

PROJET3: Feature Extraction

Membres de l'équipe :

- MAUREL Perrine (M2)
- PARMENTELAT Aaron (M1)
- SAIAG Violette (M2)
- VLACHOU-EFSTATHIOU Malamatenia (M2)

Master Humanités Numériques -ENC/PSL S2

1. Etat de l'art et perspectives/applications du projet

L'objectif de ce projet est d'explorer l'extraction (et si possible l'interprétation) des features issues d'un classificateur CNN de deux classes de manuscrits médiévaux différents : fortement abrégés et non-abrégés. L'intérêt derrière le projet était de comparer ce que la machine comprend des manuscrits abrégés et non abrégés par rapport à l'oeil humain/du paléographe.

Mis à part l'interprétation des features, il est essentiel pour la communauté de la paléographie numérique de pouvoir entraîner un classifieur efficace permettant de classer automatiquement et en masse des manuscrits fortement abrégés. Cela est particulièrement important pour des projets tels que CREMMA-Medii Aevi, qui sélectionnent leurs ensembles de données en fonction du degré d'abréviations afin d'obtenir des résultats souhaités. Avec l'accès quasi-illimité à une multitude de manuscrits disponibles dans les bibliothèques virtuelles, automatiser le processus de classification des pages manuscrites en fonction de leur degré d'abréviation sans avoir recours à une transcription ou à une vérité-terrain pour le calcul, pourrait grandement faciliter le travail des chercheurs en sciences humaines computationnelles qui souhaitent constituer des ensembles de données personnalisés, sans dépendre exclusivement de projets externes. L'interprétation des caractéristiques constitue une étape supplémentaire qui contribuerait à mieux comprendre la "boîte noire" à l'intérieur de l'algorithme de classification.

Pour mettre en place cela, l'équipe s'est inspirée du cadre et du code de DeepScript de Mike Kestemont : <https://github.com/mikekestemont/DeepScript> où l'équipe, lors d'un entraînement d'un classifieur pour 12 classes d'écritures médiévales l'équipe a observé des traces des signes d'abréviation apparaissant dans leur feature map.

2. Création du dataset

La première étape du projet consiste à choisir et construire un jeu de données adaptée et pertinent pour notre expérience. Nous avons décidé de travailler avec des jeux de données en latin et en français ancien qui sont bien documentés et dont le matériel nous a été mis à disposition. Plus précisement :

- CREMMA-fro: <https://github.com/HTR-United/cremma-medieval> :**13-15e s. 289 pages manuscrites – Segmentation avec SegmOnto**
- CREMMA-lat: <https://github.com/HTR-United/CREMMA-Medieval-LAT> :**12-16e s. - 121 pages manuscrites – écritures caractérisés – Segmentation avec SegmOnto**
- Gallo corpora_15: <https://github.com/Gallicorpora/HTR-MSS-15e-Siecle> :**13-15e s. – Segmentation avec SegmOnto**
- ECMEN: <https://github.com/oriflamms/ECMEN> :**341 pages manuscrites – écritures caractérisés – Segmentation des zones disponible mais pas SegmOnto)**
- Dataset **Marguerite Vernet** : **12 pages (46 colonnes)** d'un même manuscrit : le BnF lat. 14525. Il vient de la bibliothèque de Saint-Victor de Paris. Il a été composé sur une trentaine d'année en tout avec différentes UC de **1200 à 1230**. Ici tu as l'exemple de plusieurs mains différentes et habitudes d'abréviations proches mais avec quelques spécificités pour chaque main.

La question la plus importante consiste à définir les deux classes de manière à ce qu'elles représentent de manière significative ce que nous considérons comme des pages de manuscrits "fortement abrégées" et "non abrégées". Étant donné que les degrés d'abréviation varient parmi les pages manuscrites et qu'il n'existe pas de consensus général sur le taux d'abréviation élevé défini explicitement par la communauté, nous avons décidé de tirer parti de l'expérience des chercheurs qui entraînent des modèles de reconnaissance de caractères. Plus précisément, dans l'article suivant : <https://hal-enc.archives-ouvertes.fr/hal-03828353> publié début 2023, Clérice et al. ont observé qu'une marge de plus de 6% améliorait considérablement les performances du modèle de reconnaissance au niveau des abréviations. Cela nous amène à penser que les manuscrits qui répondent à cette condition représentent de manière satisfaisante la classe correspondant aux abréviations. Le seuil a été fixé à <1 pour les manuscrits à peine abrégés et à >6,5 pour les manuscrits fortement abrégés.

