Rapport de TX n°6115

Printemps 2019

Prédire les formes des globules rouges par machine learning

DIZIER Martin
GAROIS Sevan

Contents

1 Introduction . . . 3
2 Cadre de l'étude . . . 4
2.1 Objectifs . . . 4
2.2 Support d'étude . . . 4
2.3 Données d'entrées . . . 4
2.4 Outils utilisés . . . 5
3 Cellules étudiées . . . 5
4 Réseau de neurones convolutifs . . . 6
4.1 Présentation . . . 6
4.2 Couche de convolution . . . 7
4.3 Couche de pooling . . . 8
4.4 Fonction d'activation . . . 8
4.5 Couche *fully-connected* . . . 8
5 Pre-processing des données . . . 10
5.1 Définition . . . 10
5.2 Data pre-processing . . . 10
5.3 Image pre-processing . . . 10
6 Modèle . . . 12
6.1 Structure du CNN . . . 12
6.2 Entraînement du CNN . . . 13
7 Résultats . . . 16
7.1 Prédictions . . . 16
7.2 Critères de performance . . . 16
7.3 Visualisation des prédictions . . . 17
7.4 Détermination des intervalles des classes . . . 18
7.5 Approche "naïve" . . . 18
7.6 Recherche des paramètres optimaux . . . 23
7.7 Comparaison . . . 31
8 Perspectives . . . 33
8.1 Difficultés rencontrées . . . 33
8.2 Amélioration du réseau . . . 33

8.3 Conclusion 35

1 Introduction

Le machine learning (apprentissage automatique) a connu un grand essor ces dernieres années, avec l'émergence du Deep Learning dans les années 2010. Bien que les fondemments reposent sur des principes statistiques formalisés dans la seconde moitié du 20ième siècle; la puissance de calcul grandissante et l'accumulation massive de données ont décuplée les possibilités d'apprentissage statistique. Donner la capacité aux ordinateurs d'apprendre à partir des données; leur permettant de résoudre des tâches sans y être explicitement programmés pour chacunes, tel est le but du machine learning.

Le Deep Learning et la classification deviennent de plus en plus des solutions à des problèmes récurrents dans de nombreux domaines. La fiabilité relative de ces modèles de prédictions peuvent apporter un gain de temps sans précédent dans les études et recherches menées. Que ce soit dans la finance, dans la biologie, le médical, l'industrie etc ... Le machine learning se développe quelque soit le domaine où il y a des données en présence.

En biologie, on peut citer plusieurs applications comme la réaction des cellules aux stimulis comme les médicaments, la classification des réponses musculaires en réaction à des stimulis électriques enregitrés par capteur, et ici pour notre cas la classification des cellules de globules rouges en écoulement.

2 Cadre de l'étude

Notre projet s'inscrit dans une étude générale pour étudier les propriétés mécaniques des globules rouges.

2.1 Objectifs

L'objectif de la TX est d'appliquer la technologie du *machine learning* pour prédire les formes de modèles de globules rouges en écoulement. Il faut donc concevoir un modèle de prédiction capable de classer les images. Les performances et résultats de ce projet doivent être le plus haut possible pour envisager une étude des propriétés mécaniques après traitement et classification par notre modèle.

2.2 Support d'étude

Le support d'étude [1] est un document de recherche rédigé par Alexander Kihm, Lars Kaestner, Christian Wagner et Stephan Quint de l'université de Saarland. Il met en exergue l'utilisation d'un réseau de neurones convolutif (CNN) pour la classfication des globules rouges étudiés dans leur laboratoire avec des conditions expérimentales différentes. Le rapport, les données fournies ainsi que le code est en libre acccès sur leur site.

Nous proposons alors notre implémentation du réseau de neurones convolutif présenté afin d'approcher leur étude et d'en analyser la pertinence.

2.3 Données d'entrées

Les données en entrée sont des images en niveau de gris avec une unique taille fixée à 90x90 pixels. Ces données sont disponibles grâce au laboratoire PLoS. Nous disposons de 2 jeux séparés en 3 classes. 8000 images constituent notre jeu d'entraînement et 3100 images forment notre jeu de test. Nous enlèverons 5% du jeu d'entrainement pour créer notre jeu de validation (utilisé pour valider la cohérence de l'entrainement du réseau).

2.4 Outils utilisés

Préférant le langage Python pour coder, nous exploitons, pour cette TX, les modules suivants:

- **Jupyter** C'est un notebook en cellule exploitant facilement les modules cités ci-dessous et permettant une visualisation des données et des graphes interessante.
- **Tensorflow** Outil très populaire d'apprentissage automatique créé par Google, sous Python. Sa notoriété nous permet de résoudre nos problèmes efficacement grâce à une grande communauté.
- **Keras** Utilisant Tensorflow en *backend*, cette bibliothèque permet d'interagir facilement avec les alogrithmes proposés par l'outil de Google et notamment les réseaux de neurones que nous utilisons pour notre cas d'étude.
- **Autres modules Python** scipy, numpy, scikit-learn, searborn, matplotlib, pandas, etc ...

3 Cellules étudiées

Il existe 2 types de géométrie stable de globules rouges : les croissants et les slippers. Leurs formes dépendent des conditions d'écoulement et des caractéristiques intrinsèques de la cellule.

Les croissants ont une forme caractérisé par un axe de symétrie. On les retrouve principalement lorsque l'écoulement est à une vélocité faible. Quant aux slippers, leur forme est asymétrique et ils sont observables lorsque la vélocité est forte.

Cependant, il existe une grande phase de transition durant laquelle on ne peut déterminer avec exactidude la forme du globule rouge, c'est ce qui sera appelé un sheared et formera la 3ème classe observable de cette étude. Leur forme présentant une forte variance, il est nécessaire d'opter pour un modèle de CNN à regression avec une optique de "tolérance" sur la classification.

4 Réseau de neurones convolutifs

4.1 Présentation

Le réseau de neurones à convolution (ou *CNN*) est un réseau de neurones à deux sections comportant chacunes des couches (*layers*) :

- Section d'extraction de caractéristiques
- Section entièrement connectée

La section d'extraction du CNN comporte plusieurs couches :

- Couche de convolution (filtrage)
- Couche de pooling
- Couche de correction ou d'activation

Ces couches permettent l'extraction de caractéristiques propres à chaque image. Ces caractéristiques sont des fragments de chaque image pouvant donner lieu à des similitudes ou des différences manifestes avec d'autres images du dataset.. La section entièrement connectée (*fully-connected*) est un réseau de neurones traditionnel permettant de renvoyer la réponse du réseau.

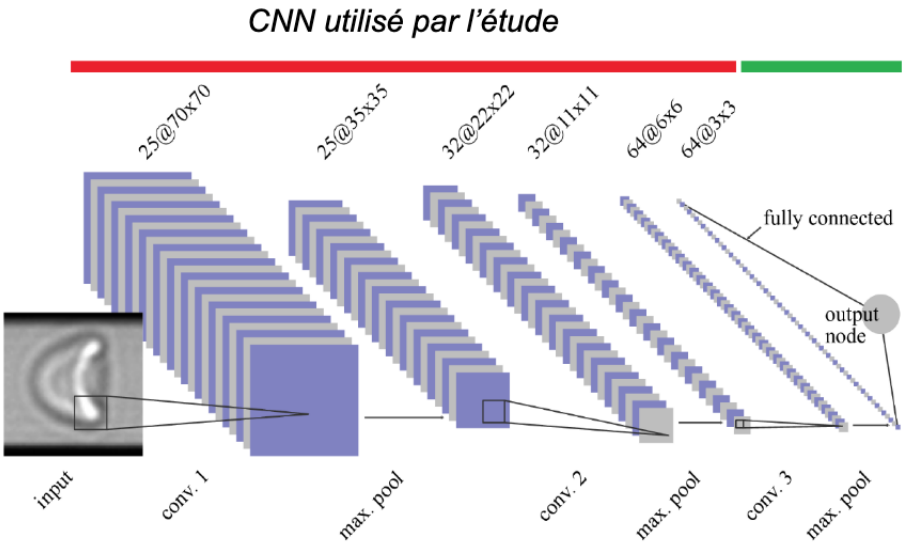


Figure 1: CNN utilisé par l'étude

Rouge : couches d'extraction de caractéristiques (features)
Vert : couches entièrement connectées donnant la réponse du réseau (réseau de neurones traditionnel)

4.2 Couche de convolution

Le but de cette couche est de détecter les caractéristiques de l'image de manière hiérarchique selon les différentes couches de convolution.

Par exemple, on peut voir qu'il y a dans le CNN de l'étude, 3 couches de convolutions. Chaque couche produit des images en sortie, appelées *features map*.

Le principe de cette couche est d'appliquer un filtre de traitement d'image (comme un flou gaussien par exemple) à l'image donnée en entrée. Il existe un grand nombre de filtres, que ce soit un filtre pouvant détecter les bords (*Edge Detection*) ou encore affiner l'image (effet *sharped*). Un filtre est une matrice qui permet d'appliquer une opération de convolution sur l'image. Cette opération linéaire effectue en quelques sorte une moyenne des caractéritiques d'une partie de l'image. Le filtre balaye ainsi l'image pour produire des *feature map*.

