

Classification de Hiéroglyphes Égyptiens avec CNN et Transformers

Jade Piller-Cammal¹, Estelle Tournassat², Théo Parris³, Alban Sarrazin⁴

Abstract— Ce projet vise à développer et comparer plusieurs modèles de Deep Learning pour la classification de hiéroglyphes égyptiens. Nous explorons différentes architectures (CNN, ResNet18, ViT) ainsi que des stratégies d'entraînement variées : apprentissage from scratch, utilisation de modèles pré-entraînés avec fine-tuning ou feature extraction... Les performances sont évaluées à l'aide d'un ensemble de métriques standards, une discussion et analyse critique des résultats et de la démarche sont fournies.

I. INTRODUCTION

Les hiéroglyphes égyptiens représentent une forme ancienne d'écriture picturale complexe. Leur identification et leur interprétation peuvent fournir des informations précieuses sur les textes anciens, mais le processus est long et nécessite une expertise humaine approfondie. Ce projet se propose de développer un modèle capable d'identifier ces symboles à l'aide de techniques de Deep Learning, posant les bases pour des analyses plus complexes comme la reconstruction de textes effacés, l'étude des structures sémantiques, traduction assistée... Cette approche s'inscrit donc dans une perspective où la reconnaissance des hiéroglyphes devient un outil facilitant des explorations plus approfondies des inscriptions égyptiennes.

II. ÉTAT DE L'ART ET INSPIRATIONS

D'une part, les modèles CNN représentent une approche classique et répandue pour la classification d'images. D'autre part, les Transformers appliqués à la vision (ViT) ont démontré d'excellentes performances notamment par le biais des travaux de Dosovitskiy et al. (2020). Le ViT y est proposé comme une alternative intéressante dans les tâches de vision supervisée avec la promesse d'utiliser moins de ressources computationnelles. Le fine-tuning de modèles pré-entraînés est devenu une méthode courante dans les cas où les données sont rares, notamment pour des domaines spécialisés tels que la reconnaissance d'écritures anciennes.

III. DESCRIPTION DE L'APPROCHE PROPOSÉE

A. Choix du jeu de données :

Nous choisissons un jeu de données d'images de hiéroglyphes égyptiens annotées, tel que le jeu de données "EgyptianHieroglyphDataset Computer Vision Project" [1]. Il contient 170 classes annotées selon la classification de *Gardiner*

[2], certaines contenant parfois moins de 10 images, nous avons conservé uniquement les 40 classes les plus fournies. Les données étaient réparties en 80% pour l'entraînement, 10% pour la validation, et 10% pour le test. Chaque image est redimensionnée à 224 × 224 pixels.

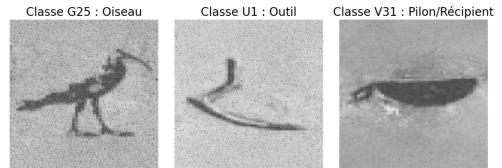


Fig. 1: Exemple des classes étiquetées présentes dans notre dataset

B. Augmentation des données

De part la faible quantité d'image disponibles pour chaque signe différent, il est nécessaire ici de faire une augmentation de nos données. Celle-ci est faite de la manière suivante : On fixe un objectif à 200 images par classes, l'ensemble des classes ayant plus de 200 images ne subissent pas d'augmentation de leur données. Pour les autres, on adapte dynamiquement le nombre d'images augmentées pour chaque image source afin d'atteindre l'objectif de 200 images par classe. Une limite à 15 images augmentées pour chaque image source a été fixée.

