Un traducteur automatique basé sur l'IA pour la langue hiéroglyphique ancienne - Des images numérisées au texte anglais

Asmaa Sobhy, Mahmoud Helmy, Michael Khalil, Sarah Elmasry,
Youtham Boules et Nermin Negied (nnegied@debi.gov.eg).

Le Caire - 16 mars 2023

RÉSUMÉ

Les avancées récentes dans les domaines de l'apprentissage automatique et de l'apprentissage profond ont entraîné une transformation considérable dans d'autres domaines qui ne sont pas liés à l'informatique. Dans cette étude, un nouveau cadre est proposé pour résoudre le problème de la traduction des anciennes écritures hiéroglyphiques égyptiennes en anglais en déployant à la fois des techniques de traitement d'images et de traitement du langage naturel combinées à des approches d'IA. Notre objectif principal est de concevoir une application qui révolutionne complètement l'expérience d'un touriste lors de la visite de sites historiques égyptiens. Cette étude utilise différentes techniques d'IA pour convertir automatiquement les photos numérisées du langage hiéroglyphique en anglais compréhensible et lisible, à travers deux sous-tâches principales : la détection et la reconnaissance automatiques des images de glyphes numérisées et leur traduction en anglais. Différentes sources de données de cette langue à faibles ressources ont été explorées et augmentées pour former et tester nos modèles. Les résultats de différents modèles et algorithmes sont évalués et analysés pour évaluer notre étude. Des résultats de pointe sont obtenus par rapport à la littérature spécialisée[1], tant en matière de reconnaissance automatique des glyphes que de traduction des glyphes vers l'anglais.

I. INTRODUCTION

Les anciens Égyptiens utilisaient notamment le langage hiéroglyphique pour enregistrer leurs découvertes en médecine[1][2], en ingénierie, en sciences, leurs réalisations et leurs opinions religieuses, en plus des faits de leur vie quotidienne. Ainsi, il est fondamentalement important de comprendre et de stocker numériquement ces écritures pour quiconque souhaite comprendre l'histoire égyptienne et en apprendre davantage sur cette grande civilisation. Dans ce travail de recherche, notre objectif est de déchiffrer cette langue remarquablement intéressante pour permettre aux touristes de comprendre plus facilement les écritures des anciens Égyptiens grâce à la détection et à la reconnaissance automatiques des hiéroglyphes puis à leur traduction en anglais. De cette façon, les gens pourront lire ce que les anciens Égyptiens ont écrit à l'époque pharaonique[3] sans avoir recours aux services des égyptologues qui sont rares et chers. Dans l'Égypte ancienne, les hiéroglyphes étaient le système d'écriture officiel. Près de mille symboles étaient utilisés dans la langue. Jusqu'à ce que Jean-François Champollion déchiffre la pierre de Rosette, la connaissance des hiéroglyphes était perdue. Les principales caractéristiques de cette langue intéressante sont indiquées ci-après.

---

[1] Le mot « Literature » utilisé régulièrement dans le texte original désigne bien évidemment les publications scientifiques sur le sujet (toutes les notes sont du traducteur).
[2] Les égyptiens n'utilisent pas la convention (Nom année-de-publication) mais désignent leurs références bibliographiques (en fin de texte) par un numéro entre crochets.
[3] « Paranoiac era » dans le texte original. Faute de frappe ou erreur de traduction automatique depuis le texte source écrit peut-être en arabe.

A. CARACTÉRISTIQUES DE LA LANGUE

1) Étant donné que chaque symbole a son propre son, il n'y a aucun lien entre la connaissance d'un hiéroglyphe et la compréhension de sa lecture. Le glyphe de la jambe inférieure, par exemple, ne signifie rien à propos de la jambe, mais il ressemble à la lettre « b » de l'alphabet anglais. Ces représentations sonores sont la « translittération » de la langue égyptienne ancienne car elles reflètent la façon dont les gens la parlaient. La translittération consiste à faire correspondre la phonétique d'une phrase à l'alphabet souhaité[4] (par exemple, l'anglais) en fonction de la similarité phonétique sans tenir compte du sens de la phrase.

2) Les hiéroglyphes peuvent apparaître dans différentes directions : horizontalement (lisibles de gauche à droite ou de droite à gauche) et verticalement de haut en bas.

3) La présence de déterminatifs joue un rôle crucial dans la langue.

Il existe des symboles qui n'ont pas de son mais qui donnent un sens aux mots, comme le type de mot ou s'il s'agit d'un mot singulier ou pluriel. Ces caractéristiques de la langue engendrent une ambiguïté dans la lecture et la compréhension de la langue. En d'autres termes, il faut beaucoup d'études dans les connaissances linguistiques pour pouvoir connaître les défis linguistiques avant d'automatiser la tâche. Nous avons beaucoup appris sur la langue pour pouvoir la comprendre avant d'essayer de donner à la machine la capacité de la comprendre, et nous avons également étudié les moyens disponibles pour simplifier la compréhension de cette étonnante langue ambigüe. Nous avons trouvé la liste des codes Gardiner qui a été introduite et compilée pour la première fois par Sir Alan Gardiner pour simplifier le processus de compréhension de la langue. La liste des codes Gardiner contient tous les glyphes égyptiens courants et leurs sous-catégories de glyphes égyptiens[2]. Les signes sont organisés en vingt-six catégories principales suivies de trois sections qui répertorient les hiéroglyphes par leur forme.

La collecte d'un ensemble de données appropriées pour cette étude n'a pas été une tâche facile car il n'est pas courant de trouver un ensemble de données complet pour la langue hiéroglyphique mais heureusement, nous avons réussi à rassembler les scripts écrits sur la pyramide d'Ounas[3]. Cette pyramide est très importante car elle contient le premier exemple de textes funéraires connus sous le nom de *Textes des Pyramides*. L'ensemble des données collectées contient cent soixante douze symboles différents.

Le reste de cet article est organisé comme suit : la section II liste le travail présenté dans les publications liées à notre étude. La section III décrit les ensembles de données utilisés dans cette étude et les étapes de traitement des données effectués pour les préparer au traitement et aux tests. La section IV explique en détail les méthodologies proposées pour la classification et la traduction des glyphes. La section V discute des résultats obtenus par les traitements proposés dans cette étude, et enfin le document

---

[4] Peu clair dans l'original. La *translittération* (décrite plus loin dans le texte pour évoquer la conversion de hiéroglyphes en codes Gardiner) est le processus de représentation phonétique des hiéroglyphes à l'aide de l'alphabet latin et de signes spécifiques, et certainement pas leur conversion en codes Gardiner. Exemple : https://www.shpylgoreih.fr/images/exemple_de_translitteration.png. La conversion de hiéroglyphes en codes Gardiner s'appelle *codification* (on dit aussi *classification*).

se conclut par la section VI qui comprend également des suggestions pour les travaux futurs.

II. REVUE DE LA LITTÉRATURE SPÉCIALISÉE

En examinant les publications, il a été facile de trouver des travaux antérieurs qui ont étudié les descripteurs visuels et la manière de segmenter et de reconnaître les caractères, tandis que d'autres chercheurs ont essayé de traduire la langue et de comprendre les textes hiéroglyphiques. Cependant, aucune recherche antérieure n'a été réalisée pour aborder le processus de bout en bout, en commençant par la reconnaissance des descripteurs visuels jusqu'à sa traduction en anglais[5].

Par exemple, Franken *et al*.[3] ont mis en œuvre une solution qui reconnaît automatiquement le texte hiéroglyphique à partir d'une image. Pour la localisation des glyphes, les auteurs ont utilisé un algorithme de détection de texte basé sur la *saillance*[4]. Ils ont ensuite utilisé une approche de correspondance d'apparence avec une version avancée de la méthode de l'histogramme des gradients orientés (HOG) : HOOSC[5]. Enfin, ils ont effectué une correspondance par paires avec un patch étiqueté. Leur approche de détection n'a détecté que 83 % des glyphes et la correspondance des glyphes n'a été réussie qu'à 74 %.

Barucci *et al*.[6] ont utilisé ResNet-50 pour développer une méthode de classification des glyphes. Ils n'ont pas trouvé d'ensemble de données suffisant pour entraîner leur approche, ils ont donc utilisé un ensemble de données différent, puis ils ont utilisé l'apprentissage par transfert. Ils ont également implémenté une nouvelle architecture appelée *Glyphnet* et l'ont entraînée sur un petit ensemble de données hiéroglyphiques conçu pour la tâche spécifique de classification des hiéroglyphes et ont formé leur réseau sur celle-ci. Le résultat a montré que *Glyphnet* a atteint un taux de précision de 96 %, ce qui est le taux de précision le plus élevé trouvé dans la littérature spécialisée. Mais les données utilisées dans leur travail ne sont malheureusement pas disponibles[6] pour que d'autres chercheurs valident les résultats.

