Implémentation sur mobile

3.1. Environnement de développement

L'application est développée pour fonctionner sur téléphone mobile avec le système d'exploitation Android. L'outil de développement utilisé est le SDK Android détaillé en Annexe 3.

3.2. Configuration requise pour l'application

Comme illustré sur le Tableau III, on a choisi de développer l'application en se basant sur une configuration de la plupart des smartphones existant sur le marché actuel.

Tableau III : Configuration minimal requise

Système d'exploitation	Android 2.3.3
Mémoire vive	128 Méga Octet
Appareil photo	5 Méga Pixel
Processeur	1Ghz

3.3. Fonctionnement générale de l'application

Figure 3.1: Synopsys générale de l'application

Le fonctionnement général de l'application est assez simple comme le montre la figure. L'image d'une équation est prise en photo avec l'appareil photo ou téléchargéedepuis la galerie du téléphone. Cette image est ensuite analysée pour déterminer le type de l'équation et la résoudre.

Comme illustré par laFigure 3.2, l'écran d'accueil de l'application comporte 3 boutons, un bouton affiche un écran de configuration pour optimiser la reconnaissance, un bouton pour prendre en photo une équation et un bouton pour lancer la reconnaissance.

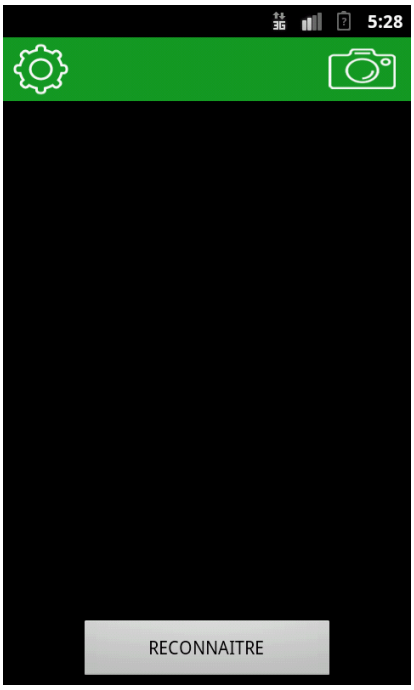

Figure 3.2 : Ecran d'accueil de l'application

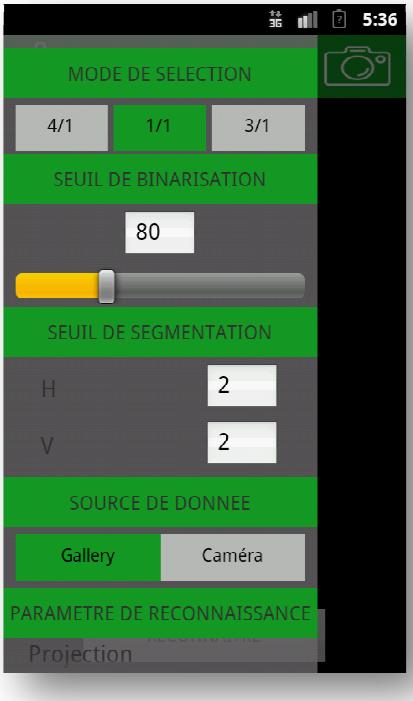

Figure 3.3 : Ecran de configuration

Après la capture (Figure 3.4), un écran de résultat affiche le type de l'équation et la solution de l'équation comme illustré sur laFigure 3.5.

