RECONNAISSANCE DE LA LANGUE DES SIGNES

Contexte du projet

Beaucoup de progrès et de recherches en IA ont été faites pour aider les personnes sourdes et muettes. L'apprentissage profond et la vision par ordinateur peuvent également être utilisés pour avoir un impact sur cette cause.

Cela peut être très utile pour les personnes sourdes et muettes dans la communication avec les autres car la connaissance de la langue des signes n'est pas quelque chose qui est commun à tous, de plus, cela peut être étendu à la création des éditeurs automatiques, où la personne peut facilement écrire par ses simples gestes

Modalités d'évaluation

- ❏ Création du jeu de données (chaque signe est associé à une lettre)
- ❏ Entraînez votre modèle sur le jeu de donnée
- ❏ Testez votre modèle sur des images afin de construire des mots ou des phrases
- ❏ Testez en temps réel avec une Webcam

LANGUE DES SIGNES

Tout comme les autres langues auxquelles nous sommes plus habitués, la langue des signes n'est pas universelle. Ainsi, les alphabets dactylologiques belge, néerlandais, suisse, allemand, danois, suédois, italien, espagnol, portugais, argentin, brésilien n'utilisent qu'une seule main pour épeler un mot. En effet, ils ont tous la même racine : l'alphabet des sourds français utilisé au 18e siècle. À contrario, les alphabets anglais, australien, irlandais, yougoslave font appel aux deux mains.

Comme une langue orale, la langue des signes s'est créée grâce à des signes arbitraires (comme nos propres mots arbitraires) afin qu'ils puissent se comprendre. Par exemple, certains mots se signent avec la forme des doigts qui imite la première lettre du mot en dactylologie ; la couleur « rose » se dit en formant la lettre « R » avec la main et en passant la main sous le menton. Certains signes de base, comme manger, boire ou laver, sont néanmoins identiques, ce qui est compréhensible. Ainsi, des sourds de pays différents et avec une langue des signes différente se comprennent presque totalement au bout de deux heures de discussion environ, quel que soit le niveau de difficulté du sujet de discussion

La grammaire de la LSF

Il existe des signes qui préfigurent une expression idiomatique du français et traduisent une phrase entière en un seul geste.

Les temps du verbe n'existent pas, en LSF, on précise seulement le moment où se déroule l'action (avant ou hier pour le passé, aujourd'hui pour le présent, après ou demain pour le futur). Mais le temps est donné par la position du corps et en particulier de l'épaule de la main maîtresse. Le positionnement des signes par rapport au corps (plus ou moins éloignés) donne aussi l'idée du temps (passé, présent, futur).

La date exacte est ensuite posée par le signeur si nécessaire (il y a deux ans, hier, tout à l'heure, demain, dans un mois, etc.). L'ordre des signes est inversé. Le verbe se met généralement en fin de phrase. En premier, on met toujours les compléments circonstanciels de temps, de lieu. C'est une logique très visuelle. Enfin, comme nous l'avons déjà mentionné, le corps et les mimiques du visage sont très importants et renseignent l'interlocuteur sur le type de phrase.

L'alphabet dactylographique consiste à représenter chaque lettre de l'alphabet par une position définie des doigts de la main (droite pour les droitiers et gauche pour les gauchers). Cet alphabet ne sert qu'à épeler des mots inconnus (noms de villes, prénoms, etc.). Le prénom des personnes est toujours épelé avec la dactylologie (l'alphabet de sourds que l'on signe avec la main). Mais en fait, en entrant dans une communauté sourde, il est d'usage de recevoir un surnom, un prénom qui se fait en un seul geste. Ce nouveau nom s'attribue généralement en fonction d'une caractéristique morale ou physique qui semble la plus importante !

Jeu de données

Après avoir constitué un jeu de données “manuel”, selon les détails figurant dans le notebook , nous choisissons de partir d’un jeu de données existant sur internet .