Elnabawy *et al*.[7] ont utilisé des techniques simples de traitement d'image pour détecter et segmenter les glyphes telles le filtre de Canny[7] et la segmentation de la région d'intérêt (ROI), puis ils ont essayé plusieurs algorithmes de correspondance d'images pour faire correspondre un glyphe segmenté avec les glyphes de l'ensemble des données. Les auteurs ont ensuite confirmé que l'histogramme des gradients orientés (HOG) obtenait les meilleurs résultats de correspondance. Les auteurs, à la fin de leur article, ont abordé la traduction des glyphes en anglais mais ils ont avoué qu'ils n'ont pas obtenu de résultats notables en raison du fait que leur concentration principale était sur la partie traitement d'image et qu'ils n'ont pas pris en compte le langage linguistique, laissant cette tâche aux chercheurs intéressés par le traitement du langage naturel. Les auteurs ont démontré leurs résultats de reconnaissance de glyphes pour chaque code Gardiner individuellement, le plus élevé atteignant 87% et le plus bas atteignant 33%. En utilisant la moyenne de leurs résultats de

---

[5] Si, en 2022 a été lancé le **Projet Loracrafft** (https://www.shpylgoreih.fr/loracrafft.htm) mais il n'a fait à l'époque l'objet d'aucune publication car aucun résultat n'avait encore été obtenu.

[6] Elles le sont maintenant : DataSet : http://iamai.nl/downloads/GlyphDataset.zip, GitHub : https://github.com/GAIA-IFAC-CNR/Glyphnet

[7] https://fr.wikipedia.org/wiki/Filtre_de_Canny

reconnaissance de glyphes, nous pouvons affirmer qu'ils ont atteint une précision de classification globale de 66,7 %.

En ce qui concerne la tâche de traduction, la traduction automatique neuronale (NMT) a été utilisée pour traduire la langue sumérienne[8]. La langue sumérienne est une autre langue ancienne. Le manque de ressources linguistiques et de compréhension complète de celle-ci entraîne l'obtention de phrases anglaises altérées issues du processus de traduction. Ce problème rend les approches semi-supervisées difficiles, car les phrases issues des grands corpus de phrases anglaises disponibles ne seront pas similaires aux phrases anglaises qui existent dans les corpus parallèles qui ont été traduits mot à mot. Cette incohérence pourrait perturber le modèle de traduction automatique. Certains chercheurs[8] ont fait référence aux langues qui souffrent d'incohérence côté cible comme étant des langues à ressources extrêmement faibles[8].

D'autres chercheurs ont essayé de traduire la langue égyptienne ancienne. Ils ont utilisé à la fois la translittération et les signes Gardiner. Il convient de mentionner qu'il existe un référentiel Open Source de Lauren Fay Rosenberg (utilisateur Fayrose) sur GITHUB dans le projet *Egyptian Translation*[9], qui contient un corpus de hiéroglyphes égyptiens translittérés.

III. ENSEMBLES DE DONNÉES ET PRÉPARATIONS DE DONNÉES

Cette section présente les différents ensembles de données utilisés dans cette étude pour entraîner et tester les différents modules de notre système tels que la détection d'objets, la classification d'images, la segmentation de mots et la traduction automatique.

A. ENSEMBLE DE DONNÉES GLYPHS

L'ensemble de données utilisé dans cette étude est appelé ensemble de données *Morris Franken* et il s'agit de l'un des rares ensembles de données accessibles au public. Il couvre 4 210 glyphes, représentant des hiéroglyphes égyptiens trouvés sur les murs de la pyramide d'Ounas, qui est caractérisée par ses colonnes verticales d'écritures hiéroglyphiques. La figure 1 (voir page suivante) montre un échantillon de la pyramide d'Ounas. Les résolutions des images de cet ensemble de données sont d'environ 1 150 × 1 600 pixels de largeur et de hauteur, respectivement. Les images sont annotées manuellement en fournissant le cadre de délimitation pour chaque glyphe. Les glyphes individuels de cet ensemble de données sont étiquetés selon leurs codes Gardiner, et chacun a une image de dimensions de 75 × 50 pixels. La figure 2 montre quelques exemples d'images de l'ensemble de données. Les images couvrent 172 codes Gardiner différents. La distribution des images entre les 172 étiquettes est déséquilibrée (voir figure 4) où la plupart des étiquettes ont moins de 10 images, tandis que certaines étiquettes ont un plus grand nombre d'images. Une augmentation de l'ensemble de données a été réalisée dans cette étude de deux manières différentes pour les tâches de détection et de classification séparément. Dans les sous-sections suivantes, une démonstration de la façon dont l'ensemble de données a été préparé pour chacune des deux tâches respectivement est présentée.

---

[8] « low-resources languages ».
[9] https://github.com/fayrose/EgyptianTranslation

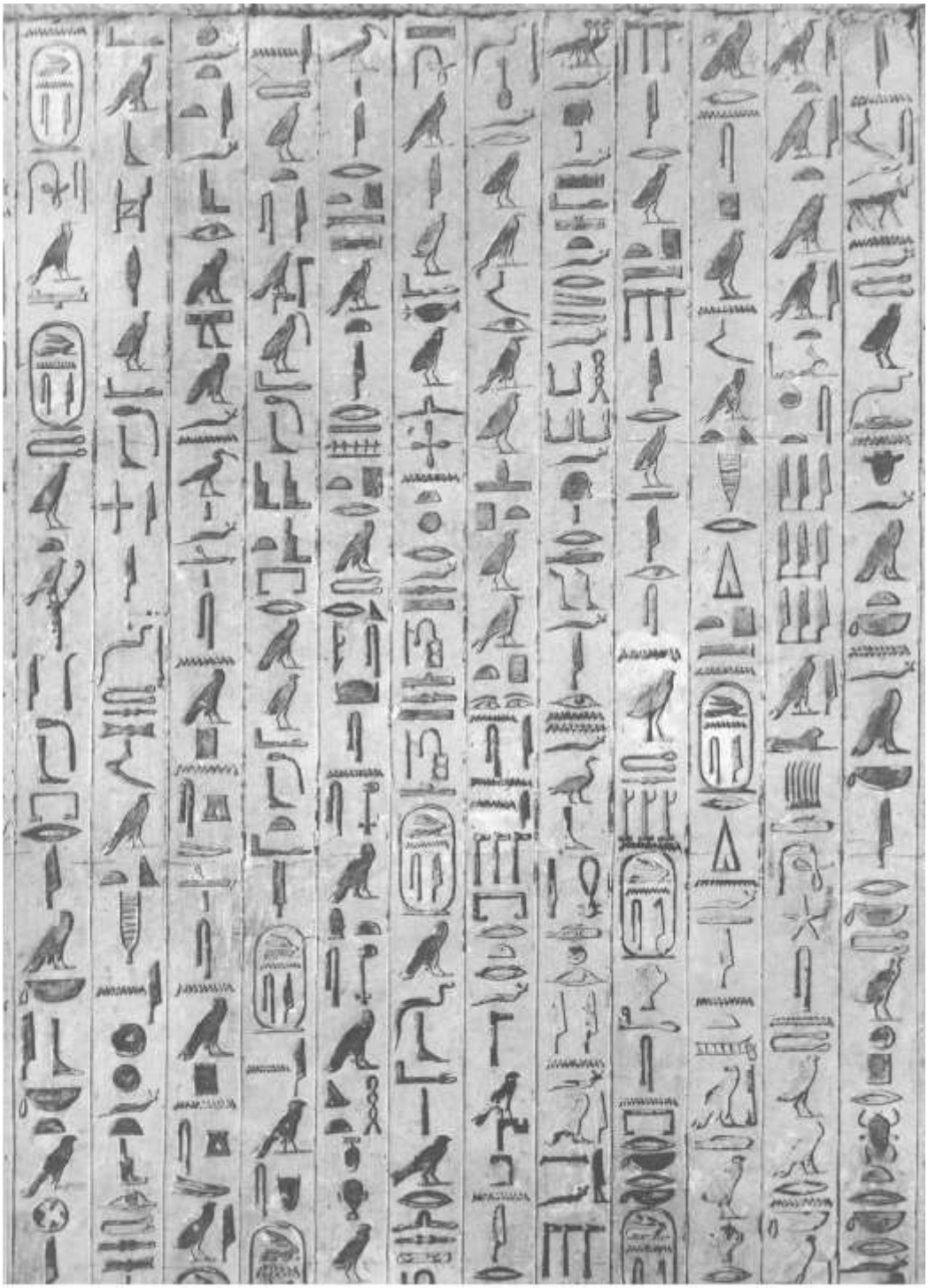

FIGURE 1. **Example piece of Unas pyramid wall.**

Image					
Gardiner Code	D1	f13	M4	M29	E1

FIGURE 2. **Sample of the dataset.**

1) DÉTECTION DE GLYPHES

Un recadrage aléatoire a été utilisé pour créer un ensemble de données augmentées à l'aide d'*Albumentations*[10][9]. L'ensemble de données résultant contenait 2 000 images.