Figure 2: Illustration de la fonction d'un filtre

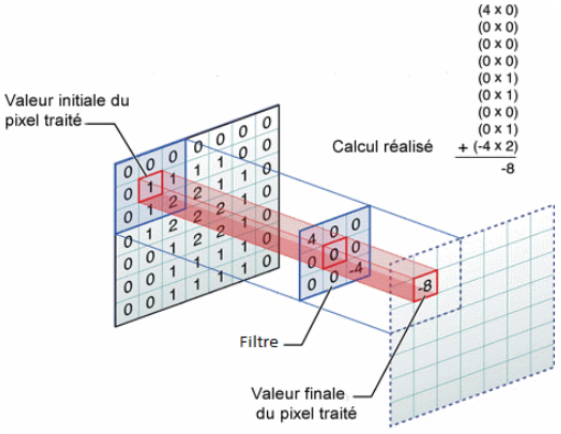


Figure 3: Illustration du principe de convolution

4.3 Couche de pooling

La couche de pooling est une forme de sous-échantillonnage de l'image. L'image d'entrée est découpée en une série de fenêtres de taille égale. On utilise le *max pooling* qui extrait le pixel d'intensité maximal dans chaque fenêtre.

Le pooling réduit ainsi la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau. Il s'insère entre deux couches convolutives successives et permet de réduire le sur-apprentissage.

4.4 Fonction d'activation

Il est usuel d'améliorer l'efficacité de tous les traitements du CNN en intercalant entre les couches de traitement, la couche de correction qui va opérer une fonction mathématique sur les valeurs de sorties. La plus connue et celle utilisée dans le réseau de l'étude est la fonction ReLU (Rectified Linear Unit) où $f(x) = max(0, x)$ celle-ci permet d'assurer une conformité mathématiques.

4.5 Couche *fully-connected*

C'est la couche finale du CNN, on a en entrée les *features map* que l'on va aplatir pour transformer ces matrices 3x3 (sortie de la partie d'extraction) en un vecteur colonne de pixels. On obtient en sortie la réponse de notre réseau. Cette couche est celle que l'on retrouve dans les réseaux de neurones classiques.

Dans l'étude, la réponse est un scalaire qui sera utilisée pour prédire la classe de l'image d'entrée du CNN. Cette réponse quantitative définie notre couche *fully-connected* comme un régresseur et non comme un classifieur où la réponse est directement la classe de l'image (réponse qualitative).

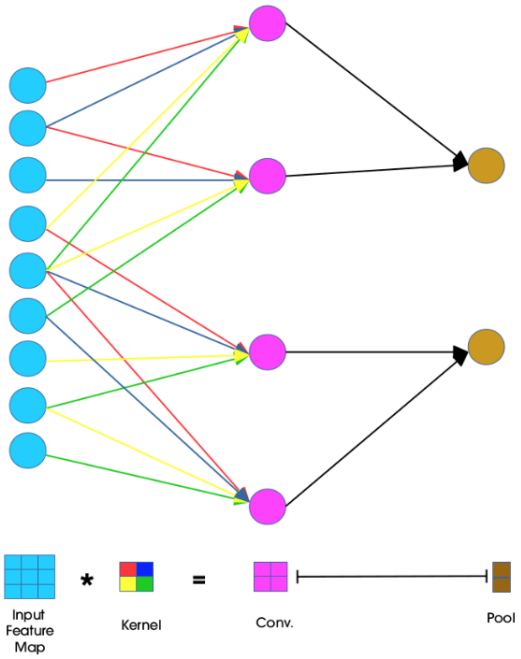


Figure 4: Représentation des couches d'un CNN sous la forme d'un réseau de neurones

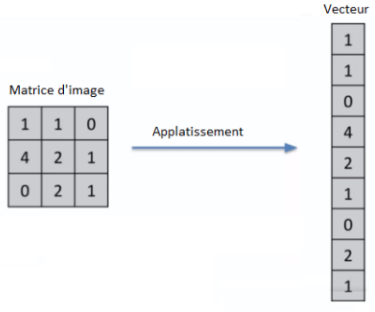


Figure 5: Applatissement pour le fully connected

5 Pre-processing des données

5.1 Définition

Le pre-processing est une phase dans le traitement des données. Elle est très importante car elle va permettre de transformer ou préparer les données pour avoir les meilleures performances avec les modèles ou algorithmes utilisés.

Dans notre problème, le *data processing* va consister à modifier et améliorer les images pour avoir des rendus plus interessants et plus importants. Cela passera par l'utilisation de filtre, pour mettre en exergue les contours de la cellule par exemple.

5.2 Data pre-processing

Definition : Data pre-processing

Pour faire du data pre-processing, il faut réaliser la création de *batch* qui sont le regroupement aléatoire d'éléments du jeu de donnée. Pour ce faire nous utilisons une fonction du module de pre-processing de Keras qui est *ImageDataGenerator*. Cette fonction nous permet de générer des batch d'entrainement, composés des images (tableaux de pixels) et de leur label associé. C'est cette formalisation des données qui permet d'appliquer d'entraîner modèle sans gêne.

Les labels des classes croissants, sheared et slippers sont définis respectivement par 0, 1 et 2. Cela nous permet dans une prochaine étape d'appliquer une transformation sur nos réponses pour obtenir la range -127, 0 et 127 proposé par le document PLoS.

5.3 Image pre-processing

Pour limiter l'influence du tube - non pertinent - sur l'apprentissage du modèle, il est intéressant de faire passer nos images dans une tuckey window.

On multiplie l'image par une fonction "fenêtre d'observation" afin de faire disparaître les bords du tube d'écoulement. En effet, cette partie de l'image n'est pas intéressante pour l'apprentissage du réseau.

$$w(y) = \begin{cases} \frac{1}{2}[1 + \cos(\frac{2\pi}{\alpha}(y - \frac{\alpha}{2})] & \text{si } 0 \leq y < \frac{\alpha}{2} \\ 1 & \text{si} \frac{\alpha}{2} \leq y < 1 - \frac{\alpha}{2} \\ \frac{1}{2}[1 + \cos(\frac{2\pi}{\alpha}(y - 1 + \frac{\alpha}{2})] & \text{si} 1 - \frac{\alpha}{2} < y \leq 1 \end{cases}$$

Ainsi, quand on multiplie l'image par cette fenêtre, on obtient une uniformisation des bords pour chaque image.

On réalise également un *mapping* des intensités de chaque image afin que 1% des pixels les plus sombres et 1% des pixels les plus clairs soit saturés. Ce processus permet une homogénéité des profiles d'intensités des images et contrebalance les possibles variations d'illuminations induites par la caméra.

6 Modèle

6.1 Structure du CNN

Le modèle implémenté reprend ainsi celui du support d'étude, adapté de *MATLAB*.

layer	kernel size [px^2]	subimage size [px^2]
input layer	-	90 × 90
conv. layer 1	21 × 21	70 × 70
reLU layer	-	70 × 70
max pooling layer	2 × 2, stride 2	35 × 35
conv. layer 2	14 × 14	22 × 22
reLU layer	-	22 × 22
max pooling layer	2 × 2, stride 2	11 × 11
conv. layer 3	6 × 6	6 × 6
reLU layer	-	6 × 6
max pooling layer	2 × 2, stride 2	3 × 3
output layer, regression type	-	1 × 1

https://doi.org/10.1371/journal.pcbi.1006278.t001

Figure 6: Structure du CNN

```
# Number of classes
num_classes = 3
## Model
cnn_model = Sequential()
## Première couche de convolution
cnn_model.add(Conv2D(25, kernel_size=(21, 21),activation=None,input_shape=(90,90,1)
cnn_model.add(BatchNormalization(epsilon=1e-5))
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D((2, 2),strides=2))
## Deuxième couche de convolution
cnn_model.add(Conv2D(32, (11, 11), activation=None))
cnn_model.add(BatchNormalization(epsilon=1e-5))
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2), strides=2))
## Dernière couche de convolution
cnn_model.add(Conv2D(64, (3, 3), activation=None))
cnn_model.add(BatchNormalization(epsilon=1e-5))
```

```
cnn_model.add(Activation('relu'))
cnn_model.add(MaxPooling2D(pool_size=(2, 2),strides=2))

## On écrase les matrices en vecteurs
cnn_model.add(Flatten())
## Couche fully-connected de regression
cnn_model.add(Dense(1, activation='linear'))
```

Notre modèle se compose alors de :

- 3 couches de convolution,
- 3 couches de pooling de taille (stride) 2x2,
- 3 couches de normalisation de batch
- 3 couches d'activations (ReLU)
- Une couche fully-connected

6.2 Entraînement du CNN

6.2.1 Principe

L'entraînement d'un CNN consiste à déterminer et à calculer empiriquement la valeur de chacun de ses poids. Le CNN traite une image du jeu d'entraînement et en sortie il fait une prédiction. Sachant que l'on connaît *a priori* la classe de chacune des images d'entraînement, on peut vérifier la véracité du résultat. Une fois que toutes les images du batch sont passées dans le modèle, le CNN met à jour ses poids et itère cette étape pour chaque batch. Une fois cette étape finie, on passe les batches du jeu de validation dans le modèle pour l'évaluer. Il classe ces images qui lui sont inconnues, ce qui permet de calculer la performance du CNN.

6.2.2 Implémentation

Pour la compilation nous utilisons un optimisateur de descente de gradient stochastique à momentum (SGDM). En effet, d'après une étude de l'Université de Cornell [3], le SGD+Momentum est un optimisateur qui permet une meilleur généralisation, i.e. évite le sur-apprentissage et permet de meilleurs prédictions. Notre choix s'est donc porté sur celui-ci.

Definition : Descente de gradient stochastique à momentum

$$\sigma_{l+1} = \sigma_l - \alpha \nabla E(\sigma_l) + \gamma(\sigma_{l+1} + \sigma_l)$$

L'algorithme du SGDM mets à jour le vecteur σ_l de manière discrète à la l-ème itération. $E(\sigma_l)$ à la fonction *loss*, dans notre cas la RMSE, $\alpha = 0.001$ étant le taux d'apprentissage et $\gamma = 0.9$ le terme du moment.

Nous implémentons une *loss* de type MSE comme le document référence du fait que le problème soit de regression.

