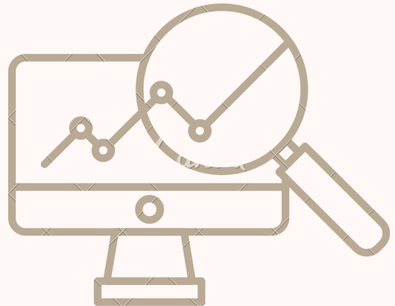
Sign to text

Ali Mahrez, Clement Lin, Stevan Bakic, Ines Taguengayte

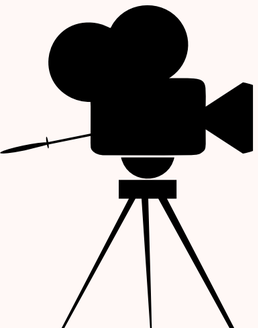
Table des Matières

- Idée du projet
- Dataset
- Méthodes
- Démonstration

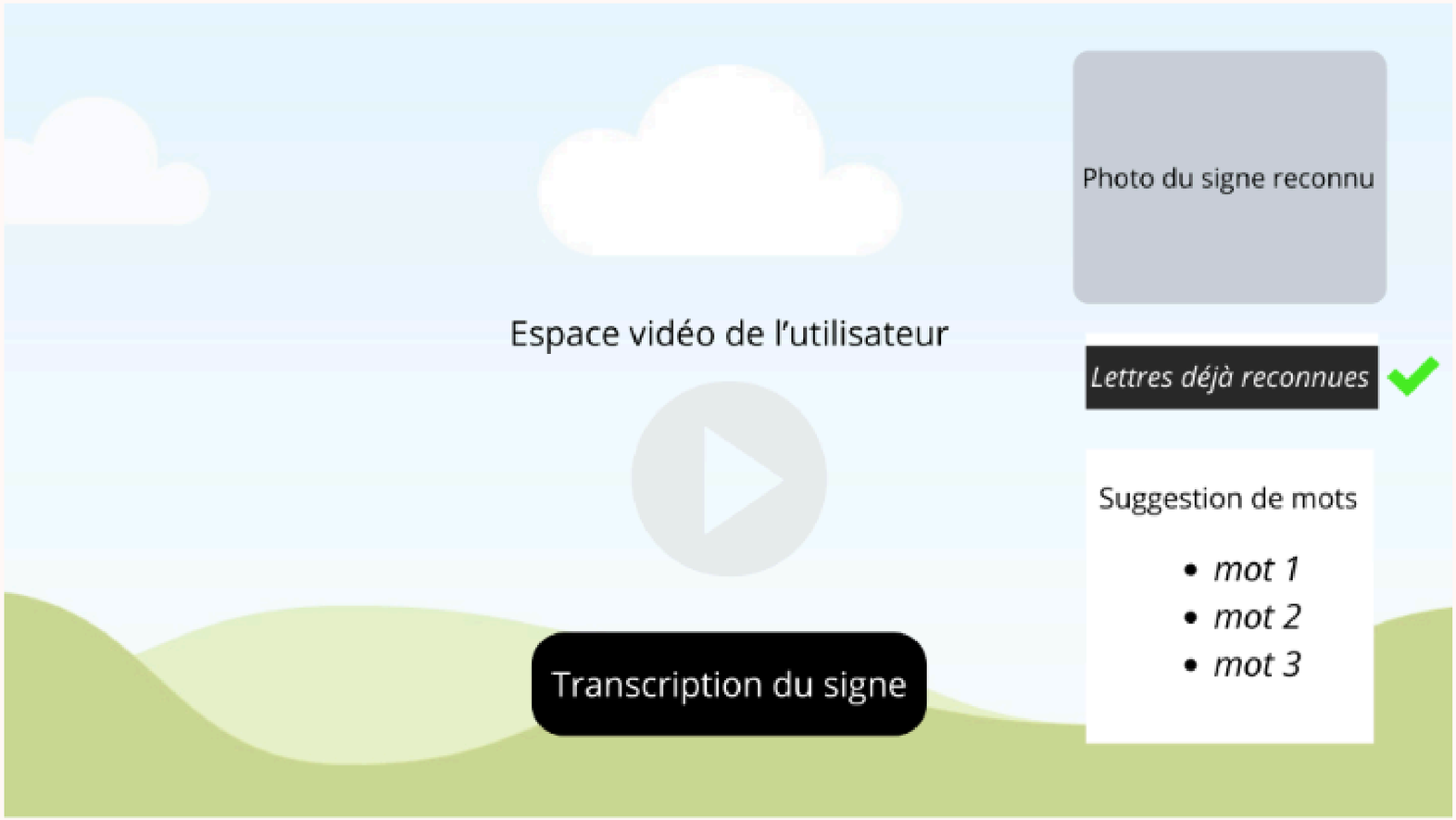


L’idée du projet

L'objectif de notre projet est de convertir en temps réel un flux vidéo des gestes du langage des signes en texte.
L’utilisateur se placera dans le cadre vidéo et représentera tour à tour les signes des lettres de son mot.
Notre modèle reconnaîtra chaque lettre et les gardera en mémoire afin qu’une fois le mot deviné, l’utilisateur puisse valider et passer à un nouveau mot.
Il suggérera également des mots pouvant correspondre au mot qu’essaye de faire deviner l’utilisateur.
