

**RENDU DU PROJET**

**PARTIE : DETECTION DE TEXTE**

**AUTEUR : GODFREE AKAKPO**



Le présent document constitue le rendu officiel du projet rassemblant les notions de détection d'objet, en occurrence dans ce projet précis la détection de texte élaboré dans le cours « Projet informatique » au sein de l'université Claude Bernard Lyon1. Ce projet a été conçu et réalisé par Godfree AKAKPO, étudiants en licence 3 informatique à l'université de Lyon 1.

Nb : Je rappelle que ce projet est l'une des parties d'un projet qui consiste à détecter et à reconnaître du texte. Une partie qui peut être utilisée indépendamment. Donc dans cette partie nous aurons à parler que de la partie concernant la détection de texte.

## **OBJECTIF DU PROJET**

L'objectif principal de ce projet était de concevoir, développer et mettre en œuvre un programme qui servira à vérifier s'il existe du texte dans un model ou pas si oui de nous ressortir les coordonnées associées aux textes. Ce programme est question n'est pas une révolution, ni une nouveauté puisque cela existe déjà et beaucoup plus performant.

## **TECHNOLOGIE UTILISEE**

Nous avons opté pour Python pour le développement de notre programme en raison de leur adaptabilité, de leur robustesse et de leur adéquation aux besoins du projet. Comme ce projet est beaucoup plus tourné vers le Deep Learning alors Python se présente comme l'outil idéal car il est beaucoup plus adapté.

## **ORGANISATION DU DOCUMENT**

Ce document est divisé en plusieurs sections pour faciliter la compréhension et l'évaluation du projet. Les sections incluent :

- Petit rappel sur les notions de base pour comprendre l'aspect technique du projet.
- Qu'est-ce qu'un Réseau de neurone et ses couches ?
- Fonction de Perte ( MSEloss)
- Les classes et fonctions principales du programme
- Les différents résultats
- Versions améliorées du projet.

En résumé, ce projet représente notre engagement envers l'excellence dans la conception et la mise en œuvre d'une solution informatique répondant aux besoins spécifiques énoncés dans le cahier des charges. Nous sommes impatients de présenter chaque aspect du projet dans les sections suivantes.

### **Petit rappel sur les notions de base pour comprendre l'aspect technique du projet.**

Tout d'abord il faut comprendre les notions mathématiques comme les matrices et tenseurs. Connaître et maîtriser les listes en python.

### **Qu'est-ce qu'un Réseau de neurone et ses couches ?**

Les réseaux de neurones, également connus sous le nom de réseaux de neurones artificiels (ANN) ou réseaux de neurones simulés (SNN) constituent un sous-ensemble de l'apprentissage automatique et sont au cœur des algorithmes de l'apprentissage en profondeur. Leur nom et leur structure sont inspirés du cerveau humain, imitant la manière dont les neurones biologiques s'envoient des signaux.

Source : <https://www.ibm.com/fr-fr/topics/neural-networks>

Généralement, chaque neurone d'une couche est lié avec tous les neurones de la couche en aval et celle-ci uniquement.

### **Fonction de Perte ( MSEloss)**

La fonction de perte ou de loss permet de calculer l'erreur entre les prédictions de ton modèle et les valeurs réelles.

**Plus la loss baisse, plus le modèle est performant !**

Pendant l'entraînement, cela permet au modèle de savoir s'il avance dans la bonne direction ou non.

L'objectif étant que loss soit le plus proche possible de zéro.

Cependant, il y a plusieurs manières de calculer l'erreur entre les prédictions et les valeurs réelles.

Dans notre programme spécialement on a utilisé la fonction MSEloss. Mais pourquoi avons-nous choisis la fonction MSEloss ? La Mean Squared Error (MSE), en français erreur quadratique moyenne, est une des fonctions de loss les plus utilisées dans les problèmes de régression.

Elle calcule la différence moyenne au carré entre les valeurs prédites et les valeurs réelles.

Donc pas de panique lorsque la valeur de retour est très éloignée 0. Cette valeur de retour est la valeur prédit par rapport à la valeur réelle.

