



















Reconnaissance de chiffres manuscrits



I	II	III	IV	V	VI	VII	VIII	IX
								
								

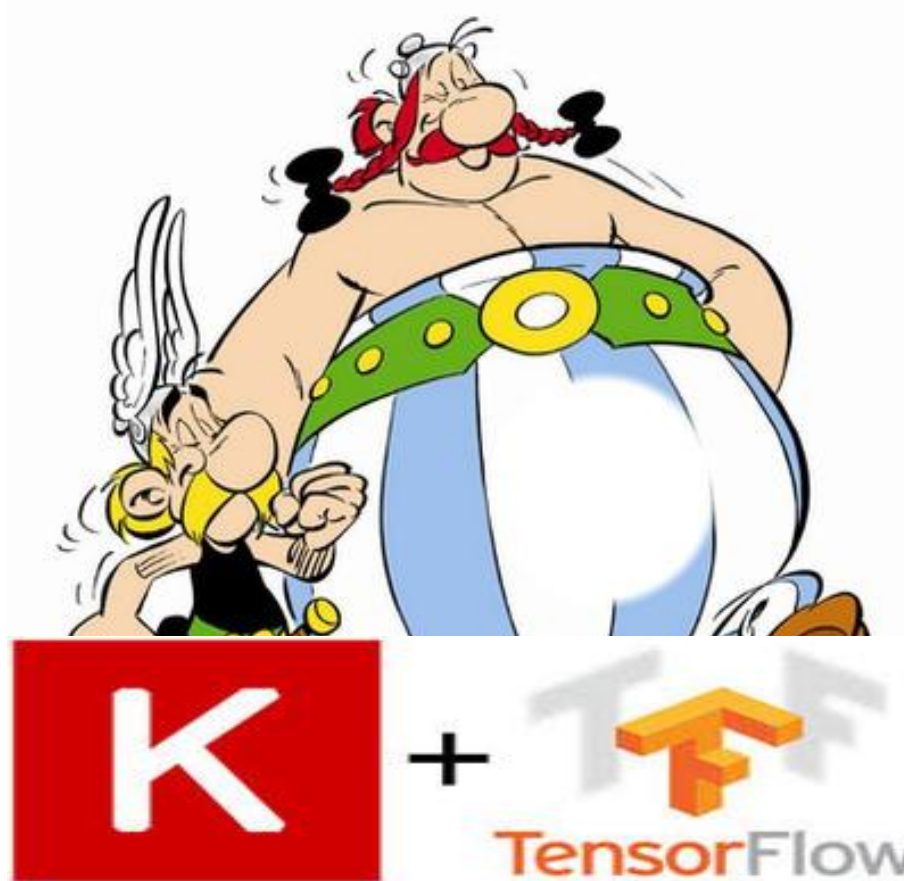
Etre devin cela ne s'improvise pas !



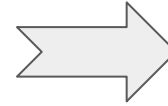
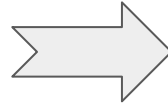
Aidons notre devin à faire de bonnes prédictions!



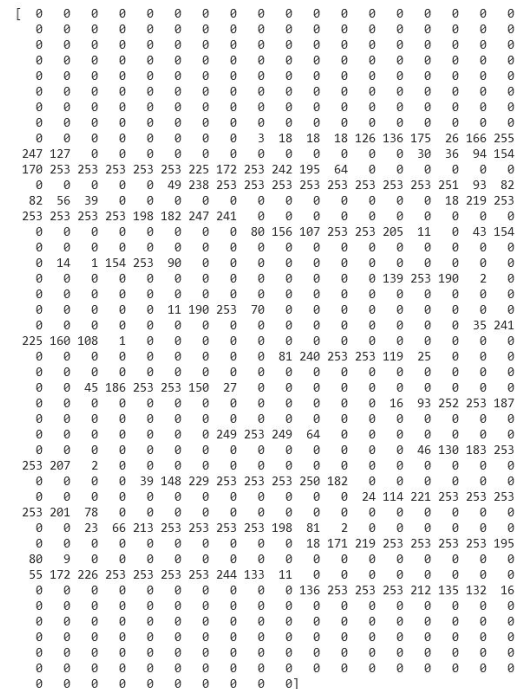
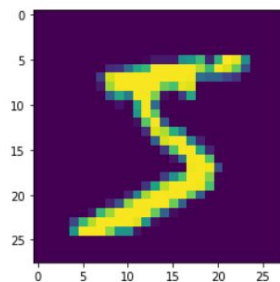
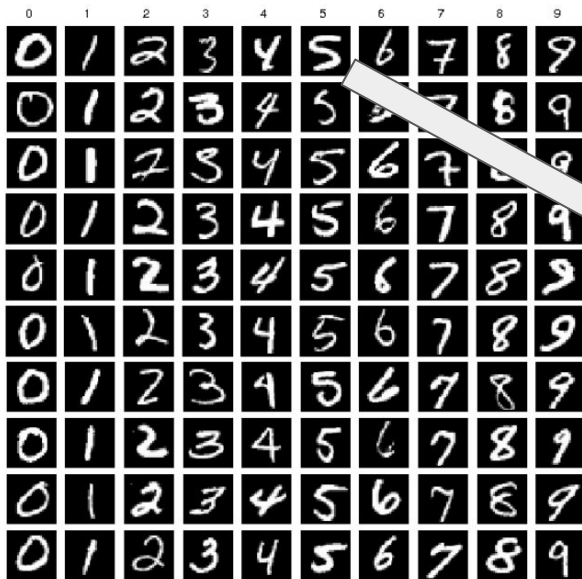
Le réseau de neurones c'est notre dolmen!