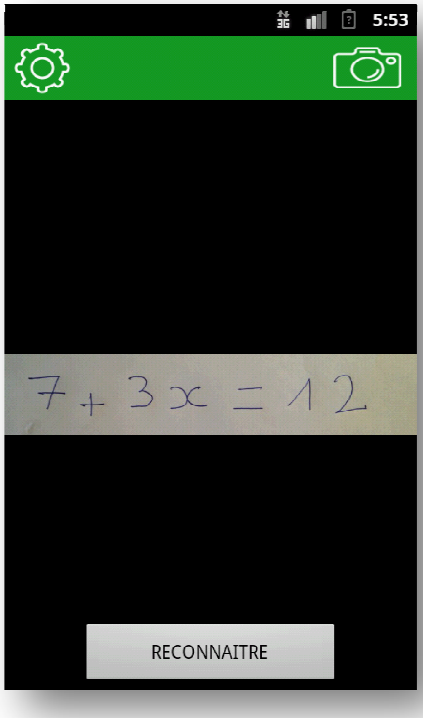

Figure 3.4 : Equation capturé

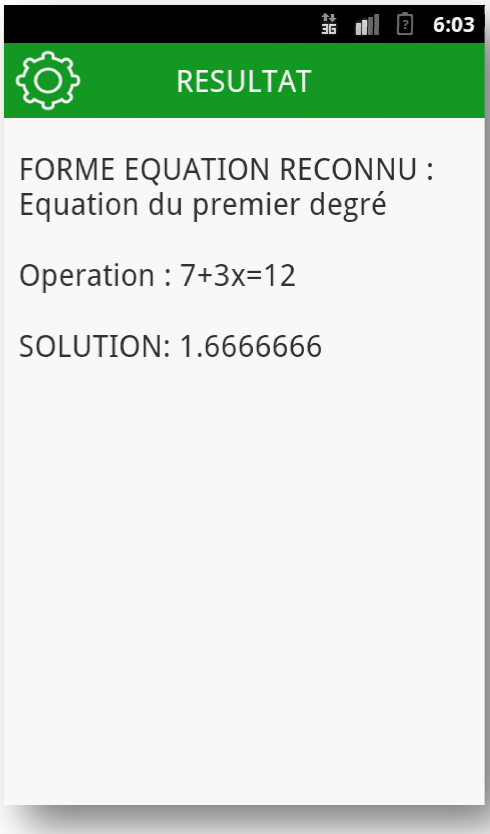

Figure 3.5 : Ecran résultat

3.4. Architecture générale de l’application

Pour optimiser la transmission et le traitement de données, on doit doter l’application d’une bonne architecture. L’architecture utilisée est l’architecture de type 3 tiers, architecture à trois niveaux ou architecture à trois couches. Comme illustré sur laFigure 3.6, l’architecture de l’application est composée de 3 couches:

- Couche présentation: toute action provenant de l’utilisateur telle que la capture, la reconnaissance sont lancées via cette couche. Elle correspond à la partie de l’application visible et interactive avec l’utilisateur.
- Couche de données : les neurones composant le réseau est modélisé dans cette couche. Ces modèles de données sont utilisés par la couche présentation et couche traitement.
- Couche de traitement : elle correspond à la mise en œuvre de l’ensemble des règles de gestion et logique applicative. Elle correspond à la partie fonctionnelle de l’application, celle qui implémente la logique et qui décrit les opérations que l’application opère sur les données en fonction des requêtes des utilisateurs, effectuées au travers de la couche présentation.

Le rôle de chacune des couches et leur interface étant bien définis, les fonctionnalités de chacune d’entre elles peuvent évoluer sans induire de changement dans les autres couches.