## Les classes et fonctions principales du programme

***Class TextDetectionModel(nn.Module)***: cette classe définit l'objet Model. Elle permet d'instancier le model. Il est formé de 3 couches de convolution et une couche linéaire

***Class CustomDataset(Dataset)***: cette classe permet d'instancier un objet Dataset qui a pour rôle de charger les données

***DataLoader(custom\_dataset, batch\_size=128, shuffle=True)*** : cette fonction est une fonction prédéfinie de la bibliothèque torch.utils.data qui permet de charger les données par lots dans le but principal d'économiser de la mémoire.

***optim.Adam(model.parameters(), lr=0.0001)*** : cette fonction est la fonction qui définit l'optimiseur qui sera utilisé pour l'entraînement du model.

***Boucle d'apprentissage*** : dans cette boucle se passe l'apprentissage des données

***nn.MSELoss()(regression\_output, coordinates)*** : comme définie plus haut cette fonction communément appelée la loss qui est une clé du bon fonctionnement de ce programme. Puisque qu'on ne parle pas d'une probabilité mais plutôt d'une valeur à prédire ici alors La MSELoss est l'idéale à utiliser. Elle calcule la différence moyenne au carré entre les valeurs prédites et les valeurs réelles.

```

class TextDetectionModel(nn.Module):
    def __init__(self):
        super(TextDetectionModel, self).__init__()

        # Couches de convolution pour l'extraction des caractéristiques
        self.conv1 = nn.Conv2d(3, 6, kernel_size=3, stride=1, padding=1)
        self.relu1 = nn.ReLU()
        self.pool1 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

        self.conv2 = nn.Conv2d(6, 6, kernel_size=3, stride=1, padding=1)
        self.relu2 = nn.ReLU()
        self.pool2 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

        self.conv3 = nn.Conv2d(6, 10, kernel_size=3, stride=1, padding=1)
        self.relu3 = nn.ReLU()
        self.pool3 = nn.MaxPool2d(kernel_size=2, stride=2, padding=0)

        # Couches entièrement connectées pour la régression
        self.fc1 = nn.Linear(38440, 512)
        self.relu3 = nn.ReLU()
        self.fc3 = nn.Linear(512, 4) # 4 valeurs pour les coordonnées (x, y,
width, height)

    def forward(self, x):
        # Propagation avant à travers les couches de convolution
        x = self.conv1(x)
        x = self.relu1(x)
        x = self.pool1(x)
        x = self.conv3(x)
        x = self.relu3(x)
        x = self.pool3(x)

        # Aplatir les caractéristiques pour les couches entièrement connectées
        x = x.view(x.size(0), -1)

        # Propagation avant à travers les couches entièrement connectées
        x = self.fc1(x)

        # Sorties pour la classification (2 classes) et la régression (4 valeurs)
        x = self.fc3(x)
        regression_output = x

        return regression_output

model = TextDetectionModel()

class CustomDataset(Dataset):
    def __init__(self, csv_file, image_folder):
        self.data = pd.read_csv(csv_file)
        self.image_folder = image_folder
        self.transform = transforms.Compose([
            transforms.Resize((250, 250)),
            transforms.ToTensor(),
        ])

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        image_name = self.data.iloc[idx, 0]
        image_path = os.path.join(self.image_folder, image_name)

```

```

csv_path = '/content/drive/MyDrive/Classroom/csv_file/coordinates.csv'
images_path = '/content/drive/MyDrive/Classroom/img/'

# Créer une instance de votre Dataset personnalisé
custom_dataset = CustomDataset(csv_path, images_path)

dataloader = DataLoader(custom_dataset, batch_size=128, shuffle=True)

```

```
optimizer = optim.Adam(model.parameters(), lr=0.0001)
```

```

#-----
checkpoint_path =
'/content/drive/MyDrive/Classroom/train_ckeck/model.pth'
if os.path.exists(checkpoint_path):
    checkpoint = torch.load(checkpoint_path)
    model.load_state_dict(checkpoint['model_state_dict'])
    optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
    epoch = checkpoint['epoch']
    loss = checkpoint['loss']
    distance = checkpoint['distance moyenne']
    # Ajoutez d'autres éléments que vous avez sauvegardés

    print(f"Chargement du point de contrôle à l'époque {epoch} avec une
perte de {loss}")
else:
    print("Aucun point de contrôle trouvé, l'entraînement commencera
depuis le début.")