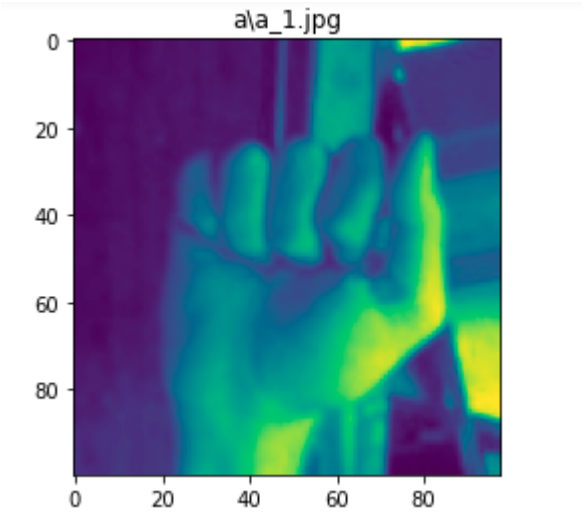
Notre dataset se compose de **40 500 photos réparties dans 27 classes** (les 26 lettres de l’alphabet et une classe inconnue). Chaque classe comprend 1500 images.

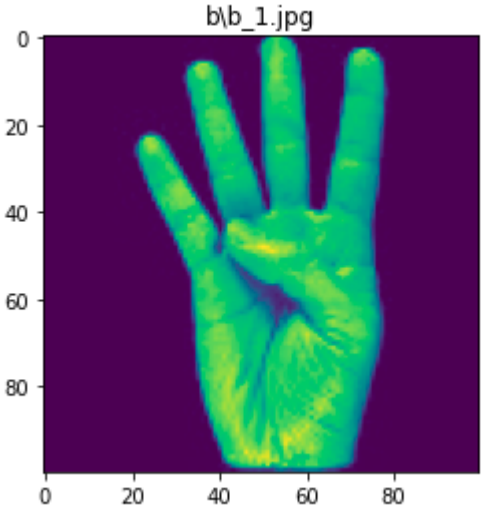
Chaque classe comprend 1500 photos.

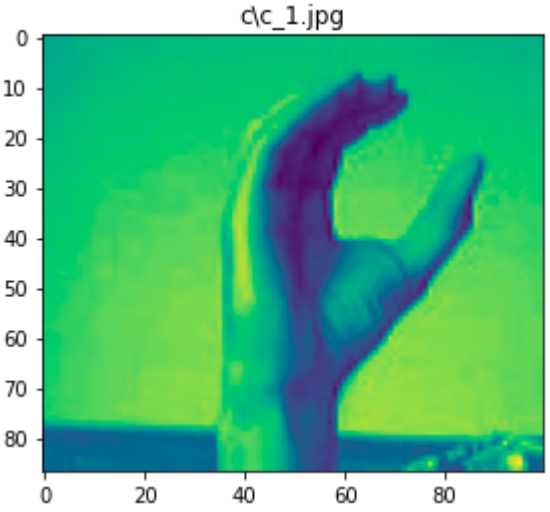
```
# Afficher le nombre de photos dans le répertoire
#os.listdir() pour obtenir tout ce qui est dans un répertoire - fichiers et répertoires.
for lettre in os.listdir(data_path):
    path = data_path + "\\" + lettre#lettre correspond à un répertoire
    print(lettre, len(os.listdir(path)))
```

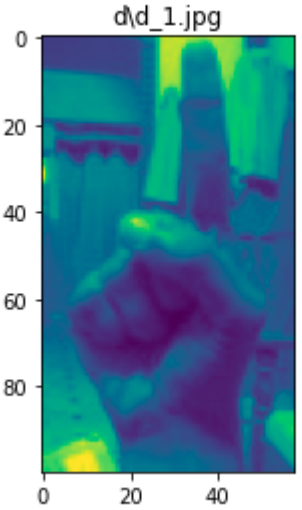
```
a 1500
b 1500
c 1500
d 1500
e 1500
f 1500
g 1500
h 1500
i 1500
j 1500
k 1500
l 1500
m 1500
n 1500
o 1500
p 1500
q 1500
r 1500
s 1500
t 1500
u 1500
unknown 1500
v 1500
w 1500
x 1500
y 1500
z 1500
```

Ci-dessous des exemples d’images :









Présentation de l'architecture utilisée

A partir de notre base de données, nous créons aléatoirement deux jeux de données composés des 27 classes

- un jeu d'entraînement (train) composé de 80% d'images (32 373 images)
- un jeu test (test) composé de 20% d'images (8127 images)

```
#insérer 80% des images du fichier lettres dans le dossier train et 20%
dans le dossier test
for lettre in os.listdir(data_path):
    path = data_path + "\\" + lettre
    taille = len(os.listdir(path))
    sequence = round(0.8 * taille)#arrondir les nombres ici 1200
    #on va dans chaque repertoire a, b... on lit les images
    train_path = train + "\\" + lettre
    test_path = test + "\\" + lettre
    for img in os.listdir(path):
        #extraction des chiffres dans le libellé
        image =  re.findall(r'\d+', img)
        #variable pour le path de l'image
        image_path = path + "\\" + img
        if int(image[0])<sequence:
            copy(image_path,train_path)
        else :
            copy(image_path,test_path)
```

Mise en place du CNN :

Dans les tâches d'apprentissage en profondeur, beaucoup de données sont nécessaires pour entraîner le modèle lorsque l'ensemble de données n'est pas assez grand, l'augmentation des données doit être appliquée.