Définition : loss
La *loss* correspond à la pénalité pour une mauvaise prédiction. Autrement dit, la *loss* est un nombre qui indique la médiocrité de la prévision du modèle pour un exemple donné. Si la prédiction du modèle est parfaite, la *loss* est nulle. Sinon, la *loss* est supérieure à zéro. Le but de l'entraînement d'un modèle est de trouver un ensemble de pondérations et de biais pour lesquels la *loss*, en moyenne sur tous les exemples, est faible.

Définition : **biais**
Un biais est un procédé qui engendre des erreurs dans les résultats d'une étude. Formellement, le biais d'un estimateur est l'espérance de la différence entre la valeur de son espérance et la valeur de la variable aléatoire qu'il est censé estimer

Dans notre cas, la fonction de perte utilisé est la MSE (*Mean Squarred Error*) définie telle que :

$$MSE = \frac{1}{N} \sum_{y \in DataSet} (y_{true} - y_{pred})^2$$

Pour l'entrainement, nous nous fions aux paramètres explicites du code source du document PLoS.

Nous utilisons le jeux d'entrainement et de validation prévu à cet effet pour entraîner le modèle vu précédemment.

6.2.3 Résultats d'entrainement

On voit clairement sur la figure 7 une convergence de l'erreur vers des valeurs très basses. Cette convergence se stabilise à partir plus ou moins 40 epochs ce qui diffère beaucoup de ce que nous avions lu dans le document référence où il annonçait une

convergence franche à partir de 10 epochs. On visualise qu'à ce stade il n'y a clairement pas de convergence.

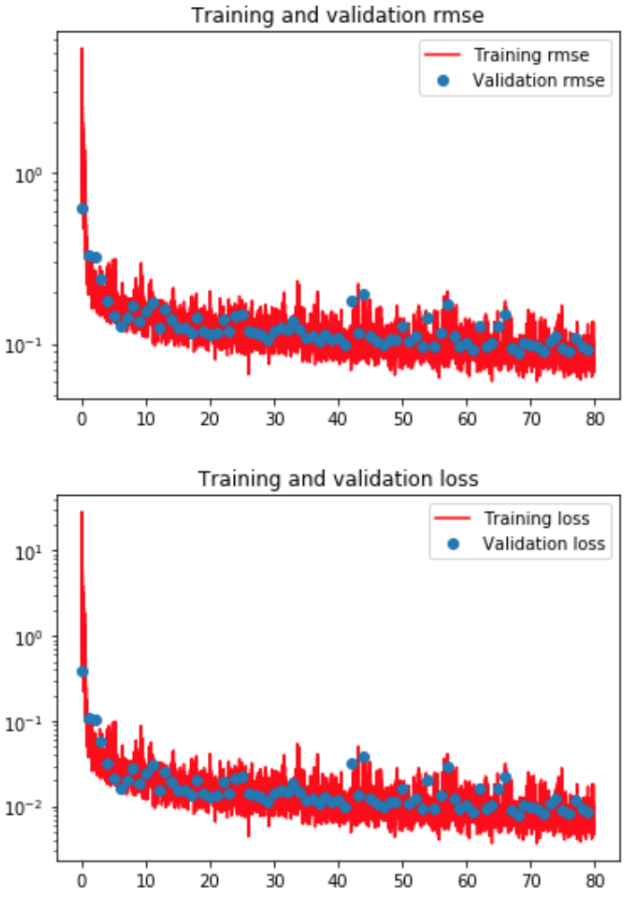


Figure 7: Graphe de la loss pour le train et la validation

7 Résultats

7.1 Prédictions

Pour cette étape, nous utilisons le jeu de données de test pour évaluer les performances du modèle. Chaque image de notre jeu de test (3090 images) passe à travers notre réseau, donnant ainsi la réponse prédite (label) pour l'image.

7.2 Critères de performance

On définit des critères qui permettent d'interpréter la performance de notre modèle sur différents aspects qui ne se valent pas. Par exemple, si un modèle prédit 98% des réels croissants comme croissants mais dans le même temps classe 40% des sheared comme croissant, peut-on vraiment affirmer que ce modèle est bon ?

Pour se faire, introduisons les éléments d'une matrice de confusion obtenue avec les prédictions d'un modèle. On définit pour chaque classe les 3 critères de performance ci dessous (recall, precision, f1-score). La classe étudiée sera considérée comme positive.

Introduisons les termes :
vp : vrai positif
fn : faux negatif
vn : vrai negatif
fp : faux positif
Par exemple, si la classe étudiée est les croissants, un vrai positif sera un croissant classé par le modèle comme croissant. Un faux positif sera un sheared (ou un slippers) classé par le modèle comme croissant. Un faux négatif sera un croissant classé comme sheared ou slippers et un vrai négatif sera un slippers ou un sheared "bien classé".

Definition : **Recall**
Le recall est le nombre de résultats correct (vp) sur le nombre de résultats qui aurait dû être retourné, on a :

$$R = \frac{vp}{vp + fn}$$

Definition : **Precision**
La précision est le nombre de résultats correct(vp) sur le nombre total de résultats renvoyé, on a :

$$P = \frac{vp}{vp + fp}$$

Definition : **Fβ Score**
Le Fβ Score est la précision d'un test. Il utilise les deux critères précédents avec un paramètre β. On utilisera dans notre étude, $\beta = 1$ (F1 Score), i.e. la précision et le recall ont la même importance ou le même équilibre dans la formule. Ainsi on a avec la formule simplifiée :

$$F_1 = 2 * \frac{P * R}{P + R}$$

Definition : **Accuracy**
L'accuracy (traduit par *justesse*) est un critère général qui mesure la proportion de prédictions correctes (vp) du modèle. Elle est définie par :

$$Acc = \frac{vp + vn}{vp + vn + fn + fp}$$

7.3 Visualisation des prédictions

Nous obtenons un histogramme des prédictions qui nous permet de visualiser la distribution. Celui-ci ressemble fortement à celui tracé dans le document d'étude.

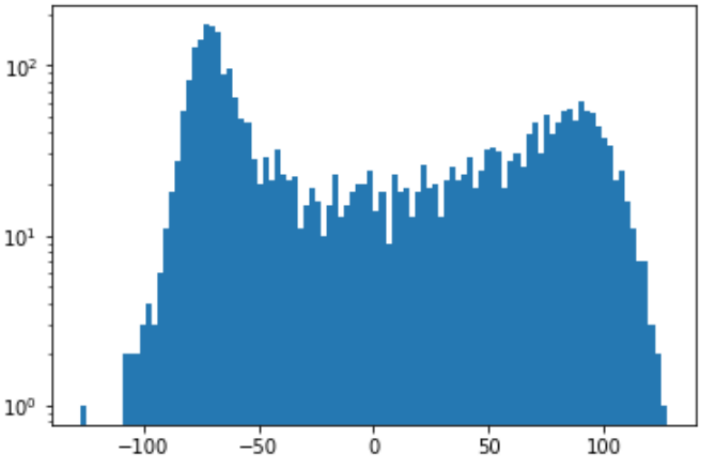


Figure 8: Histogramme des individus prédis

Nous devons alors définir dans quelle range de valeurs chaque classe appartient.

7.4 Détermination des intervalles des classes

On utilise un Gaussian Mixture Model (GMM) pour *fit* la distribution des labels prédis.

Le Gaussian Mixture Model (GMM) est une approche statistique non supervisé qui permet de représenter des sous-populations en les considérants gaussiennes pour estimer leur distribution.

L'utilité d'un GMM pour notre problème est d'obtenir automatiquement les paramètres μ, la moyenne, et σ, l'écart-type, de chaque gaussienne de la distribution.

Ces paramètres nous permettent de réaliser les intervalles de confiance nécessaires pour déduire la classe de chaque prédiction. Chaque classe sera donc associée à une range de valeurs donnant au modèle une tolérance plus flexible qu'un simple classifieur.

7.5 Approche "naïve"

Nous avons tout d'abord *fit* les prédictions avec 3 gaussiennes, une pour chaque classe.

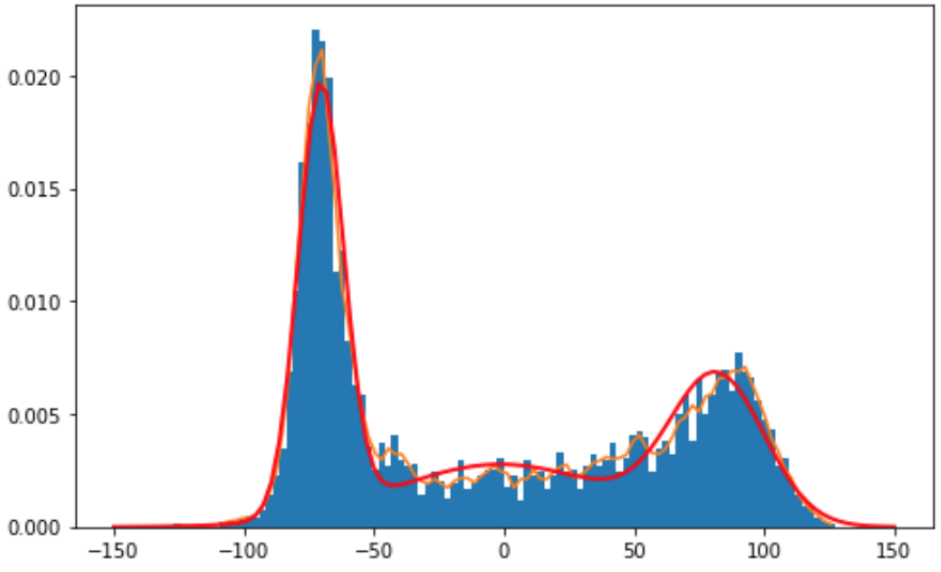


Figure 9: Histogramme et GMM

On peut voir que le *fit* n'est pas très bon. En effet, les courbes, notamment sur les

croissants (pic vers 80), ne coincident pas parfaitement avec la courbe orange qui *fit* "naivement" l'histogramme figure 8.