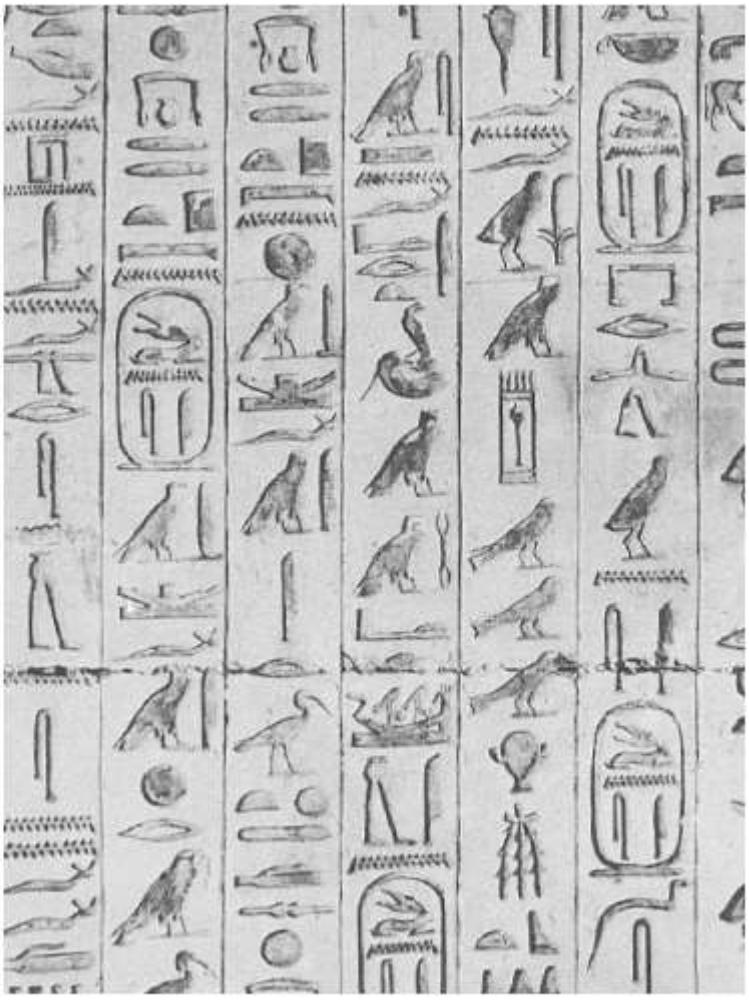

FIGURE 3. Example of a generated crop.

Nous avons défini les rapports hauteur/largeur des patches recadrés comme les rapports hauteur/largeur les plus courants pour les appareils photo des smartphones dans les orientations portrait et paysage, qui sont respectivement 4:3 et 16:9. La distribution des rapports d'aspect est égale et ils couvrent chacun trois résolutions différentes. Les résolutions du rapport d'aspect 4:3 sont (512, 384), (640, 480), (800, 600), tandis que le rapport d'aspect 16:9 a les résolutions (426, 240), (640, 360), (854, 480).

La résolution a été choisie pour conserver un nombre utile de glyphes dans le recadrage.

La figure 3 montre un exemple des recadrages générés en plus des cadres de délimitation qu'il contient.

[10] Librairie Open Source permettant l'amélioration d'images. https://github.com/albumentations-team/albumentations

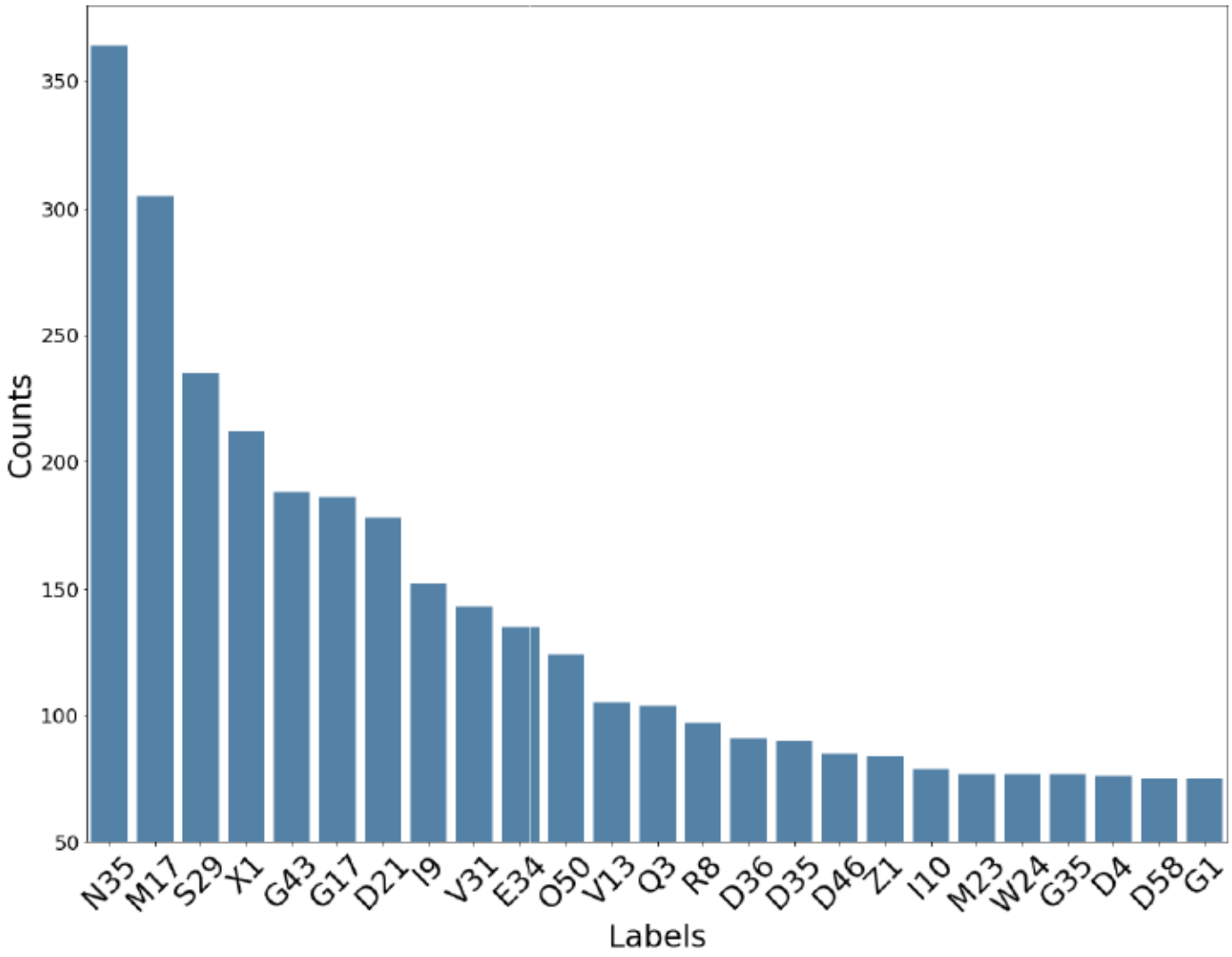

FIGURE 4. Distribution of the dataset.

2) CLASSIFICATION DES GLYPHES

Pour la tâche de classification, les 4 210 recadrages d'images ont été divisés en données d'entraînement et de test avec des rapports 70:30 respectivement de manière stratifiée. Étant donné la petite taille des données, une augmentation des données a été appliquée aux données d'entraînement. L'augmentation des données a été réalisée à l'aide de réseaux de neurones convolutifs (CNN) pour augmenter la taille des données d'entraînement, car ils sont très efficaces pour augmenter la taille des données et éviter en même temps le problème de surajustement. En plus de cela, un zoom aléatoire avec un zoom minimum de 0% et un zoom maximum de 15% a également été utilisé pour agrandir l'ensemble de données. En plus d'une rotation entre −11◦ et +11◦, différents niveaux de luminosité de l'image avec une moyenne commençant de −21% à 21% ont également été utilisés dans le même but, et quelques pixels de bruit (3%) ont également été ajoutés à certaines images.

À la fin, nous avons réussi à augmenter la taille de l'ensemble de données d'entraînement de 2 947 images à 8 226 images. La figure 5 montre quelques exemples d'images augmentées.

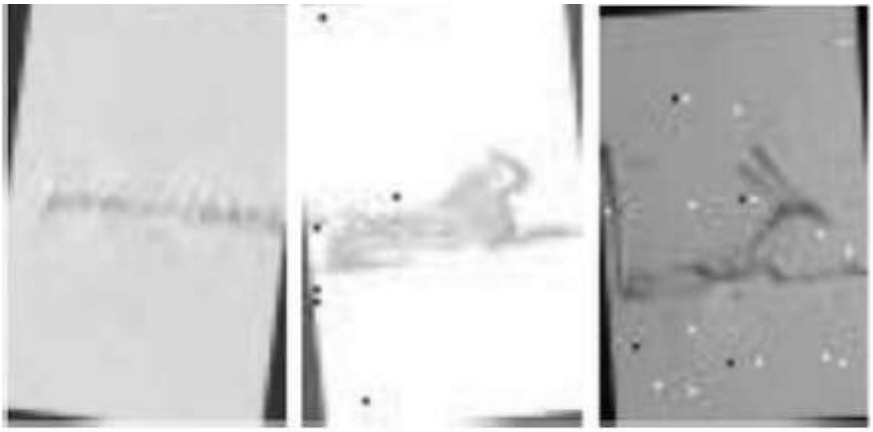

FIGURE 5. Augmented data-set samples.