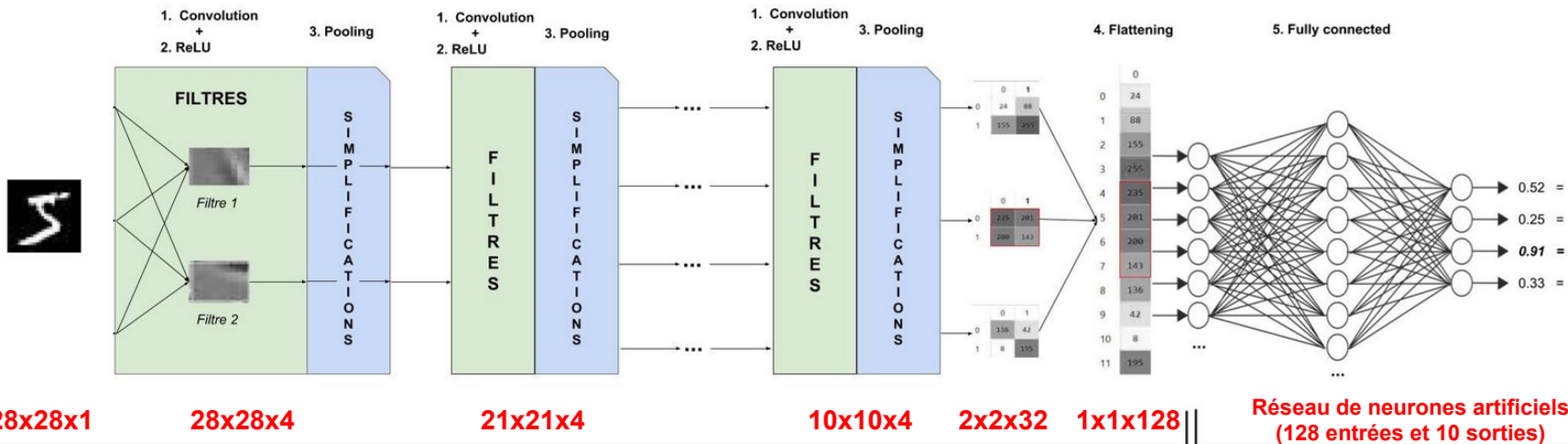
Entraînons notre Devin



Mais comment ces données sont lues?