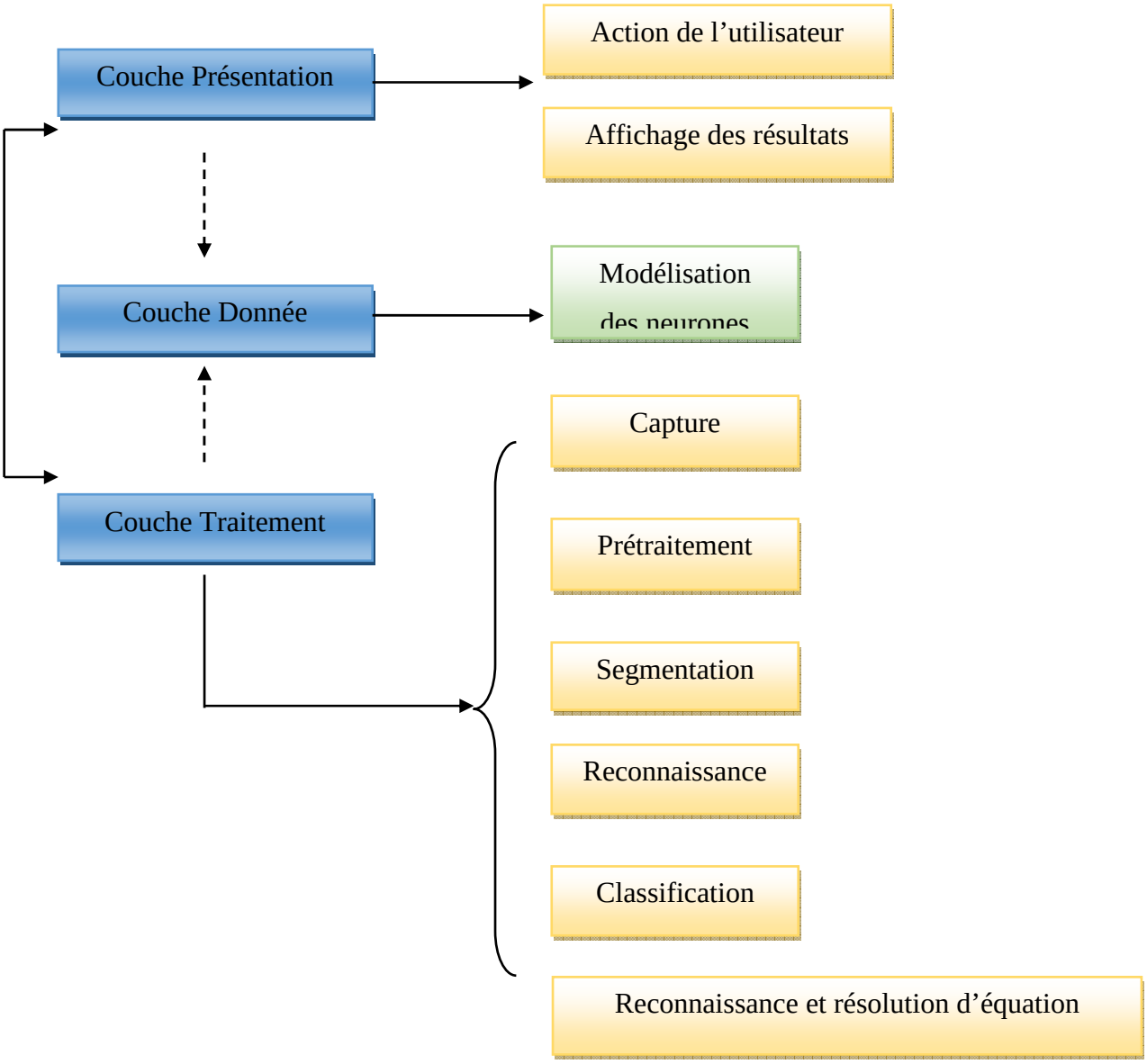

Figure 3.6 Architecture de l'application

3.5. Implémentation proprement dit de l'application sur mobile

3.5.1. Capture de l'image de l'équation

L'image de l'application capture est assez grande et peut comporter d'autres informations qui ne sont pas utiles. Un système de sélection de l'équation a été mis en place pour diminuer la taille de l'image à traiter et aussi pour cibler facilement l'équation à traiter comme illustré sur laFigure 3.7.

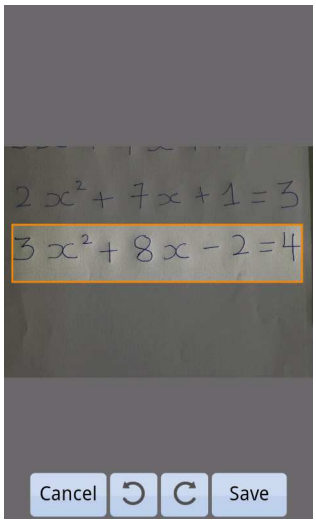

Figure 3.7 : Sélection de l'équation

3.5.2. Le seuillage de l'image

Le seuillage de l'image est assez difficile parce que le niveau du seuil choisi détermine les données qui vont être traitées. La formule (3.1) utilisée sur chaque pixel de l'image initial pour obtenir l'image binarisée est:

$$In(x,y)=\begin{cases} 1, \text{ si } (Ir(x,y)<seuil \mid Ig(x,y)<seuil \mid Ib(x,y)<seuil) \\ 0, \text{ autre} \end{cases} \qquad (3.1)$$

In(x,y) : valeur du pixel (x,y) dans l'image binarisée.

Ir(x,y) : valeur du niveau de rouge pour le pixel (x,y) dans l'image initial.

Ig(x,y) : valeur du niveau de vert pour le pixel (x , y) dans l'image initial.

Ib(x,y) : valeur du niveau de bleu pour le pixel (x , y) dans l'image initial.

D’après les tests effectués, différents paramètres peuvent affecter le choix du seuil, parmi ceux-ci :

- La lumière : les niveaux RGB d’une image varient en fonction du niveau de lumière ambiante. Pour pallier à ce problème de variation de lumière, la valeur du seuil a été rendue accessible depuis le l’écran menu de l’application (Fig. 3.3).
- L’homogénéité du support

Apres l’opération de seuillage, l’image est symbolisée par un tableau de booleancomme illustré en Annexe 2 ceci dans le but de minimiser l’espace mémoire occupé par chaque caractère car un boolean n’occupe que 1 octet comme illustré sur le Tableau IV.

Tableau IV : Les 8 type primitif du languate

Type	**Signification**	**Taille (octet)**
Char	Caractère unicode	2
Byte	Entier très court	1
Short	Entier court	2
Int	Entier	4
Long	Entier long	8
Float	Nombre simple réel	4
Double	Nombre décimale	8
Boolean	Valeur logique	1

3.5.3. Segmentation de l’image

La segmentation d’un caractère dans l’image nécessite une segmentation de colonne et une segmentation de ligne. Dans cette étude, les textes à segmenter seront des équations mathématiques, donc pour des cas particuliers de forme d’équation, il sera nécessaire de réitérer ces deux opérations pour isoler les variables et constantes composant les fonctions. Après une opération de segmentation de ligne et de colonne, on a des blocs d’éléments de l’équation qui nécessitent encore plusieurs niveaux de segmentation. Prenons par exemple une équation comportant une fraction illustrée par laFigure 3.8, tous les éléments de la fraction

seront englobés en un même bloc après une opération de segmentation de ligne et de colonne par projections horizontale et verticale. Il sera donc nécessaire de refaire la segmentation du bloc de fraction pour localiser le numérateur et le dénominateur ainsi que la barre de fraction.