La classe ImageDataGenerator dans Keras est utilisée pour **implémenter l'augmentation d'image** en implémentant la rotation d'image, le décalage, le redimensionnement,.. Le principal avantage de la classe Keras ImageDataGenerator est sa capacité à produire une augmentation d'image en temps réel. Cela signifie simplement qu'il peut générer des images augmentées de manière dynamique pendant la formation du modèle, ce qui rend le mode global plus robuste et précis.

Nous chargeons les données via **ImageDataGenerato**r à travers lequel nous pouvons utiliser la fonction **flow_from_directory** pour charger les données du train et de l'ensemble de test, et chacun des noms des dossiers numériques sera le nom de classe des images chargées.

```
train_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.pr
eprocess_input).flow_from_directory(directory=train,
target_size=(64,64), class_mode='categorical',
batch_size=10,shuffle=True)
test_batches =
ImageDataGenerator(preprocessing_function=tf.keras.applications.vgg16.pr
eprocess_input).flow_from_directory(directory=test, target_size=(64,64),
class_mode='categorical', batch_size=10, shuffle=True)
```

Nos images seront redimensionnées en matrice de classe binaire de taille 64*64 et par lot de 10.

```
(10, 64, 64, 3)
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0.]
```

Création du CNN

Un CNN, qu'est ce que c'est ?

Un CNN, c'est avant tout un réseau de neurones. Une donnée d'entrée, en l'occurrence une image, parcourt les différentes couches du réseau qui apprend à en extraire des caractéristiques, les features, qui représentent la donnée d'entrée à un niveau sémantique de plus en plus haut, jusqu'à permettre de pouvoir par exemple la classifier avec une probabilité associée : « cette image représente un camion », ou « un arbre ».

La particularité d'un réseau de neurones à convolution vient en fait de l'hypothèse d'invariance spatiale des caractéristiques utilisées dans l'image : on s'intéresse aux mêmes motifs à reconnaître dans les différentes parties de l'image. Techniquement, cela revient à faire partager à l'ensemble des neurones d'une même couche des poids similaires, ce qui permet de réduire considérablement le nombre de paramètres du réseau. A noter que cette hypothèse peut néanmoins limiter l'exploitation de structures bien spécifiques dans une image, comme la géométrie d'un visage

```
#Create the model
model = Sequential()


#couche de convolution, input shape correspond à nos images
redimensionnées en 64x64:
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(64,64,3)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',
padding = 'same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu',
padding = 'valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

#Ecrasement de l'input pour qu'il soit d'une seule dimension
model.add(Flatten())

# Passage aux couches dense
model.add(Dense(64,activation ="relu"))
```

```
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.2))
model.add(Dense(128,activation ="relu"))
#model.add(Dropout(0.3))

#27 pour nos 27 classes, softmax car multiclasses
model.add(Dense(27,activation ="softmax"))
```

Ajustement (fit) et enregistrement(save) du modèle pour l'utiliser lors de la détection video.

Les callbacks sont des méthodes de keras qui vont nous faire gagner du temps. On les met dans une liste et l'on passera cette liste à la méthode d'apprentissage fit() lorsqu'on entraînera le modèle.

- Le Early Stopping est un callback qui va stopper l'apprentissage afin d'éviter le suraprentissage.