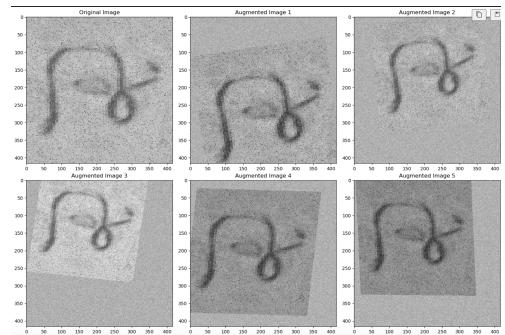


Fig. 2: Exemple d'augmentation pour une image source

Les transformations suivantes ont été appliquées de manière aléatoire à chaque itération:

- Rotation, à plus ou moins 16°
- Translation, de 0 à 20 % sur les axes x et y
- Rétrécissement, jusqu'à une taille de 80% de la taille originale

¹ Université Laval, NI: 537306695

² Université Laval, NI: 537305301

³ Université Laval, NI: 537306690

⁴ Université Laval, NI: 537304112

- Modification de l'éclairage et du contraste

C. Architectures Implémentées

Nous avons testé de nombreux paramètres, régularisations et couches différentes (pour les modèles pretrained). Nous présentons ainsi nos 7 architectures couplées à leurs paramètres les plus satisfaisants.

1. CustomCNN (from scratch): Ce modèle convolutionnel a été implémenté from scratch comme baseline. Il est constitué de trois blocs convolutifs simples, suivis d'un bloc de classification :

- Trois couches Conv2d avec 32, 64 et 128 canaux
- Chaque couche est suivie de BatchNorm2d, ReLU et AvgPool2d
- Un AdaptiveAvgPool2d($1, 1$) condense les features en un seul vecteur
- Le bloc de classification contient Dropout(0.3), Linear(128 → 64), ReLU, puis Linear(64 → 40)

Cette architecture simple avait initialement souffert d'un extrême sur-ajustement. L'ajout de normalisation par batch et de dropout a amélioré la régularisation.

2. ResNet18 (préentraîné, frozen): Nous avons utilisé le ResNet18 préentraîné sur ImageNet, et avons remplacé la dernière couche fc par une Linear(512, 40) :

- Architecture résiduelle en 4 blocs, avec skip connections
- Extraction de caractéristiques via convolutions de 64 à 512 canaux
- Seule la couche finale est fine-tunée ; les poids du backbone sont figés

Cette approche offre ainsi le double avantage d'exploiter des traits visuels universels et de minimiser le surajustement.

3. ResNet50 (préentraîné, frozen): Nous avons utilisé le ResNet50 préentraîné sur ImageNet, et avons remplacé la dernière couche fc par une Linear(512, 40) :

- Architecture résiduelle avec 4 blocs principaux composés de blocs bottleneck
- Connexions de saut (skip connections) facilitant la rétro-propagation du gradient
- Extraction hiérarchique de caractéristiques de bas niveau à haut niveau (64 à 2048 canaux)

Ce modèle profond reste facile à optimiser et permet une généralisation efficace grâce aux connexions résiduelles.

4. Inception-v3 (préentraîné, frozen): Nous avons utilisé le modèle Inception-v3 préentraîné sur ImageNet, et avons remplacé la couche finale par une Linear(2048, 40) :

- Architecture modulaire composée de blocs Inception multi-chemin (convolutions 1×1 , 3×3 , 5×5)
- Factorisation des convolutions pour réduire les coûts de calcul
- Présence d'une tête auxiliaire pour stabiliser l'entraînement en profondeur

Cette structure permet une extraction efficace de caractéristiques à plusieurs échelles tout en limitant le surajustement.

5. Vision Transformer (from scratch): Nous avons implémenté un Vision Transformer léger :

- Projection en patchs 16×16 via une convolution $3 \rightarrow 256$
- Ajout d'un token [CLS] et d'un embedding positionnel appris, initialisé via trunc_normal
- Encodeur Transformer composé de 5 couches avec 8 têtes et $\text{dim_feedforward} = 760$
- Tête de classification : LayerNorm suivi d'une Linear(256, 40)

Malgré une implémentation conforme aux principes classiques, l'apprentissage n'a pas permis d'atteindre convergence. Il s'agit tout de même d'un architecture difficile à entraîner et la pauvre efficacité de ce modèle est probablement due à la faible quantité de données.

