SIMPLON
OUTRE-MER

Société Publique Locale
afpar

BRIEF 1A 10/2022

DOCUMENTATION TECHNIQUE

Ugo FOK-YIN 03

DIA AFPAR

Lien GitHub du projet: [UgoDIA/AiToy (github.com)]

1. Contexte du projet

En tant que développeur en Intelligence Artificielle, vous venez d'intégrer YAYA2IST une petite entreprise spécialisée dans la conception de jouets, et notamment de petits robots.

Les ingénieurs de l'entreprise pensent qu'ils pourraient être extrêmement intéressant d'équiper NoNo d'une caméra et de le rendre capable de reconnaître plusieurs dizaines d'objets communs, afin d'améliorer son interactivité avec les utilisateurs.

Pour étudier la faisabilité de ce projet, ils vous demandent de réaliser une POC (Proof Of Concept), sur une base web simple (navigateur + webcam), de détection et reconnaissance d'objet, avant de se lancer plus en avant dans ce projet .

A terme, ce projet pourrait embarquer une série d'activités ludiques fortement interactives pour jeunes enfants : son, caméra, jeu ...

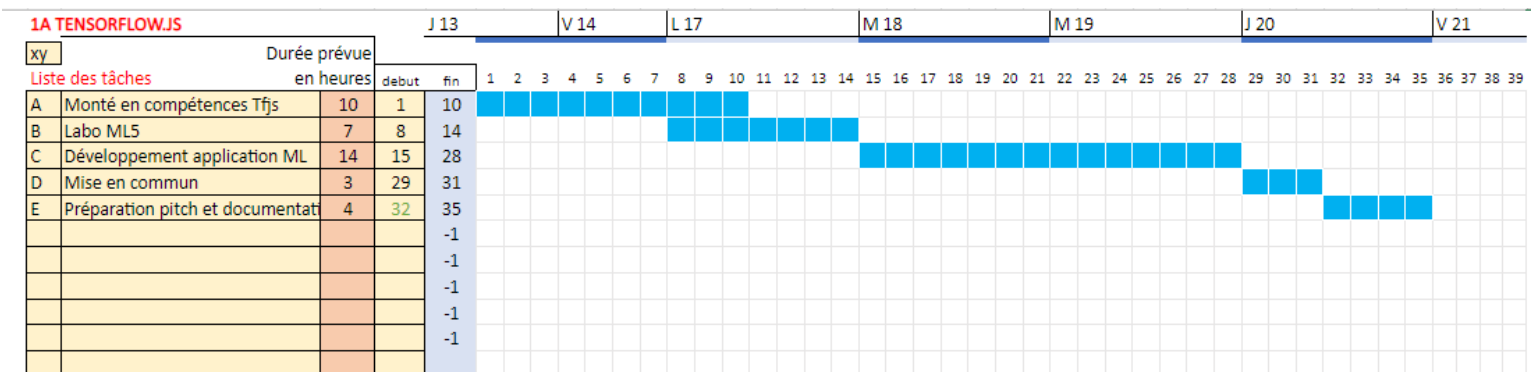
2. Objectifs et plan d'action

Objectif:

Mon objectif pour ce projet était de développer un classifieur de formes afin de pouvoir reconnaître des formes simples dessinées sur du papier et afficher les résultats .

Comme tâche bonus j'ai voulu que mon classifieur reconnaisse aussi les couleurs ainsi que des petits dessins simples.

Plan d'action:



3. Ressources et outils utilisées

ML5.js:

ML5.js est une librairie JavaScript basée sur TensorFlowJS et permet de rendre le machine learning beaucoup plus accessible.

ml5js·Friendly Machine Learning For The Web

P5.js:

P5 est une librairie JavaScript qui permet de simplifier JavaScript et le rend plus accessible et peu faire pas mal de choses avec ML5.js.

reference | p5.js (p5js.org)

Processing

Processing:

Processing est une librairie et IDE basé sur du Java et est plus utilisé pour du design visuel.

Reference / Processing.org

Daniel shiffman de Coding Train:

A Beginner's Guide to Machine Learning with ml5.js

Je me suis beaucoup inspiré des vidéos de Daniel Shiffman sur ml5.js.