```

```

#-----
train_losses = []
iteration = 0
train_distances = []
n = 50
for epoch in range(n):
    model.train()

    for batch in dataloader:

        images = batch['image']
        coordinates = batch['coordonnees']
        """ print(f'Type of images: {type(images)}')
        print(batch)
        print(type(batch))
        print("images = ", type(images))
        print("coordinates = ", type(coordinates))"""

```

```

optimizer.zero_grad()

# Obtenez les sorties du modèle
regression_output = model(images)

# Calcul de la perte
#print(type(regression_output))
#print(regression_output)

coordinates = torch.stack([coord.float() for coord in coordinates],
dim=1)
#print(coordinates.shape)
#print(coordinates)
loss = nn.MSELoss()(regression_output, coordinates)

distance = torch.mean(torch.sqrt(torch.sum((regression_output -
coordinates)**2, dim=1)))
train_distances.append(distance.item())

# Rétropropagation et mise à jour des poids
loss.backward()
optimizer.step()
train_losses.append(loss)

print(f'Epoch {epoch+1}/{n}, Iteration {iteration}, Loss: {loss.item()},
Distance : {distance.item()}')

```

```

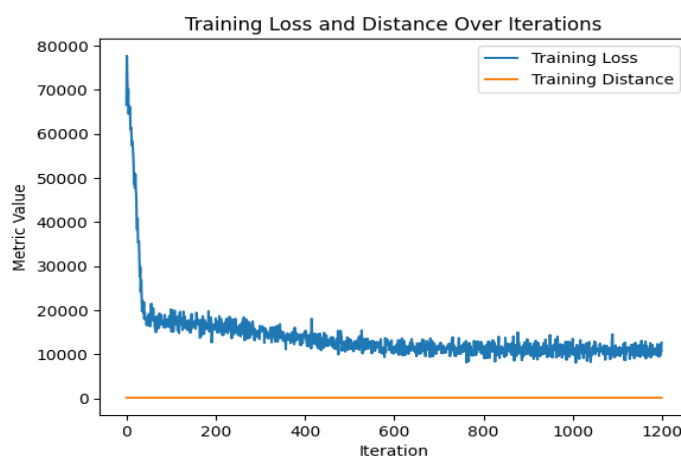
iteration += 1
# À la fin de votre boucle d'entraînement
checkpoint_path =
'/content/drive/MyDrive/Classroom/train_ckeck/model.pth'
torch.save({
    'epoch': epoch,
    'model_state_dict': model.state_dict(),
    'optimizer_state_dict': optimizer.state_dict(),
    'loss': loss,
    'distance moyenne': distance,
    # Ajoutez d'autres informations que vous souhaitez sauvegarder
}, checkpoint_path)
plt.plot([loss.item() for loss in train_losses], label='Training Loss')
plt.plot([distance.item() for dis in train_distances], label='Training
Distance')
plt.xlabel('Iteration')
plt.ylabel('Metric Value')
plt.title('Training Loss and Distance Over Iterations')
plt.legend()
plt.show()