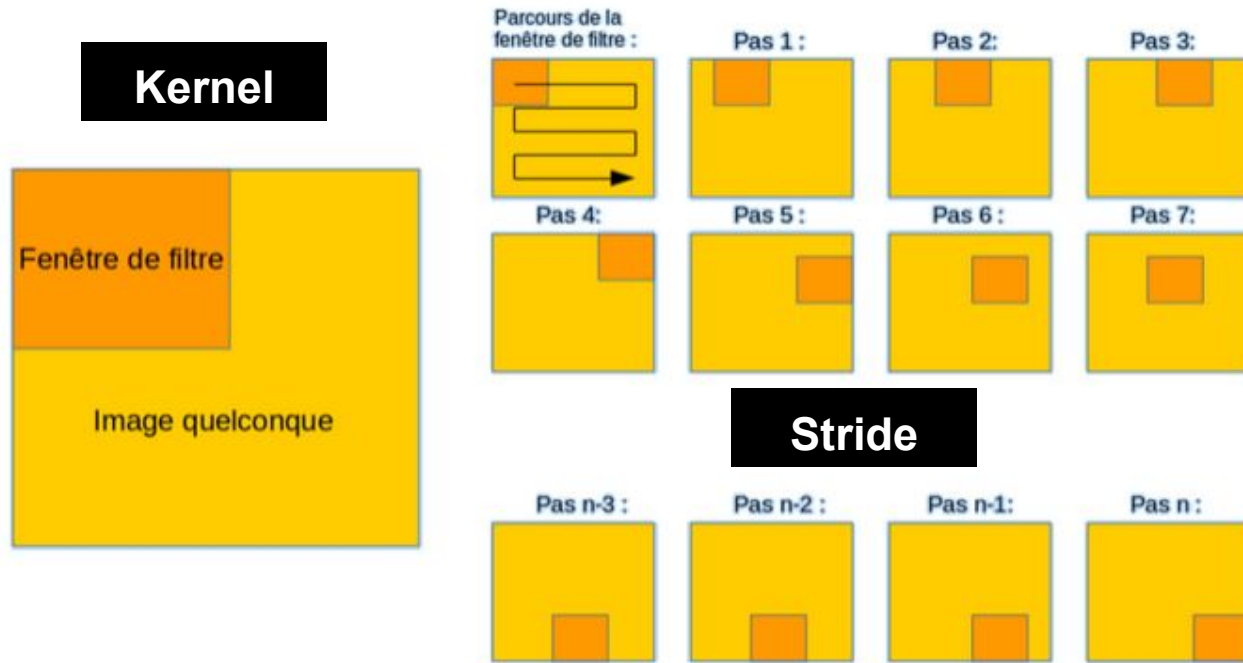
Schémas du fonctionnement du modèle CNN



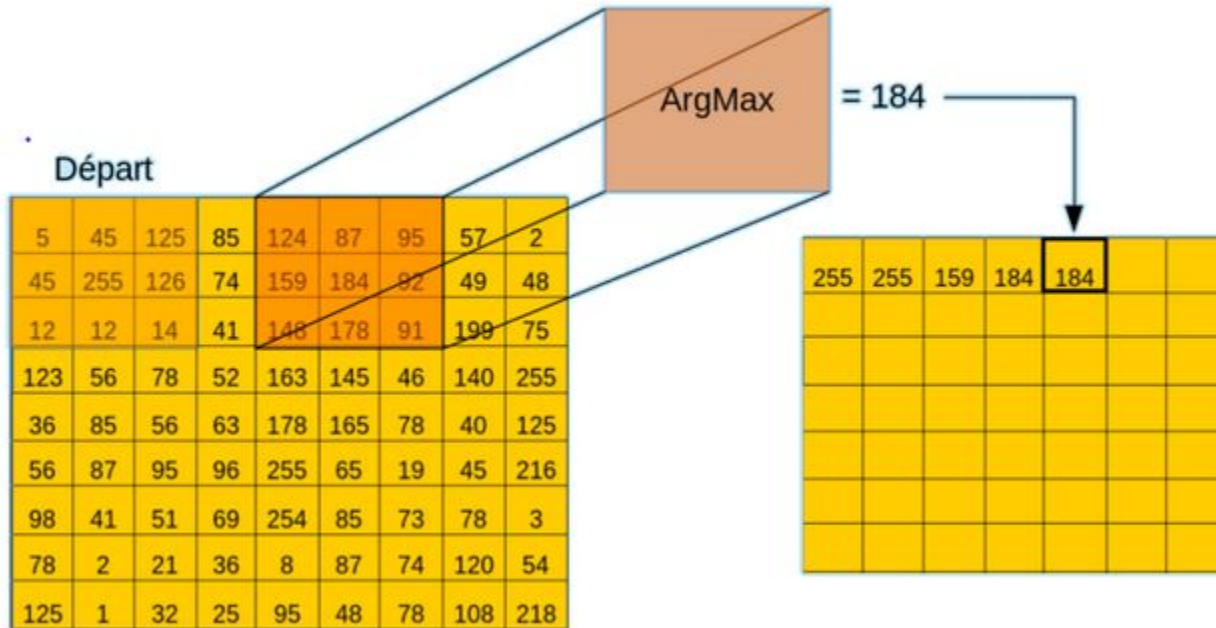
Extraction des informations de l'image grâce à un enchaînement de filtres (et de simplifications)

Prédiction de la classe de l'image


Le principe de la convolution



Exemple de la convolution Arg max



De nombreux filtres de convolution

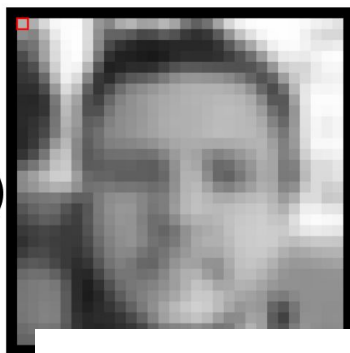



$$\begin{pmatrix} 206 & 205 & 247 \\ \times 0.0625 & \times 0.125 & \times 0.0625 \\ + 244 & 161 & 137 \\ \times 0.125 & \times 0.25 & \times 0.125 \\ + 192 & 154 & 75 \\ \times 0.0625 & \times 0.125 & \times 0.0625 \end{pmatrix}$$

= 178

kernel: blur

input image

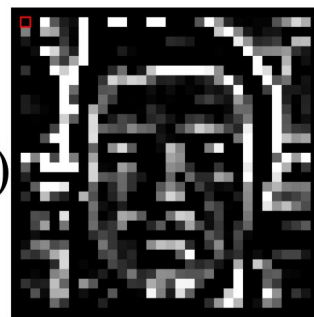




$$\begin{pmatrix} 206 & 205 & 247 \\ \times -1 & \times -1 & \times -1 \\ + 244 & 161 & 137 \\ \times -1 & \times 8 & \times -1 \\ + 192 & 154 & 75 \\ \times -1 & \times -1 & \times -1 \end{pmatrix}$$