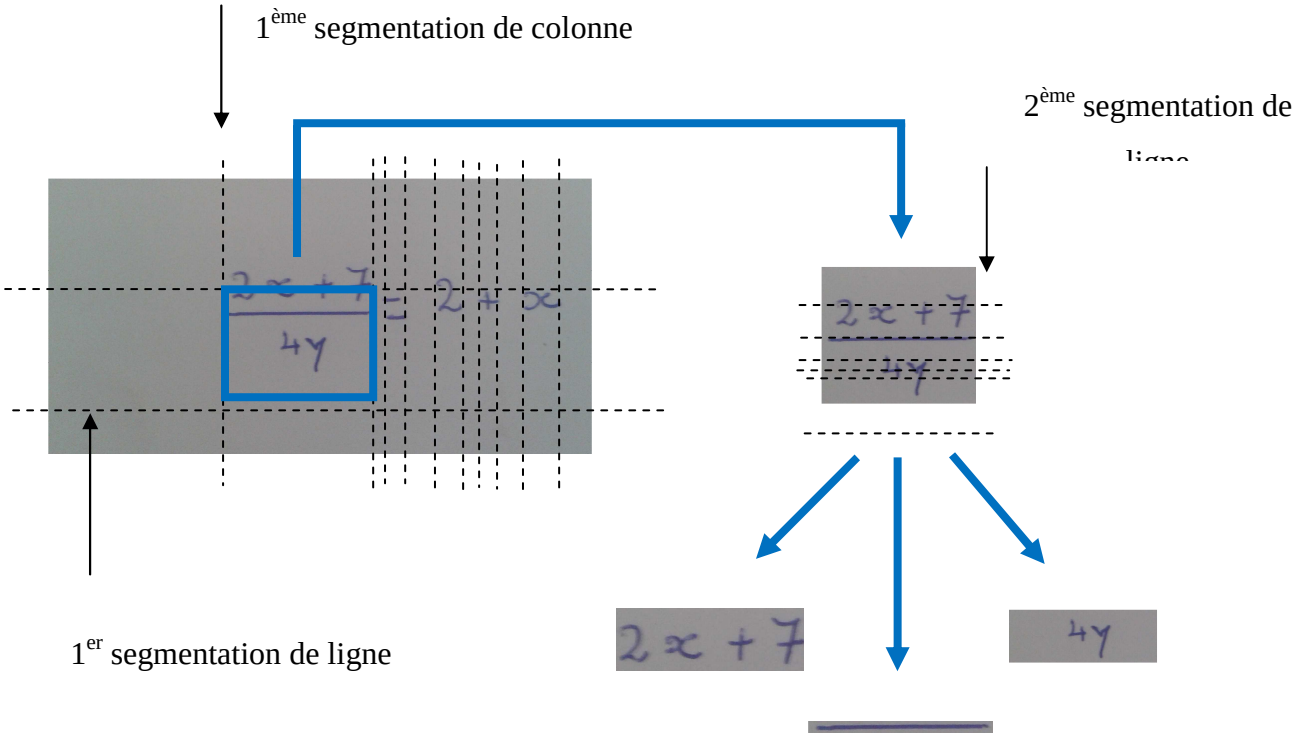

Figure 3.8 : Itération de la segmentation sur une équation

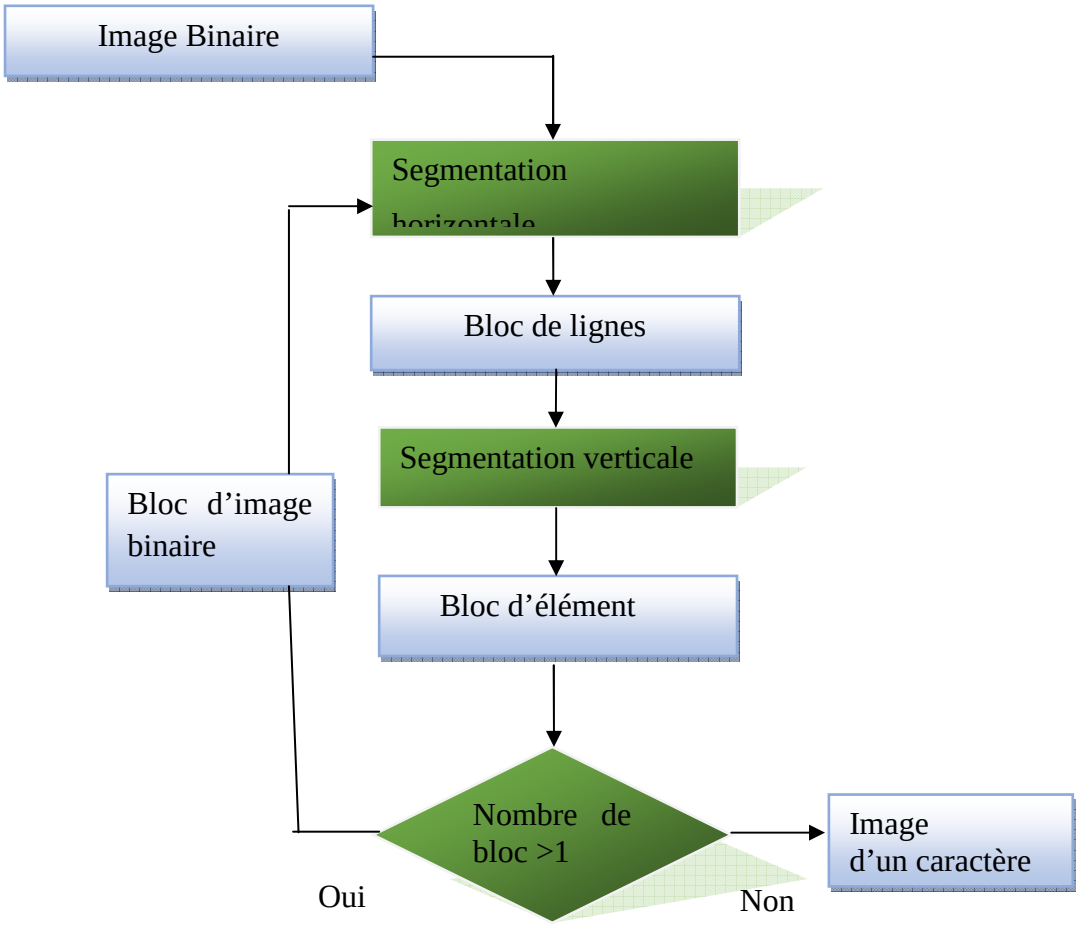

Figure 3.9 :Organigramme de segmentation

La Figure 3.9 présente l'organigramme de segmentation effectuée sur l'image binaire obtenue après prétraitement de l'image. A la fin de cette étape, l'algorithme de segmentation donne en sortie la liste des éléments segmentés comme illustrés en Annexe 2.

3.5.4. Reconnaissance des caractères segmentés

a. Extraction des caractéristiques

Nous avons vu au chapitre précédent les différentes caractéristiques qu'on va utiliser pour différencier un caractère d'un autre. Pour leur implémentation, on va réduire au maximum les données à traiter sans pour autant perdre les informations nécessaires.

- La dimension des caractères