```

## Les différents résultats

### 50 premières epoch de notre apprentissage en temps réel

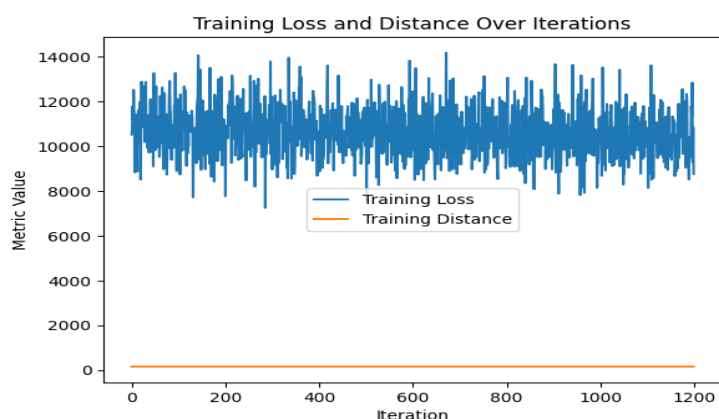
+ Code	+ Texte	+ Copier sur Drive
Epoch 19/50, Iteration 18, Loss:	(12959.53125,,), Distance :	209.23538288007812
Epoch 20/50, Iteration 19, Loss:	(12643.263671875,,), Distance :	201.4491729236328
Epoch 21/50, Iteration 20, Loss:	(11327.5625,,), Distance :	194.4766387939453
Epoch 22/50, Iteration 21, Loss:	(15455.089765625,,), Distance :	219.5759735107422
Epoch 23/50, Iteration 22, Loss:	(11859.685546875,,), Distance :	195.99618530273438
Epoch 24/50, Iteration 23, Loss:	(13246.15625,,), Distance :	185.1132823486328
Epoch 25/50, Iteration 24, Loss:	(10221.818359375,,), Distance :	177.1907958984375
Epoch 26/50, Iteration 25, Loss:	(10478.568359375,,), Distance :	180.0505828857422
Epoch 27/50, Iteration 26, Loss:	(10402.3974609375,,), Distance :	181.06103315625
Epoch 28/50, Iteration 27, Loss:	(11449.2763671875,,), Distance :	187.69180297851562
Epoch 29/50, Iteration 28, Loss:	(11543.830078125,,), Distance :	188.6119384765625
Epoch 30/50, Iteration 29, Loss:	(11701.830078125,,), Distance :	194.78163146972656
Epoch 31/50, Iteration 30, Loss:	(11857.8564453125,,), Distance :	194.72335815429688
Epoch 32/50, Iteration 31, Loss:	(8411.1474609375,,), Distance :	160.36753845214844
Epoch 33/50, Iteration 32, Loss:	(11249.5400390625,,), Distance :	187.5104237529297
Epoch 34/50, Iteration 33, Loss:	(12666.7470703125,,), Distance :	180.36027087402344
Epoch 35/50, Iteration 34, Loss:	(11573.8408203125,,), Distance :	189.57601928710938
Epoch 36/50, Iteration 35, Loss:	(9982.587890625,,), Distance :	167.52511596679688
Epoch 37/50, Iteration 36, Loss:	(10954.1323046875,,), Distance :	177.31967163885938
Epoch 38/50, Iteration 37, Loss:	(10330.2392578125,,), Distance :	179.52560520320312
Epoch 39/50, Iteration 38, Loss:	(12595.6240234375,,), Distance :	195.7135772705078
Epoch 40/50, Iteration 39, Loss:	(12339.373046875,,), Distance :	203.7322998046875
Epoch 41/50, Iteration 40, Loss:	(10176.5771484375,,), Distance :	174.8807398453125
Epoch 42/50, Iteration 41, Loss:	(8117.9189453125,,), Distance :	160.72680947753906
Epoch 43/50, Iteration 42, Loss:	(10234.8837890625,,), Distance :	177.5340118408203
Epoch 44/50, Iteration 43, Loss:	(11100.9287109375,,), Distance :	187.1370391845703
Epoch 45/50, Iteration 44, Loss:	(10414.1921828125,,), Distance :	175.52988941503906
Epoch 46/50, Iteration 45, Loss:	(11409.578125,,), Distance :	189.8236541748047
Epoch 47/50, Iteration 46, Loss:	(10575.8564453125,,), Distance :	182.14680110351562
Epoch 48/50, Iteration 47, Loss:	(9427.3125,,), Distance :	174.44878603964844
Epoch 49/50, Iteration 48, Loss:	(9314.666015625,,), Distance :	166.8095703125
Epoch 50/50, Iteration 49, Loss:	(12709.5146484375,,), Distance :	191.6496278554688



