PROJET KAGGLE : CHEST X-RAY IMAGES (PNEUMONIA)

Bénédicte Noblet - Laura Xénard M2BI - 27/10/2021

Ayant suivi le module concerné en 1ère année de Master, notre intérêt s'est porté sur le sujet *Chest X-Ray* comportant de l'analyse d'images. Le jeu de données suggéré est celui proposé par Paul Mooney sur la plateforme Kaggle (https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia).

Le contexte de ce jeu de données se situe en pratique clinique en soutien à l'équipe médicale. Des images de radiographies aux rayons X de thorax de patients (agés de 1 à 5 ans) pris en charge au Centre Médical Femme et Enfants de Guangzhou (Chine) sont mises à disposition afin de déterminer si le patient est atteint d'une infection pulmonaire (classification sous le terme de *PNEUMONIA*) ou non (groupe *NORMAL*). Les travaux initiaux ont été publiés dans la revue *Cell* (Kermany *et al.*, 2018).
Les images proposées pour les individus atteints de pneumonie peuvent être divisées en deux catégories selon le type identifié par des experts : bactérien ou viral. Dans notre étude, nous nous sommes limitées pour une première approche à la discrimination du statut malade ou non du patient.

EXPLORATION DES DONNÉES

Le téléchargement des données, disponibles sur la plateforme Kaggle après inscription, se révèle plus volumineux que le gigabit de données annoncé. En effet, l'archive contient le dossier d'images en trois exemplaires dont deux issus d'un travail sous environnement MacOS. Le dossier unitaire se décompose en trois sous-dossiers nommés *train*, *test* et *val*, pour les trois échantillonnages de données requis dans une phase d'analyse de données supervisée. Chacune de ces catégories est composée de dossiers NORMAL et PNEUMONIA.

La répartition effective du nombre d'images disponible dans chaque dossier et pour chaque sous-dossier est montrée dans la table ci-contre. Les effectifs proposés dans l'échantillon de validation (*val*) étant très faibles (8 images par catégorie à discriminer), nous avons affecté ces données à l'échantillon d'apprentissage (*training*) par la suite. Un sous-échantillon de validation est créé à l'aide de l'option `validation_split` directement en phase d'apprentissage (`model.fit()`).

class_str	NORMAL	PNEUMONIA
subset		
test	234	390
train	1341	3875
val	8	8

Nous observons par ailleurs le déséquilibre important entre les deux catégories dans le jeu de données d'apprentissage avec 25,78% d'images non pathologiques pour 74,22% d'images de pneumonie.

Une analyse globale des caractéristiques des 5856 images du dossier permet de mettre en évidence une grande disparité des tailles d'images, variant de 127 pixels (plus petite hauteur) à 2916 pixels (plus grande largeur), malgré un format d'image unique (`jpeg`). De même, la proportion largeur par hauteur

n'est pas harmonisée : quelques formats portraits (ratio minimal à 0,84) sont présents aux côtés de nombreux formats paysages (ratio maximal à 3,38). La moyenne (`np.mean()`) des ratios est de 1,443.

En ce qui concerne la couleur des images, nous observons deux catégories : nuances de gris (1 canal, 5573 images soit 95% environ de l'ensemble) et couleurs RGB (*Red Green Blue*, 3 canaux, 283 images). La profondeur de l'image est déterminée par la valeur de dimensions d'un objet image (`np.ndim`) et est validée par le nombre de canaux donné en dernière position du tuple retourné par *`np.shape`*.

La comparaison observable en figure 1 de la distribution de 3 paramètres statistiques pour les valeurs prises par les pixels au sein des images (importées en Noir et Blanc) permet de valider l'équivalence des images entre lots entraînement et test. Les échantillons se situent dans le même espace de données, un modèle établi sur le premier peut être repris pour analyser le second.