4. Étapes de développement de l'appli

Pour ce projet je souhaite donc réaliser une application qui permet de reconnaître 3 formes(cercle, carré et triangle) et 3 couleurs(rouge, vert, bleu) à partir d'une webcam.

A. Dataset

Pour commencer, la première étape importante en machine learning c'est d'avoir un dataset, j'utilise donc Processing pour générer 100 images pour chacune des 3 formes et je fais la même chose pour les 3 couleurs.

Démonstration vidéo Processing Formes

B. Création et entraînement d'un réseau de neurones pour le modèle

Chargement des formes générées:

J'utilise la fonction preload et la méthode loadImage qui viennent de P5.js afin de récupérer les 100 images pour chaque forme avec une simple boucle for et je les place dans 3 variables.

```
function preload(){
    for(let i = 0; i < 100; i++) {
        let index = nf(i+1,4,0)
        cercle[i] = loadImage(`data/cercle${index}.png`);
        carré[i] = loadImage(`data/carré${index}.png`);
        triangle[i] = loadImage(`data/triangle${index}.png`);
    }
}
```

Paramétrage du réseau de neurones et ajout des données:

J'utilise la méthode neuralNetwork de ML5.js et je précise dans options les dimensions qui me conviennent et je précise aussi que l'on veut utiliser ce réseau de neurones pour faire de la classification d'images.

J'ajoute ensuite toutes les données qui sont stockées dans les 3 variables et les ajoute au réseau et donne un label pour chaque classe.

```
let options={
        inputs:[64,64,4],
        task: 'imageClassification',
        debug: true,
    };
    formesClassifier = ml5.neuralNetwork(options);
    for (let i = 0; i<cercle.length; i++){
        formesClassifier.addData({image:cercle[i]},{label:"Cercle"});
        formesClassifier.addData({image:carré[i]},{label:"Carré"});
        formesClassifier.addData({image:triangle[i]},{label:"Triangle"});
    }
```

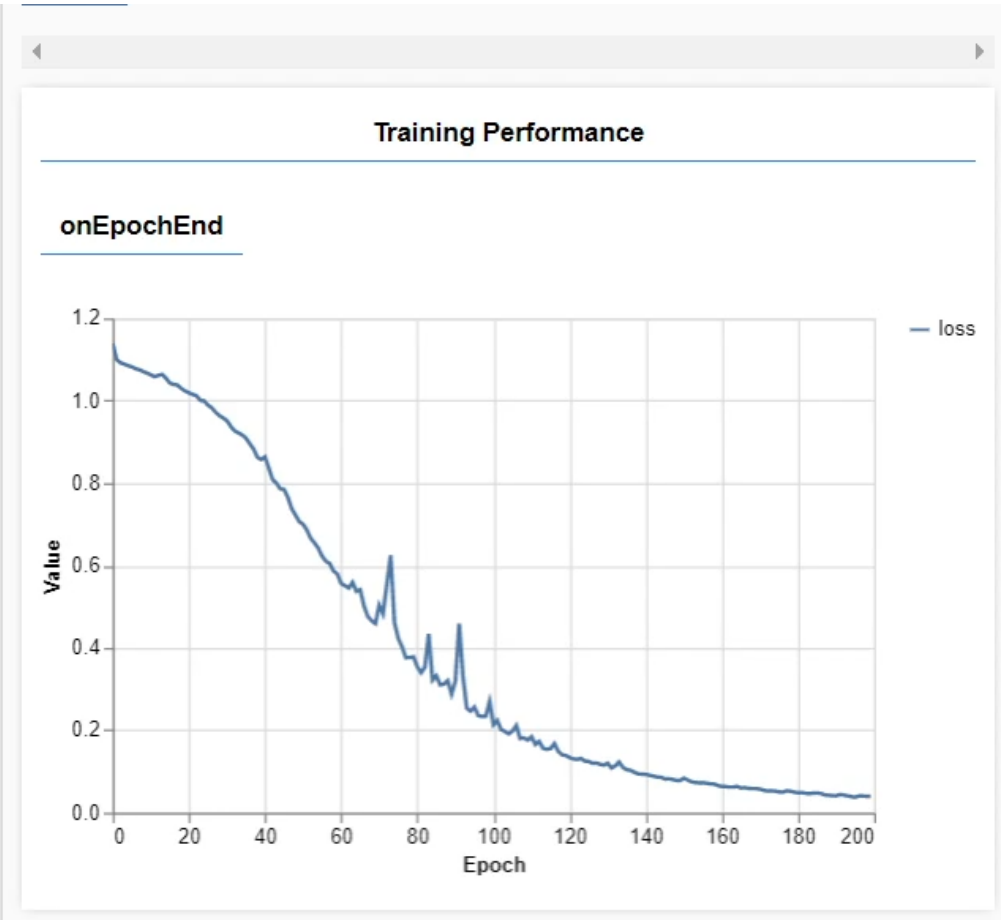
Normalisation des données puis lancement de l'entraînement et enfin sauvegarde du modèle obtenu:

