ECOLE DE MANAGEMENT ET D'INNOVATION TECHNOLOGIQUE

NOM et Prénoms: RAVOSON Tantelinirina Charlin
Numéro Matricule: 3838
Niveau: M1 INFO

Projet Deep Learning avec Réseau de Neurone à Convolution

THÈME : RECONNAISSANCE D'UNE FORME DES DIGITS DE 0 à 9

I - L'objectif de cette application:

L'objectif de cette application étant de créer un modèl de réconnaissance de forme en Deep learning en utilisant le réseau de neurone à convolution CNN .
Pour l'entreinnement du réseau , on utlise le DataSet Mnist , on s'interesse à connaître les digits de 0 à 9 déssiné manuellement dans une image.

II - Réseau de neurone à convolution CNN

CNN **[** Convolution, RELU, MAX POOLING **][** Fully Connected **]**

III - Langage de programmation utilisé:

JAVA «DeepLearning4J»:

- DeepLearning4J est une framework open source (licence Apache) qui permet de construire ,entraineret tester une grande diversité d'algorithmes de Deep Learning (depuis les réseaux standart,jusqu'aux réseaux à convolutions , en passant par des architecture plus complexes).

- Il se basé sur sa structure de données (ND4j) permettant d'effectuer les operations de l'algèbre linéaires sur les architectures massivements parallèles GPU et les architectures distribuées.

- **ND4j** utilise du code natif (Cuda oblige) et alloue de l'espace or du tas java . Ceci est imperativement à prendre en compte lorsque la volumétrie des données est importante.

- DL4J utilise **DataVec** pour la vectorisation et la transformation des données.

ND4J:

- **ND4J** est une bibliothèque de calcul scientifique et d'algèbre linéaire en langange de programmation Java et compatible avec d'autre langage des tels que Scala et Cotlin. Le ND4J a été versé à la fondation en onctobre 2017.

- ND4J perment de réaliser des manipulations des calculs linéaires et matricièl dans un environnement de production.

- Elle s'integre à apache Hadoop et **Spark pour** fonctionner avec des unités centrales de traitemant(UC) ou des unités de traitement graphiques(**GPU**). Il prend en charge les tableaux n-dimentionnels pour les langages basés sur JVM.

ND4J est un logiciel gratuit et à code source ouvert , publié sous Apache Licence 2.0 et développé principalement par le groupe basé à San Francisco qui a créé DeepLearning4J, sous une licence Apache.

IV - Démarches du projet:

Initialisation des variables utilisées dans le model :
On initialise plusieurs variables pour utiliser dans le model

```
String basePath = System.getProperty("user.home")+"/mnist";
System.out.println(basePath);
int height=28;int width=28;
int channels=1;// signe channel for graysacle image
int outputNum=10;// 10 digits classification
int batchSize=54;
int epochCount=1;
int seed =1234;
Map<Integer,Double> learningRateByIterations=new HashMap<>();
learningRateByIterations.put(0,0.06);
learningRateByIterations.put(200,0.05);
learningRateByIterations.put(600,0.028);
learningRateByIterations.put(800,0.006);
learningRateByIterations.put(1000,0.001);
double quadraticError=0.0005;
double momentum=0.9;
Random randomGenNum=new Random(seed);
```