6. ViT préentraîné (backbone figé): Nous avons utilisé le modèle ViT-B/16 préentraîné sur ImageNet-21k. La tête de classification a été remplacée par : nn.Linear(768, 40). Les poids du backbone ont été figés (requires_grad = False) pour n'entraîner que la tête. Cette approche, peu coûteuse en calcul, a démontré une bonne robustesse à travers l'apprentissage tout en limitant les risques de surajustement.

7. ViT préentraîné (fine-tune complet): Ce modèle reprend ViT-B/16 avec un entraînement complet : la tête de classification est similaire à la version figée : nn.Linear(768, 40) C'est le modèle qui a obtenu les meilleures performances, avec une convergence rapide, sans sur-apprentissage et une généralisation efficace.

D. Paramètres d'entraînement

Les modèles sont entraînés avec la fonction de perte **CrossEntropyLoss**. Nous avons choisi l'optimiseur AdamW car il offre une meilleure régularisation qu'un Adam, ce qui était essentiel pour éviter le surapprentissage sur notre dataset de taille plutôt réduite, en particulier avec des modèles profonds comme les Vision Transformers (ViT). Résumons ainsi les hyperparamètres utilisés:

TABLE I: Hyperparamètres d'entraînement

Modèle	LR	Ep.	BS	Rég.	Sched.
CustomCNN	5e-4	12	32	DO(0.3), BN	RLRP
ResNet18	1e-3	20	64	FT tête seule	–
ResNet50	1e-3	15	64	–	–
Inception	1e-3	15	64	–	–
ViT scratch	1e-4	20	32	LN, DO(0.1)	–
ViT frozen	1e-3	15	64	MLP tête	–
ViT finetune	1e-5	30	32	DO(0.1)	RLRP

IV. RÉSULTATS EXPÉRIMENTAUX

Nous avons évalué tous les modèles sur l'ensemble de test, en mesurant l'accuracy, le F1-score, la précision et le rappel. Nous incluons également les figures suivantes: matrices de confusion ainsi que les courbes de perte et de précision pour l'entraînement et la validation.

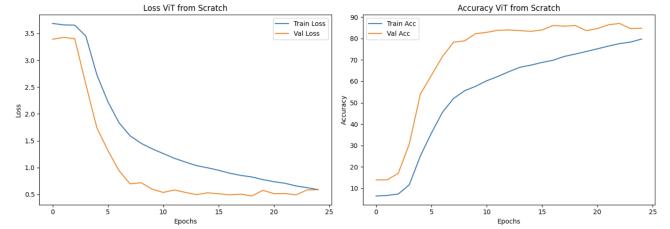
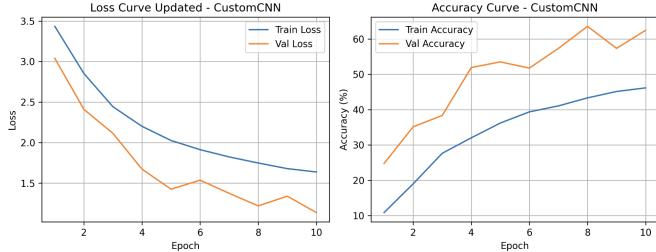
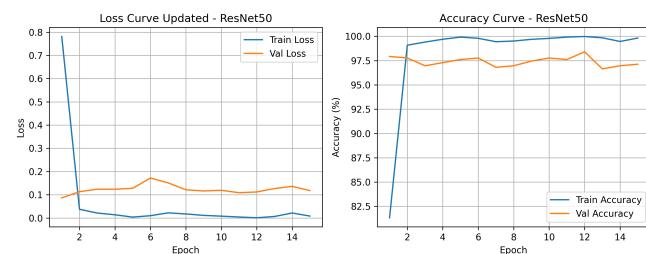
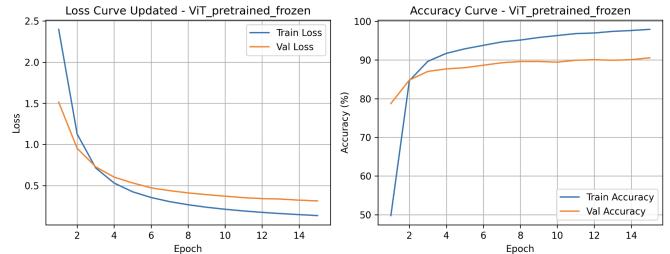
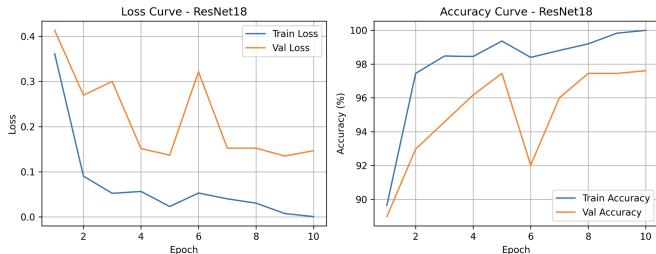
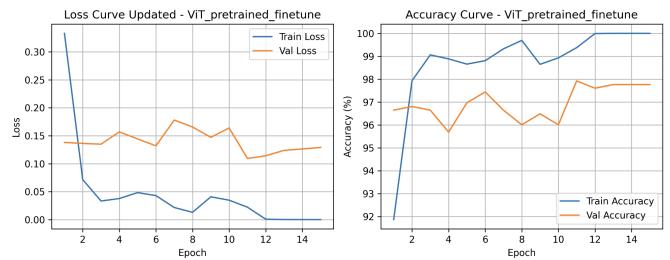