- Le Reduce LR on Plateau va réduire le learning rate lorsque la val_accuracy atteint un plateau afin que le modèle continue à apprendre des features interessantes :

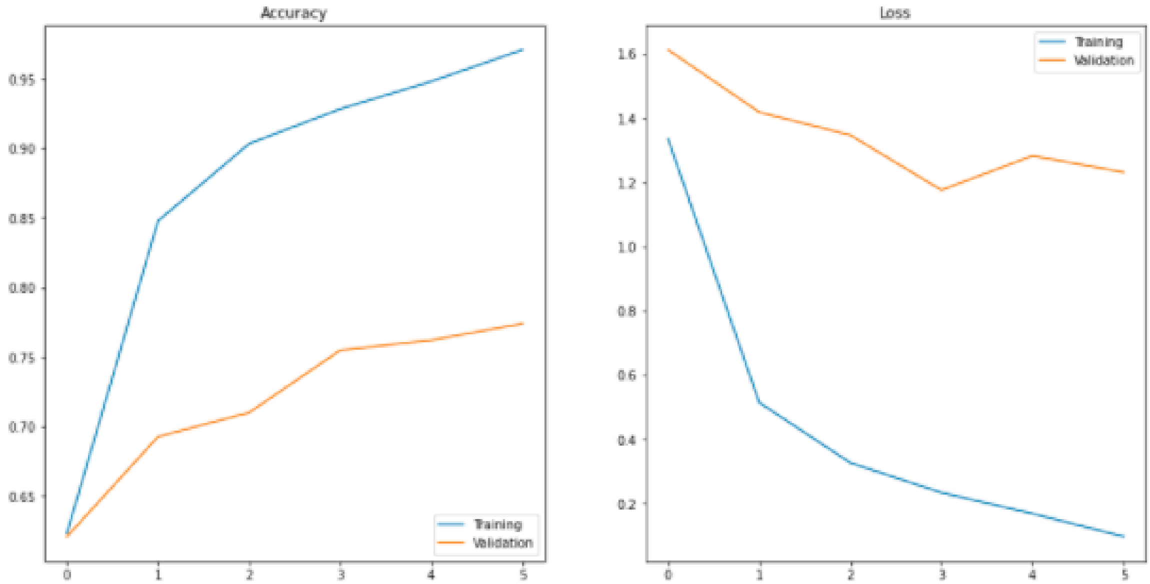
Lors de la formation denotre modèle, nous utiliserons ReduceLRonplateau and earlystopping. Les 2 callbacks dépendent de la val_loss

Après chaque epoch, l'accuracy(précision) et la loss (perte)sont calculées à l'aide de l'ensemble des données de validation. Si val_loss ne diminue pas, le LR du modèle est réduit à l'aide de la reduce_lr pour empêcher le modèle de dépasser les minima de perte et nous utilisons également EarlyStopping algorithm de sorte que si la validatiion accurany continue de diminuer pendant certaines epochs, l'apprentissage est arrêté.

L'exemple contient les callbacks utilisés, ainsi que les deux algorithmes d'optimisation différents utilisés - SGD (descente de gradient stochastique, ce qui signifie que les poids sont mis à jour à chaque instance d'entraînement) et Adam est utilisé.

Nous avons trouvé que le modèle SGD semblait donner des précisions plus élevées.

Evaluation du modèle :



visualisation des tests

```
F   R   I   D   D   A   A   S   D   I
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```

```
Actual labels
F   R   I   D   O   A   A   A   D   I
```

Prédire les gestes.

Nous allons reconnaître les gestes de la main à partir d'une séquence vidéo. Pour reconnaître ces gestes nous devons d'abord retirer la région de la main seule en supprimant toutes les parties indésirables de la séquence vidéo. Après avoir segmenté la région de la main, nous déterminerons quelle est la lettre identifiée pour faire un mot.

Cette étape va nous permettre de vérifier si notre modèle fonctionne en temps réel .
2 fenêtres vont s'ouvrir :
- une fenêtre qui propose un cadre de détection dans lequel nous devons insérer notre main

- la deuxième fenêtre qui va prédire la lettre.

> Pour un fonctionnement optimal , nous devons nous placer devant un fond uni car le modèle ne permet pas la détection automatique.

etapes :

Chargement de notre modèle
Mise en place du dimensionnement du cadre qui permettra la prédiction de l'image (endroit qui détecte notre main)
Fonction pour calculer la moyenne pondérée accumulée en arrière-plan (pour séparer le premier plan de l'arrière-plan) Puis nous soustrayons celle-ci des cadres qui contiennent un objet devant l'arrière plan (ou premier plan)

calcul de la valeur seuil :

la fonction va nous permettre de calculer la valeur seuil Thresholded pour chaque image et déterminer le contour grâce à la fonction cv2.findContoursà l'aide du segment de fonction. En d'autres termes, s'il y a ou non une main dans le ROI.

Détection de la main

```
print(lst)
```

```
['E', 'I']
```

```
" ".join(lst)
```

```
'E I'
```