Creation du model:
Voici une capture d'écran qui décrit les codes de la configuration du model .
Le model est enregistré dans un fichier .zip « model.zip ».

```
MultiLayerConfiguration configuration=new NeuralNetConfiguration.Builder()
        .seed(seed)
        .l2(quadraticError)
        .optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
        .updater(new
Nesterovs(newMapSchedule(ScheduleType.ITERATION,learningRateByIterations),moment
um))
        .weightInit(WeightInit.XAVIER)
        .list()
          .layer(0,new ConvolutionLayer.Builder()
                .kernelSize(3,3)
                .nIn(channels)
                .stride(1,1)
                .nOut(20)
                .activation(Activation.RELU).build())
           .layer(1, new SubsamplingLayer.Builder()
                 .poolingType(SubsamplingLayer.PoolingType.MAX)
                 .kernelSize(2,2)
                 .stride(2,2)
                 .build())
          .layer(2, new ConvolutionLayer.Builder(3,3)
                .stride(1,1)
                .nOut(50)
                .activation(Activation.RELU)
                .build())
          .layer(3, new SubsamplingLayer.Builder(SubsamplingLayer.PoolingType.MAX)
                .kernelSize(2,2)
                .stride(2,2)
                .build())
          .layer(4, new DenseLayer.Builder()
                .activation(Activation.RELU)
                .nOut(500)
                .build())
          .layer(5,new
OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
                .activation(Activation.SOFTMAX)
                .nOut(outputNum)
                .build())
          .setInputType(InputType.convolutionalFlat(height,width,channels))
        .backpropType(BackpropType.Standard)
        .build();
 System.out.println(configuration.toJson());
 MultiLayerNetwork model=new MultiLayerNetwork(configuration);
```

```
model.init();

UIServer uiServer=UIServer.getInstance();
 StatsStorage statsStorage=new InMemoryStatsStorage();
 uiServer.attach(statsStorage);
 model.setListeners(new StatsListener(statsStorage));

 logger.info("Total params:"+model.numParams());

 for (int i = 0; i < epochCount; i++) {
    model.fit(trainDataSetIterator);
    Evaluation evaluation=model.evaluate(testDataSetIterator);
    trainDataSetIterator.reset();
    testDataSetIterator.reset();
 }

 logger.info("Enregistrement model ....");
 ModelSerializer.writeModel(model,new File(basePath+"/model.zip"),true);
```

Entrainnement du model

Voici l'extrait du code de l'entrainnement du model

```
File trainDataFile=new File(basePath+"/trainingSet");
FileSplit trainFileSplit=new FileSplit(trainDataFile,
NativeImageLoader.ALLOWED_FORMATS,randomGenNum);
ParentPathLabelGenerator labelMarker=new ParentPathLabelGenerator();
ImageRecordReader trainImageRecordReader=new
ImageRecordReader(height,width,channels,labelMarker);
trainImageRecordReader.initialize(trainFileSplit);
int labelIndex=1;

DataSetIterator trainDataSetIterator=new
RecordReaderDataSetIterator(trainImageRecordReader,batchSize,labelIndex,outputNum
);
DataNormalization scaler=new ImagePreProcessingScaler(0,1);
scaler.fit(trainDataSetIterator);
trainDataSetIterator.setPreProcessor(scaler);
```

Evaluation du model

Voici l'extrait du code de l'évaluation du model

```
File testDataFile=new File(basePath+"/testing");
FileSplit testFileSplit=new FileSplit(testDataFile,
NativeImageLoader.ALLOWED_FORMATS,randomGenNum);
ImageRecordReader testImageRecordReader=new
ImageRecordReader(height,width,channels,labelMarker);
testImageRecordReader.initialize(testFileSplit);
DataSetIterator testDataSetIterator=new
RecordReaderDataSetIterator(testImageRecordReader,batchSize,labelIndex,outputNum);
trainDataSetIterator.setPreProcessor(scaler);
```