Fig. 3: Courbes de perte et de précision CustomCNN



Note: Nous pouvons remarquer que modèle ViT_pretrained_finetune montre de légers signes visibles de sur-apprentissage (petit écart visible entre les courbes de loss et accuracy entraînement/validation). Toutefois, ce sur-apprentissage reste limité/contrôlé, puisque les performances sur l'ensemble de validation restent élevées et stables. De plus, l'évaluation finale par nos différentes métriques nous confirmera que la généralisation n'est pour autant pas compromise, contrairement à ce que l'on observe généralement dans des cas de sur-apprentissage sévère. De son côté, bien que le modèle ViT_pretrained_frozen semble moins sur-ajuster, il présente une accuracy plus basse.

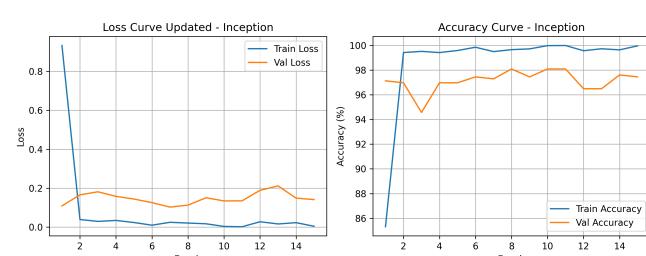


TABLE II: Performances sur le jeu de test (40 classes)

Modèle	Accuracy	F1 macro	Recall	Précision
CustomCNN	62.74%	32.1%	35.3%	35.2%
ResNet18	98.7%	94.8%	94.8%	95.3%
ResNet50	96.2%	89.6%	90.2%	91.3%
Inception	97.8%	91.6%	91.4%	0.93.1%
ViT scratch	87.90%	74.18%	78.40%	75.07%
ViT frozen	91.1%	90.4%	89.9%	91.0%
ViT finetune	94.7%	94.2%	94.0%	94.6%