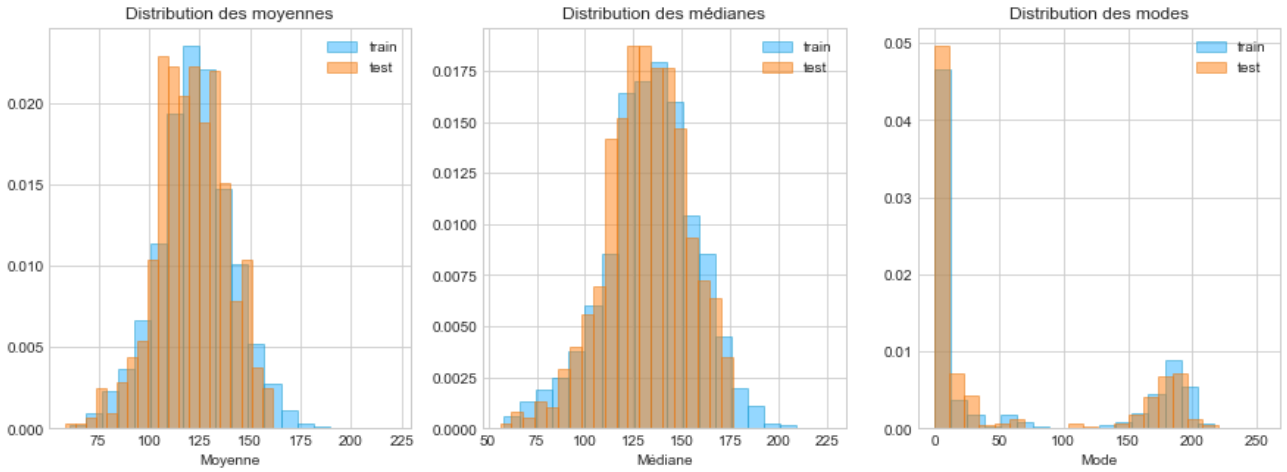


Figure 1 : Distribution des valeurs moyennes, médianes et des modes pour les valeurs de pixels des images étudiées. En bleu, l'échantillon d'apprentissage (*n*=5232), en orange l'échantillon de test (*n*=624).

En se basant sur la nomenclature renseignée par Kermany *et al.*, nous observons que plusieurs images existent pour un même patient : 1 à 16 images par patient sauf 30 images pour *person23_bacteria*. En parallèle, parmi les personnes atteintes de pneumonie (nomenclature *person*) un numéro de patient peut correspondre à une infection bactérienne comme à une infection virale comme c'est le cas avec les numéros 23 et 30 (3 images *bacteria* pour 1 image intitulée *virus*). En l'absence de données annexes, telle que la temporalité des radiographies, et d'expertise pour la distinction manuelle des images, nous poserons deux hypothèses de travail. La première suppose qu'une seule image est prise par journée et par patient (économiquement vrai, incertain en pratique). La seconde simplifie le problème en considérant qu'un patient ne peut être observé avec deux types d'infections différentes (pourtant pas impossible d'un point de vue biologique).

L'inégalité de représentation des patients dans l'échantillon d'apprentissage (*train* et *val*) pourra ainsi être corrigée avec une pondération du poids des images pour qu'un patient ne compte que pour 1 fois dans le réglage des poids du modèle prédictif.

Outils et méthodes

Modalité d'importation des images en entrée d'analyse

De nombreuses modalités de récupération des images sous Python pour analyse existent. Parmi celles-ci, deux ont été utilisées : la création d'un *array* avec la librairie `numpy` à partir d'une liste de

chemins de fichiers et le recours à un générateur à l'aide de la librairie `keras`. La première approche utilise alternativement les modules `Image` et `ImageOps` de la librairie `PIL` (Pillow) ou directement la librairie `matplotlib.image`, basée sur la librairie `PIL`.

Dans les deux cas, les données d'entraînement ont été mélangées avant chargement , méthode `pandas_dataframe.sample(frac=1)` et `generator.flow_from_directory(shuffle=True)` respectivement, pour assurer une homogénéité des lots (*mini-batches*) lors du réglage des poids. Les données de test ne sont à l'inverse pas randomisées, l'évaluation du résultat se faisant individuellement et sans échantillonnage.

L'utilisation d'un réseau de neurones comme d'une Analyse en Composantes Principales (ACP) nécessite l'harmonisation des dimensions des individus étudiés (ici, les images). Par souci de simplicité, notamment pour les architectures de réseaux par convolution (CNN), nous choisissons d'importer les images avec une définition carré, à l'image également de l'exemple réalisé en cours sur les données MNIST. Le retrait de pixels (de données en général) étant préférable à l'extrapolation de données manquantes, une taille minimale d'image est recherchée. Quatre images sont affichées avec cinq résolutions différentes (256x256, 128x128, 96x96, 64x64 et 32x32) pour s'assurer visuellement que le format retenu reste lisible. En deçà de 96x96, l'image perd visiblement en détails et la plus petite dimension de notre lot d'images étant de 127 pixels (hauteur), nous réglons à 96x96 le paramètre `image_size` de la méthode `resize()` de la librairie Pillow. La méthode de redimensionnement est laissée par défaut à une interpolation *nearest-neighbour*.

Analyses de données et réseaux

L'analyse en composantes principales est réalisée avec la librairie *Scikit-learn* (`sklearn`), plus précisément la fonction PCA du module `decomposition`. Les réseaux sont créés à l'aide de la librairie `keras`. Deux fonctions de *callbacks* sont mises en place pour la phase d'apprentissage du réseau avec `keras.callbacks` : `EarlyStopping` permet un arrêt adéquat lorsque la `val_loss` remonte sur 3 époques et restaure les poids avec le meilleur résultat, `ReduceLROnPlateau` opère lorsque la `val_loss` ne varie plus sur 5 époques pour réduire de 10% la valeur de *learning rate*.