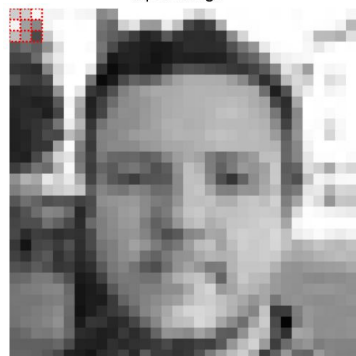
= -172

kernel: outline



output image

<https://setosa.io/ev/image-kernels/>

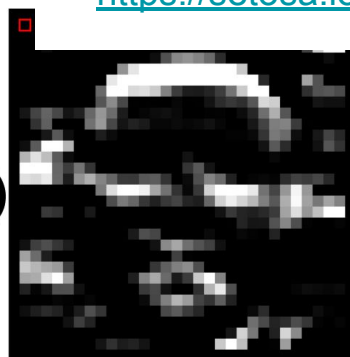


$$\begin{pmatrix} 206 & 205 & 247 \\ \times -1 & \times -2 & \times -1 \\ + 244 & 161 & 137 \\ \times 0 & \times 0 & \times 0 \\ + 192 & 154 & 75 \\ \times 1 & \times 2 & \times 1 \end{pmatrix}$$

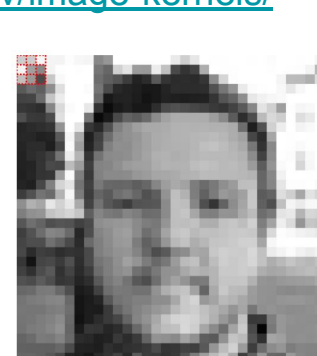
= -288

kernel: bottom sobel

input image



output image

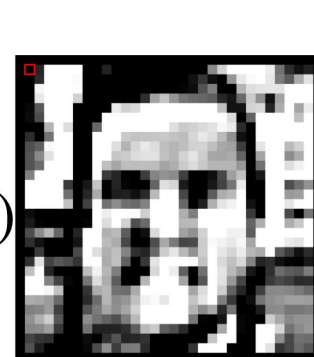


$$\begin{pmatrix} 206 & 205 & 247 \\ \times -2 & \times -1 & \times 0 \\ + 244 & 161 & 137 \\ \times -1 & \times 1 & \times 1 \\ + 192 & 154 & 75 \\ \times 0 & \times 1 & \times 2 \end{pmatrix}$$