Les ensembles de données choisis présentent l'avantage d'être accompagnés de transcriptions et d'une segmentation au niveau des zones, ce qui permet d'extraire uniquement la partie correspondant aux zones et d'éviter ou de réduire le bruit provenant de l'arrière-plan ainsi que des initiales/figures qui pourraient influencer les caractéristiques des manuscrits. Afin de diviser de manière objective les jeux de données en deux catégories distinctes, nous avons eu l'idée d'exploiter les transcriptions et de calculer le pourcentage total/ratio de signes abréviatifs par page. Cette approche a permis d'obtenir des classes bien définies, comme présenté dans le notebook du dossier des données disponible ici : <https://github.com/malamatenia/FeatureExtractionProject/tree/main/data/notebooks>

3. Feature extraction

3.1. Data augmentation

La data augmentation était nécessaire dans notre cas en raison de deux difficultés principales : en premier lieu, nous avons peu de données initiales ; aussi, nous ne travaillons que sur deux catégories, ce qui est assez rare (par exemple, le code de Mike Kestemont dont nous nous sommes inspiré.e.s travaillait sur douze catégories). Ainsi, la data augmentation était nécessaire pour avoir un dataset plus conséquent et pour augmenter la variance des deux classes.

Nous avons choisi d'utiliser la librairie Albumentations car elle offre une large gamme de transformations d'images prédéfinies telles que les rotations, les translations, les redimensionnements, les crops, les changements de contraste, les changements de luminosité, etc. Cette bibliothèque offre également une grande flexibilité en permettant de combiner aisément plusieurs transformations, ce qui nous permet de créer des pipelines de transformations complexes et adaptées à nos données pour répondre à des besoins spécifiques. Aussi, Albumentations est compatible avec Keras, que nous comptons utiliser pour l'entraînement.

Nous avons adapté un code de Chahan Vidal-Gorene à nos besoins et notre dataset.

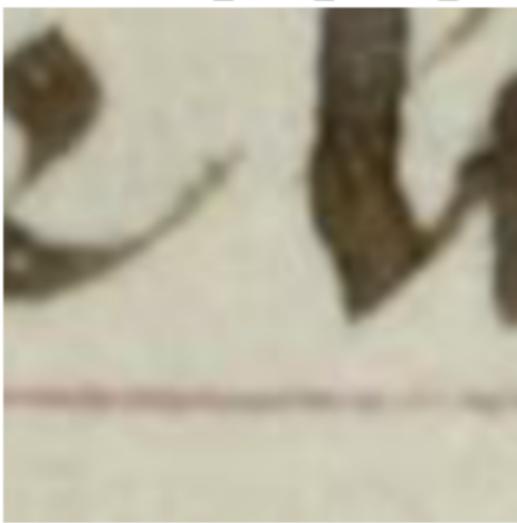
Nous avons appliqué cinq transformations à nos données :

- “RandomBrightnessContrast” qui ajuste aléatoirement la luminosité et le contraste de l'image avec une probabilité de 80% ;
- “RandomCrop”, qui effectue un recadrage aléatoire de l'image à une taille spécifiée, est une transformation toujours appliquée (avec une probabilité de 1.0) ;
- “RandomResizedCrop” effectue un recadrage aléatoire de l'image à une taille spécifiée également, puis redimensionne l'image : cette transformation est également toujours appliquée. Pendant ces crops, nous avons pris soin de conserver le “aspect ratio” afin de ne pas compresser l'image et d'éviter les distorsions ;
- “Equalize”, appliquée une fois sur deux, égalise l'histogramme de l'image ;
- “GaussianBlur” applique un flou gaussien dont nous définissons les limites grâce aux paramètres “blur_limit” et “sigma_limit”.