Le suivi de l'apprentissage du réseau est réalisé à l'aide d'un fonction de `loss` de type `crossentropy` (`binary` ou `categorical`), le modèle mis en place visant une classification en catégories discrètes. La valeur de juste *accuracy* est prise avec précaution du fait du déséquilibre observable dans les effectifs des deux groupes étudiés (NORMAL et PNEUMONIA). En complément, les score F1 et Mathews Correlation Coefficient (MCC) sont considérés en phase de test. Ces fonctions ainsi que celle pour créer la matrice de confusion proviennent du module `metrics` de `sklearn`.

La librairie `pyplot` de `matplotlib` permet de représenter l'ensemble des données générées pendant notre projet et présentées ici.

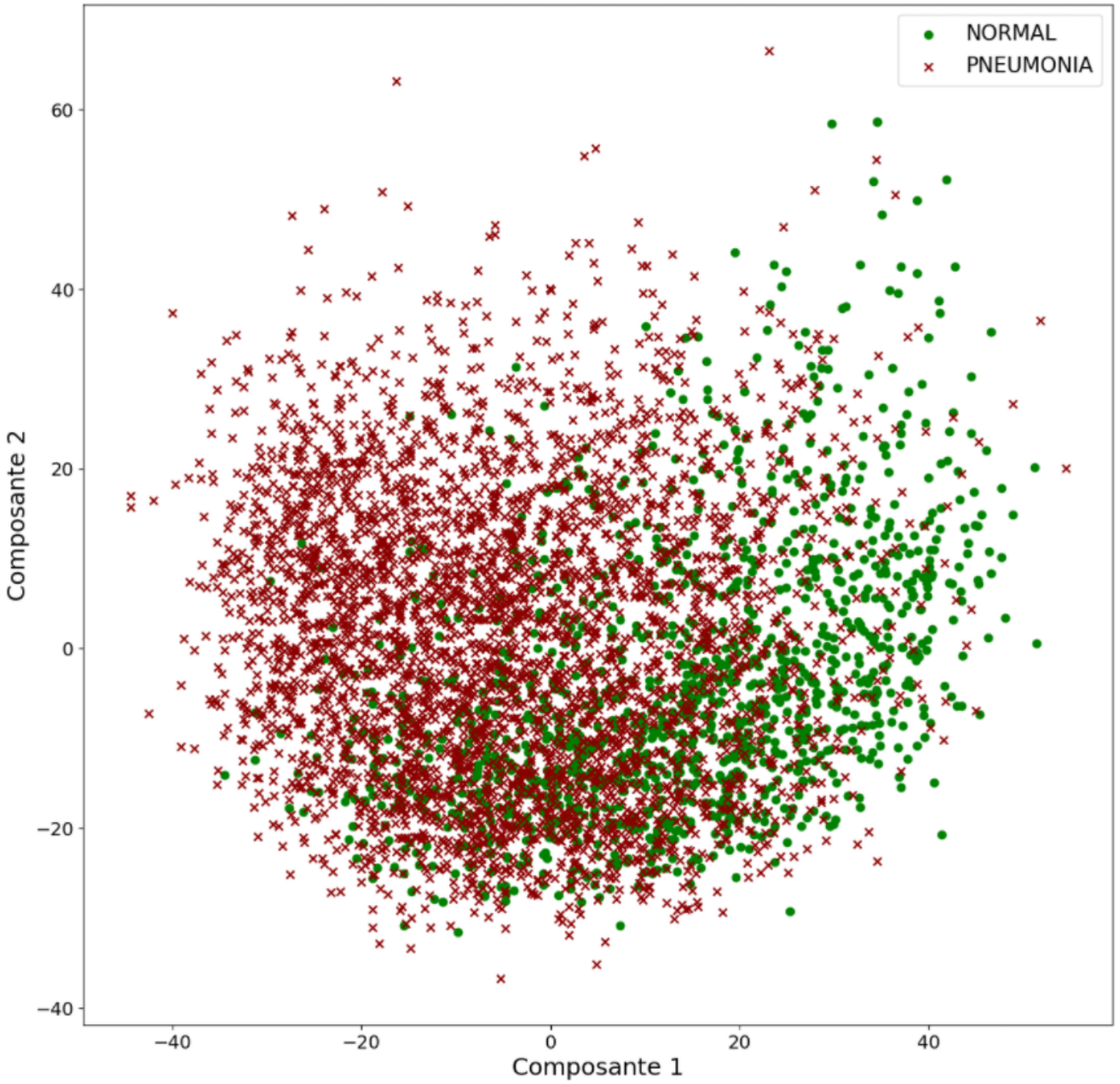
Analyse en composantes principales et réseaux denses

Dans cette partie, on propose de réduire la dimensionnalité des données via une ACP puis de montrer l'impact de cette réduction sur des réseaux denses simplistes.

ACP ET PREMIER MODÈLE

L'ACP s'effectue sur des images aplaties. Elle a été réalisée sur les images redimensionnées en 500x500 d'une part et en 96x96 d'autre part. Dans les deux cas, la projection sur les deux premières composantes est visuellement identique.

Figure 2 : Projection des composantes 1 et 2 pour les images en 96x96 sur ensemble *train*. Ces 2 composantes expliquent une variance cumulée de 31%.