En se basant sur le rapport entre la largeur et la hauteur d'un caractère, on peut isoler certain caractère comme illustré ci-après :

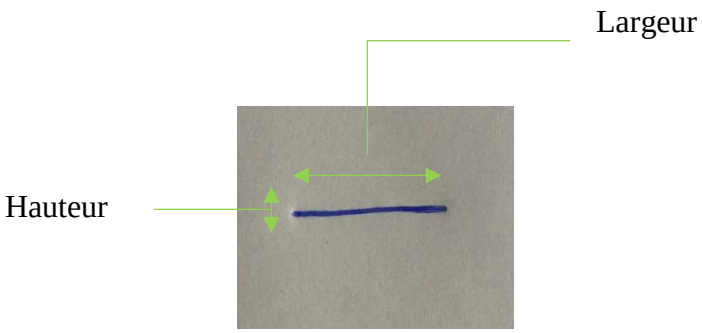

Figure3.10:Rapport largeur / hauteur

On peut ainsi voir une base d'apprentissage liée au rapport largeur/hauteur de l'image comme représenté sur le Tableau V.

Tableau V : Base d'apprentisage isssue du rapport Largeur /hauteur

Caractère	
-	Largeur = 10 * Hauteur
(	Hauteur =10 *Largeur
|	Hauteur = 10 *Largeur

- Projection horizontale et verticale de l'histogramme

Pour déterminer les traits dominants dans un caractère, on a établi un algorithme de réduction des projections horizontale et verticale en deux niveaux, tels qu'on le voit sur l'illustration en Figure 3.11.

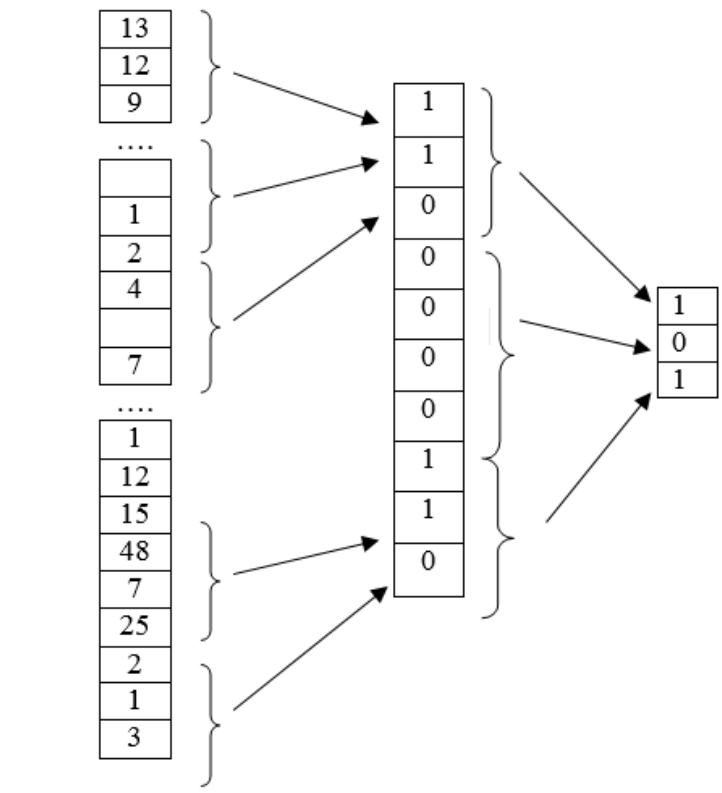

Figure 3.11 : Réduction de la projection horizontale et verticale de l'histogramme