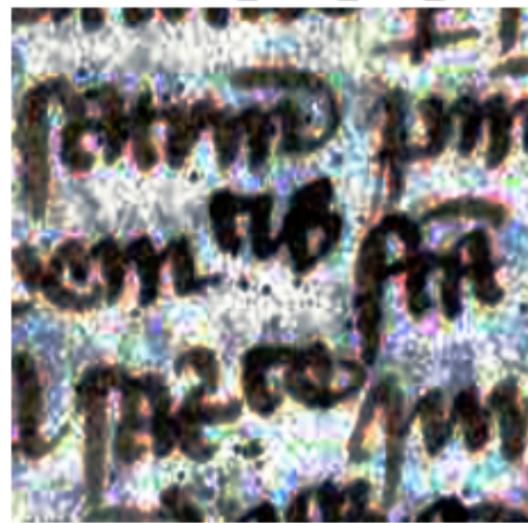
Grâce à cette data augmentation, nous sommes passé.e.s de 177 images fortement abrégées et 185 images peu abrégées selon le seul déterminé à 34.776 images fortement abrégées et 36.384 images faiblement abrégées pour un total de 71.160 images dont 14.232 seront utilisées comme set de validation.

Nous avons imprimé aléatoirement dans notre notebook une image pour chacune des deux classes :

`transformed_final_faible_abbr`



`transformed_final_fort_abbr`



3.2. Entraînement

Nous avons split nos données en utilisant 80% des images pour l'entraînement et 20% pour la validation. Nous avons entraîné deux modèles différents : un ResNet50 et un modèle que nous avons fait nous-mêmes. Notre batch_size est de 32 pour tous les entraînements : nous avons en effet gardé les mêmes paramètres pour pouvoir comparer les deux modèles différents.

Voici le summary du ResNet50 :

Model: "resnet50"			
Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 256, 256, 3 0)]	0	[]
conv1_pad (ZeroPadding2D)	(None, 262, 262, 3) 0	0	['input_1[0][0]']
conv1_conv (Conv2D)	(None, 128, 128, 64 9472)	9472	['conv1_pad[0][0]']
conv1_bn (BatchNormalization)	(None, 128, 128, 64 256)	256	['conv1_conv[0][0]']
conv1_relu (Activation)	(None, 128, 128, 64 0)	0	['conv1_bn[0][0]']
pool1_pad (ZeroPadding2D)	(None, 130, 130, 64 0)	0	['conv1_relu[0][0]']
pool1_pool (MaxPooling2D)	(None, 64, 64, 64) 0	0	['pool1_pad[0][0]']
conv2_block1_1_conv (Conv2D)	(None, 64, 64, 64) 4160	4160	['pool1_pool[0][0]']
...			
Total params:	23,591,810		
Trainable params:	23,538,690		
Non-trainable params:	53,120		

Nous ne souhaitions garder que l'architecture du modèle pour entraîner le ResNet50 avec nos propres données et générer nos propres poids, processus qui serait plus pertinent dans le cas de notre projet de visualisation de features. Nous avons ainsi modifié le modèle pour ne lui donner que deux classes à détecter et n'avons pas récupéré les poids.

Nous avons choisi, dans les paramètres du modèle, de l'entraîner avec un optimizer Adam, un learning rate de 0.001 (pour éviter des sauts excessifs dans l'espace des paramètres et pour atteindre le global minimum de la loss de manière progressive), pour 50 epochs, et nous avons conservé la métrique par défaut : l'accuracy. Pour la fonction de perte, nous avons choisi la “sparse_categorical_crossentropy” qui était la plus adaptée à nos données et à la tâche de classification.

Voici le summary du modèle de réseau de neurones convolutifs (CNN) à architecture séquentielle que nous avons construit nous-mêmes à l'aide de la bibliothèque Keras :

Model: "sequential"

Layer (type)	Output Shape	Param #
<hr/>		
conv1 (Conv2D)	(None, 254, 254, 8)	224
conv2 (Conv2D)	(None, 252, 252, 16)	1168
maxpool1 (MaxPooling2D)	(None, 126, 126, 16)	0
dropout1 (Dropout)	(None, 126, 126, 16)	0
conv3 (Conv2D)	(None, 124, 124, 16)	2320
conv4 (Conv2D)	(None, 122, 122, 16)	2320
maxpool2 (MaxPooling2D)	(None, 61, 61, 16)	0
dropout2 (Dropout)	(None, 61, 61, 16)	0
conv5 (Conv2D)	(None, 59, 59, 16)	2320
conv6 (Conv2D)	(None, 57, 57, 16)	2320
maxpool3 (MaxPooling2D)	(None, 28, 28, 16)	0
dropout3 (Dropout)	(None, 28, 28, 16)	0
conv7 (Conv2D)	(None, 26, 26, 16)	2320
conv8 (Conv2D)	(None, 24, 24, 16)	2320
maxpool4 (MaxPooling2D)	(None, 12, 12, 16)	0
dropout4 (Dropout)	(None, 12, 12, 16)	0
conv9 (Conv2D)	(None, 10, 10, 16)	2320
conv10 (Conv2D)	(None, 8, 8, 16)	2320
maxpool5 (MaxPooling2D)	(None, 4, 4, 16)	0
dropout5 (Dropout)	(None, 4, 4, 16)	0
flatten (Flatten)	(None, 256)	0
dense1 (Dense)	(None, 64)	16448
<hr/>		
Total params: 36,400		
Trainable params: 36,400		
Non-trainable params: 0		