Ce projet permettra aux personnes sourdes, malentendantes ou muettes d’interagir plus facilement avec leur environnement.



L’idée du projet



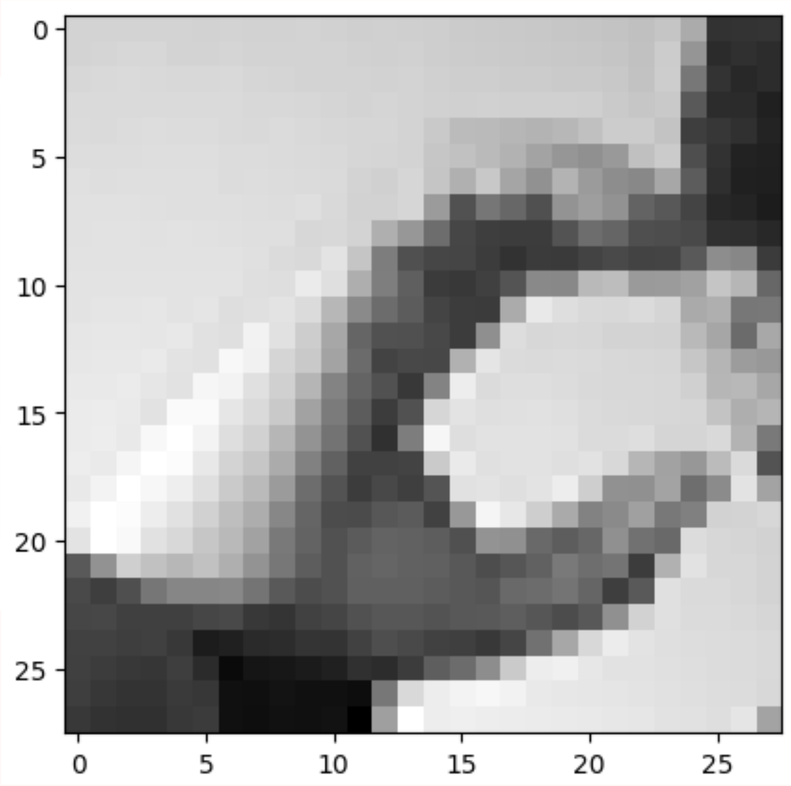
Dataset

Data augmentation

In [ ]:
```
# Configuration de la data augmentation
datagen = ImageDataGenerator(
    rotation_range=20,       # Rotation de l'image jusqu'à 20 degrés
    width_shift_range=0.2,   # Translation horizontale jusqu'à 20% de la largeur de l'image
    height_shift_range=0.2,  # Translation verticale jusqu'à 20% de la hauteur de l'image
    shear_range=0.2,         # Cisaillement jusqu'à 20%
    zoom_range=0.2,          # Zoom avant/arrière jusqu'à 20%
    horizontal_flip=True,    # Inversion horizontale (très utile pour les images de mains)
    fill_mode='nearest'      # Stratégie de remplissage des pixels manquants
)
```