On peut voir que même si les deux classes NORMAL et PNEUMONIA ne sont pas séparables, une certaine tendance au regroupement se dégage.

		% cumulé de variabilité conservée									
		20	30	40	50	60	70	80	90	95	100
Taille des images	500x500	3	4	7	11	18	34	88	348	935	5205
	96x96	3	4	6	10	15	27	60	199	455	5205

Un premier modèle très simple, avec un réseau uniquement constitué de deux couches denses, a été entraîné sur les 455 premières composantes issues de l'ACP sur les images en 96x96. Celles-ci permettent de conserver 95% de variabilité. L'optimiseur utilisé est NADAM afin de bénéficier non seulement d'une prise en compte de l'inertie mais aussi de l'accélération de Nestorov.

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 455)]	0
dense_9 (Dense)	(None, 128)	58368
dropout_6 (Dropout)	(None, 128)	0
dense_10 (Dense)	(None, 64)	8256
dropout_7 (Dropout)	(None, 64)	0
dense_11 (Dense)	(None, 2)	130

Total params: 66,754
Trainable params: 66,754
Non-trainable params: 0

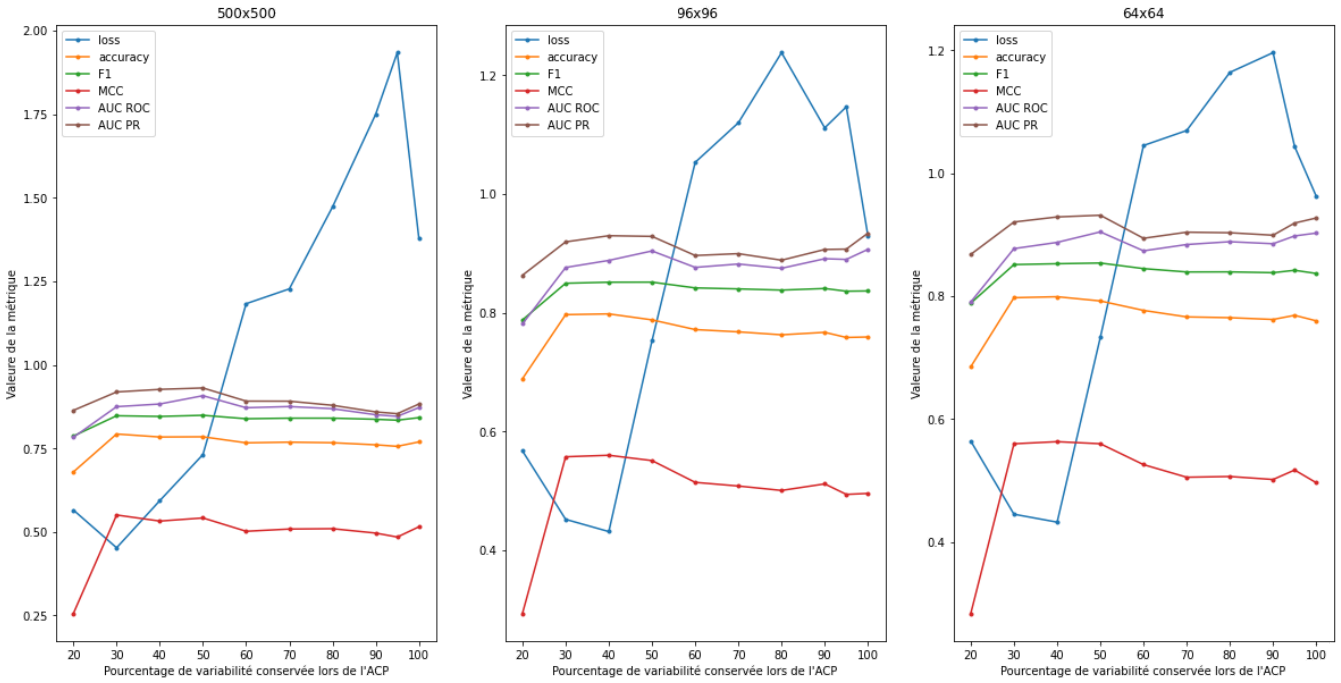
Ce réseau a ensuite été modifié progressivement afin de trouver le modèle le plus performant en validation (représenté ci-contre) : changement du nombre de couches, de leur taille, ajout de dropout, variation du taux de dropout.

En validation, ce modèle permet d'obtenir une *loss* de 0,0772 et une *accuracy* de 0,9733 en 9 époques et de manière quasi-instantanée. En test, le MCC est de 0,5189 et le F score de 0,8425.

INFLUENCE DU POURCENTAGE DE VARIABILITÉ CONSERVÉ

Le modèle obtenu ci-dessus a été repris et appliqué sur 3 formats d'images différents (500x500, 96x96, 64x64) et 10 pourcentages différents de variance cumulée conservée en sortie d'ACP.

Figure 3 : Influence du pourcentage de variabilité conservée après ACP sur les métriques du modèle en test.