B. CORPUS TEXTUEL

Le corpus textuel est un ensemble de données hybrides construit à l'aide de deux sources : Lauren Fay Rosenberg sur Github[11], et l'ensemble de données obtenu à partir des ensembles de données *Hugging Face*[8]. Dans l'ensemble de données de Rosenberg, chaque échantillon est une ancienne phrase égyptienne au format de translittération, accompagnée de la traduction en anglais correspondante. L'ensemble de données *Hugging Face* est le même que celui de Rosenberg, mais la traduction est en allemand pour la plupart des échantillons. Certains échantillons ont une traduction en anglais au lieu de l'allemand, et d'autres ont des traductions qui sont un mélange d'allemand et d'anglais. L'ensemble de données comporte un champ supplémentaire pour les codes Gardiner associés à l'échantillon, mais malheureusement, ce champ est vide pour la plupart des échantillons.

L'un des problèmes ici est que les formats de translittération sont différents entre les deux ensembles de données. De plus, l'ensemble de données *Hugging Face* comporte des interprétations et des translittérations qui sont endommagées, manquantes ou peu claires. Par exemple, elles peuvent entourer une partie de la translittération de points d'interrogation pour indiquer que cette partie manquait dans le source.

Pour unifier les données, certaines étapes de prétraitement des données ont été appliquées à l'ensemble de données *Hugging Face*. Tout d'abord, une traduction de l'allemand vers l'anglais a été effectuée, puis les crochets ont été supprimés ainsi que la translittération à l'intérieur. Plusieurs auteurs de publications[10] ont confirmé qu'il est préférable de conserver autant d'informations que possible, mais grâce à ces recherches, il a été découvert que manipuler les données d'une manière différente pourrait être très utile, donc un mappage de certains caractères de l'ensemble de données *Hugging Face* pour correspondre à l'autre format a été effectué.

Pour la partie traduction automatique, différentes sources de données ont été utilisées, voici les ensembles de données utilisés à cette étape et comment ils ont été augmentés :

[11] https://github.com/fayrose/EgyptianTranslation

1) Ensemble de données Hugging Face[10]
2) Ensemble de données de Rosenberg[11]
3) L'ensemble du corpus : combinaison de (1) et (2)
4) Ensemble de données augmentées BERT
5) Ensemble de données augmentées des synonymes
6) Ensemble de données augmentées de rétrotraduction
7) Ensemble de données entièrement augmentées (« All-augmented »)

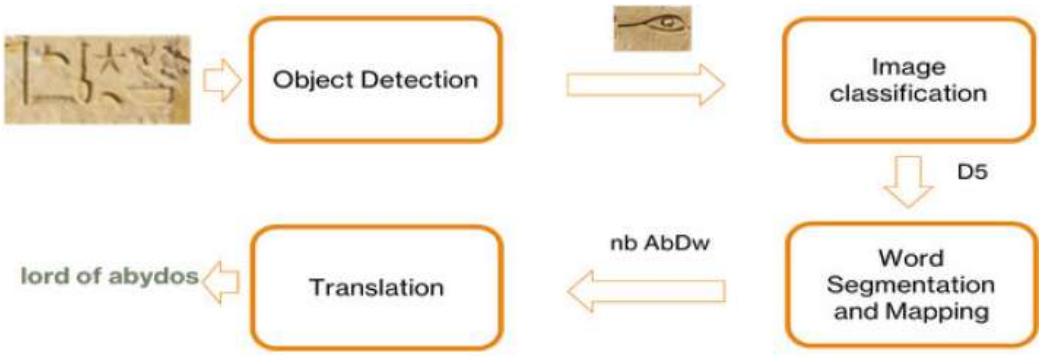

FIGURE 6. The overall diagram of the project's workflow.

La bibliothèque *nlpaug*[12] a été utilisée pour augmenter[13] les données. L'ensemble de données BERT[14] augmenté a été construit en insérant des mots supplémentaires qui n'affectent pas le sens de la phrase. L'ensemble de données augmenté des synonymes a été construit à l'aide de *WordNet*[15] pour remplacer certains mots par leurs synonymes. L'ensemble de données augmenté par rétrotraduction a été construit en traduisant la phrase cible dans une autre langue, puis en la traduisant à nouveau en anglais afin que la formulation change mais que le sens ne change pas. L'ensemble de données *All-augmented* a été construit en fusionnant les trois ensembles de données d'augmentation. Toutes les augmentations ont été effectuées sur l'ensemble de données Rosenberg pour augmenter sa taille. L'augmentation n'est pas effectuée sur l'ensemble de données complet mais uniquement sur un sous-ensemble de l'ensemble de données qui a été choisi au hasard.

C. DICTIONNAIRE

Le dictionnaire créé par le site Web *Ancient Egypt and Archaeology*[12] a également été utilisé dans cette étude. L'ensemble de données est un fichier CSV qui contient plusieurs colonnes de mots hiéroglyphiques, leur code Gardiner, leur translittération et leur traduction. Ainsi, dans la phase de prétraitement, nous avons supprimé l'anglais, les hiéroglyphes et les doublons, nous avons ainsi un dictionnaire de 10 596 mots.

IV. MÉTHODOLOGIE

L'approche proposée dans cette étude a été divisée en deux tâches principales comme

[12] https://github.com/makcedward/nlpaug
[13] https://www.datacamp.com/tutorial/complete-guide-data-augmentation
[14] https://www.intelligence-artificielle-school.com/ecole/technologies/abert-modele-nlp/
[15] https://wordnet.princeton.edu/

mentionné précédemment. Les deux tâches sont la *reconnaissance des caractères hiéroglyphiques* et la *traduction des hiéroglyphes vers l'anglais*. Comme le montre la figure 6, l'entrée est une image contenant un groupe de hiéroglyphes qui passe par le bloc de détection d'objets. Le rôle de la partie détection d'objets est de détecter le cadre de délimitation de chaque glyphe et de le recadrer en convertissant l'image en un ensemble de recadrages ordonnés à transmettre chacun au bloc de classification. Le bloc de classification prend ensuite les images de glyphes uniques et les classe avec leurs codes Gardiner correspondants. La tâche de reconnaissance de caractères a été divisée en deux étapes car l'ensemble de données utilisé, contenant 4 210 recadrages, était insuffisant pour résoudre le problème en une seule étape.

La deuxième tâche, qui est la traduction de l'hiéroglyphe vers l'anglais a également été divisée en deux étapes. L'ensemble des codes Gardiner classés et ordonnés entre dans le bloc de segmentation et de mappage pour segmenter les codes en mots car les mots de la langue hiéroglyphique ne comportent pas d'espaces entre eux. Enfin, le bloc de traduction traduit ensuite les mots en anglais. Différentes bibliothèques Open Source pour l'apprentissage automatique et l'apprentissage profond ont été utilisées ici, comme *Tensorflow* 2.0[16] qui sert à créer une architecture pour les réseaux convolutifs s'appuyant sur ses puissants modules. *OpenNMT*[17] a également été utilisé pour la partie traduction automatique, ainsi que la bibliothèque Open Source de META[18], et *Detectron*[19] a été principalement utilisé pour la partie vision par ordinateur.

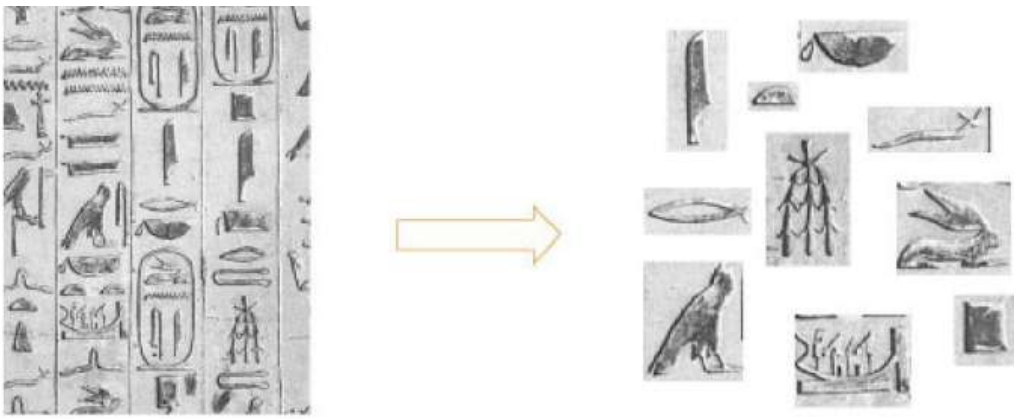

FIGURE 7. Example Input/Output for the glyph detection part.