In [ ]:
```
print('Min label:', np.min(y_train))
print('Max label:', np.max(y_train))
```

```
Min label: 0
Max label: 24
```



Dataset

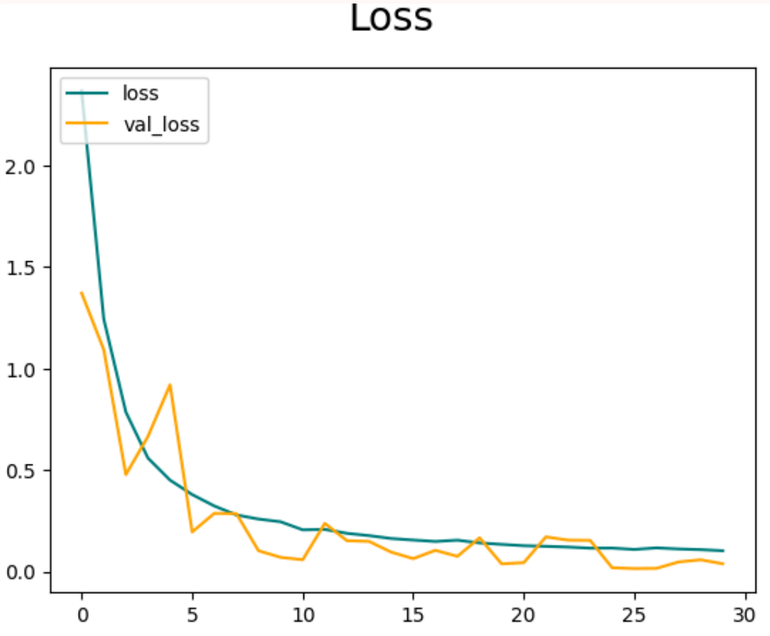
CNN

Création du modèle CNN

In [ ]:

```
# Création du modèle
model = Sequential()

model.add(Conv2D(75 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu' , input_shape = (28,28,1)))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))
model.add(BatchNormalization())

model.add(Conv2D(50 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(Dropout(0.2))
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))

model.add(Conv2D(25 , (3,3) , strides = 1 , padding = 'same' , activation = 'relu'))
model.add(BatchNormalization())
model.add(MaxPool2D((2,2) , strides = 2 , padding = 'same'))


model.add(Flatten())
model.add(Dense(units = 512 , activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(units = 24 , activation = 'softmax'))

model.compile(optimizer = 'adam' , loss = 'categorical_crossentropy' , metrics = ['accuracy', tf.keras.metrics.
model.summary()
```

```
loss: 0.0379 - accuracy: 0.9845 - recall_4: 0.9841
```