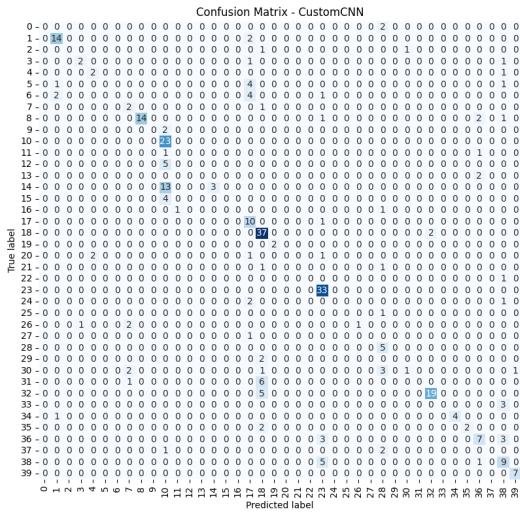


Fig. 10: Matrice de confusion CustomCNN

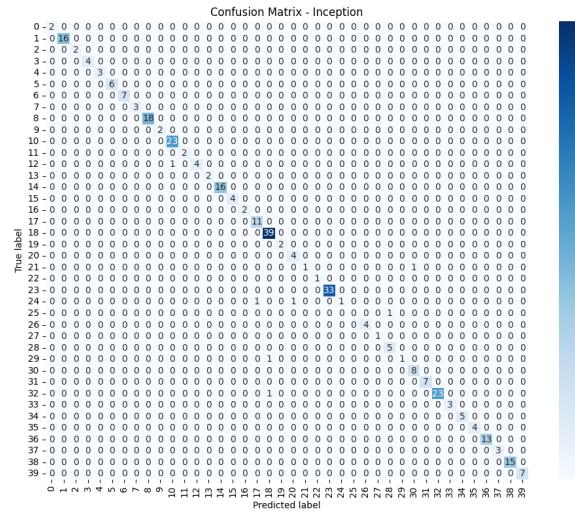


Fig. 13: Matrice de confusion Inception-v3

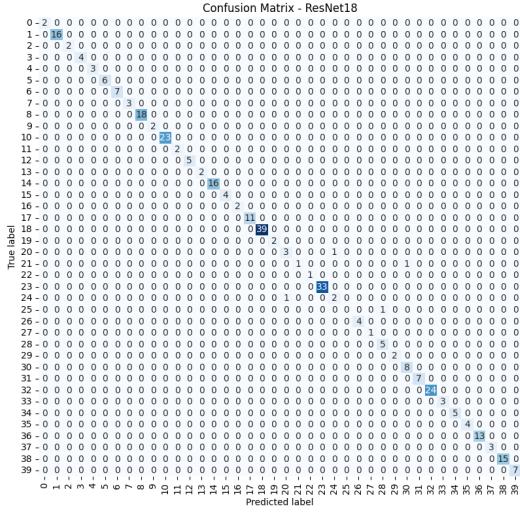


Fig. 11: Matrice de confusion ResNet18

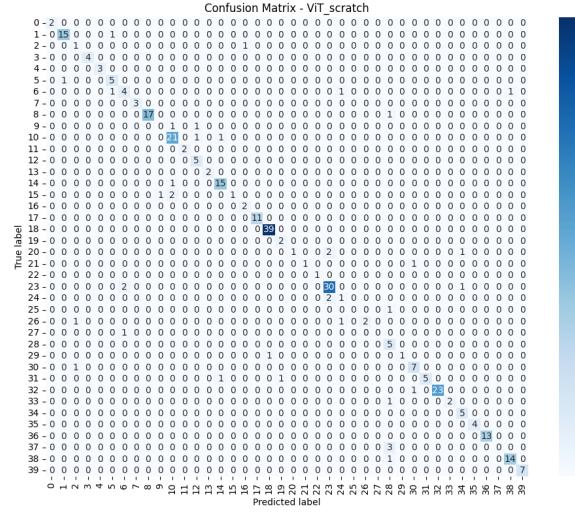


Fig. 14: Matrice de confusion ViT (scratch)