A. DÉTECTION DE GLYPHES

Le but de cette partie est de détecter les glyphes dans une image afin qu'ils puissent d'abord être classés puis traduits en texte anglais. Étant donné une image en entrée, la sortie doit représenter les glyphes présents dans les coordonnées du cadre de délimitation, comme indiqué dans la figure 7. Pour la partie détection d'objet, l'algorithme *R-CNN*[13] a été utilisé. L'algorithme adopte un processus d'apprentissage multitâche à quatre voies : recherche de propositions de région, prédiction d'un score d'objectivité (l'appartenance à un ensemble de classes d'objets par rapport à la classe d'arrière-plan), estimation des probabilités de classe et enfin correction des coordonnées du cadre de délimitation proposé. Ce qui nous a fait choisir cet algorithme

[16] https://www.tensorflow.org/
[17] https://opennmt.net/
[18] https://opensource.fb.com/
[19] https://ai.meta.com/tools/detectron/

plutôt que d'autres, c'est qu'il fonctionne mieux pour les objets plus petits. L'apprentissage par transfert à partir des poids pré-entraînés d'*ImageNet*[20] a été utilisé, tout en gelant les deux premières étapes de son ResNet[21] à 5 étapes.

B. CLASSIFICATION

Dans la partie suivante, les architectures utilisées dans ce projet sont discutées en détail. Les réseaux siamois[22] et RestNet 50[23] ont été utilisés pour les tâches de classification d'images.

1) RESNET 50

ResNet a été développé pour la première fois par les laboratoires Microsoft Research en 2015[14]. Il est également proposé comme réseau à 34, 50 et 101 couches, mais nous avons choisi le réseau profond à 50 couches qui est pré-entraîné sur 23 millions de paramètres et a une entrée pour traiter des images de dimension 224 × 224 pixels. Le réseau représente l'apprentissage résiduel, qui résout le problème du gradient qui s'annule dans les réseaux de neurones profonds en permettant un chemin de raccourci supplémentaire pour que le gradient puisse passer. Les réseaux peuvent être formés à partir de zéro ou en utilisant l'apprentissage par transfert[15]. Dans cette étude, de nouvelles couches ont été ajoutées, telles que : une couche d'entrée de 70 × 75 pour s'adapter aux images, une couche dense de 256 neurones et une couche d'activation *SoftMax*[24] comme sortie pour la classification (nombre d'étiquettes). Ensuite, le modèle a été compilé : une descente de gradient stochastique (SGD) avec une impulsion de 0,9 et un taux d'apprentissage de 0,001, une taille de lot de 64 et 400 itérations et la fonction de perte utilisée est l'entropie croisée catégorique.

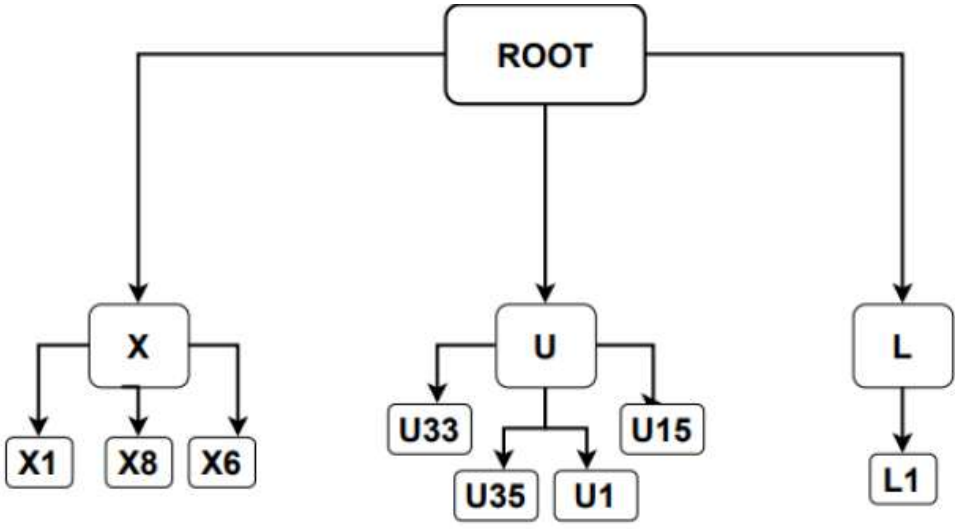

FIGURE 8. Part of the hierarchical classification.

[20] https://www.image-net.org/
[21] Réseau neuronal résiduel, une création de Microsoft.
[22] Un *réseau neuronal siamois* (Siamese Network), parfois appelé *réseau neuronal jumeau*, est un réseau neuronal artificiel qui utilise les mêmes poids tout en travaillant en tandem sur deux vecteurs d'entrée différents pour calculer des vecteurs de sortie comparables (Wikipedia US).
[23] https://www.innovatiana.com/post/discover-resnet-50
[24] Une *couche d'activation Softmax* est une fonction d'activation utilisée principalement dans les problèmes de classification multi-classes en apprentissage profond. Elle transforme un vecteur de valeurs réelles en un vecteur de probabilités où la somme des sorties est égale à **1** (Wikipedia).

2) RESNET 50 HIÉRARCHIQUE

Étant donné que 172 étiquettes sont désormais disponibles, ce qui est un grand nombre par rapport au petit nombre d'images mentionné ci-dessus dans l'ensemble de données, une nouvelle approche qui utilise la classification hiérarchique à l'aide de ResNet 50 est proposée dans cette étude. Fondamentalement, la classification hiérarchique de la figure ci-dessus a divisé la classification en deux étapes principales. La première étape a commencé à partir de la racine, de sorte que le modèle prédit la lettre Gardiner qui représente l'image. La deuxième étape consiste à représenter chaque lettre Gardiner à l'aide d'un petit modèle qui prédit en particulier le groupe Gardiner, de cette façon un groupe de modèles au lieu d'un seul modèle comme indiqué dans la figure 8 est disponible. Dans l'approche hiérarchique, l'architecture précédente avec les mêmes trois nouvelles couches a été utilisée. Le changement essentiel apporté ici est de permettre une taille variable des sorties par classe, c'est-à-dire que la classe X a une activation *SoftMax* de 3 sorties et la classe G a une activation *SoftMax* de 4 sorties, etc.

3) RÉSEAU SIAMÉRISÉ

Comme mentionné précédemment, l'ensemble de données est petit et déséquilibré. Pour cette raison, les méthodes traditionnelles de classification n'ont pas bien fonctionné, en particulier pour les classes les moins représentées. Comme solution, le réseau siamois[16] a été déployé ici. Le terme « jumeau siamois » désigne un jumeau identique. Le réseau a reçu ce nom car il prend deux entrées, les alimente dans deux CNN identiques avec les mêmes poids, en extrayant deux tenseurs de sortie qui seront ensuite soustraits, obtenant le tenseur de différence. Ce tenseur de différence a ensuite été alimenté dans deux couches denses avec 512 neurones et 256 neurones et une dernière couche avec un neurone et une fonction d'activation sigmoïde. Le réseau génère 1 si les deux entrées sont similaires et 0 sinon. Le réseau siamois est généralement utilisé dans les tâches de reconnaissance de signature et de reconnaissance faciale, et il est également connu sous le nom d'apprentissage en une seule fois. L'architecture du réseau est présentée dans la figure 9.

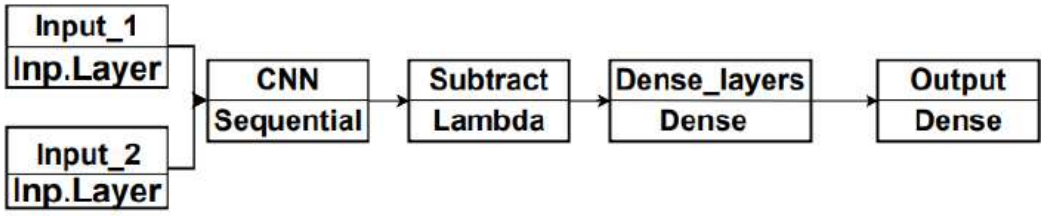

FIGURE 9. The Siamese network's architecture.

Le modèle a été compilé avec *Adam Optimizer*[25] avec un taux d'apprentissage de 0,001 perte d'entropie croisée binaire. Le réseau a été formé avec l'ensemble de données sans augmentation pendant 20 000 itérations avec une taille de lot de 128.