= -259

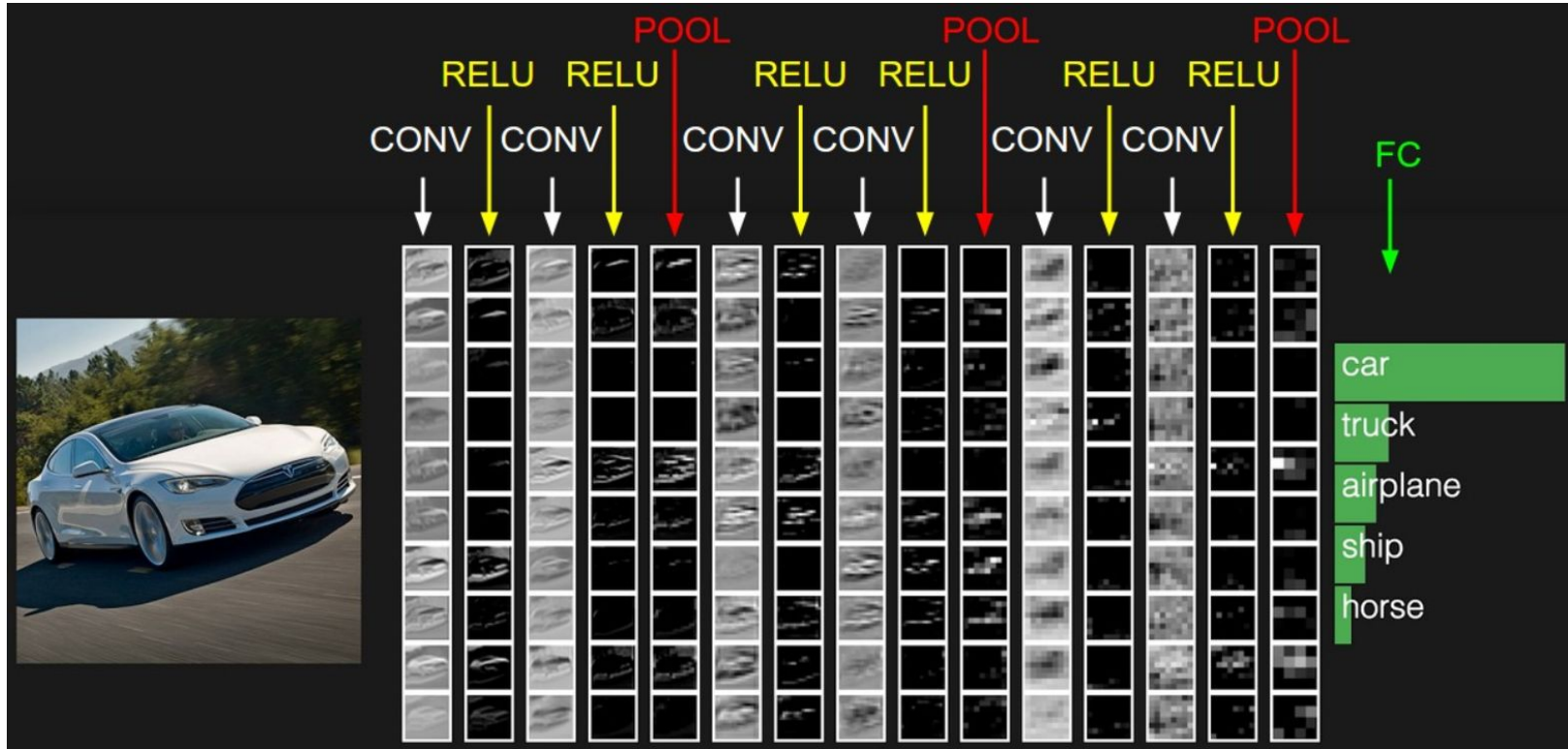
kernel: emboss

input image

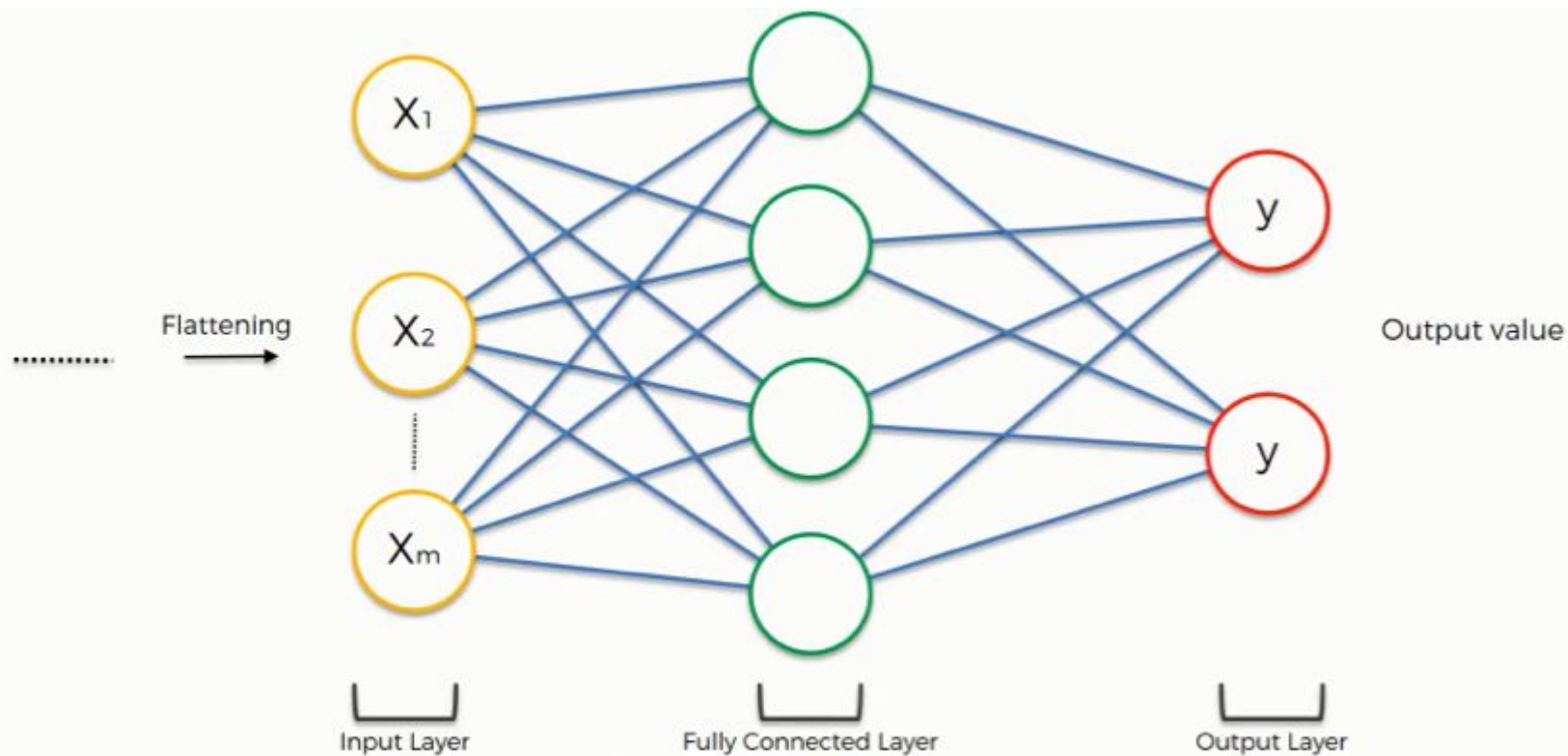


output image

Application des 3 couches: Convolution + Relu+ Pool



Dense

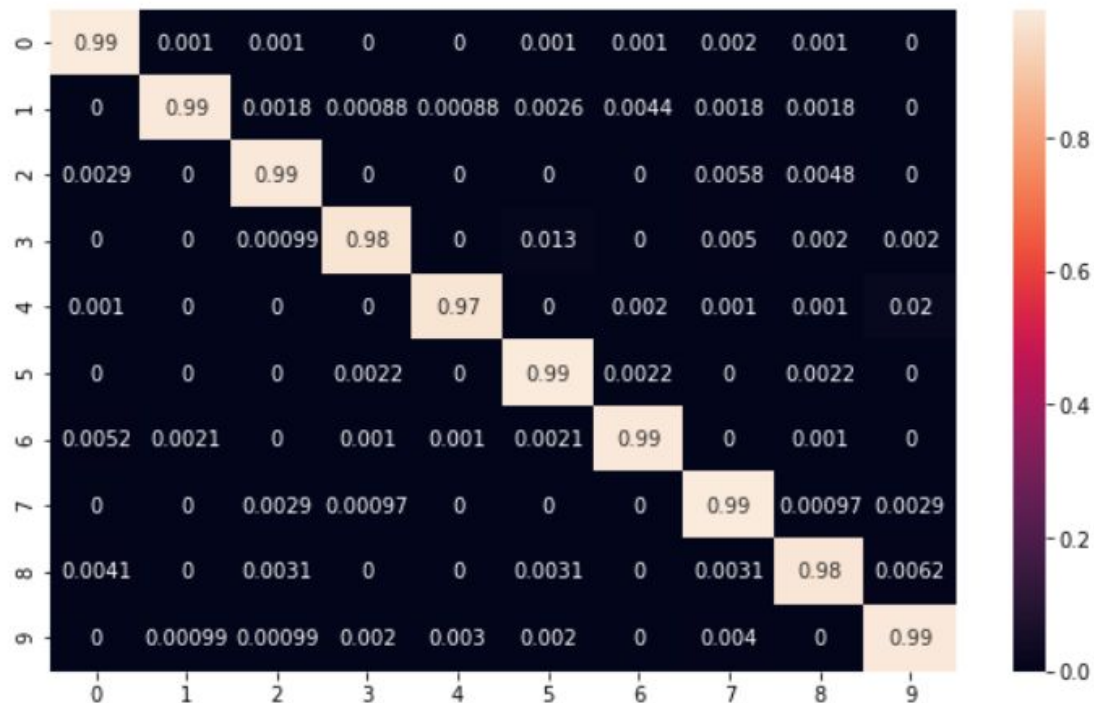


Codification du modèle CNN

```
1 # Importation
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten
```

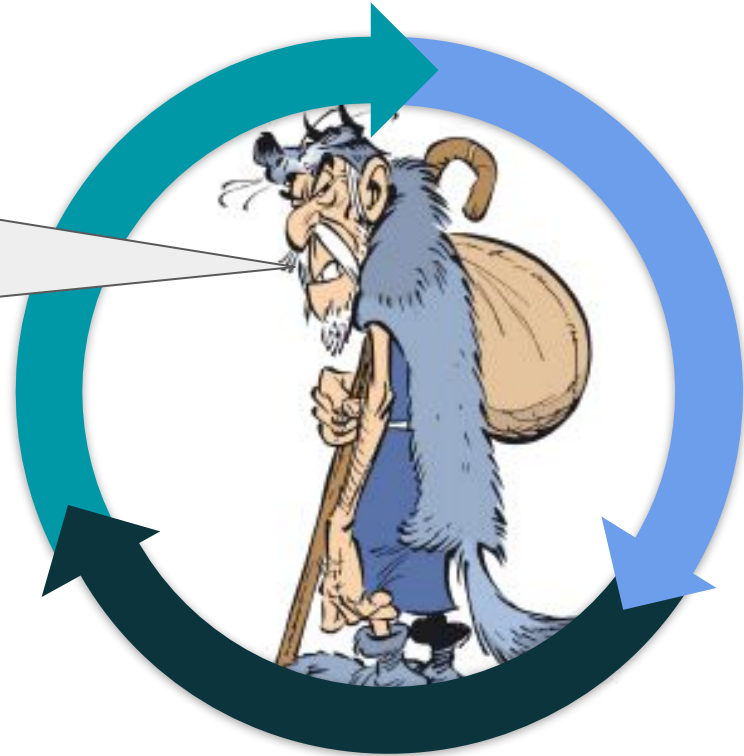
```
1 # création du modèle séquentiel
2 model = Sequential()
3 # couche de convolution
4 model.add(Conv2D(filters=32, kernel_size=(4,4), strides=(1,1), input_shape=(28,28,1), activation='relu'))
5 # couche de Pooling
6 model.add(MaxPool2D(pool_size=(2,2)))
7 # Applatis les images en une dimension
8 model.add(Flatten())
9 # Couche d'entrée "dense": fonction d'activation
10 model.add(Dense(128, activation='relu'))
11 # couche de sortie: fonction d'activation 'softmax'
12 model.add(Dense(10, activation='softmax'))
13 # phase de compilation d'une classification multiclasse.
14 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
15 |
```


Le CNN est le modèle le plus précis



Notre Devin est prêt à faire ses prédictions!

*Montre moi un chiffre et
je te dirai lequel c'est !!!*



Ouvrons l'oeil !