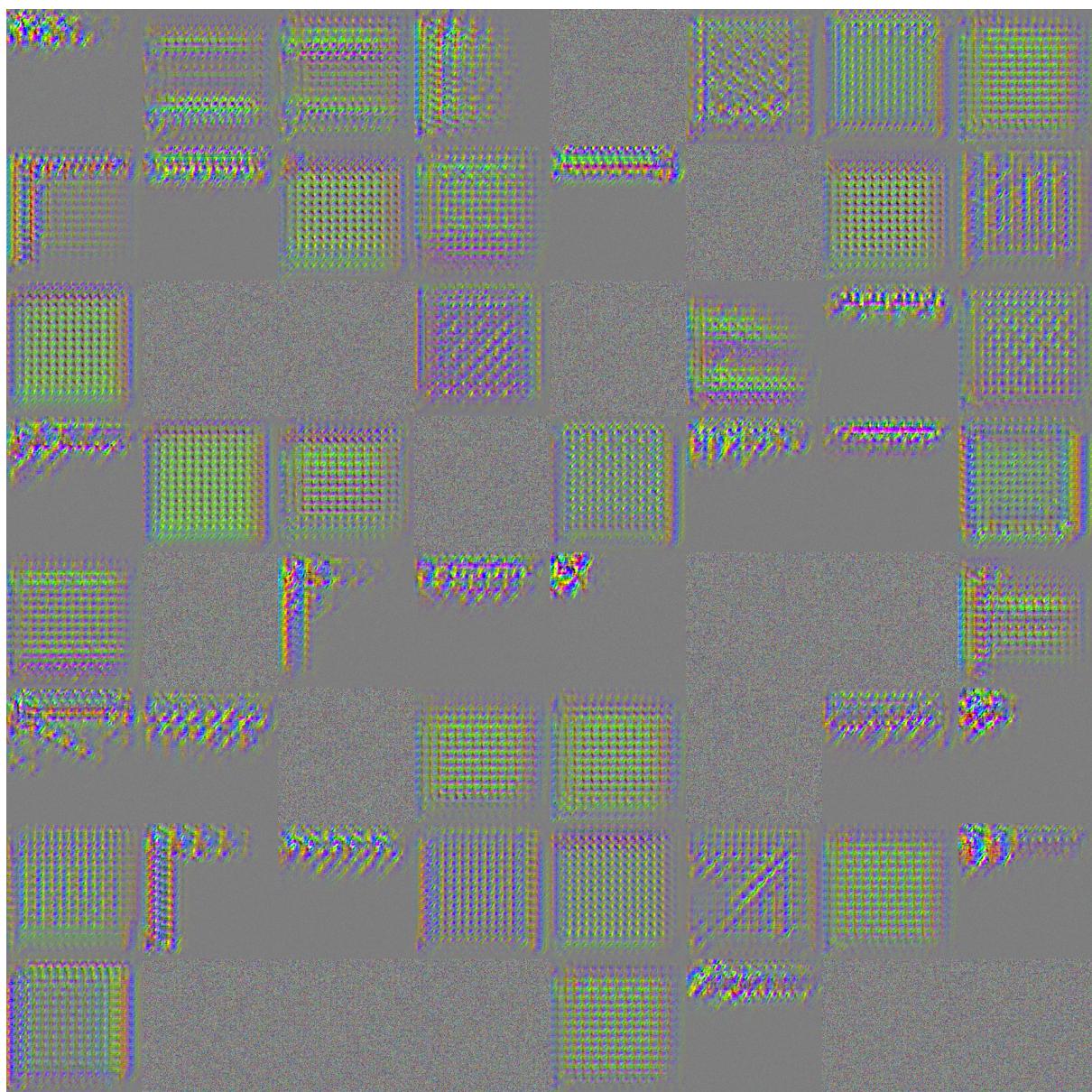
Ce modèle comprend des couches de convolution avec la fonction d'activation ReLU, de MaxPooling (couche de mise en commun qui réduit la taille spatiale de la représentation en prenant le maximum dans chaque région), de Dropout (couche de régularisation qui désactive aléatoirement un certain pourcentage des neurones de la couche précédente pendant l'entraînement afin d'éviter le surapprentissage), une couche Flatten (qui convertit les données en sortie des couches précédentes en un vecteur unidimensionnel pour être utilisé comme entrée pour les couches denses) et une couche Dense (qui produit les sorties finales