On observe très clairement que la même tendance se dessine quelle que soit la taille des images. La *loss* est faible lorsque le pourcentage de variabilité est bas avec un minimum aux alentours de 30-40%, puis elle augmente avec le pourcentage et atteint un maximum vers 80-90% de variabilité. Pour les autres métriques, les valeurs sont faibles à pourcentage bas, augmentent pour atteindre un plateau maximal entre 30 et 50% puis redescendent un peu et varient plus légèrement.

Dans la mesure où l'on cherche à obtenir une *loss* la plus faible possible et une *accuracy*, un F score, un MCC ainsi qu'une aire sous les courbes ROC et PR les plus grandes possibles, il apparaît plus pertinent de ne conserver que 40% de la variabilité. De plus, les valeurs de métriques sont très similaires d'une dimension à une autre, et il vaut mieux donc travailler avec la plus petite taille d'image afin de limiter le nombre de paramètres du réseau ainsi que le temps d'exécution de l'ACP comme de l'apprentissage.

Sans surprise, lorsque l'apprentissage est effectué sans ACP préalable, le nombre de paramètres

explose et le modèle est bien moins performant. Sur des images en 500x500, seulement 3 images en test ne sont pas classées en PNEUMONIA. Pour les deux autres tailles, les prédictions sont un peu meilleures mais restent très déséquilibrées en faveur de PNEUMONIA.

Modèle final

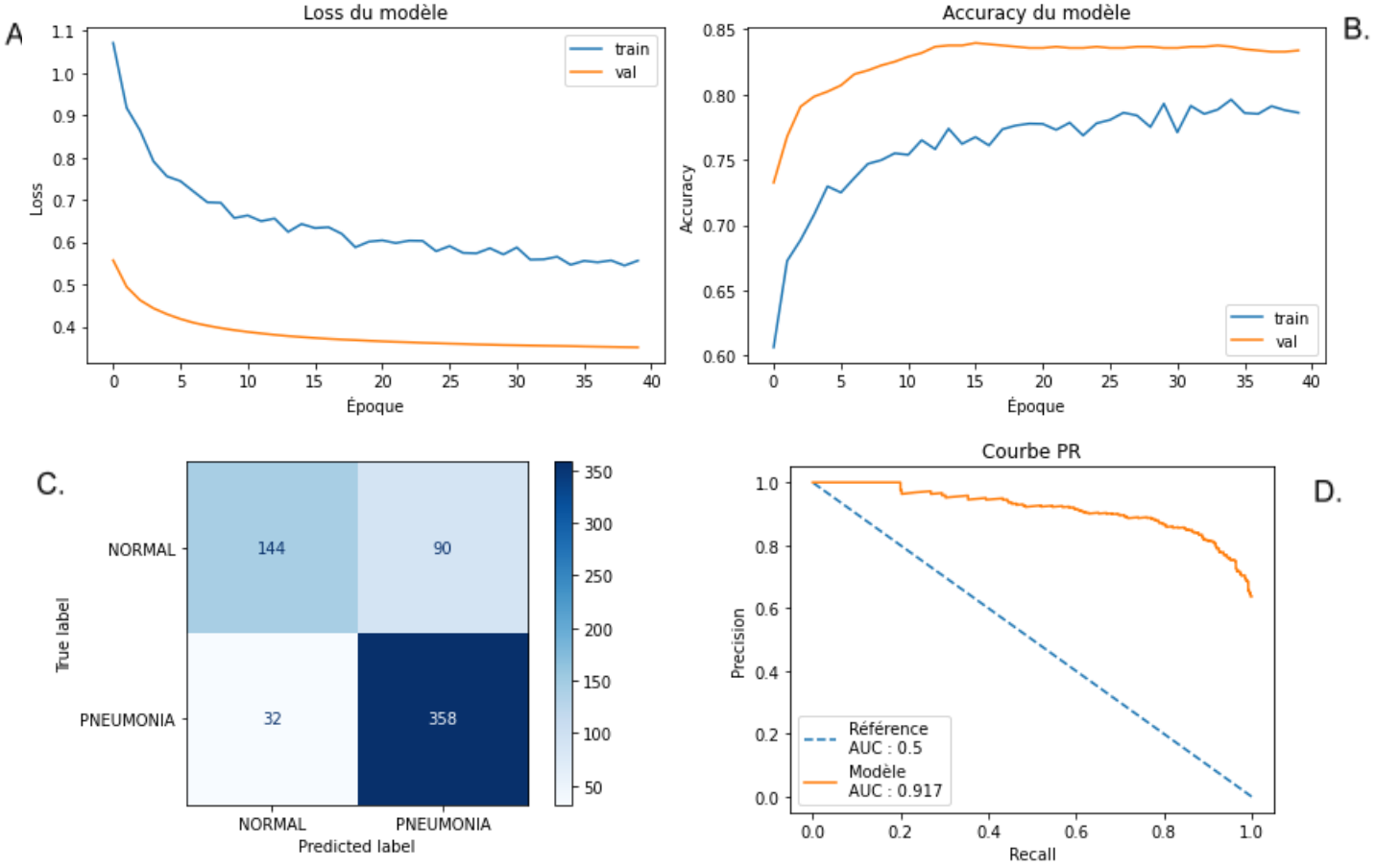
On reprend donc le modèle précédent et on lance un apprentissage sur les 6 premières composantes issues de l'ACP des images en 64x64 (conservation de 40% variabilité cumulée).