Grâce à cet algorithme, on a pu établir une base d'apprentissage de quelque caractère(Tableau VI)

Tableau VI : Base d'apprenstissage issue des projections horizontale et verticale

Caractère	Projection horizontale	Projection verticale
0	101	101
1	001	010
2	101	010
3	111	001
4	010	010
5	111	101
6	111	101
7	100	001
8	111	101
9	111	101
+	010	010
-	111	111
x	101	101

- Concentration de pixel

Lors des tests effectués, on s'est aperçu qu'il y a des cas où l'utilisation de l'histogramme pour localiser les traits dominants dans un caractère manuscrit est insuffisante surtout quand les lignes qui devraient se rapprocher de l'horizontale ou de la verticale sont plus ou moins obliques comme illustré sur la Figure3.12.

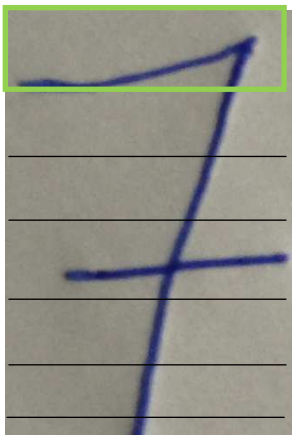

Figure3.12 : Concentration horizontale d'un caractère

On se retrouve alors avec un tableau de concentrations verticale et horizontale de pixel pour un caractère. On utilise le même algorithme de réduction de données utilisé pour les projections horizontale et verticale pour réduire les données, on a aussi la même base d'apprentissage, mais le taux de reconnaissance augmente en combinant les données provenant de la projection de l'histogramme avec les données de la concentration de pixel.

- Détection des concavités

A ce stade, il y a encore des confusions entre certains caractères. Ces confusionsse posent surtout dans le cas des nombres 3, 5, 6, 8, 9. Pour pallier à ce problème, on a implémenté un algorithme de détection des concavités. Comme évoqué au chapitre 2, les concavités peuvent être localisées en envoyant un certain nombre de « sondes » vers le caractère. Pour l'implémentation, on détecte le premier pixel trouvé à gauche et à droite du caractère, c'est-à-dire à 30%, 60 %, 90 % de sa hauteur comme illustrée sur la Figure3.13.

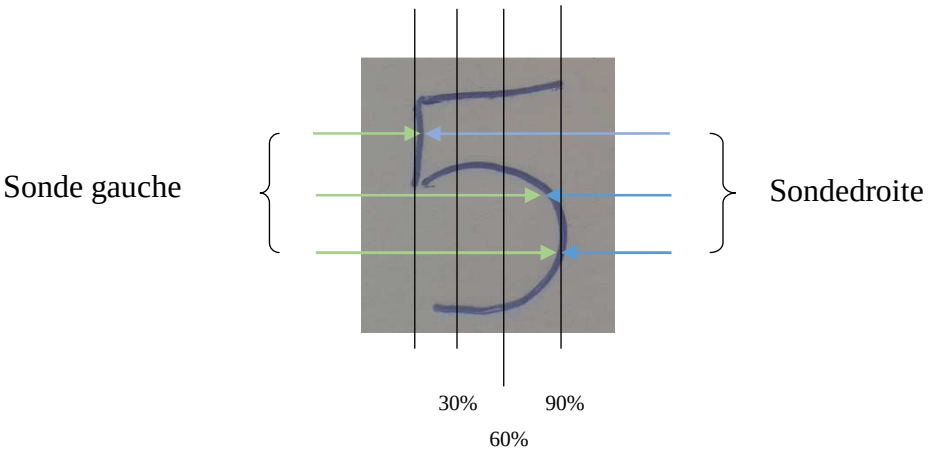

Figure3.13 : Détection des concavités

Dans cet algorithme, plus une sonde progresse dans un caractère, plus sa valeur augmente. Pour cette illustration de la Figure3.13 on a les valeurs suivantes :

- Sonde gauche : 0 2 2
- Sonde droite : 2 0 0

On a alors une base d'apprentissage pour différencier les caractères avec les mêmes projections d'histogramme et de concentration de pixel.