On peut noter également une concentration plus prononcée des prédictions par rapport aux résultats de l'article de référence.

En effet, avec ces courbes nous obtenons les résultats illustrés par la matrice de confusion ci-dessous. Ces résultats sont largement en déça de ce que nous pouvions espérer. Avec une approche naïve du problème, on se base surtout sur le critère du recall, c'est à dire le pourcentage sur la diagonale.

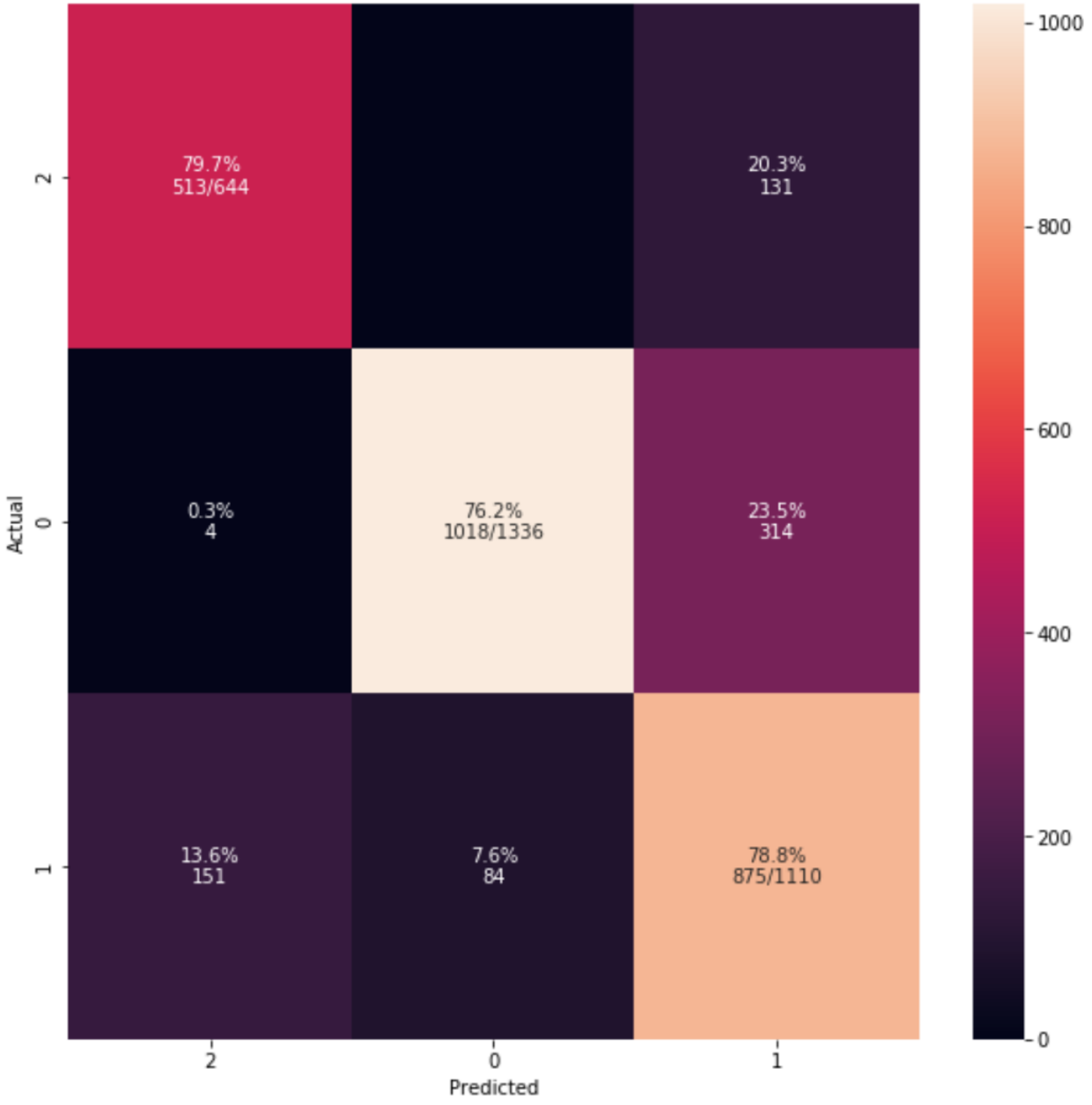


Figure 10: Matrice de confusion avec σ

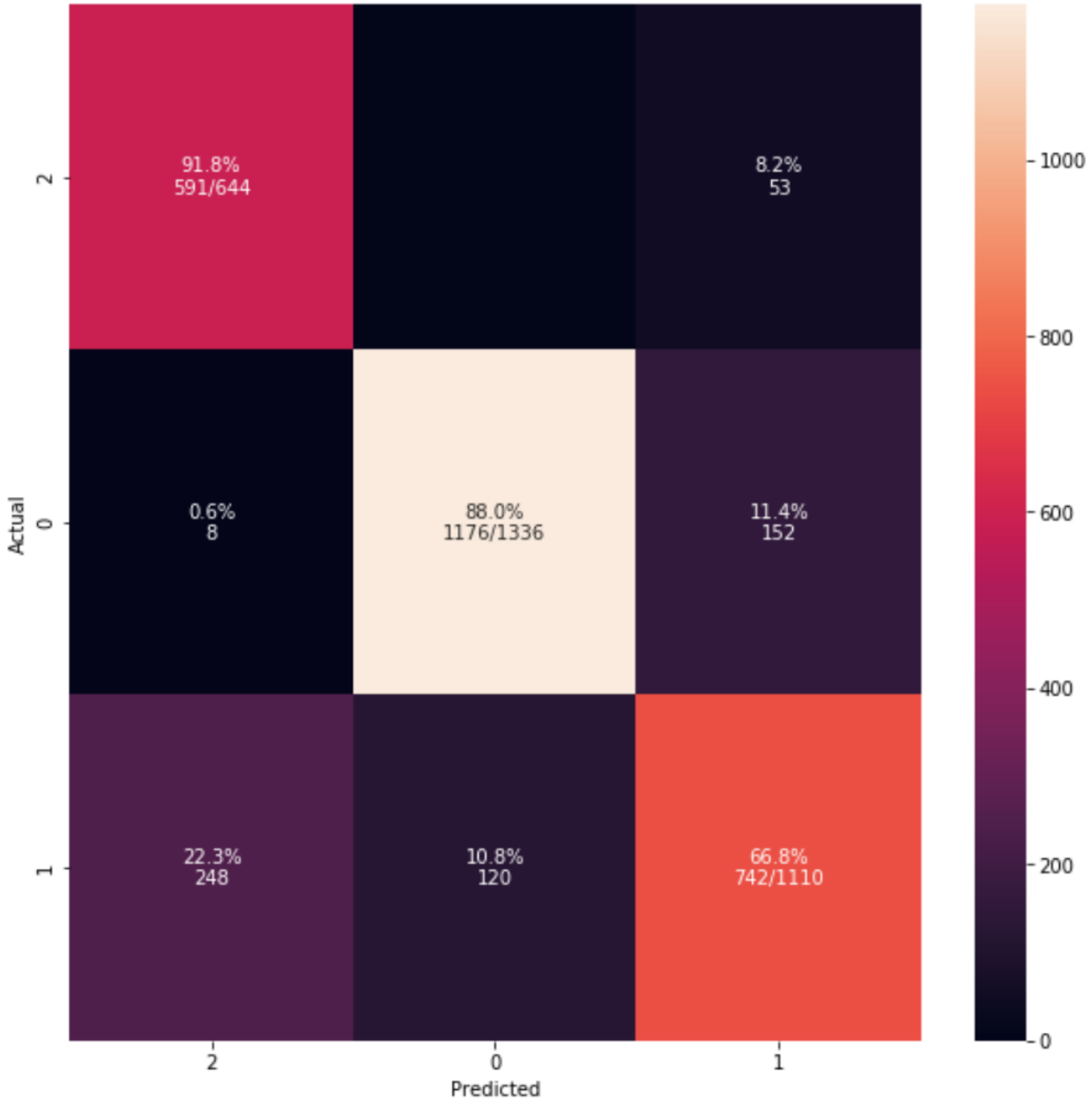


Figure 11: Matrice de confusion avec 2σ

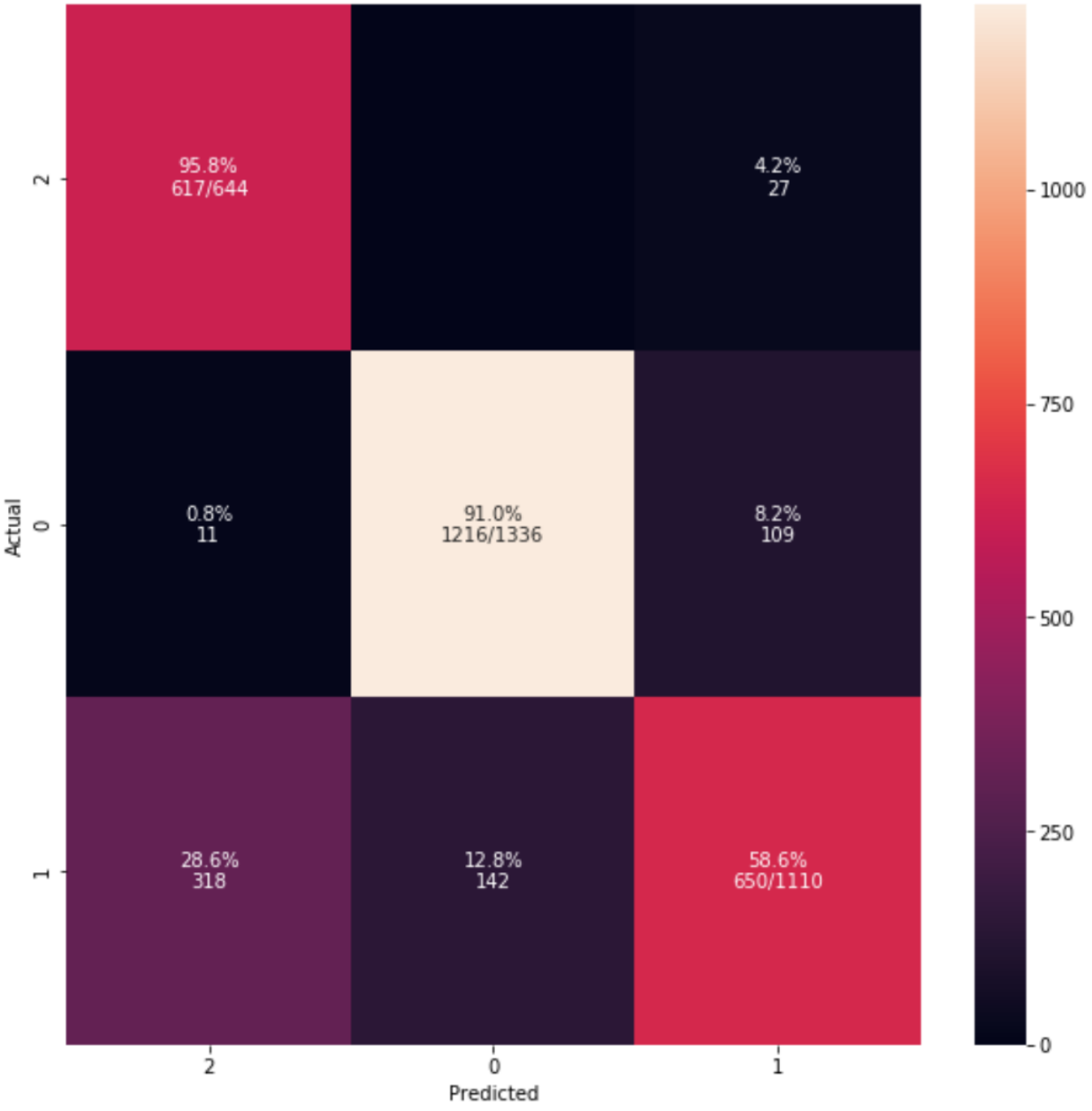


Figure 12: Matrice de confusion avec 3σ

7.6 Recherche des paramètres optimaux

Une grille de recherche est mise en place afin de trouver - selon les critères de performance définis plus haut - les paramètres optimaux :

- Le nombre de gaussiennes
- L'intervalle de confiance des croissants
- L'intervalle de confiance des slippers

Le nombre de gaussiennes permet d'obtenir des courbes qui *fit* au mieux la distribution des prédictions.

7.6.1 Stratégie de décision

Nous avons 3 critères de performance qui définissent chacun un aspect différent de la réussite de la prédiction. Le recall privilégie le fort taux de vrai positifs i.e le nombre d'individus prédis qui sont effectivement de cette classe. Une bonne précision limite le nombre de faux positifs i.e les membres qui ne devraient pas être de cette classe. Le f1-score est quand a lui la moyenne des deux.