Les résultats obtenus sont encourageants mais les courbes d'*accuracy*, notamment en validation, sont très ondulantes. On divise donc par 10 le *learning rate* (passage de 0,001 à 0,0001) afin de ralentir l'apprentissage et stabiliser l'*accuracy*. On obtient finalement le modèle suivant ci-contre, plutôt performant pour un modèle ne comptant que 9282 paramètres et à l'exécution quasi instantanée, mais au prix d'un pré-traitement des données plus lourd. On atteint 80% de bien prédit.

Layer (type)	Output Shape	Param #
input_13 (InputLayer)	[(None, 6)]	0
dense_36 (Dense)	(None, 128)	896
dropout_24 (Dropout)	(None, 128)	0
dense_37 (Dense)	(None, 64)	8256
dropout_25 (Dropout)	(None, 64)	0
dense_38 (Dense)	(None, 2)	130

Total params: 9,282
Trainable params: 9,282
Non-trainable params: 0

Figure 4 : Performance du modèle dense final. **A.** et **B.** *Loss* et *accuracy* en train et validation. **C.** Matrice de confusion en test. **D.** Courbe Precision-Recall en test.



CNN

Le premier réseau convolutionnel de neurones a été repris du modèle découvert en séance de Travaux Pratiques sur les données MNIST. La couche d'*input* la dimension de l'image d'entrée 28x28 est remplacée par celle utilisée, à savoir 96x96, et la taille du *batch* est adaptée pour assurer l'existence de 100 lots environ, constitué chacun de 52 images, dans une époque. En phase d'apprentissage, les courbes de *loss* et d'*accuracy* présentent des profils satisfaisants comme visible en figure 5.

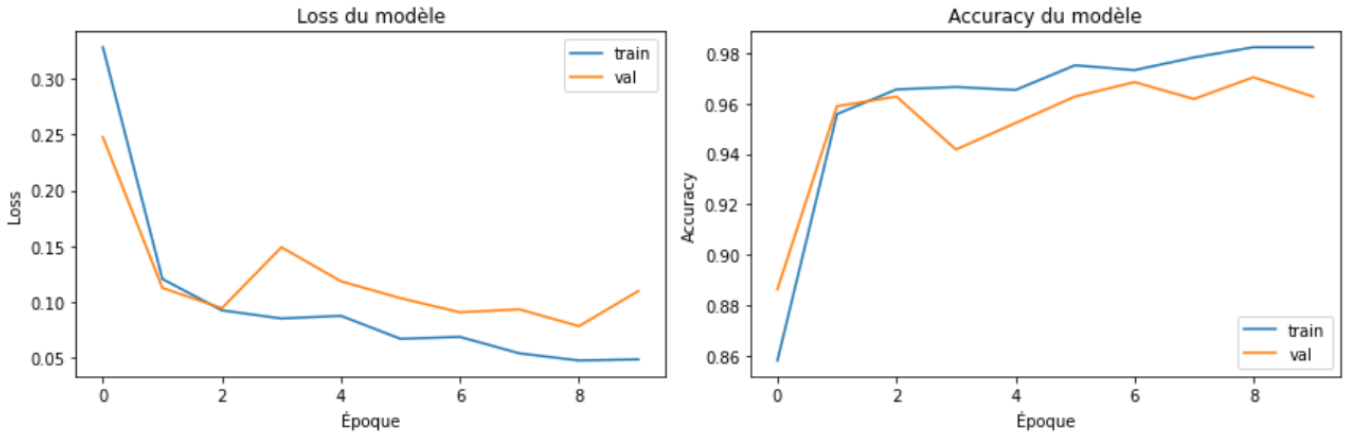


Figure 5 : Courbes de *loss* et d'*accuracy* obtenues en phase d'apprentissage pour le modèle utilisé en Travaux Pratiques pour les données MNIST et un `batch_size` de 52.

Le modèle semble ainsi bien se régler sur l'échantillon d'apprentissage avec une diminution de la valeur de perte atteignant 0,05. Toutefois, la valeur de *loss* obtenue pour le sous-échantillon de validation (`validation_split` de 20%) ne diminue plus dès la deuxième époque. Le surapprentissage est visible au niveau de la valeur de justesse (*accuracy*), celle de l'échantillon *train* est supérieure de 2% à celle observée en validation. Les résultats obtenus avec ce premier essai en phase de *test* sont en pratique peu concluants : la matrice de confusion montre que la quasi-totalité des images sont classées en *PNEUMONIA* et seuls 38 vrais négatifs sont détectés, soit une attribution juste à 69,4% environ. Par ailleurs, la *loss* calculée dans cette même phase atteint la valeur de 417.