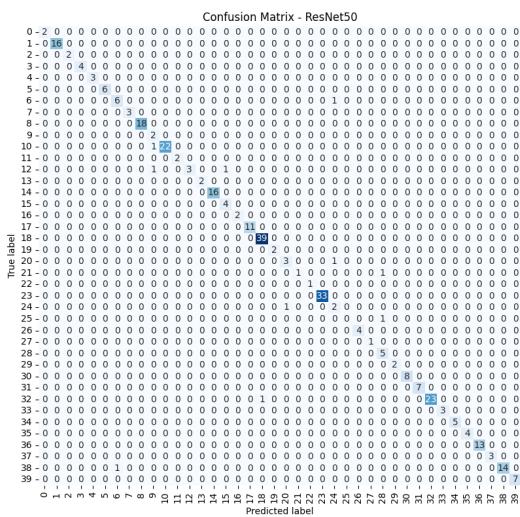


Fig. 12: Matrice de confusion ResNet50

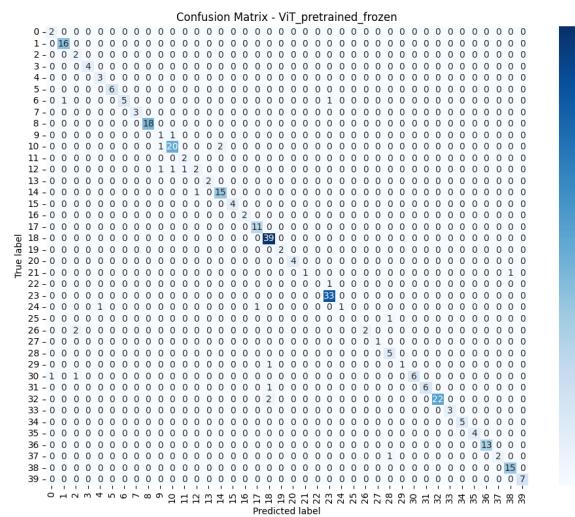


Fig. 15: Matrice de confusion ViT pretrained (frozen)

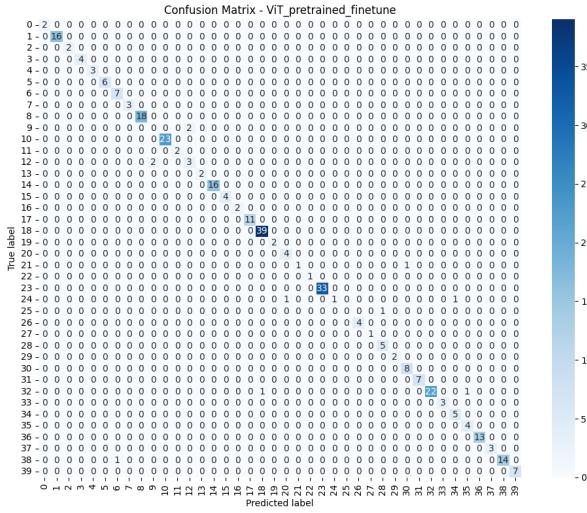


Fig. 16: Matrice de confusion ViT pretrained (finetuned)

V. ÉTUDE D'ABLATION

Nous réalisons finalement un étude d'ablation sur le vision transformeur custom. Pour cette étude, nous avons entraîné le modèle sans augmentation des données, avec une couche de moins. Nous avons aussi testé l'ajout de test-time augmentation (TTA), consistant à augmenter les données de test, puis de faire une moyenne des probabilités du modèle pour les images originales + augmentées, afin d'avoir une prédiction finale. Nous avons cependant obtenus des résultats catastrophiques concernant le TTA (1.5% d'accuracy) relevant probablement de problèmes d'implémentations que nous n'avons pas su régler, nous le laissons donc de côté.

TABLE III: Performances des modèles de l'étude d'ablation)

Modèle	Accuracy	F1 macro	Recall	Précision
ViT complet	87.90%	74.18%	78.40%	75.07%
ViT - augmentation	85.35%	70.63%	71.40%	71.76%
ViT - augmentation 4l	85.99%	70.85%	73.79%	70.46%
ViT, 4l	85.99%	70.61%	70.87%	72.41%

Par l'étude d'ablation, on a pu voir que la configuration actuelle du vision transformeur custom est la meilleure, bien qu'en accuracy, les modèles se valent, il y a un changement drastique en score F1, en recall et en précision pour les modèles non-complets. Ce constat s'explique par la difficulté des modèles à reconnaître les classes plus complexes ou moins représentées sans augmentation de données. On aperçoit aussi un sur-apprentissage caractérisé pour les modèles sans augmentation des données.

