : 131 sec
- **AUC : 0.984**



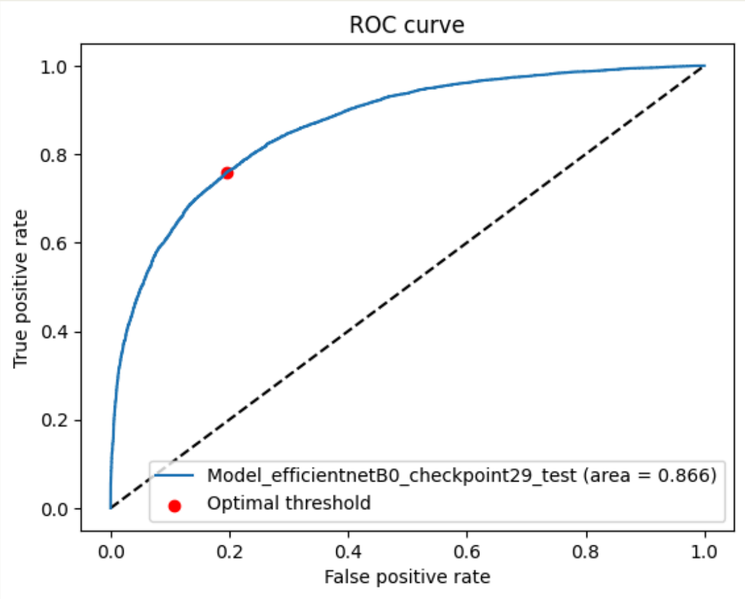
RESULTATS

Inference (+StyleGAN CelebA)

Nombre d'images : 26 997
Nombre de classes : {Real : 9 000, Fake : 17 997}

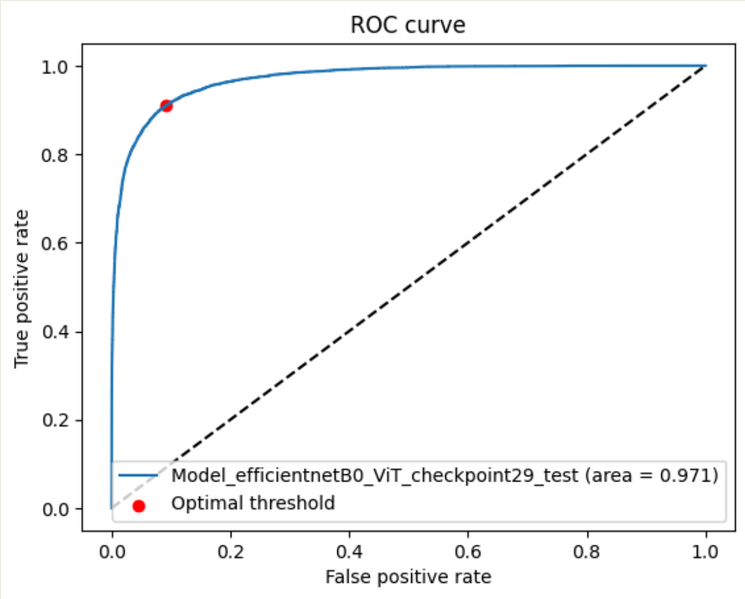
EfficientNet-B0 (Baseline)

- Temps d'inference : 179 sec
- AUC : 0.880 --> 0.866 (**-0.014**)



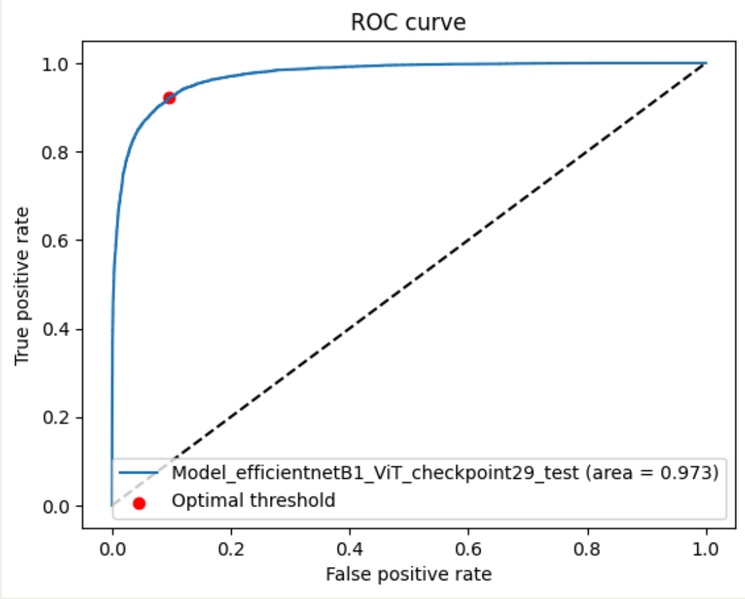
EfficientNet-B0 + ViT

- Temps d'inference : 195 sec
- AUC : 0.981 --> 0.971 (**-0.010**)



EfficientNet-B1 + ViT

- Temps d'inference : 215 sec
- AUC : 0.984 --> **0.973** (**-0.011**)



RÉFÉRENCES

[1] Davide Coccomini, Nicola Messina, Claudio Gennaro, Fabrizio Falchi, Combining EfficientNet and Vision Transformers for Video Deepfake Detection, 2022

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021

[3] Mingxing Tan, Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2020

[4] Tero Karras, Samuli Laine, Timo Aila, A Style-Based Generator Architecture for Generative Adversarial Networks, 2018

[5] Hao Dang, Feng Liu, Joel Stehouwer, Xiaoming Liu, Anil Jain, On the Detection of Digital Face Manipulation, 2020

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, Xiaoou Tang, Deep Learning Face Attributes in the Wild, 2015

[7] https://github.com/NVlabs/ffhq-dataset

[8] https://github.com/NVlabs/stylegan

[9] Dlib C++ Library

Thank you !

MODÉLISATION SYSTÈMES VISION

CAMILLE KURTZ

Taeyeon Kim, M2 Vision et Machine Intelligente

20 Juin, 2024