Pour prévenir le surapprentissage, la première approche a consisté à ajouter les deux fonctions d'arrêt anticipé (*callback*) au sein de la méthode `model.fit()` sur ce même modèle remis à son état initial (`model.set_weights(weights)`, l'objet `weights` étant issu de `model.get_weights` après construction du modèle). Si 6 vrais négatifs de l'échantillon *test* sont détectés en plus, les valeurs d'*accuracy* comme de *loss* demeurent inchangées.

Un essai d'architecture arbitraire a été réalisé à l'aide de 3 couches convolutionnelles 2D avec glissement des filtres (*kernels*) de 2 en 2 pixels (`stride = 2`), d'une couche DropOut et d'une couche d'aplatissement avant la couche dense de sortie. Alors que la taille du batch a été remontée à la valeur de 128 utilisée en Travaux Pratiques, les résultats obtenus avec l'échantillon *test* se sont révélés similaires à ceux précédents observés : *accuracy* de 69,47% et *loss* de 249 malgré les deux fonctions de *callbacks* utilisées.

La seconde approche envisagée a consisté à réaliser plusieurs variations unitaires du réseau initialement utilisé en séances de Travaux Pratiques. Les poids initiaux ayant changé depuis le premier essai, ce modèle est testé à nouveau avec la valeur de *batch* proposée en séance (`batch_size = 128`). Les courbes d'apprentissage (*loss* comme *accuracy*) semblent plus stables qu'avec une taille de lot

réduite (`batch_size = 52`) et l'écart de justesse d'attribution entre les deux lots *train* et *val* est moindre, comme visible en figure 6.

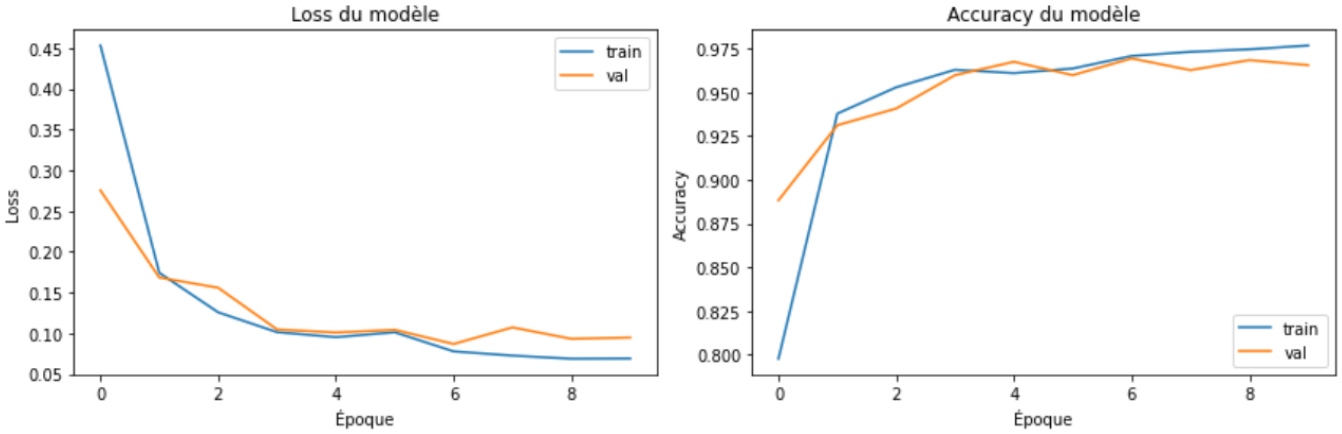


Figure 6 : Évolution des métriques lors de la phase d'apprentissage du modèle de Travaux Pratiques MNIST (même `batch_size` de 128) avec 2 fonctions `callbacks`, sans paramétrage `class_weight`.

En phase de test, les mesures qualitatives sont légèrement améliorées avec une *loss* à 127,8 et une valeur d'*accuracy* de 75,16%. Deux fois plus de vrais négatifs sont ainsi détectés (figure 7).

A.

```
Model: "Model_du_cours_exact"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 96, 96, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 94, 94, 32)        320
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 47, 47, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 30976)             0
_________________________________________________________________
dropout (Dropout)            (None, 30976)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 61954
=================================================================
Total params: 80,770
Trainable params: 80,770
Non-trainable params: 0
_________________________________________________________________
None
```

B.

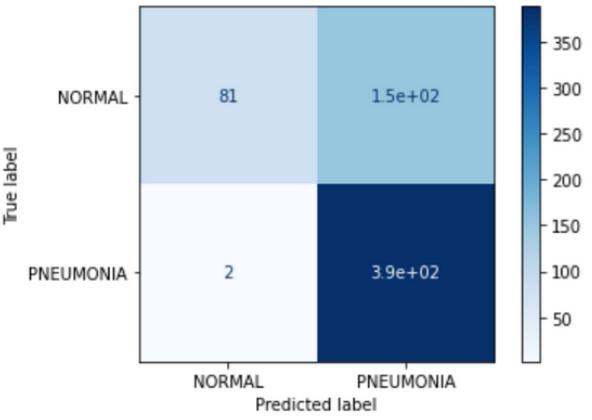


Figure 7 : Modèle exact issu de la séance de TP et résultats pour les images de radiographie. **A.** Architecture du réseau utilisé . **B.** Représentation de la table de confusion des valeurs de *test* attribuées.