Interface de DL4J Training UI pour visualiser l'évolution du model

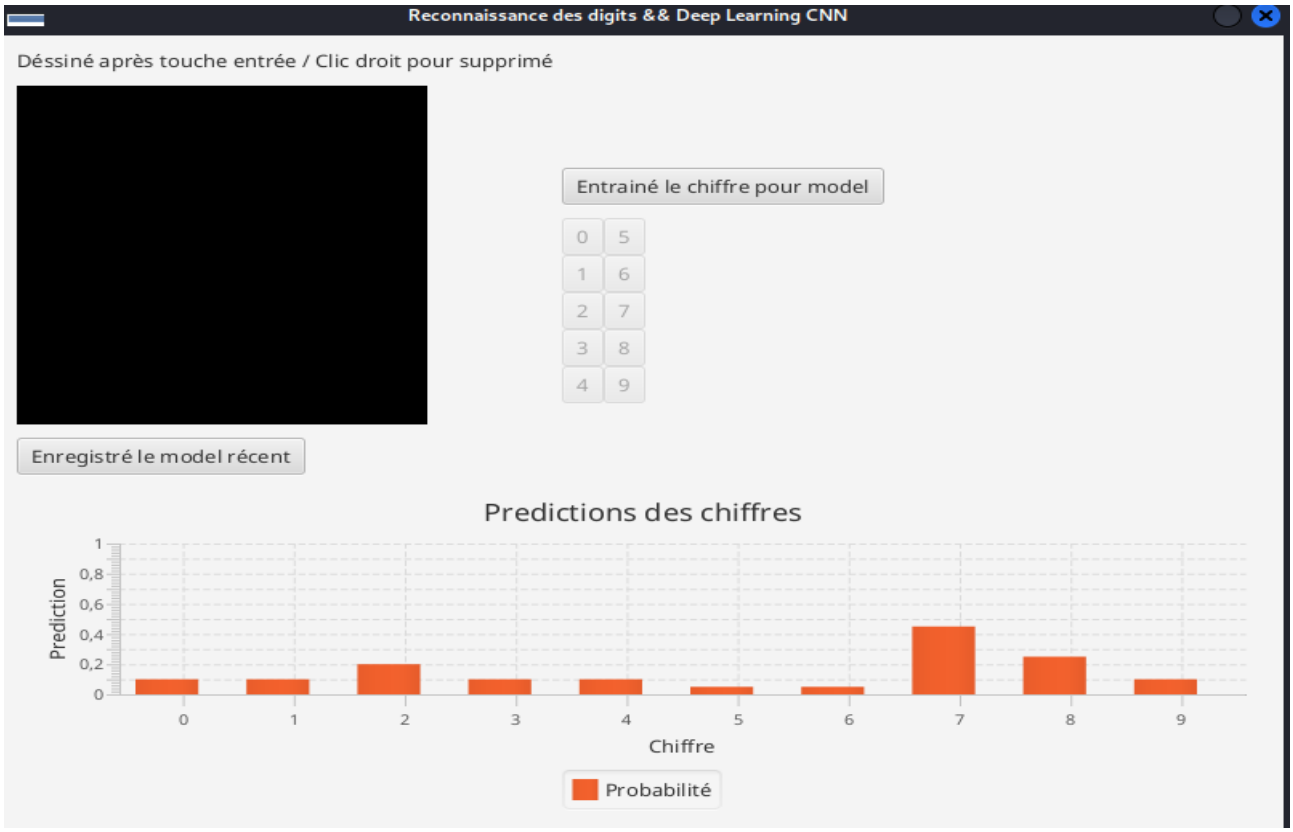
Visualisation d’évolution du model via le terminale

```
========================Evaluation Metrics========================
 # of classes:    10
 Accuracy:        0,9703
 Precision:       0,9713
 Recall:          0,9701
 F1 Score:        0,9701
Precision, recall & F1: macro-averaged (equally weighted avg. of 10 classes)
```