Tableau VII : Base d'apprentissage issue de la détection des concavite

Caractère	Sonde gauche	Sonde droite
3	010	000
5	022	200
6	000	100
8	010	010
9	022	000

- Intersection du caractère avec l'horizontale et la verticale

Pour renforcer la base d'apprentissage, on détecte l'intersection d'un caractère avec l'horizontale et la verticale comme illustrée sur la Figure3.14 en se basant sur la même technique des sondes. On a alors en sortie pour chaque sonde le nombre d'intersection avec le caractère. Pour cet exemple de la Figure3.14, les valeurs de sonde sont :

- Sonde verticale : 222
- Sonde horizontale : 212

Sonde verticale Intersections

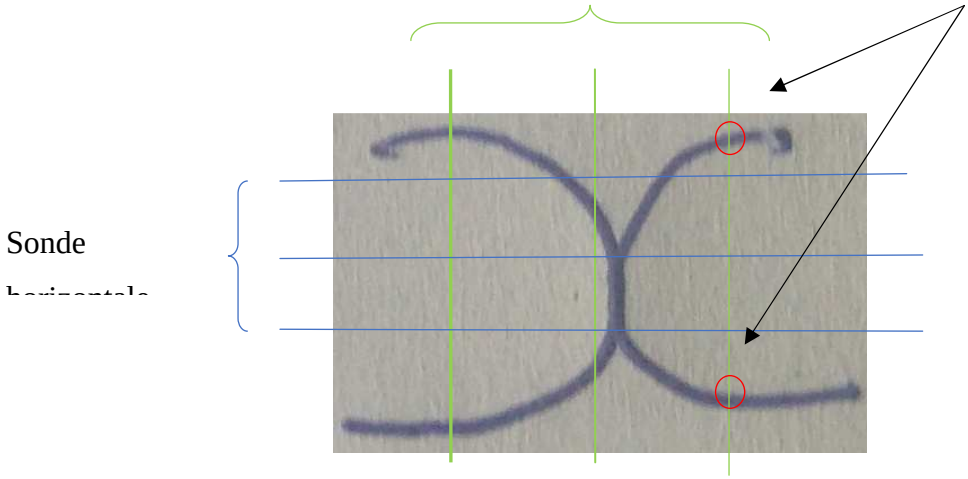

Figure3.14 : Intersection d'un caractère avec l'horizontale et la verticale

On a alors une base d'apprentissage basée sur l'intersection d'un caractère avec l'horizontale et la verticale (Tableau VIII).

Tableau VIII : Base d'apprentissage issue de l'intersection d'un caractère avec l'horizontale et la verticale

Caractère	Intersection avec l'horizontale	Intersection avec la verticale
0	222	222
3	111	234
5	111	333
9	221	333
X	222	222

b. Mis en place du réseau de neurone

Comme évoqué précédemment, on va utiliser un réseau de neurone pour la reconnaissance des caractères. Plus précisément, on met en place un réseau de neurone monocouche non bouclé (Figure 3.15) composé de:

- Neurone caché : il a pour entrée la représentation binaire du caractère à reconnaître et a pour sortie les caractéristiques du caractère.
- Neurones de sorties : ils ont pour entrées la sortie du neurone caché. Chaque neurone de sortie représente un caractère à reconnaître et donne en sortie la probabilité que le caractère à reconnaître est la sienne. On obtient le caractère en prenant la plus grande des sorties.