J'utilise la méthode normalizeData pour normaliser les données puis je lance l'entraînement avec la méthode train et je précise dans les paramètres que l'entraînement va durer 150 epochs et enfin j'attribue une fonction callback appelé finishedTraining qui se lance quand l'entraînement sera terminé, et dans cette fonction callback on utilise la méthode save afin de sauvegarder le modèle final.

```
    formesClassifier.normalizeData();
    formesClassifier.train({epochs:150}, finishedTraining);
}
function finishedTraining(){
    console.log("finished training!");
    formesClassifier.save()
```

Courbe d'apprentissage obtenu :

Lors de l'entraînement on peut observer la courbe d'apprentissage en train de se dessiner et elle permet de savoir si on obtient un bon modèle si la fonction coût est très proche de zéro.
À la fin de l'entraînement on récupère 3 fichiers json qui représentent le modèle, je répète le processus pour avoir le modèle des couleurs.



Demo training:

Démonstration vidéo Training ML5

C. Programmation de l'appli de classification

Chargement des modèles:

Je charge les 2 modèles(formes et couleurs) obtenu dans un nouveau sketch avec la méthode .load et j'attribue comme paramètres les chemins d'accès aux fichiers json et une fonction callback modelLoaded qui se déclenche lorsque les modèles auront fini de charger.

```
formesClassifier = ml5.neuralNetwork(options);
const modelDetails = {
  model: 'model/model.json',
  metadata: 'model/model_meta.json',
  weights: 'model/model.weights.bin'
}
CouleurClassifier = ml5.neuralNetwork(options);
const modelDetailsCoul={
  model:'modelCouleur/model.json',
  metadata :'modelCouleur/model_meta.json',
  weights:'modelCouleur/model.weights.bin',
}
background(255);
formesClassifier.load(modelDetails, modelLoaded);
CouleurClassifier.load(modelDetailsCoul,modelLoaded);
```

Lancement de la classification:

Une fois le modèle prêt on peut lancer la fonction keyPressed qui vient de P5.js et permet de lancer la fonction lorsqu'on appuie sur n'importe quelle touche du clavier, on utilise la méthode classify pour les 2 modèles avec comme input la webcam qui permettra donc de faire une classification à partir de la webcam et enfin 2 fonctions callback pour les résultats de chaque classification.

```
function modelLoaded(){
  console.log("modèle prêt")
  keyPressed();
}

function keyPressed(){
  formesClassifier.classify({image:video},gotResults);
  CouleurClassifier.classify({image:video},gotResultsCoul);
}
```

Affichage des résultats:

On récupère le label et le pourcentage de confiance pour les afficher sur le html

```
function gotResults(err, results){
  if(err) {
    console.error(err);
    return;
  }
  let label=results[0].label;
  let confidence=nf(100*results[0].confidence,2,0 );
  document.getElementById("results").innerHTML= label +" "+ confidence + "%" ;
  console.log(results);
}
```

```
function gotResultsCoul(err, results){
  if(err) {
    console.error(err);
    return;
  }
  let label=results[0].label;
  let confidence=nf(100*results[0].confidence,2,0 );
  document.getElementById("ResultsCoul").innerHTML= label + " " + confidence
+"%";
  console.log(results);
}
```

5.Démonstration

[Démonstration Vidéo]

6. Conclusion et TO DO LIST

Objectif principal accompli

TO DO LIST:

- Améliorer les modèles
- Ajouter plus de formes
- Ajouter plus de couleurs
- Classification de dessins simples