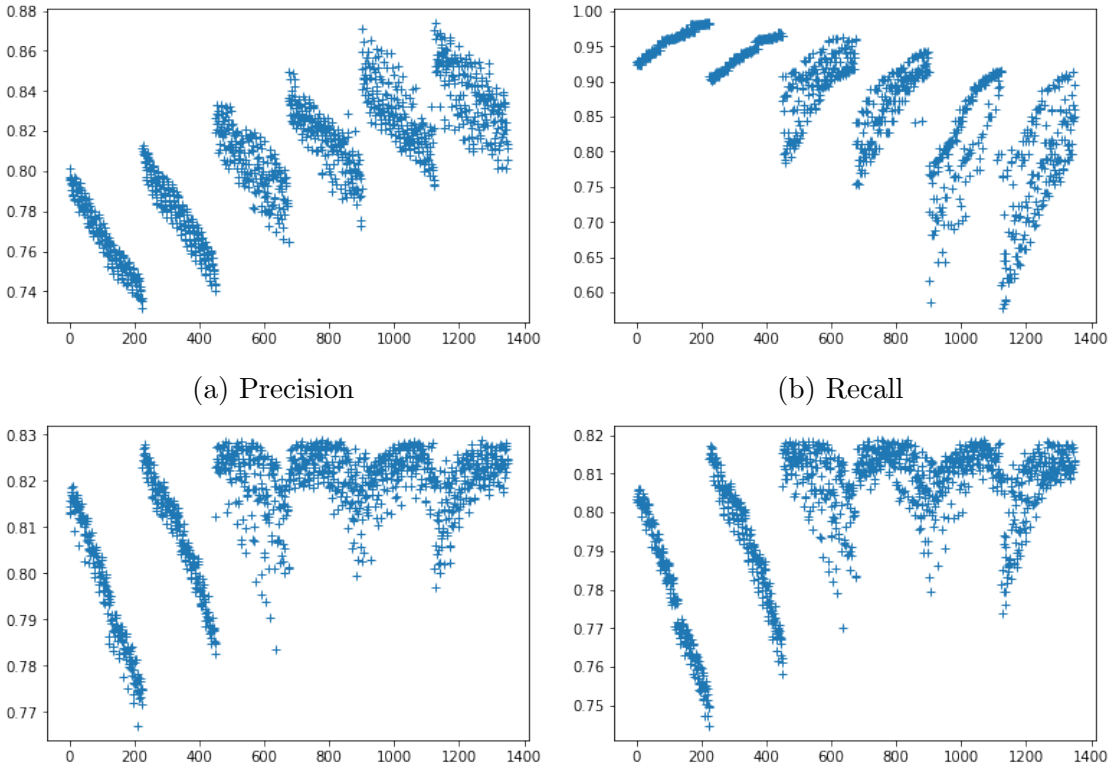
Un exemple de stratégie pourrait être le suivant : on pourrait se dire qu'il vaut mieux avoir un pourcentage de perte de croissants confondus avec des sheared que des sheared considérés comme des croissants. En effet, si l'on désire faire des études sur les croissants il est préférable d'avoir une perte de certains individus plutôt que d'avoir des sheared dans l'analyse des croissants.

7.6.2 Recherche paramètres optimaux par grille de recherche

On obtient 1350 modèles par notre grille de recherche en faisant varier les intervalles de confiance de 2σ à 3.5σ (pas de 0.1) pour les croissants et les slippers. Le nombre de gaussiennes varie de 4 à 9. Pour chaque modèle on récupère la précision, le recall et le f1-score de chacune des classes ainsi que l'accuracy générale. On peut alors visualiser la précision, le recall et le f1-score moyen (moyenne générale des croissants et slippers) pour chaque modèle.

On note que pour chaque critère, les paramètres optimaux sont assez différents. La précision et le recall évolue de façon inversé : le nombre de gaussienne tend à augmenter la précision et inversement diminue le recall.

On récupère ainsi les paramètres optimaux pour chaque critère maximisé, nous permettant alors de produire une matrice de confusion.