[25] https://keras.io/api/optimizers/adam/

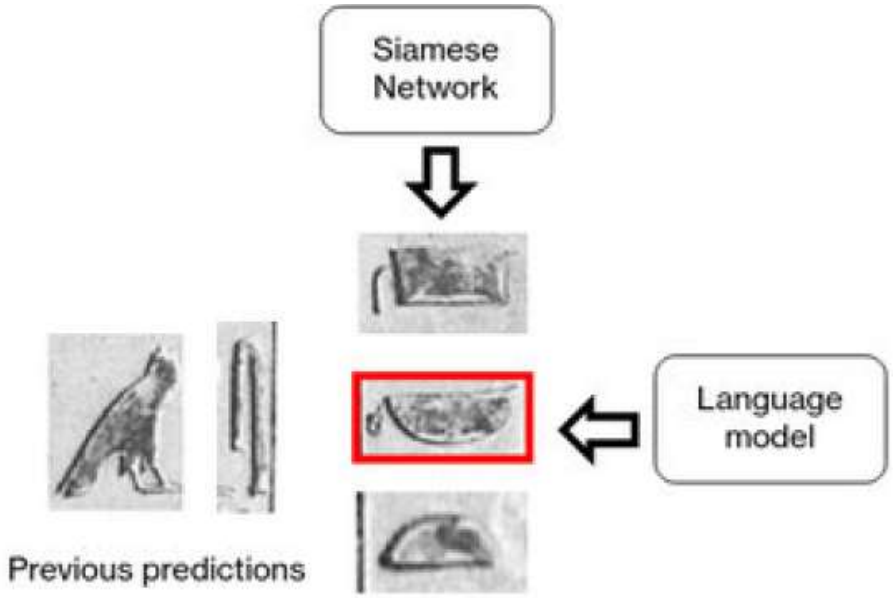

FIGURE 10. Prediction after adding the language model.

Le modèle a ensuite été affiné avec les classes qui ont moins de dix images dans les données d'entraînement pendant 1 500 itérations avec un taux d'apprentissage réduit de 1e-6 et une taille de lot de 32. L'étape d'affinement a augmenté la généralisation du modèle aux classes les moins représentées car cet ensemble moins représenté contenait des classes qui avaient une plus grande similarité, augmentant ainsi la difficulté de la formation, ce qui a donné lieu à un modèle plus performant.

Un modèle de langage au niveau des caractères a ensuite été ajouté au réseau siamois pour classer les trois prédictions les plus élevées du modèle comme indiqué dans la figure 10. Un modèle de langage tri-gramme a été utilisé pour diviser le corpus de phrases en un nouveau corpus de trois caractères dans chaque entrée. Il faut deux caractères précédents pour calculer la probabilité du caractère proposé actuel avec l'équation 1 (ci-dessous), où G1, G2 et G3 représentent les codes Gardiner dans une phrase proposée dans le bon ordre et C représente le nombre de la séquence dans notre corpus. Ce modèle de langage au niveau des caractères a été ajouté pour prendre en compte les prédictions précédentes.

Les trois scores les plus élevés du réseau siamois ont ensuite été normalisés avec la normalisation standard, puis les scores de fréquence de chaque prédiction sont calculés avec le modèle de langage donné dans les deux prédictions précédentes. Les scores de fréquence du modèle de langage ont ensuite été normalisés à l'aide de la fonction *SoftMax* 2 pour éviter de diviser par zéro si aucune des prédictions n'était rencontrée. Les scores du réseau siamois et les scores de fréquence sont ensuite additionnés pour obtenir le code Gardiner prédit.

$$P(G3|G1, G2) = \frac{C(G1, G2, G3)}{C(G1, G2)} \qquad (1)$$

$$p_i = \frac{exp(F_i)}{\sum_j exp(F_j)} \qquad (2)$$

C. SEGMENTATION ET MAPPAGE À L'AIDE D'UN DICTIONNAIRE

L'entrée de ce module est constituée des caractères hiéroglyphiques sous forme de codes Gardiner obtenus à partir du module précédent. La phrase est ensuite segmentée en mots de sorte que les mots de sortie sont mappés dans la langue de translittération pour démarrer la phase de traduction en anglais.

L'objectif est d'essayer différentes techniques de segmentation, l'une basée sur le dictionnaire et l'autre utilisant des techniques de *tokenisation* de sous-mots[26]. Le travail ici est divisé en deux catégories : tout d'abord, la segmentation de mots à l'aide des algorithmes basés sur des règles[17], où la segmentation est effectuée sur la base d'un dictionnaire, et l'autre utilise la *tokenisation* de sous-mots ou *Sentencepiece*[27].

Vous trouverez ci-dessous une explication plus détaillée des deux méthodes et une comparaison entre elles.

1) ALGORITHME BASÉ SUR UN DICTIONNAIRE

Les algorithmes basés sur des règles sont utilisés pour segmenter les mots en fonction d'un dictionnaire. Les sous-sections suivantes décrivent deux algorithmes basés sur des règles différents.

a) Correspondance maximale avant[18] : cet algorithme fonctionne simplement en prenant les m caractères les plus longs de la séquence qui correspond à un mot dans le dictionnaire. Une recherche du mot dans le dictionnaire est effectuée ; s'il est trouvé, il sera supprimé de la séquence, sinon le dernier caractère de la séquence sera supprimé et une nouvelle séquence sera créée. L'algorithme itère pour segmenter tous les caractères ou mots en conséquence.

b) Correspondance maximale inverse[19] : l'algorithme de correspondance inverse fonctionne comme l'algorithme avant, mais supprime uniquement un caractère du début de la séquence, si la recherche ne trouve pas de séquence correspondante.

2) TOKENIZER DE SOUS-MOTS

Sentencepiece est un *tokenizer-detokenizer* de sous-mots indépendant de la langue, conçu pour le traitement de texte basé sur les neurones y compris la traduction automatique neuronale. *Sentencepiece* peut entraîner des modèles de sous-mots directement à partir de phrases brutes.

Dans la partie *Sentencepiece*, le modèle a été entraîné sur des phrases d'une série de codes Gardiner comme : (M11N5A13.. .). La segmentation n'a pas été exclue car elle segmentait les lettres et les chiffres séparément, par exemple : « M » et « 11 » étaient considérés comme deux tokens distincts, bien que les vrais codes Gardiner devraient ressembler à « 11 », « 5 », etc.

---

[26] Les algorithmes de tokenisation en sous-mots reposent sur le principe selon lequel les mots fréquemment utilisés ne doivent pas être divisés en sous-mots plus petits, mais les mots rares doivent être décomposés en sous-mots significatifs. https://huggingface.co/learn/nlp-course/fr/chapter2/4

[27] https://github.com/google/sentencepiece

Une autre raison est le petit nombre de phrases qui n'étaient pas suffisantes pour le modèle, car l'entraînement de ce modèle nécessite un ensemble de données riche où le modèle peut facilement trouver une combinaison entre des séquences de caractères.

Rien dans la littérature sur ce problème n'a été trouvé, et aucune tentative n'a été faite pour le résoudre au mieux de nos connaissances, donc une étape de prétraitement a été mise en œuvre pour remplacer chaque chiffre de la séquence par une combinaison de lettres correspondante provenant d'un dictionnaire créé.

Le dictionnaire est le suivant : (1 : ab, 2 : ac,. . .), donc la séquence sera « MagNae ». En conséquence, la segmentation s'est beaucoup augmentée et le modèle a surmonté le problème des combinaisons de lettres et de chiffres.

D. TRADUCTION AUTOMATIQUE

Pour pouvoir traduire le texte translittéré résultant des étapes de segmentation et de mappage, l'architecture du transformateur a été utilisée ici. Elle a été formée à l'aide des différents paramètres de jeu de données du corpus textuel mentionnés précédemment. Les hyper-paramètres utilisés sont les mêmes que ceux utilisés dans [10]. Cependant, il reste encore beaucoup de travail à faire dans la phase de traduction automatique pour obtenir un modèle complet qui traduit les hiéroglyphes.