VI. DISCUSSION

Les résultats expérimentaux confirment l'intérêt des modèles préentraînés pour la classification de hiéroglyphes dans un contexte de données limitées. Les modèles ResNet18, ResNet50 et Inception-v3, tous préentraînés sur ImageNet et utilisés en extraction de features, ont atteint une accuracy identique de 98.7 % avec d'excellents scores de F1, précision

et rappel ($\sim 95\%$). Cela montre que des caractéristiques visuelles génériques peuvent être efficacement transférées à notre tâche spécifique.

Le modèle CustomCNN, entraîné from scratch, souffre en revanche de performances nettement inférieures (F1 macro à 32.1%), malgré l'ajout de régularisations comme le dropout ou la normalisation par batch. Ce résultat met en évidence la difficulté à entraîner un réseau convolutionnel profond sur un jeu de données restreint et déséquilibré.

Les modèles Vision Transformer montrent des résultats plus contrastés. Le modèle entraîné from scratch ne parvient pas à rivaliser avec les versions pré-entraînées, atteignant une accuracy de 87.9%, ce qui reste notable mais inférieur. L'utilisation du ViT pré-entraîné, avec fine-tuning complet, a permis d'obtenir un très bon compromis entre performance (94.7% d'accuracy) et généralisation. Bien que de légers signes de sur-apprentissage soient visibles sur les courbes de perte et d'accuracy, les performances sur l'ensemble de test restent stables, montrant une bonne robustesse globale. Le modèle ViT frozen, quant à lui, présente une accuracy de 91.1%, mais des performances légèrement inférieures sur les autres métriques, suggérant une moindre capacité d'adaptation au domaine spécifique.

L'étude d'ablation vient appuyer ces conclusions. La suppression de l'augmentation de données ou la réduction du nombre de couches a systématiquement entraîné une baisse des performances, en particulier du F1-score. Cela confirme l'importance des données augmentées dans un contexte multi-classes où certaines classes sont faiblement représentées. L'ajout de test-time augmentation n'a pas été concluant dans notre cas (1.5 % d'accuracy), probablement à cause de problèmes d'implémentation non résolus.

En somme, le modèle ViT pré-entraîné avec fine-tuning complet constitue notre approche la plus performante en tenant compte à la fois de la précision, de la capacité de généralisation et de la stabilité à l'entraînement. Les architectures CNN classiques demeurent néanmoins compétitives et efficaces pour ce type de tâche, notamment lorsqu'elles sont pré-entraînées.

Github du projet: <https://github.com/EstelleTournassat/HieroglClassification>

REFERENCES

- [1] Jeu de données contenant 3 584 images de hiéroglyphes égyptiens, organisées en 170 classes, <https://universe.roboflow.com/sameh-zaghliou/egyptianhieroglyphdataset>
- [2] Liste des signes et annotations utilisées pour la classification, <https://www.egyptianhieroglyphs.net/gardiners-sign-list/>