du modèle grâce à la couche d’activation “softmax” qui va classer les images dans les deux catégories de notre dataset).

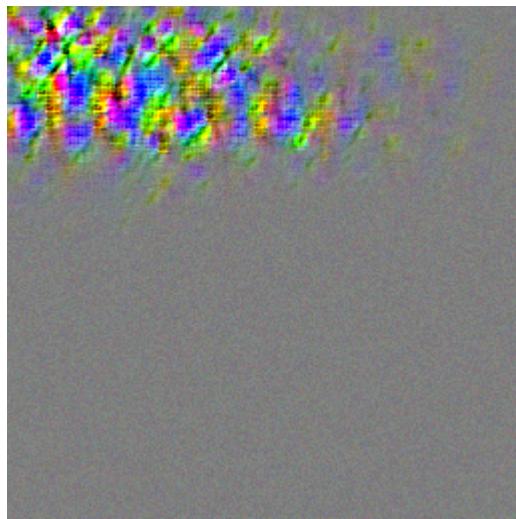
Les résultats de ces entraînements sont très prometteurs pour la tâche de classification en elle-même : le ResNet50 était arrivé à 95% d’accuracy, et le réseau fait maison atteint les 81%. Les modèles ont été très performants, nous pouvons en déduire qu’ils sont tous les deux utilisables dans le cadre d’une classification de manuscrits.

3.3. Visualisation des filtres

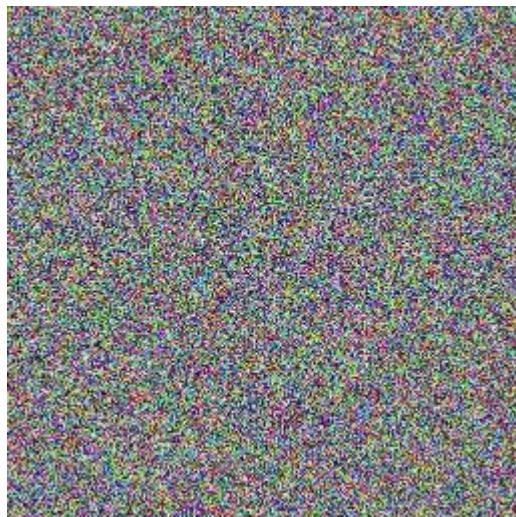
Voici l’output de la visualisation de 64 filtres de la couche conv3block4_out :



Nous zoomons sur le premier filtre de la couche. A première vue, celui-ci n'est pas interprétable par l'œil humain. Nous n'arrivons pas à y trouver un pattern pour expliquer la répartition de la feature map :



Pour la visualisation du modèle homemade, nous avons pris la couche "conv10" qui est la dernière couche convulsive. Ce résultat est encore moins interprétable, ou potentiellement non-pertinent car cette image ne semble être que du bruit :



4. Conclusion :

Si nous avons réussi à classifier nos données de manière efficace, l'interprétation des features est malaisée. Du fait du manque de littérature à ce sujet et de l'absence de projets similaires,

nous ne pouvons pas comparer nos résultats à ceux de classifications semblables pour mieux comprendre les filtres que nous obtenons.

Ce travail nous a été bénéfique car il nous a permis de nous confronter à des problèmes de compatibilité de packages et donc d'apprendre à les régler. Nous avons également travaillé avec un GPU pour la première fois.

“It's the not the Destination, It's the journey.”

-Ralph Waldo Emerson

(ft. “Bon bah euh, tant que vous gardez le sourire, hein :)”

- Chahan Vidal Gorène, 2023)



#DropTheMike(Kestemont)