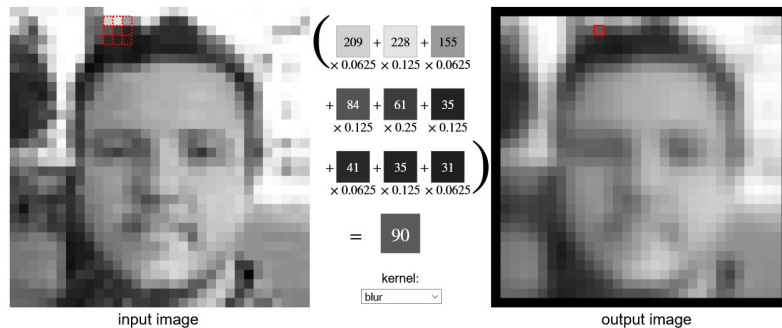


Initialisation de la reconnaissance d'images

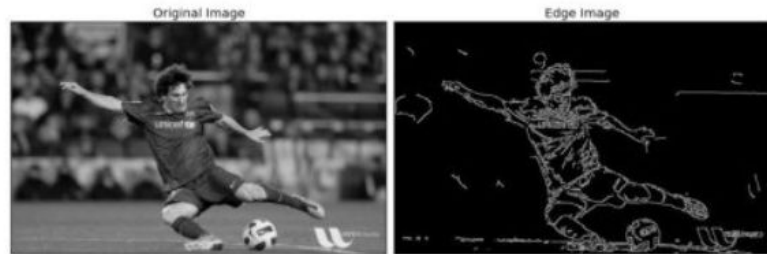
cvtColor



Gaussianblur



canny

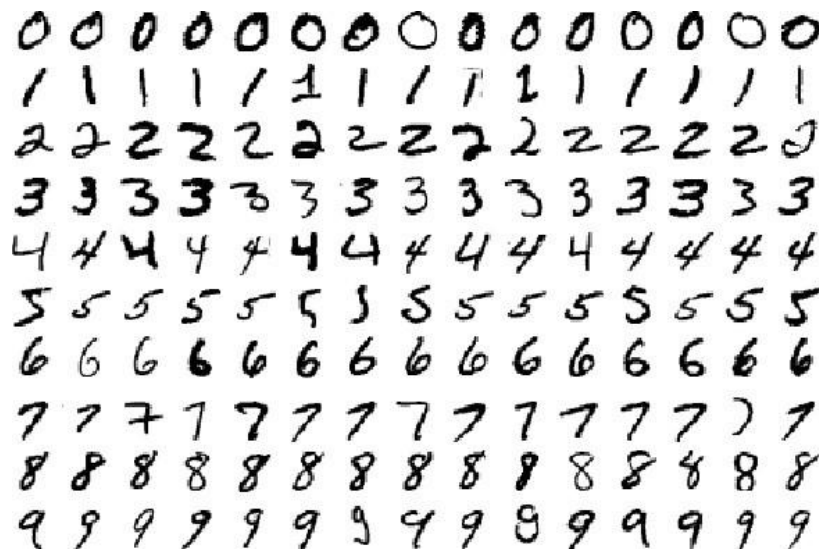
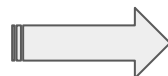
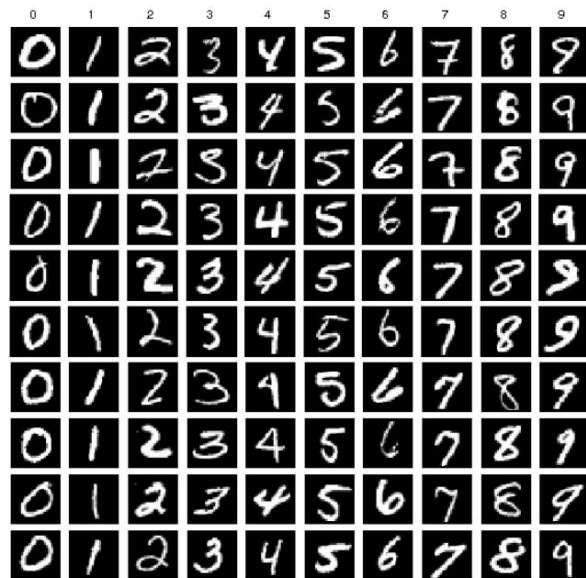


image

Threshold



Thresh_Binary_Inv + Thresh_Otsu



Fonction pour définir et améliorer la lecture des caractères

```
1 def main():
2     # Ouvrir la caméra pour traiter l'image:
3     cap = cv2.VideoCapture(0)
4
5     # pour faire tourner la caméra:
6     while (cap.isOpened()):
7         # Capture des images de la webcam
8         ret, img = cap.read()
9         # Appliquer la fonction get_img_contour_thresh sur le cadre
10        img, contours, thresh = get_img_contour_thresh(img)
11        ans = ''
12
13        if len(contours) > 0:
14            # Trouver un contour avec une surface maximale
15            contour = max(contours, key=cv2.contourArea)
16            # Contour évolutif
17            if cv2.contourArea(contour) > 1500 and cv2.contourArea(contour) < 5000 :
18                # Dimensions du rectangle
19                x, y, w, h = cv2.boundingRect(contour)
20                # Créer une nouvelle image contenant un contour pour la classification
21                newImage = thresh[y:y + h, x:x + w]
22                # Redimensionner une nouvelle image
23                newImage = cv2.resize(newImage, (28, 28))
24                newImage = np.array(newImage) / 255
25                # Effectuer la classification
26                hog_ft = hog(newImage, orientations=9, pixels_per_cell=(14, 14), cells_per_block=(1, 1))
27                #hog_ft = scaled_image(np.array([hog_ft], 'float64'))
28                ans = np.argmax(model.predict(newImage.reshape(1,28,28,1)),axis=-1)
29
30
31
```