Les essais suivants voient varier plusieurs paramètres, un à la fois, portant sur la correction de la valeur de loss (`class_weight` et `sample_weight`), l'utilisation d'optimiseur alternatif à Adam (Nadam, RMSProp), la réduction du taux d'apprentissage de l'optimiseur Adam (0,0001 vs 0,001 par défaut), l'utilisation de la `binary_crossentropy` ou encore la variation de la taille du *batch* (56 et 256 par rapport à 128). Les valeurs de métriques obtenues en généralisation sont reportées dans la figure D ci-dessous, les remarques en bas de tableau concernent la phase d'apprentissage.

				with class_weight						
Variation	-	class weigth	sample weight	nadam	RMSprop	Adam (LR/10)	binary_cross entropy loss	-	batch_size = 56	batch_size = 256
Loss	167,87	153,13	189,86	218,61	221,39	134,57	81,16	179,83	151,85	106,06
Accuracy	0,7516	0,7644	0,7196	0,7356	0,6907	0,7003	0,8205	0,702	0,7388	0,788
True Negative	81	81	81	81	71	71	71	71	71	71
MCC	0,4842	0,4842	0,4862	0,4862	0,4493	0,4493	0,4493	0,4493	0,4493	0,4493
F1	0,8335	0,8335	0,8335	0,8335	0,8247	0,8247	0,8247	0,8247	0,8247	0,8247

Note		curve splitting		atypical curve					

Figure D : Valeurs de métriques obtenues en phase de test pour les différentes variations du réseau modèle utilisé (2 lots de Conv2D - MaxPooling, `batch_size` de 128, *loss* de type `categorical_crossentropy` et optimiseur `adam`). Coloration des valeurs selon l'importance par ligne (la coloration verte est recherchée). Les remarques portent sur les courbes de loss et d'accuracy en phase d'apprentissage.

L'ajout du paramètre `class_weight` permet de corriger le déséquilibre de représentation des deux classes. Les courbes de *loss* et *validation_loss* sont ainsi visuellement séparées, ce qui permet de mettre en évidence le phénomène de surapprentissage ou surajustement du modèle aux données d'apprentissage. L'utilisation de la `binary_crossentropy` améliore les valeurs de *loss* et d'*accuracy* obtenues mais ne change pas la répartition en phase de test. Par ailleurs, la taille du batch size est un paramètre important, une valeur plus grande assure une meilleure estimation des paramètres sans pour autant influencer la phase de test.

D'autres tests portant sur la variation du nombre de couches, l'ajout de couches de *batch normalization* ou encore de couche DropOut ont également été réalisés. Toutefois, ceux-ci ont été effectués un autre jour par rapport à ceux présentés ci-dessus et le réseau dans sa forme basale présente de meilleures performances (tous les paramètres ci-dessus à l'exception de la *loss* qui est équivalente). Au final, il semble que la graine de génération des poids initiaux soit un point clé des performances d'un modèle de réseaux de neurones artificiels, dans le cadre des essais réalisés.

Conclusion

Dans cette étude deux types de réseaux ont été utilisés pour la discrimination d'images pathologiques (pneumonie) et saines de radiographies de poumons : les réseaux denses dits *fully-connected* (FC) et les réseaux convolutionnels (CNN) avec couche à 2 dimensions.

Le recours à l'analyse en composantes principales avant réseau dense permet d'établir une bonne prédiction en généralisation. A l'inverse, l'architecture CNN, bien que destinée au traitement d'images, n'a entre nos mains pas permis d'obtenir une discrimination correcte. En effet, celle-ci est biaisée par l'enrichissement en données de patients atteints de pneumonie. Il serait souhaitable de recourir à de la *data augmentation* pour contrer ce problème sur le modèle de certains auteurs reconnus sur la plateforme Kaggle.

Alternativement, il serait intéressant de changer de valeur de référence pour le suivi de la phase d'entraînement en réglant si possible la méthode `fit()` non sur l'*accuracy* mais sur le score F1 ou la valeur de MCC.

On notera par ailleurs que la reproduction des résultats d'un réseau est en pratique difficile (données non montrées sur nos réseaux comme sur les propositions d'auteurs vus sur le web) compte tenu de

l'initialisation des poids à la création du modèle. Pouvoir maîtriser le facteur aléatoire (la graine) semble être un facteur essentiel de la réussite d'une mise au point de modèle de réseaux de neurones.

Bibliographie

Kermany DS., Goldbaum M., Cai WJ., *et al.* (2018). Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*. 172(5): 1-10.