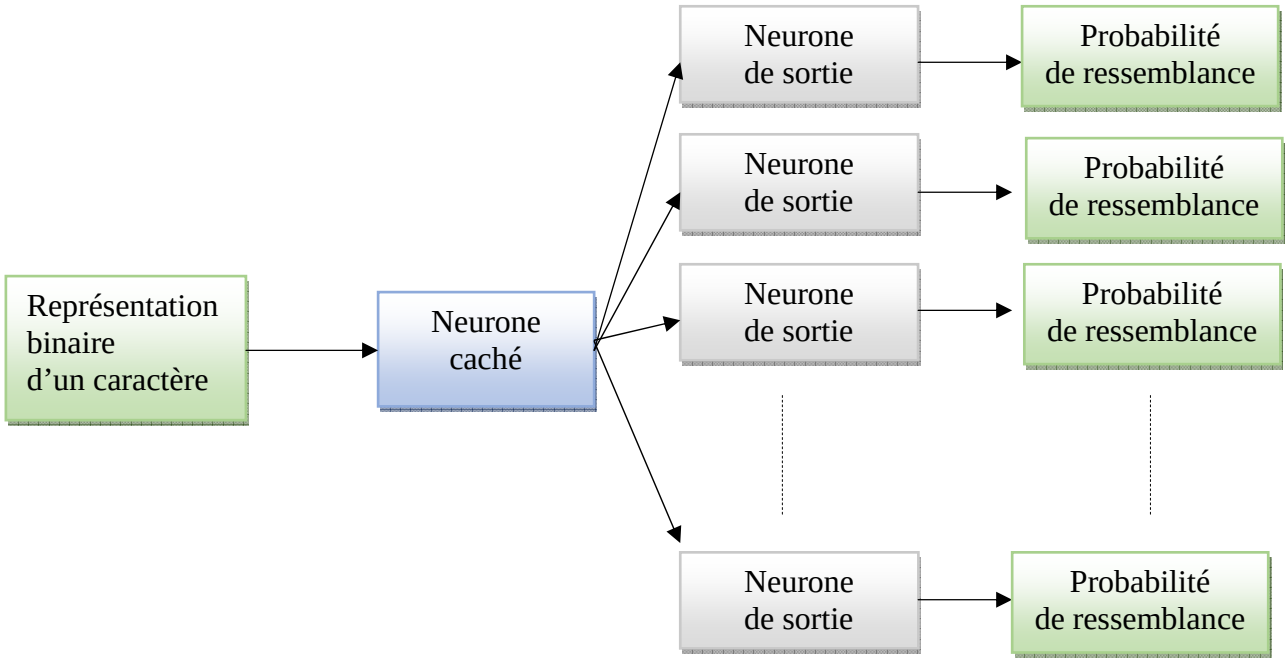

Figure 3.15 : Réseau de neurones utilisé

L'objectif est d'apprendre au réseau à reconnaître un caractère en entraînant d'abord sur une base d'apprentissage et ensuite le tester sur d'autre caractère. On a par exemple le taux de reconnaissance d'un caractère auTableau IX.

Tableau IX : Taux de reconnaissance du chiffre 6

Caractère	Neurones finale	Taux de reconnaissance
	0	10%
	1	0%
	2	0%
	3	10%
	4	0%
	5	10%
	6	100%
	7	0%
	8	10%
	9	10%
	+	0%
	-	0%
	/	0%
	x	0%

3.5.5. Reconnaissance et résolution d'équation

Le type d'une équation est défini suivant les éléments qui la composent comme illustré en Annexe 2. Les types d'équation et les formules pour les résoudre étant déjà définis au préalable. L'algorithme procède alors à une identification à partir d'une base existante suivant la forme de l'équation. On peut dire alors que l'algorithme responsable de la reconnaissance et la résolution d'une équation est un système expert. L'organigramme de cette étape est montré par la Figure 3.16

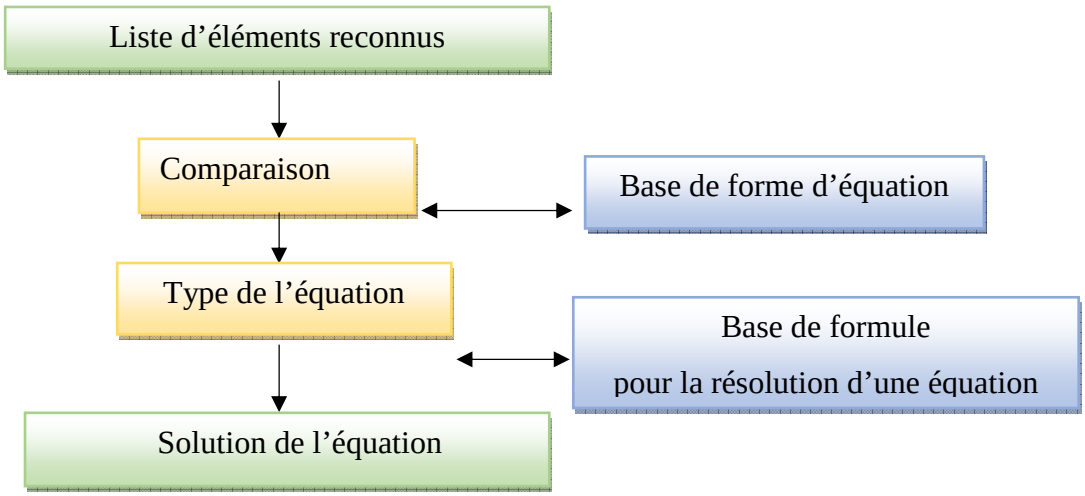

Figure 3.16: Organigramme de reconnaissance et résolution d'équation

3.6. Exemple de cas d'utilisation de l'application

Dans l'exemple qui suit, on va prendre une équation de second degré écrit sur une feuille de papier. On va suivre les étapes suivantes de la Figure3.17 :

- Une fois installée, on lance l'application depuis le dashboard du téléphone (Figure 3.7 a);
- On ouvre l'application et on choisit le mode capture par appareil photo dans les paramètres de l'application (Figure 3.7 b);
- On lance la capture et une fois l'équation dans le champ de capture, on enregistre (Figure 3.7 c);
- On arrive sur l'écran qui permet de sélectionner la partie de l'image qui contient vraiment l'équation à traiter (Figure 3.7 d);
- On revient sur l'écran d'accueil et le bouton « reconnaître » devient actif (Figure 3.7 e);
- Quand on clique sur le bouton « reconnaître », on passe à l'écran résultat qui affiche le type de l'équation et la solution de l'équation (Figure 3.7 f).

a) Dashboard b) Paramètre c) Capture

f) Sélection e) Reconnaissance d) Résultat

Figure3.17 : Etape d'utilisation de l'application