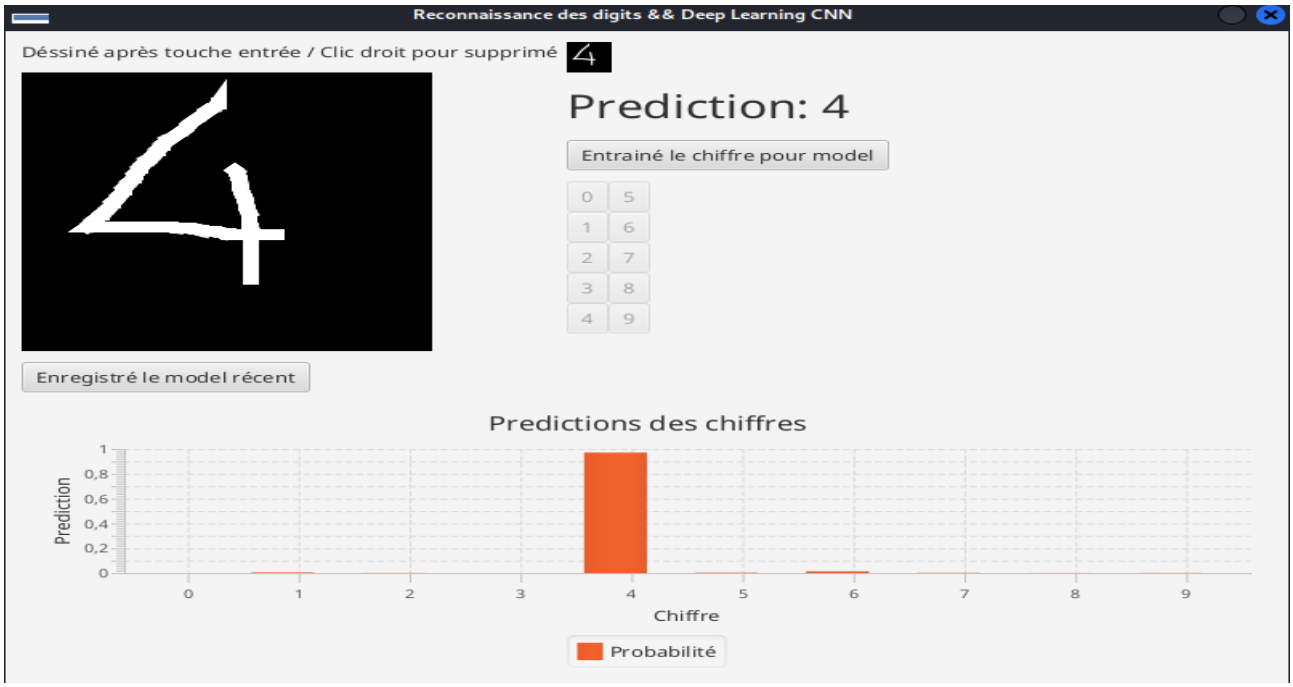
```
 =========================Confusion Matrix=========================
    0    1    2    3    4    5    6    7    8    9
 ------------------------------------------------------
  4122    0    0    0    4    0    4    0    1    1 | 0 = 0
     0 4668    2    0   10    0    1    0    3    0 | 1 = 1
    16   37 4030    1   37    0    7   21   23    5 | 2 = 2
    25   10   25 4116    6   46    1    6  100   16 | 3 = 3
     1    5    0    0 4059    0    5    1    0    1 | 4 = 4
    28    4    0    2   15 3629   41    3   58   15 | 5 = 5
    13    4    0    0   15    1 4096    0    8    0 | 6 = 6
     4   50   13    1   43    0    0 4261   13   16 | 7 = 7
    11   28    3    0   48    1    7    0 3960    5 | 8 = 8
    15   10    0    0  316    2    2   14   19 3810 | 9 = 9

 Confusion matrix format: Actual (rowClass) predicted as (columnClass) N times
 ==================================================================
```

V - Démostration de l'application

On utilise JAVAFX pour l'interface graphique de l'application .
Pour utiliser l'application , déssinez dans le zone de déssin coloré en noire puis tapez sur le touche entrer pour visualiser le resultat sous forme de diagramme.