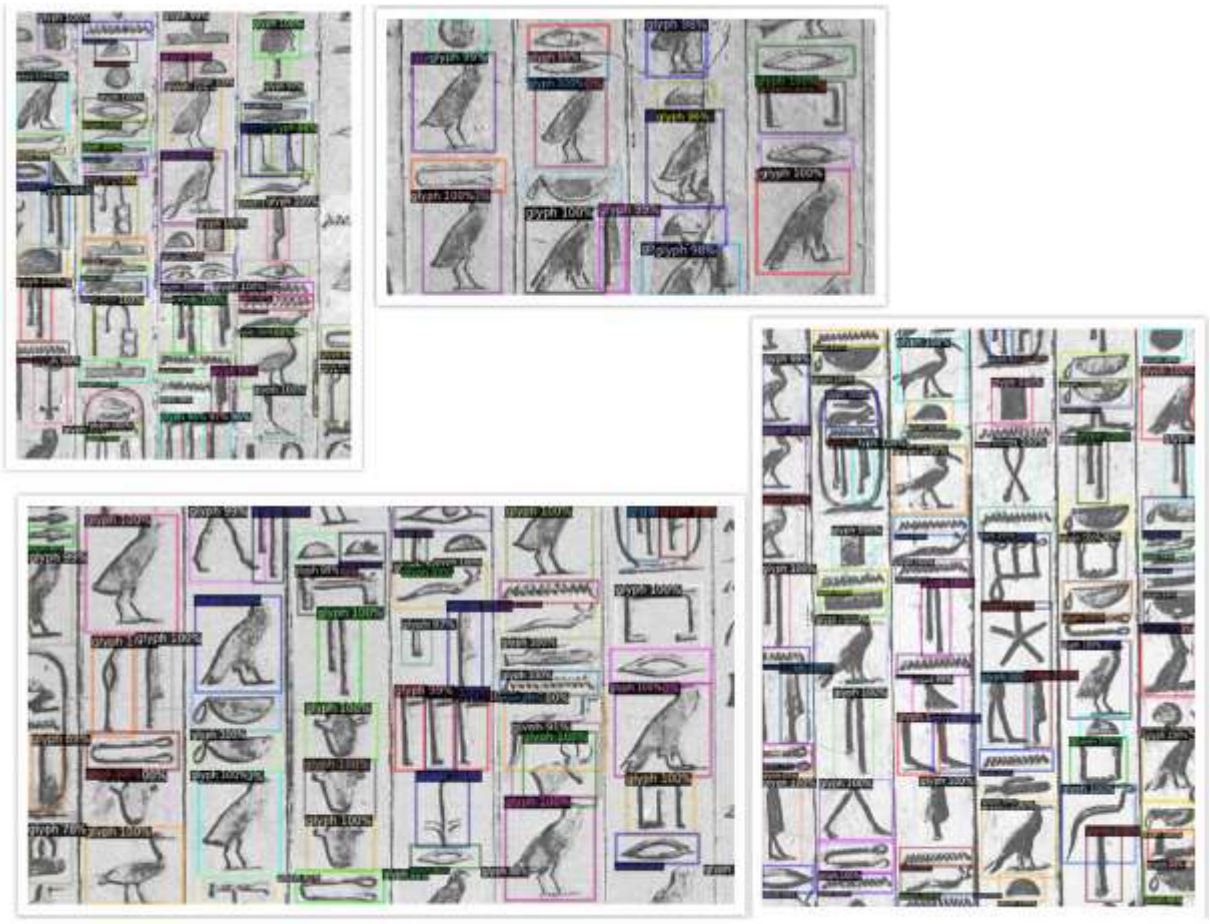

FIGURE 11. Example glyph detection output.

V. RÉSULTATS

Dans cette section, une discussion des résultats obtenus par chaque module séparément est présentée, chaque module comme précédemment discuté résout un problème spécifique indépendant avec son ensemble de données spécifiques. Nous passons ensuite en revue l'évaluation de l'ensemble du flux de travail intégré.

Pour évaluer l'efficacité du module de détection de glyphe, l'ensemble de données susmentionné a été divisé de manière aléatoire en ensembles d'entraînement et de validation dans un rapport de division de 80:20. Le processus d'entraînement a été exécuté pendant 1000 itérations avec un taux d'apprentissage de 0,01 qui est réduit d'un facteur de 0,1 aux 400e, 600e et 800e itérations ainsi qu'en utilisant une préparation du taux d'apprentissage en 100 étapes. L'entraînement a pris environ une heure sur un GPU NVIDIA Tesla V100. Les expériences ont donné une précision moyenne finale de 95,9 % et un rappel moyen de 74,4 % à l'intersection sur l'union de 0,5.

La figure 11 montre quelques images de sortie après les avoir introduites dans le modèle obtenu.

En utilisant le modèle siamois, une comparaison par paires avec un tableau étiqueté est effectuée. Les tableaux d'ancrage ont été créés pour contenir une image d'ancrage et son étiquette correspondante par classe d'instance et son code Gardiner correspondant. Pour l'évaluation, l'image de test a été comparée à chaque entrée du tableau *anchor_img* pour calculer le score de similarité avec le réseau siamois, puis le code Gardiner correspondant à l'entrée avec le score de similarité le plus élevé. Nous avons ensuite calculé la précision du nombre d'images de test correctement classées par rapport au nombre total d'images testées.

La précision du modèle après un réglage fin avec les classes les moins représentées est de 85 %. Étant donné que la méthode d'évaluation est très sensible à notre choix des images dans le tableau d'ancrage, un tableau d'ancrage multicouche avec trois images par classe a été créé. L'image de test contenait les trois images de chaque entrée et nous avons additionné les scores du modèle pour signaler le score moyen le plus élevé en correspondance avec le code Gardiner approprié. Cette méthode a augmenté la précision à 87,5 %. Enfin, l'ajout du modèle de langue a permis d'obtenir la précision la plus élevée de 88,5 %.

Pour l'étape de classification, différentes approches ont été menées et comparées. ResNet 50 présentant une certaine limitation en raison de la taille des données, nous avons donc déplacé le réseau siamois.

De nombreuses approches pour augmenter la précision des modèles ont été déployées et une comparaison des résultats de chaque approche est présentée dans le tableau 1. Le réseau siamois est mieux adapté au problème de la classification des langues à faibles ressources.

Cependant, pour un plus grand nombre de classes et plus de données pour la formation, ResNet 50 pourrait être un meilleur choix car comparer l'image de test avec plus de classes prendrait beaucoup de temps.

TABLE 1. Comparison of the methods used for classification.

Model	Accuracy
ResNet50 before data augmentation	61%
ResNet50 after data augmentation	72%
Hierarchical Classification	68%
Siamese Network	85%
Siamese with Multi-anchor array	87.5%
Siamese after adding a Language model	88.5%

Pour garantir que la sortie du dictionnaire soit similaire à l'ensemble de données d'origine, des algorithmes de distance de *Hamming* et de correspondance de séquence ont été utilisés pour évaluer la similarité entre les phrases produites et les phrases d'origine.

1) La distance de Hamming[20] :

Cet algorithme compare deux phrases, en comparant le nombre de positions de caractères dans lesquelles les deux phrases sont différentes à l'aide d'un simple XOR.

2) SEQUENCE MATCHER[21] :

SequenceMatcher est un algorithme qui compte le nombre de caractères correspondants entre deux chaînes, puis génère un ratio compris entre 0 et 1, où 1 signifie que les deux chaînes sont parfaitement similaires et 0 signifie le contraire.

$$Ratio = \frac{T}{M}$$

où M est la séquence correspondante et T est le nombre total d'éléments dans les deux séquences. Après avoir supprimé toutes les phrases qui ne contenaient pas les codes Gardiner de l'ensemble de données, nous avions un total de 2 538 phrases, et après avoir supprimé les doublons et les valeurs NAN[28], nous avions 2 350 phrases. Le tableau suivant présente les résultats des techniques de segmentation de mots.

TABLE 2. Comparison of results for word segmentation techniques.

Algorithm	Hamming distance	sequence Matcher
Forward Matching	6.6	0.58
Inverse Matching	6.7	0.57
Sentencepiece	3	0.21

Comme le montre le tableau 2, l'algorithme champion est le *Forward max matching* car il a atteint un ratio de segmentation correcte de 60 % de la phrase d'origine

[28] https://developer.mozilla.org/fr/docs/Glossary/NaN

correctement mappée.

Le problème est que les phrases codées Gardiner contiennent des caractères inconnus, donc la suppression des symboles nécessitera un égyptologue pour interpréter les caractères manquants.

Le principal défi était la dépendance au dictionnaire pour faire une bonne segmentation, il est donc absolument nécessaire d'avoir soit un dictionnaire riche pour couvrir la majeure partie du vocabulaire, soit un ensemble de données avec un grand nombre de phrases bien traitées et ne comportant aucun caractère manquant, ce qui est impossible dans une langue à si faible ressources.

Pour évaluer la partie traduction automatique, le score BLEU a été calculé de quatre manières différentes :

- BLEU de phrase de NLTK,
- GLEU de NLTK,
- BLEU de corpus de NLTK et
- sacreBLEU - BLEU.

Pour le BLEU de NLTK, une fonction de lissage a été utilisée lors du calcul du score BLEU.

Le tableau suivant montre les résultats de la traduction.

TABLE 3. Evaluation of translation results using different BLEU score techniques.

Setting	NLTK's BLEU	NLTK's GLEU	NLTK's corpus BLEU	sacreBLEU-BLEU
1	**49.73**	47.83	**59.19**	28.12
2	31.56	30.73	34.59	42.73
3	47.36	44.94	58.73	42.73
4	48.71	47.44	58.69	**70.71**
5	49.52	**48.34**	58.39	48.55
6	47.83	46.85	57.59	50.00
7	46.71	46.81	56.32	35.36

Le score le plus élevé a été mis en évidence pour chaque méthode de score BLEU afin de pouvoir voir quel paramètre a donné les meilleurs scores BLEU obtenus. Le modèle de traduction proposé dans cette étude a obtenu le score BLEU le plus élevé par rapport aux travaux précédents dans la littérature.

Comparé au travail de Rosenberg, le corpus BLEU du NLTK atteint 59,19 dans le paramètre 1, ce qui est plus que le maximum actuel du corpus BLEU du NLTK de Rosenberg de 42,22. Voici quelques analyses des résultats à utiliser à l'avenir pour améliorer les performances de cette tâche.