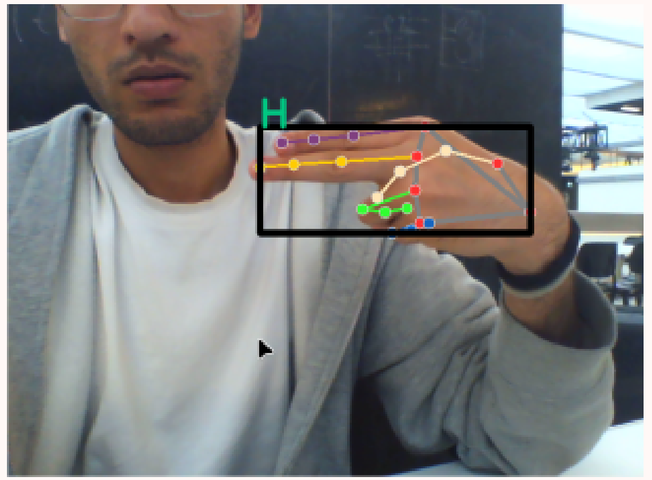
Pas d’overfitting

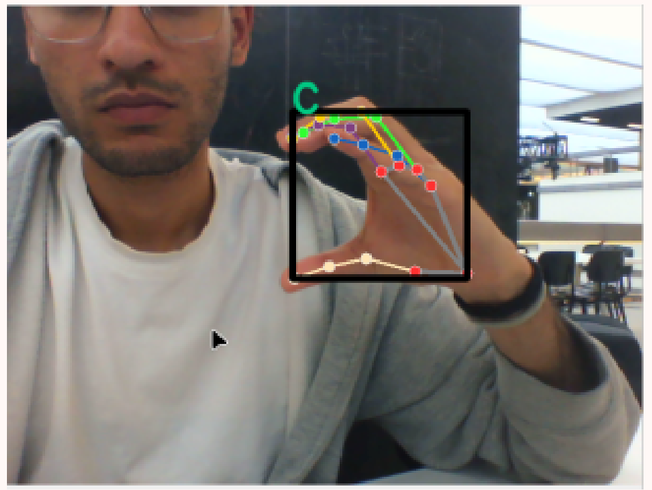
Dataset

Dataset

```python
# Boucle pour chaque lettre
for i in range(nb_lettres):
    letter_path = os.path.join(DATA_DIR, lettres[i])
    if not os.path.exists(letter_path):
        os.makedirs(letter_path)

    print(f'En cours de création de données pour la lettre : {lettres[i]}')

    # Affichage du message jusqu'à ce que l'utilisateur appuie sur 's'
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Échec de la capture d'une image")
            continue

        cv2.putText(frame, "Appuyer sur la lettre 's' pour commencer la collection", (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.3,
        cv2.imshow('frame', frame)

        if cv2.waitKey(25) == ord('s'):
            break

    # Capture des images pour la lettre en cours
    compteur = 600
    while compteur < class_size:
        ret, frame = cap.read()
        cv2.imshow('frame', frame)
        cv2.waitKey(25)
        cv2.imwrite(os.path.join(DATA_DIR, lettres[i], '{}.jpg'.format(compteur)), frame)
```