### En route vers la 100 -ème epoch de notre apprentissage en temps réel

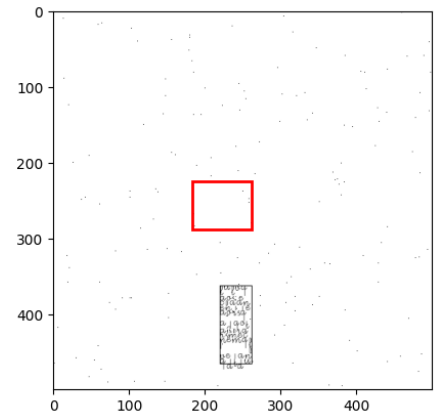
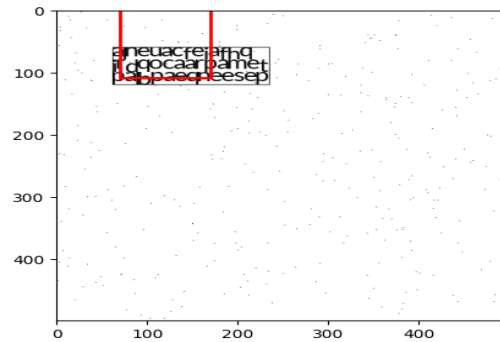
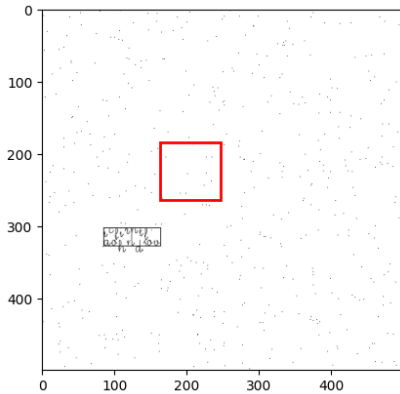
Epoch 49/50, Iteration 48, Loss:	(12518.21071,,), Distance :	190.3489394457812
Epoch 45/50, Iteration 44, Loss:	(11041.1357421875,,), Distance :	187.98165893554688
Epoch 46/50, Iteration 45, Loss:	(8997.84375,,), Distance :	162.07872009277344
Epoch 47/50, Iteration 46, Loss:	(9610.134765625,,), Distance :	176.58334350585938
Epoch 48/50, Iteration 47, Loss:	(9564.857421875,,), Distance :	166.0132598876953
Epoch 49/50, Iteration 48, Loss:	(12549.1943359375,,), Distance :	197.8041229248047
Epoch 50/50, Iteration 49, Loss:	(8754.6865234375,,), Distance :	157.46446228027344

Training Loss and Distance Over Iterations



Malheureusement le model peine à apprendre plus et on a dû l'arrêter mais cela affecte les résultats attendus.

### Place aux résultats de nos test



Ça se rapproche quand même de la zone surtout pour le deuxième teste mais cela reste moins précise.

Le cadre en noir est le résultat attendu et le cadre en rouge est le résultat prédit.

Pour être plus précis dans les chiffres :

**Loss** : 8754.6865234375 (a interpreter selon le contexte)

**Distance moyenne entre la valeur predict et la valeur réelle** : 157.46446228027344

En conclusion, le projet « Détection de text » a représenté un défi stimulant qui a mis en lumière notre capacité à concevoir, développer et déployer une solution informatique robuste. À travers une analyse approfondie des besoins, une conception réfléchie basée sur cette architecture, et une implémentation méticuleuse avec les technologies basées sur l'intelligence artificielle, nous avons réussi à répondre aux objectifs initiaux. Les phases de test ont confirmé la fiabilité de notre programme, tout en soulignant des opportunités d'amélioration. Au-delà des aspects techniques, ce projet a renforcé notre collaboration en équipe et a été une source d'apprentissage continu. Nous sommes reconnaissants envers les enseignants pour leur soutien, et nous sommes confiants que les compétences acquises seront des atouts précieux dans nos futures entreprises