Fonction pour un contour

```
# Rendre la région rectangulaire de taille x, y, w, h
x, y, w, h = 0, 0, 300, 300
# dimension du rectangle
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# putText = texte qui affiche la prediction sous le rectangle
# FONT_HERSHEY_SIMPLEX = police de style d'écriture à la main
cv2.putText(img, "svm : " + str(ans), (10, 320), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

# Affichage du cadre et du seuil
cv2.imshow("Frame", img)
cv2.imshow("Contours", thresh)

# Touche Echap pour arrêter
k = cv2.waitKey(10)
if k==27:
    break

def get_img_contour_thresh(img):
    x, y, w, h = 0, 0, 300, 300

    # Changer l'espace colorimétrique de RGB -> Gris
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Appliquer un flou
    blur = cv2.GaussianBlur(gray, (35, 35), 0)
    # Créer une image à seuil binaire
    ret, thresh = cv2.threshold(blur, 255, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    # Création d'une image de seuil de taille x, y, w, h = 0, 0, 300, 300
    thresh = thresh[y:y + h, x:x + w]

    # Trouver les contours à partir de l'image seuillée
    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)[-2:]

    return img, contours, h
```



.exe



Flask



Nos Déboires....

TensorFlow et environnement Anaconda.

Comprendre le CNN.

Améliorer détection des contours sur Open CV.

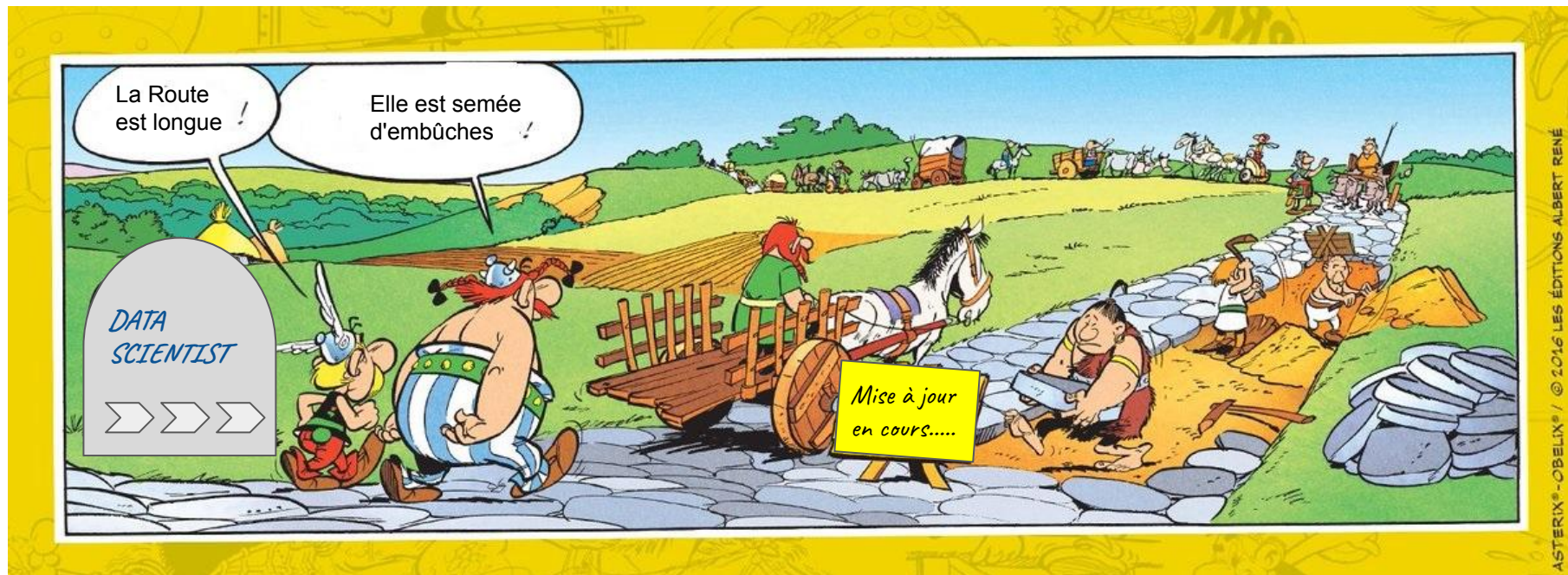
Déploiement avec Flask.

Partie chronophage de notre travail.

Supporter les blagues de certains.



Bref....



Si vous avez des questions, on les a déjà devinées!