(a) Precision (b) Recall

(c) Accuracy (d) F1-Score

Figure 13: Recherche des meilleurs modèles selon les critères de performance

7.6.3 Meilleure accuracy

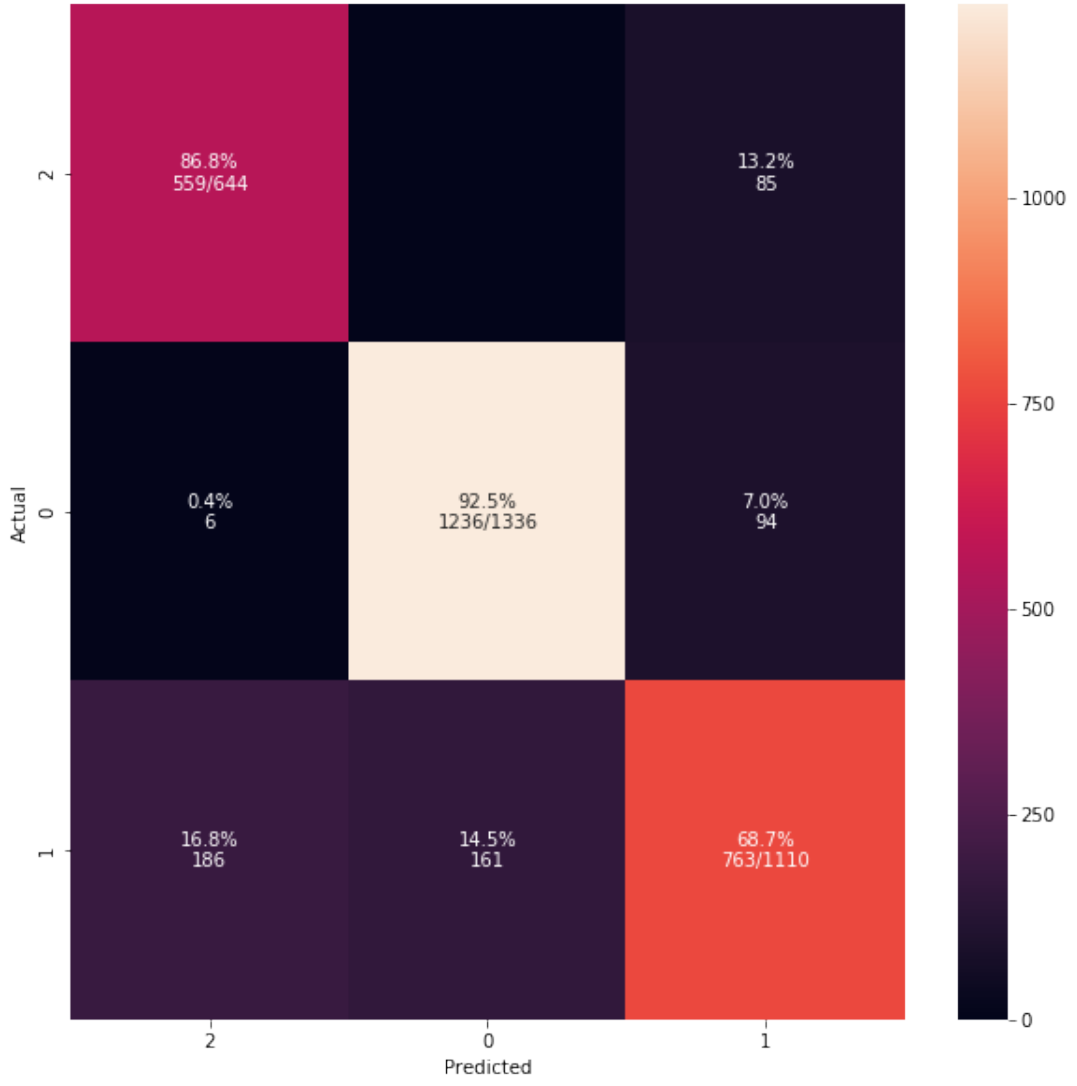


Figure 14: Modèle avec la meilleure accuracy moyenne

Paramètres :

- Nombre de gaussienne : 7
- Ic croissants : 2.6σ
- Ic slippers : 3.2σ

En optimisant ce critère, on s'assure d'une bonne classification des slippers et des

croissants ainsi que 16.8% de faux positifs venant des sheared. C'est un bon compromis entre vrai positifs et faux positifs car on garde une bon nombre d'individus bien classé pour un relativement faible pourcentage d'erreur parmis ce groupe.

7.6.4 Meilleur recall

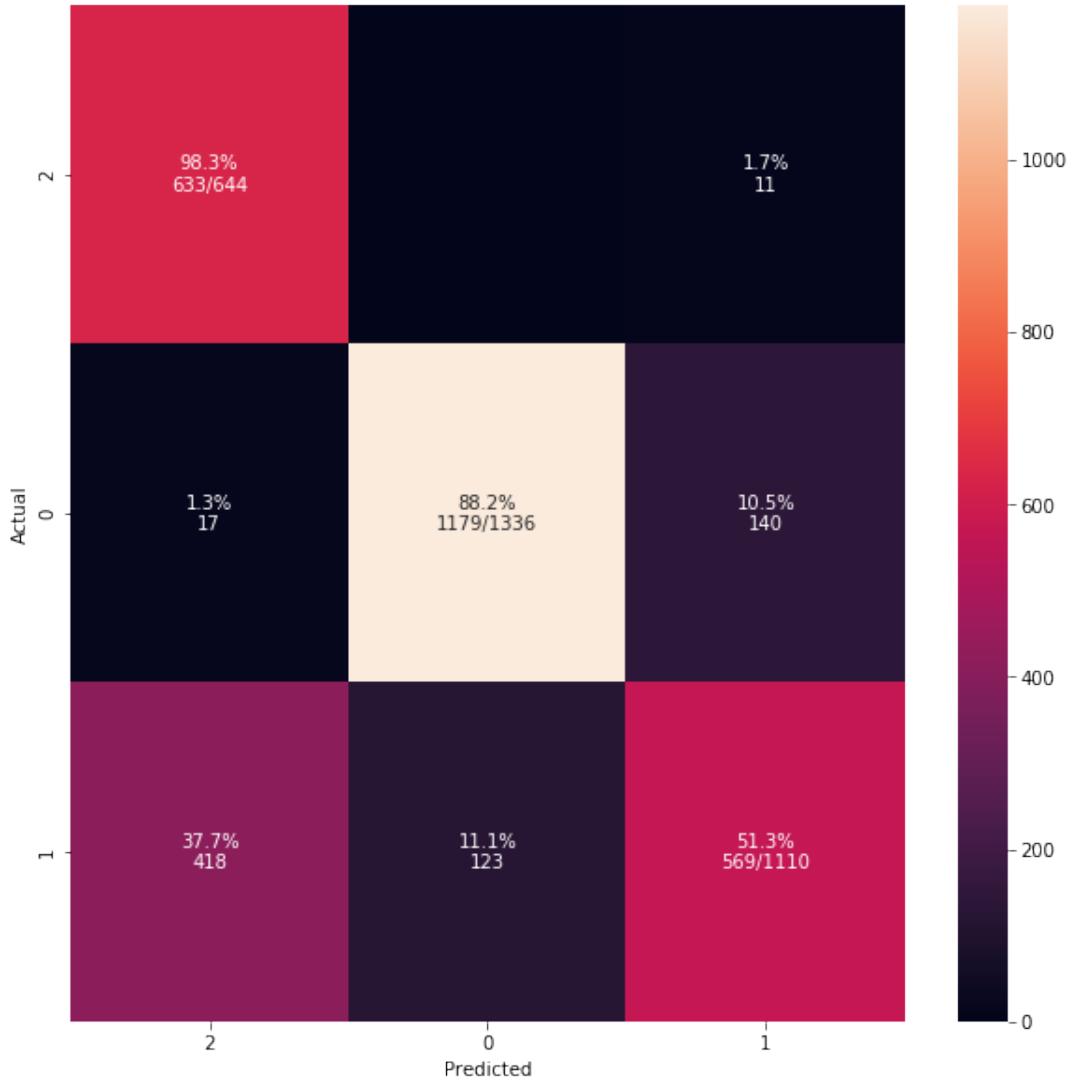


Figure 15: Modèle avec le meilleur recall moyen

Paramètres :

– Nombre de gaussienne : 4

– Ic croissants : 3.4σ

– Ic slippers : 2σ

Le recall nous donne de très bons résultats pour les vrai positifs (98% de croissants bien classés) cependant on dénombre à coté 37% de sheared chez les croissants ce qui n'est pas acceptable en vu d'une étude ultérieur.

7.6.5 Meilleure précision

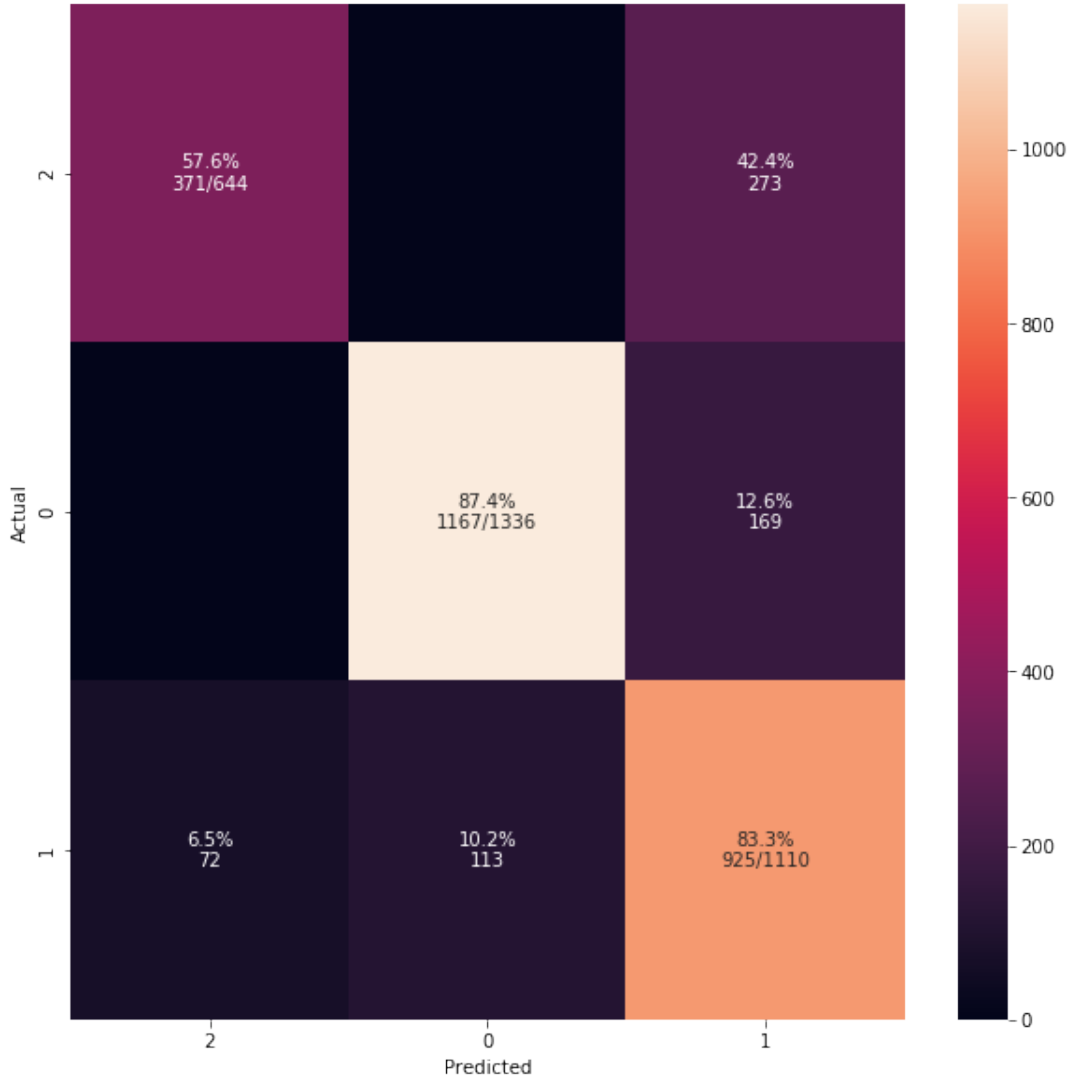


Figure 16: Modèle avec la meilleure précision

Paramètres :

- Nombre de gaussienne : 9
- Ic croissants : 2σ
- Ic slippers : 2.1σ

La classification des croissants est bien moins bonne on a seulement 57% des croissants classés comme tel,mais à coté, il n'y qu'un faible nombre de sheared (6.5%)dans cette population. Ainsi notre population de croissants prédis ne comptera pratiquement que des croissants, ce qui semble idéal si l'on veut s'appuyer sur ces résultats pour entamer par la suite une étude sur le comportement des cellules.