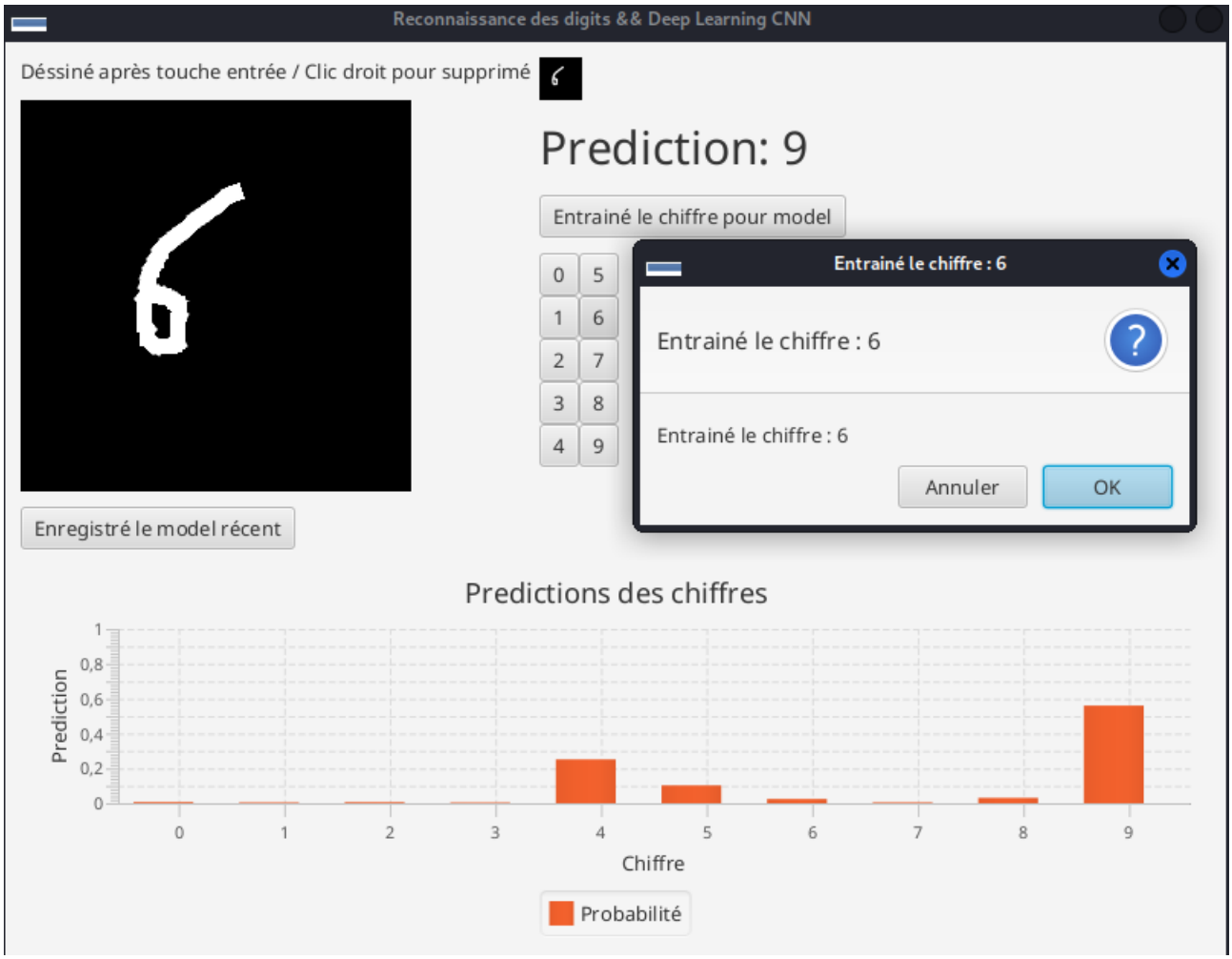


Résultat après :
On prédite le chiffre 4 est voilà le resultat , il connaît que c'est un 4 et il affiche sous forme de diagramme . Clic droit pour effacé le dessin dans le zone .



En cas de tromper le model :

Si le model trompe comme dans l'image ci-dessous , On va cliquer sur le bouton « Entrainer le chiffre pour model » c'est à dire qu'on va dire au model que c'est un 6 mais pas 9 , après cliquer sur le bouton « ok » pour enregistrer.



En résumé:

cette application est crée pour reconnaître la forme d'un digit . L'objectif pour la réalisation du projet c'est de reconnaître les fonctionnements du CNN (Réseau de neurone à convolution).

Donc , CNN est un type de réseua de neurone artificiels , dans lequel le motif de connexion entre les neurones est inspiré par le cortex visuel des annimaux.

Utilisés dans la reconnaissance d'image et vidéo, les systèmes de recomandation et le traitement du langage naturel.

Ils reçoivent des images en entrée , détectent les features (caracteristiques) de chacune d'entre elles , puis entrainent un classifieur.

Un CNN se compose des différentes couches suivantes :

- Couches de convolution qui consiste à appliquer un filtre de convolution à l'image pour détecter des caracteristiques de l'images.
- Couche de correlation RELU (Rectiied Linear Unit) qui consiste à remplacer les nombres négatifs des images filtrées par des zéros.
- Couches de Pooling qui consiste à résuire la taille de l'image en ne gradant les informations les plus importantes . Par exemple pour chaque groupe de 4 pixels ayant la valeur mximale (Max Pooling).
- Couche entièrement connectée (Fully Connected) qui reçoie un vecteur en entrée contenant les pixels aplatis ed toutes les images filtrées et reduites par le pooling .