Extrait de code pour la construction de notre dataset



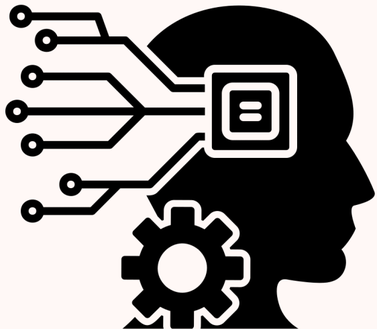


Machine learning

Utilisation du modèle Random Forest

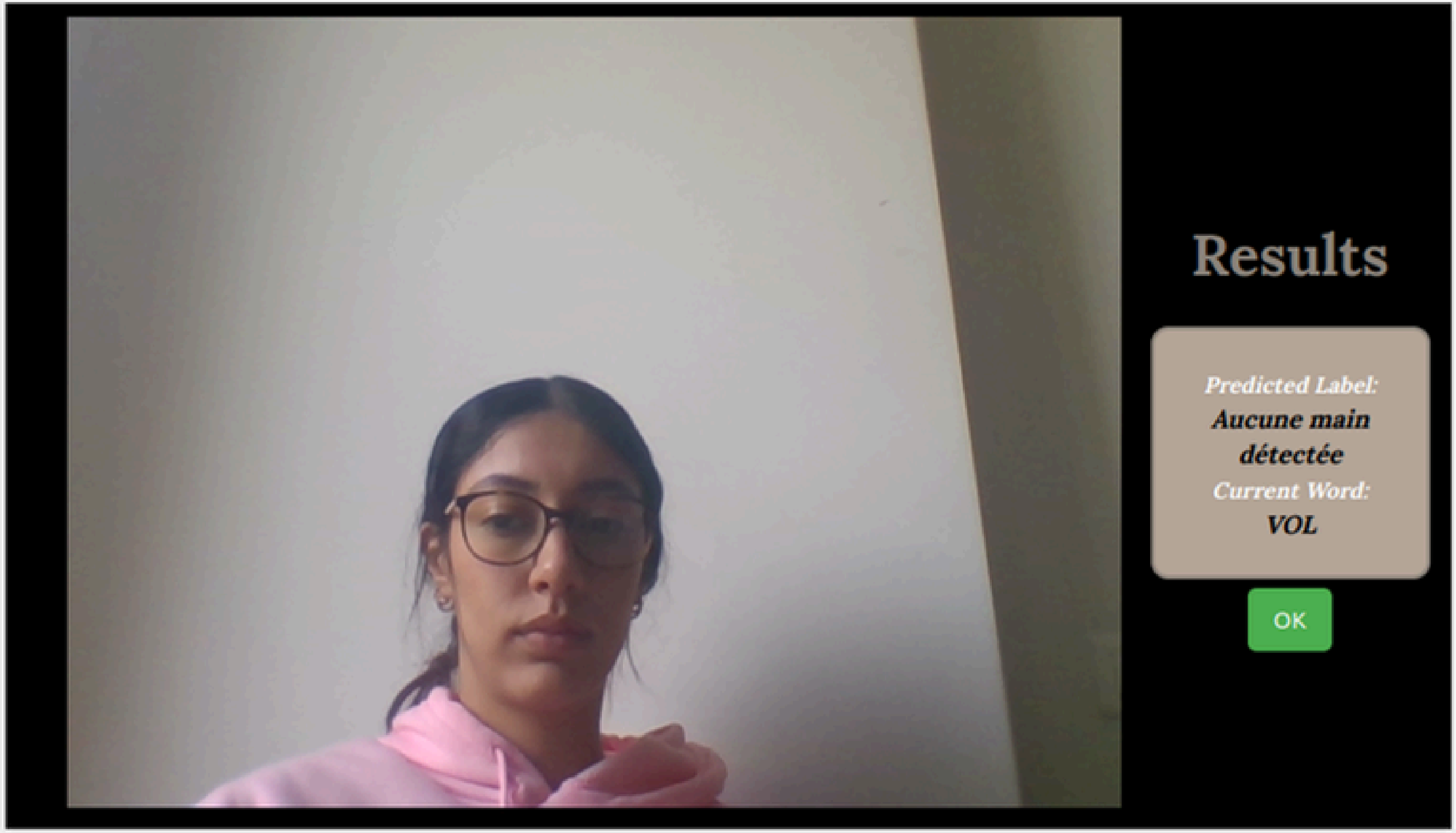
```
data_dict = pickle.load(open("./data.pickle", 'rb'))
data = data_dict['data']
labels = data_dict['labels']

# Initialiser le LabelEncoder
label_encoder = LabelEncoder()

# Encoder les labels
encoded_labels = label_encoder.fit_transform(labels)

# Filtrer les indices où les vecteurs de caractéristiques ont la longueur correcte
valid_indices = [i for i, item in enumerate(data) if len(item) == 63]

# Filtrer X et y en utilisant ces indices
X_filtered = [data[i] for i in valid_indices]
y_filtered = [encoded_labels[i] for i in valid_indices]

# Convertir en tableaux numpy
X = np.array(X_filtered)
y = np.array(y_filtered)

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True, stratify=y)

# Initialiser et entraîner le RandomForestClassifier
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
Accuracy: 0.9994367783722895
```



Démonstration

Real Time Hand Sign Recognition



volatile | volume | volunteer

Validate

Real Time Hand Sign Recognition

Sentence : volume bgw

Validate