7.6.6 Meilleur f1-score

Paramètres :

- Nombre de gaussienne : 7
- Ic croissants : 2.5σ
- Ic slippers : 2.5σ

Surement le meilleur compromis avec l'accuracy. On obtient une bonne classification avec un nombre plus ou moins acceptable de faux positifs dans les deux classes principales (en moyenne 13%).

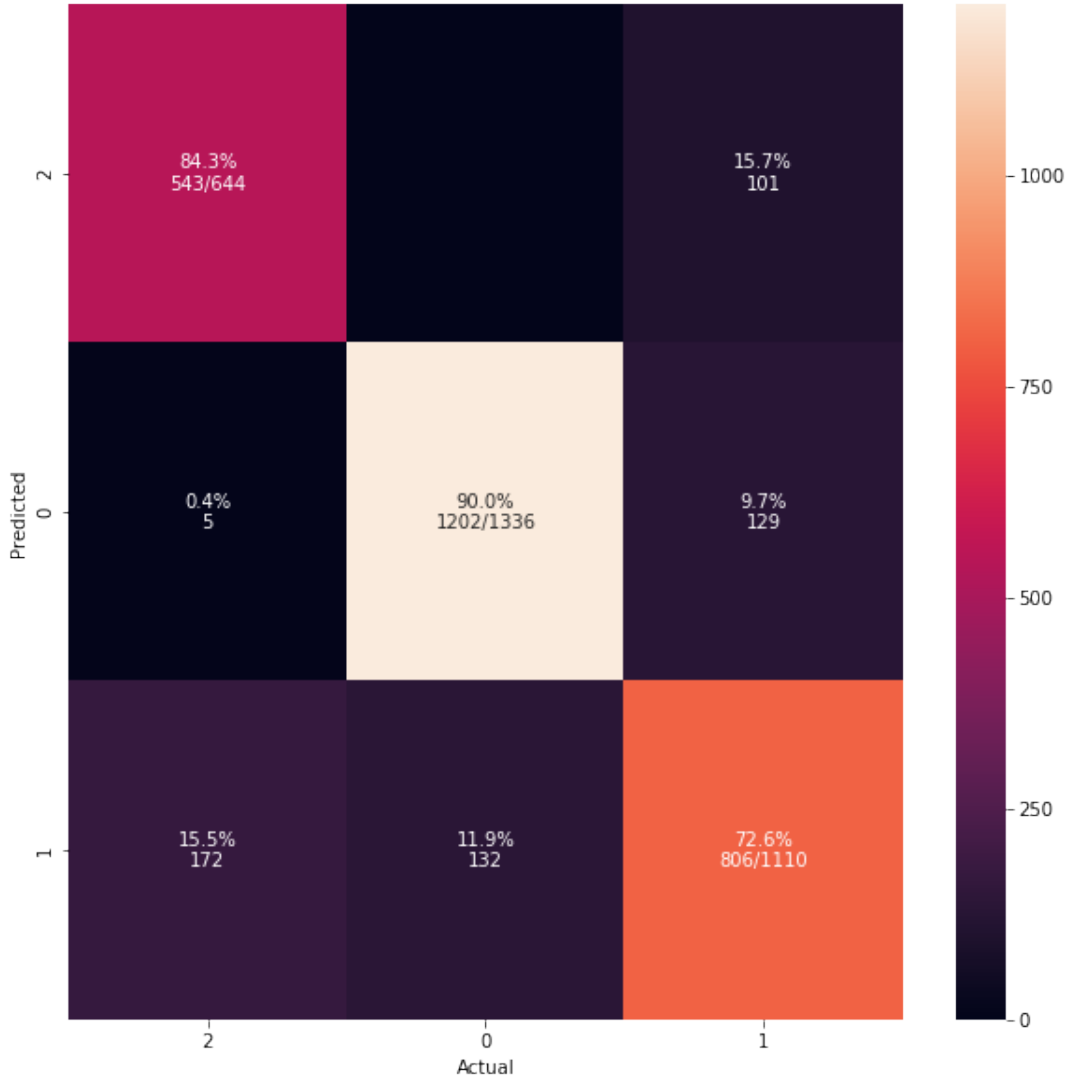


Figure 17: Modèle avec le meilleur F1

7.6.7 Observations

Quelque soit le critère de performance choisi, les slippers sont globalement prédis avec une présence de faux positifs d'environs 10% (i.e 10% des sheared sont classés comme slippers). L'analyse et la décision prise du modèle optimal se portent donc majoritairement sur les résultats de classification des croissants.

En conclusion, si on se base sur la précision, on limite les faux positifs mais on perd beaucoup de vrai positif tandis qu'en se basant principalement sur le recall on

obtient plus de vrai positifs mais également plus de faux positifs, ce qui n'est pas vraiment acceptable selon la stratégie de décision énoncée.

C'est pourquoi il serait judicieux d'accorder de l'importance à l'un comme l'autre et donc maximiser le critère de performance qui moyenne les deux i.e. le f1-score. On peut aussi se tourner vers l'accuracy qui donne des résultats similaires.