Le premier paramètre de données a obtenu de meilleurs résultats que les autres dans deux méthodes, ce qui suggère qu'une source de données unique sans augmentation pourrait être la meilleure pour les langues à faibles ressources. De plus, les résultats du 5ème paramètre sont proches du premier paramètre avec le score GLEU le plus élevé, ce qui indique que l'augmentation à l'aide de synonymes pourrait être une bonne option à explorer davantage. Le 6ème paramètre s'est avéré être le pire. Cela peut être dû au problème TSIC[29], car la rétrotraduction traduit la phrase dans une autre langue, puis de nouveau en anglais, mais les phrases du corpus d'entraînement ne correspondent pas à l'anglais moderne, donc ces traductions peuvent être trompeuses et déroutantes pour la machine. L'analyse a montré que le paramètre n° 6 est la raison pour laquelle les mauvaises performances sont obtenues par rapport au paramètre n° 7, qui utilise toutes les techniques d'augmentation. Le paramètre n° 4 a le score le plus élevé dans sacreBLEU, mais les résultats de sacreBLEU semblent instables avec beaucoup de variance par rapport aux autres. Pour le paramètre n° 3, l'analyse a montré que les résultats sont inférieurs à ceux du paramètre n° 1 pour deux raisons : tout d'abord, les performances de l'ensemble de données dans le paramètre 2 ne sont pas très bonnes. Ensuite, l'ensemble de données a été traduit de l'allemand vers l'anglais, un anglais qui serait différent de l'anglais de l'ensemble de données du paramètre n° 1 en raison du TSIC, puisque les moteurs de traduction ont tendance à créer une phrase qui ressemble plus à l'anglais moderne.

En conclusion, l'augmentation des données entraînerait une baisse des performances de traduction, mais une augmentation telle que l'utilisation de synonymes qui remplacent simplement un mot par un mot différent de même signification mérite d'être explorée. L'utilisation d'autres ensembles de données pour compléter l'ensemble de données d'origine mérite également d'être explorée, mais elle doit utiliser la même langue cible que l'ensemble de données d'origine.

VI. CONCLUSION ET TRAVAUX FUTURS

Dans ce projet, un système complet de traduction des symboles hiéroglyphiques numérisés en anglais lisible a été proposé. De nombreux algorithmes d'apprentissage automatique et d'apprentissage profond ont été utilisés dans cette étude et une analyse des résultats a été menée pour déterminer les modèles champions et les meilleurs paramètres de données. Ce problème complexe a été divisé en deux sous-problèmes : reconnaître les glyphes sur une photo, puis les traduire en anglais. En parlant de reconnaissance des glyphes, le système proposé dans cette étude a surpassé les résultats de pointe précédemment publiés, car notre approche a été capable de détecter des glyphes avec un mAP de 95% et un AR de 75%, et de classer avec précision 88,5% des glyphes par rapport au taux de classification moyen de 66,6% obtenu par les recherches les plus récentes effectuées dans ce domaine en utilisant le même ensemble de données[7].

En ce qui concerne la tâche de traduction, le travail proposé a réussi à obtenir un score BLEU de 59,19, ce qui indique que nos modèles de traduction surpassent également les modèles équivalents trouvés dans les publications. En tant que travail futur, un score plus élevé pourrait être obtenu en traduction en utilisant un nettoyage des données plus exhaustif. Des données d'entraînement de plus grand volume amélioreraient le

[29] Translation Similarity Issue or Concept.

biais et la variabilité. Mais comme la langue est une langue à faibles ressources[30], il sera difficile d'obtenir plus de données. Comme la langue est TSIC, il n'y a pas beaucoup de variation dans les phrases du corpus textuel. Même si les phrases peuvent être différentes, de nombreux mots sont communs entre les phrases. L'analyse des résultats a également montré que le manque de variation entre les phrases a entraîné un certain niveau de biais. Par conséquent, une analyse plus poussée des données et des recherches sur les similitudes entre les ensembles d'entraînement et de test devraient être effectuées. Dans le cadre d'un travail futur, il faudrait également trouver d'autres sources d'ensembles de données qui correspondent à l'ensemble du processus, de la reconnaissance des glyphes à la traduction des glyphes. Il serait peut-être utile de trouver une autre source de données écrites horizontalement pour analyser les performances de nos modèles qui y font face, car l'évaluation du travail proposé s'est limitée à la Pyramide d'Ounas dans laquelle les écritures sont orientées verticalement.

REMERCIEMENTS

Les auteurs souhaitent remercier et apprécier le soutien fourni par la société Microsoft en Égypte, en particulier le Dr Ahmed Tawfik et la *Digital Egypt Builders Initiative* (DEBI), tout au long de leur voyage avec nous sur ce projet.

RÉFÉRENCES

[1] C. J. van Schaik, L. L. Boer, J. M. Draaisma, C. J. van der Vleuten, J. J. Janssen, J. J. Futterer, L. J. S. Kool, and W. M. Klein, ‘‘The lymphatic system throughout history: From hieroglyphic translations to state of the art radiological techniques,’’ Clin. Anatomy, vol. 35, no. 6, pp. 701–710, 2022.
[2] B. Manley, Egyptian Hieroglyphs for Complete Beginners, 1st ed. London, U.K.: Thames and Hudson, 2012, pp. 1–100.
[3] M. Franken and J. C. van Gemert, ‘‘Automatic Egyptian hieroglyph recognition by retrieving images as texts,’’ in Proc. 21st ACM Int. Conf. Multimedia, 2013, pp. 765–768.
[4] B. Epshtein, E. Ofek, and Y.Wexler, ‘‘Detecting text in natural scenes with stroke width transform,’’ in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., Jun. 2010, pp. 2963–2970.
[5] E. Roman-Rangel, C. P. Gayol, J.-M. Odobez, and D. Gatica-Perez, ‘‘Searching the past: An improved shape descriptor to retrieve maya hieroglyphs,’’ in Proc. 19th ACM Int. Conf. Multimedia, 2011, pp. 163–172.
[6] A. Barucci, C. Cucci, M. Franci, M. Loschiavo, and F. Argenti, ‘‘A deep learning approach to ancient Egyptian hieroglyphs classification,’’ IEEE Access, vol. 9, pp. 123438–123447, 2021, doi: 10.1109/ACCESS.2021.3110082.
[7] R. Elnabawy, R. Elias, and M. Salem, ‘‘Image based hieroglyphic character recognition,’’ in Proc. 14th Int. Conf. Signal-Image Technol. Internet-Based Syst. (SITIS), Nov. 2018, pp. 32–39, doi:10.1109/SITIS.2018.00016.

[30] Une langue à faibles ressources (ou *low-resource language* en anglais) est une langue qui dispose de peu de données linguistiques numériques telles que des corpus écrits, des dictionnaires, des traductions parallèles ou des outils de traitement du langage naturel (NLP). Ces langues manquent souvent des ressources nécessaires pour entraîner des modèles linguistiques performants, en particulier dans des domaines comme la traduction automatique, la reconnaissance vocale ou l'analyse syntaxique (ChatGPT)

[8] R. Bansal, H. Choudhary, R. Punia, N. Schenk, J. L. Dahl, and E. Page-Perron, "How low is too low? A computational perspective on extremely low-resource languages," 2021, arXiv:2105.14515.
[9] A. Buslaev,V. I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A. A. Kalinin, "Albumentations: Fast and flexible image augmentations," Information, vol. 11, no. 2, p. 125, Feb. 2020.
[10] P. Wiesenbach and S. Riezler, "Multi-task modeling of phonographic languages: Translating middle Egyptian hieroglyphs," in Proc. 16th Int. Conf. Spoken Lang. Transl., 2019, pp. 1–7.
[11] (2021). Machine Translation for Middle Egyptian-English. [available online]. https://github.com/fayrose/EgyptianTranslation .
[12] (2021). Ancient Egypt Dictionary. [available online]. http://www.ancient-egypt.co.uk/transliteration/ancient_egypt_dictionary.pdf
[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Computer Vision and Pattern Recognition. 2016.
[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Las Vegas, NV, USA, Jun. 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
[15] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2014, pp. 1–9.
[16] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "DeepFace: Closing the gap to human-level performance in face verification," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2014, pp. 1701–1708.
[17] (2021). Natural Language Processing - Rule Segmentation. [available online]. https://programmer.group/natural-language-processing-rulesegmentation.html
[18] J. Tang, Q. Wu, and Y. Li, "An optimization algorithm of Chinese word segmentation based on dictionary," in Proc. Int. Conf. Netw. Inf. Syst. Comput. (ICNISC), 2015, pp. 259–262.
[19] J. Wu and Z. Tu, "Reverse image segmentation: A high-level solution to a low-level task," in Proc. Brit. Mach. Vis. Conf., 2014, pp. 1–13.
[20] M. Norouzi, J. D. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in Proc. Neural Inf. Process. Syst. Conf., 2012, pp. 1–9.
[21] T. Mondal, N. Ragot, J.-Y. Ramel, and U. Pal, "Flexible sequence matching technique: Application to word spotting in degraded documents," in Proc. 14th Int. Conf. Frontiers Handwriting Recognit., Sep. 2014, pp. 210–215.