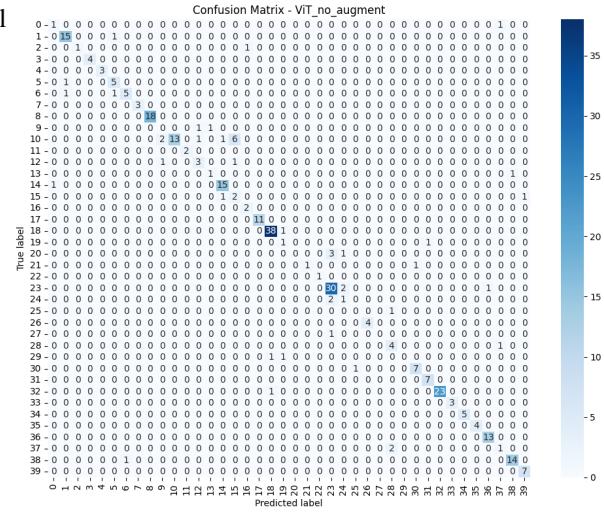


Fig. 20: Matrice de confusion ViT sans augmentation

APPENDIX

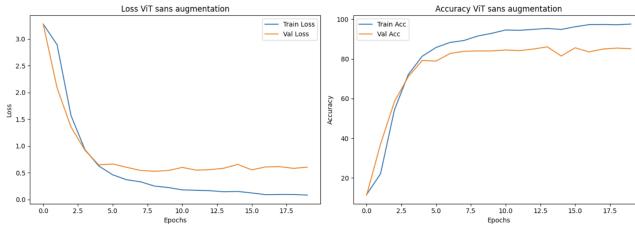


Fig. 17: Courbes de perte et précision ViT sans augmentation

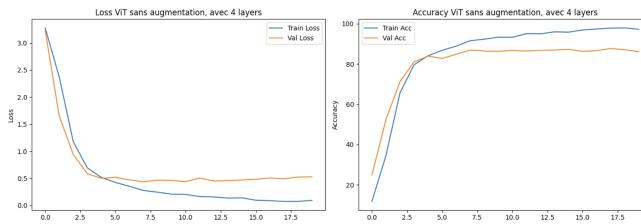


Fig. 18: Courbes de perte et précision ViT sans augmentation, avec 4 layers

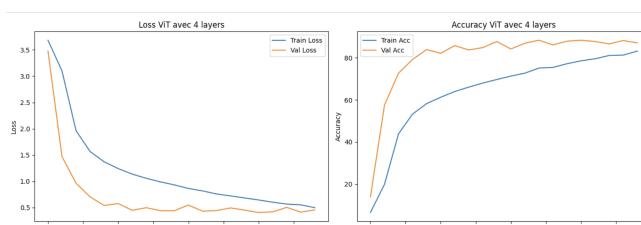


Fig. 19: Courbes de perte et précision ViT avec 4 layers

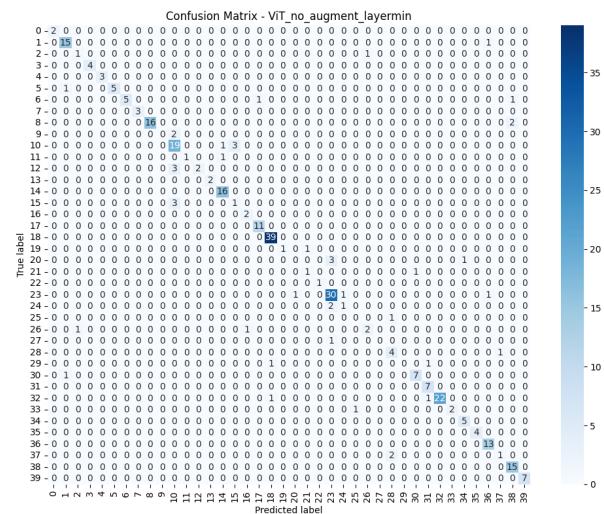


Fig. 21: Matrice de confusion ViT sans augmentation, avec 4 layers

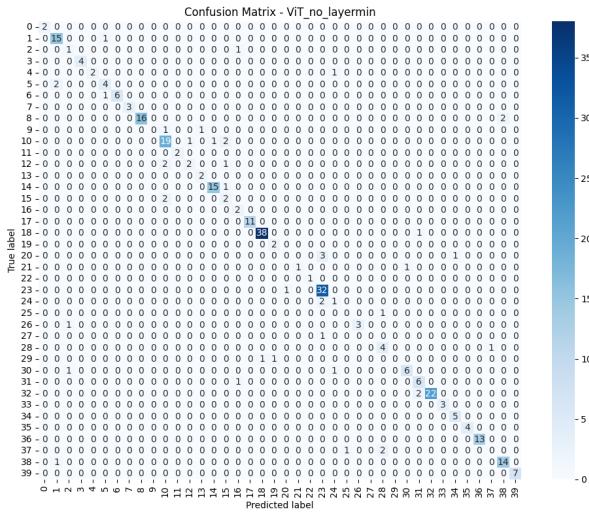


Fig. 22: Matrice de confusion ViT avec 4 layers