7.7 Comparaison

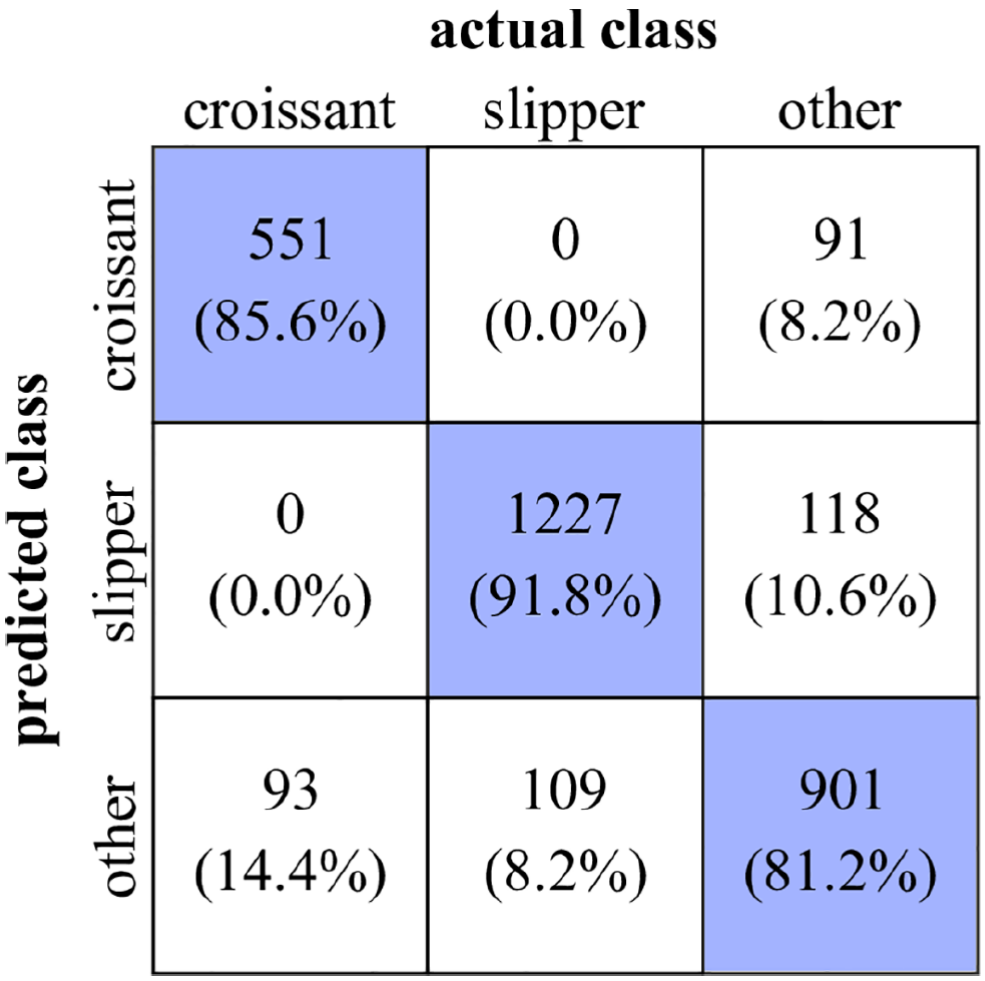


Figure 18: Matrice de confusion du document PLoS

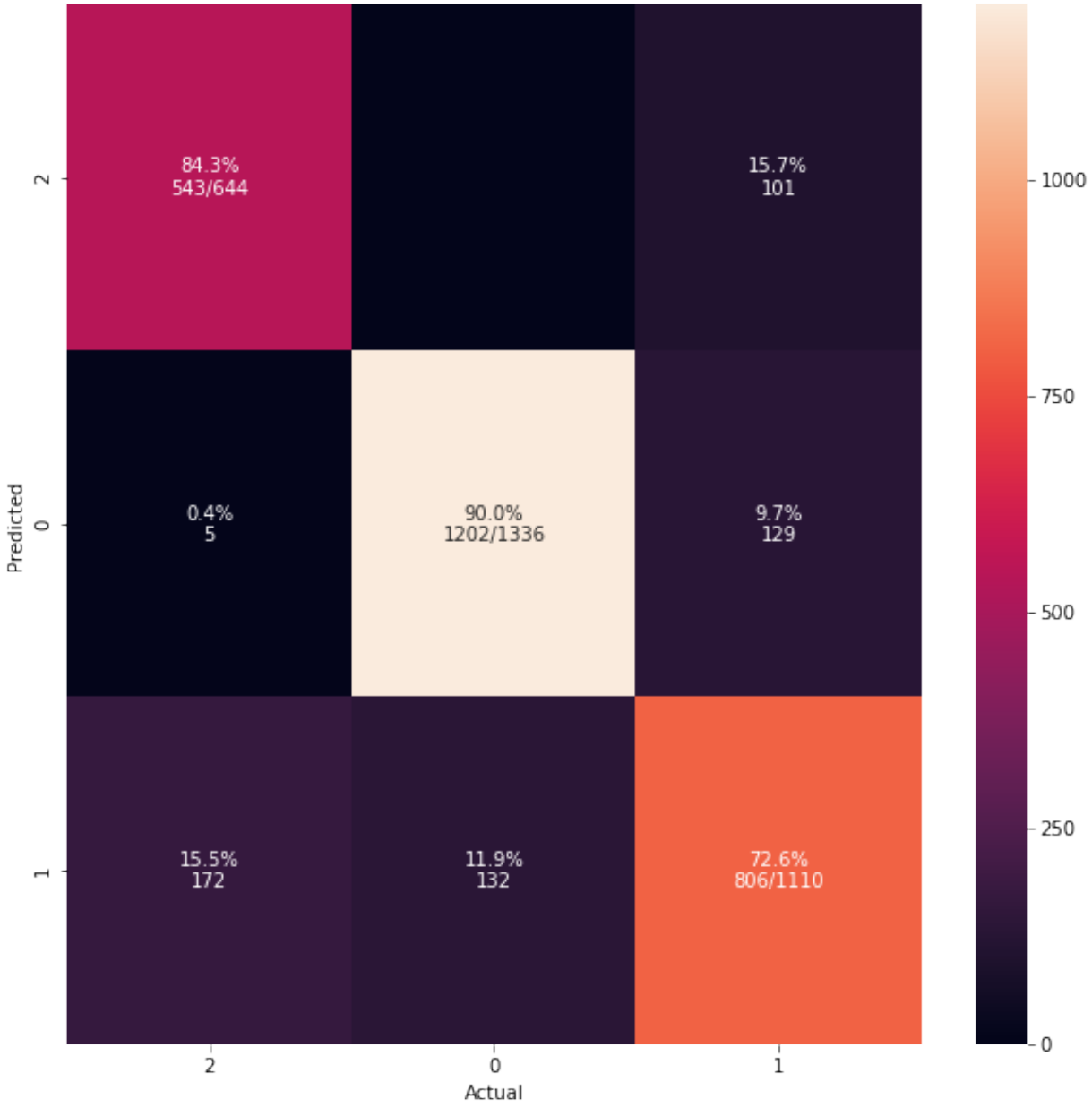


Figure 19: Notre matrice de confusion optimale

Nous obtenons sensiblement, à 2% près, les mêmes résultats qu'eux pour les croissants et les slippers au niveau du recall. Cependant, notre classification de sheared est moins bonne que la leur. En effet, une plus grande partie des sheared se retrouve prédis comme étant des croissants. Cela est surement dû à l'aspect très compact de notre distribution qui favorise la confusion entre ces deux classes. La précision entre

slippers et sheared est sensiblement la même sur les deux matrices. Il faudrait donc trouver un moyen d'éclater la distribution pour avoir une meilleur précision entre les croissants et les sheared et ainsi nous rapprocher de leur résultats. Toutefois, nous pouvons juger notre modèle comme ayant des performances acceptables.

8 Perspectives

8.1 Difficultés rencontrées

8.1.1 Techniques

La première difficulté a été la prise en main technique des différentes bibliothèques python et de les utiliser correctement pour implémenter notre modèle. Le nombre conséquent de différents paramètres à prendre en compte a nécessité un recul constant sur le travail déjà effectué. Par exemple Keras est à la fois puissant et *user-friendly*. Cependant, il n'a pas offert une configuration assez poussée pour notre data pre-processing lorsque nous avons voulu mettre des valeurs de classes à -127, 0 et 127. Nous n'avons pu que mettre 0, 1 et 2.

8.1.2 Décisionnelles

Une fois que nous avions eu nos résultats, il a été difficile de bien savoir comment les optimiser et pour quelle stratégie nous devions opter pour satisfaire les objectifs des suiveurs du projet. De même, nous nous sommes questionnés sur la rigueur du projet amené par le document d'étude. Il a fallut prendre du recul par rapport à notre projet ainsi qu'au leur pour parvenir à un résultat vraiment performant et optimisé. Il a souvent fallu discuter de la marche à suivre lorsque les résultats obtenus avec l'aide des méthodes du document d'étude n'était vraiment pas à la hauteur de nos espérances, ce qui a entrainé parfois des moments de flou durant le projet.

8.2 Amélioration du réseau

8.2.1 Dropout

Une première idée que nous avons eu a été de rajouter des couches de Dropout pendant la phase d'apprentissage. La méthode du dropout consiste à désactiver

aléatoirement des sorties du réseau de neurones avec une probabilité arbitraire donnée en paramètre. Cela permet de créer artificiellement de nouveaux réseaux à chaque itération d'apprentissage. Attention cependant, le dropout est une méthode pouvant généralement permettre une acceleration de l'apprentissage et une meilleure performance de celui. Néanmoins ce n'est pas toujours le cas et c'est pourquoi des tests doivent être fait pour vérifier son utilité.

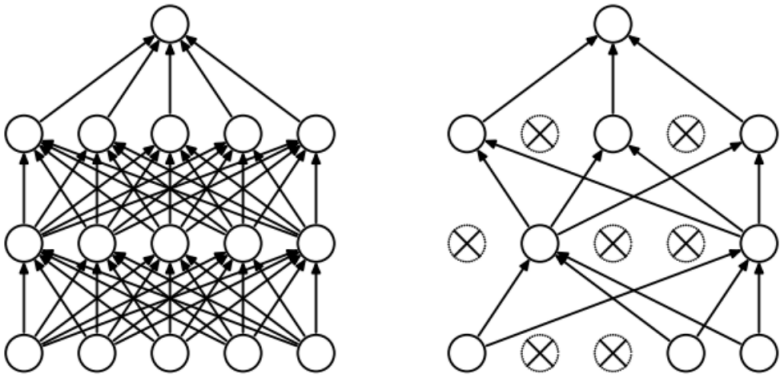


Figure 20: Illsutration du Dropout

8.2.2 DropConnect

Le DropConnect est une alternative au dropout consistant à inhiber une connexion (l'équivalent de la synapse), et ce de manière toujours aléatoire. Les résultats sont similaires (rapidité, capacité de généralisation de l'apprentissage) au dropout, mais présentent une différence. En effet la méthode sur la coupure d'une connexion certains nodes plutôt que la coupure d'un node entier.

8.2.3 Augmentation du nombre de couches

L'augmentation du nombre de couches peut sembler être une solution intéressante pour améliorer le modèle. Néanmoins, étant donné la petite taille des images, rajouter des couches ne seraient pas forcément pertinent, augmentant la complexité du modèle pour de faibles résultats.

8.3 Conclusion

Pour le moment le modèle est performant mais n'est pas aussi efficace que le document d'étude. Il faut savoir que l'utilisation d'intervalle de confiance avec un GMM qui est déjà un modèle d'approximation rajoute *a priori* de l'incertitude pour les prédictions malgré le fait que nous prenions l'intervalle de confiance optimal. Il serait intéressant de prendre du recul et de comparer ces résultats avec un modèle basé uniquement sur un classifieur. Ces différences de performances avec le document d'étude peuvent s'expliquer par le choix des bibliothèques et de la manière dont elles gèrent les calculs, particulièrement pour l'initialisation des poids dans le réseau. Mais également par le fait que nous avons un intervalle de base compact [0,1,2] qui peut expliquer le fait que l'histogramme l'est également malgré l'eclatement de cet intervalle à [-127,127] *a posteriori* des prédictions.

Il existe beaucoup de méthodes pour améliorer le modèle en lui-même. Nous avions vu que les couches de Dropout ou encore de DropConnect peuvent être une alternative efficace. Le nombre de couches ou tout simplement le nombre de données à disposition est aussi un facteur pouvant améliorer le modèle.

De même, il pourrait être intéressant de comparer sur un diagramme de phase similaire à celui du document d'étude, la classification du modèle et celle manuelle. Cela donnerait un aperçu de la puissance des prédictions et de la robustesse du modèle en pratique.

Bibliography

[1] Kihm A, Kaestner L, Wagner C, Quint S (2018) *Classification of red blood cell shapes in flow using outlier tolerant machine learning*, PLoS Comput Biol 14(6): e1006278. https://doi.org/10.1371/journal.pcbi.1006278

[2] @CharlesCrouspeyre (2017) *Comment les réseaux de neurones à convolution fonctionnent*, Article Medium. https://medium.com/@CharlesCrouspeyre/comment-les-réseaux-de-neurones-à-convolution-fonctionnent-b288519dbcf8

[3] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, Benjamin Recht (2018) *The Marginal Value of Adaptive Gradient Methods in Machine Learning* arXiv:1705.08292v2

[4] Documentation Keras https://keras.io/

[5] Julien Dejasmin (2018) *Entrainement des réseaux de neurones convolutifs* https://www.natural-solutions.eu/blog/entrainement-dun-rseau-de-neurones-convolutif