



## Travail pratique 4

ARN

Anthony David, Alejandro Stadlin

12 mai 2024

# Table des matières

<b>Introduction</b>	<b>2</b>
<b>Reconnaissance des chiffres à partir des données brutes</b>	<b>3</b>
Réseau neuronal superficiel . . . . .	3
<b>Reconnaissance des chiffres à partir des caractéristiques des données d'entrée</b>	<b>4</b>
Réseau neuronal superficiel . . . . .	4
Hyper-paramètres . . . . .	4
Configuration du réseau : . . . . .	4
Calcul des poids : . . . . .	4
Poids totaux : . . . . .	4
Expérimentations . . . . .	5
4 pixels, 8 orientations, 9 neurones . . . . .	5
4 pixels, 8 orientations, 50 neurones . . . . .	5
4 pixels, 8 orientations, 100 neurones . . . . .	5
4 pixels, 4 orientations, 9 neurones . . . . .	6
4 pixels, 4 orientations, 50 neurones . . . . .	6
7 pixels, 8 orientations, 9 neurones . . . . .	6
7 pixels, 4 orientations, 9 neurones . . . . .	7
Commentaires . . . . .	7
<b>Reconnaissance de chiffres par réseau neuronal convolutif</b>	<b>8</b>
<b>Reconnaissance des chiffres par réseau neuronal convolutionnel</b>	<b>9</b>
Réseau neuronal convolutionnel profond . . . . .	9
Expérimentation . . . . .	10
25 neurones . . . . .	10
200 neurones . . . . .	10
500 neurones . . . . .	10
<b>Questions générales</b>	<b>11</b>
<b>Conclusion</b>	<b>12</b>

# Introduction

# Reconnaissance des chiffres à partir des données brutes

Quel est l'algorithme d'apprentissage utilisé pour optimiser les poids des réseaux de neurones ? Quels sont les paramètres (arguments) utilisés par cet algorithme ? Quelle fonction de coût est utilisée ? Veuillez fournir l'équation (ou les équations).

L'algorithme utilisé pour optimiser les poids est RMSprop (Root Mean Square Propagation).

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1 - \beta) \left( \frac{\partial C}{\partial w} \right)^2 \quad w_t = w_{t-1} - \frac{\eta}{\sqrt{E[g^2]_t}} \frac{\partial C}{\partial w}$$

où  $\eta$  est le taux d'apprentissage,  $w_t$  le nouveau poids,  $\beta$  est le paramètre de moyenne mobile,  $E[g]$  est la moyenne mobile des gradients carrés et  $\frac{\partial C}{\partial w}$  est la dérivée de la fonction de coût par rapport au poids

La fonction de coût est la fonction de perte par entropie croisée catégorique :

$$\text{CE} = -\frac{1}{N} \sum_{k=0}^N \log \vec{p}_i[y_i]$$

où  $N$  est le nombre d'échantillons,  $\vec{p}_i$  est la sortie du réseau de neurones et  $y_i$  est l'index de la classe cible.

## Réseau neuronal superficiel

# Reconnaissance des chiffres à partir des caractéristiques des données d'entrée

## Réseau neuronal superficiel

Dans cette expérience, nous utilisons les caractéristiques du Histogramme des gradients (HOG) pour classer les chiffres.

### Hyper-paramètres

- Nombre d'orientations : 8
- Pixels par cellule : 4
- Nombre de cellules par image :  $\frac{28}{4} \times \frac{28}{4} = 7 \times 7 = 49$  cellules
- Caractéristiques HOG totales :  $49 \times 8 = 392$

### Configuration du réseau :

- Couches cachées : 9 neurones, puis 50 neurones, puis 100 neurones
- Fonction d'activation de sortie : softmax pour 10 classes (chiffres de 0 à 9)

### Calcul des poids :

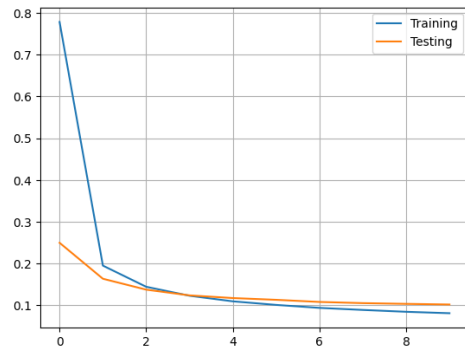
1. Première couche cachée (9 neurones) :
  - Poids :  $392 \times 9 + 9(biais) = 3528 + 9 = 3537$
2. Deuxième couche cachée (50 neurones) :
  - Poids :  $9 \times 50 + 50(biais) = 450 + 50 = 500$
3. Troisième couche cachée (100 neurones) :
  - Poids :  $50 \times 100 + 100(biais) = 5000 + 100 = 5100$
4. Couche de sortie :
  - Poids :  $100 \times 10 + 10(biais) = 1000 + 10 = 1010$

### Poids totaux :

- Total :  $3537$  (première couche cachée) +  $500$  (deuxième couche cachée) +  $5100$  (troisième couche cachée) +  $1010$  (couche de sortie) =  $10147$

## Expérimentations

## 4 pixels, 8 orientations, 9 neurones



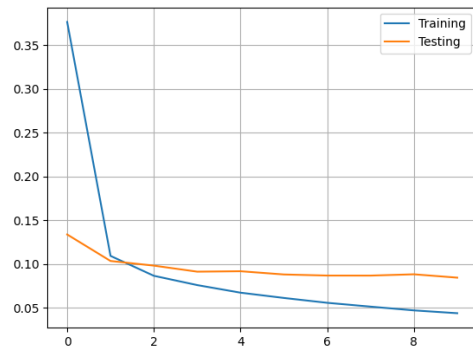
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	969	0	3	0	1	2	2	1	1	1
1	0	1120	3	1	1	0	2	3	5	0
2	6	4	996	4	2	0	2	8	10	0
3	0	0	2	982	0	11	0	4	10	1
4	3	4	4	1	954	0	1	3	2	10
5	3	0	2	13	0	855	6	0	7	6
6	10	1	1	0	4	2	937	0	3	0
7	0	9	10	3	5	1	0	988	3	9
8	7	2	5	11	1	2	2	3	933	8
9	2	4	0	9	9	6	0	13	6	960

(b) Matrice de confusion

FIGURE 1 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 8 orientations et 9 neurones

## 4 pixels, 8 orientations, 50 neurones



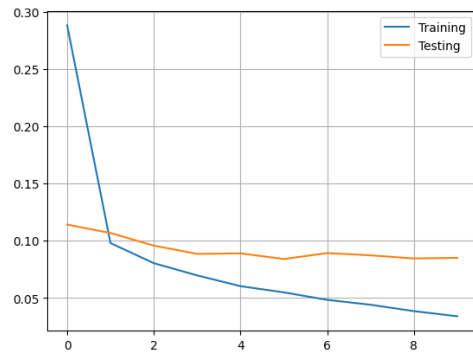
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	968	0	2	0	0	1	5	2	1	1
1	1	1124	2	0	1	0	3	2	2	0
2	3	3	1014	0	1	0	2	4	5	0
3	0	0	2	979	0	9	1	5	13	1
4	2	1	2	0	968	0	1	1	3	4
5	2	1	0	14	0	865	7	0	3	0
6	2	2	1	0	5	3	945	0	0	0
7	0	3	8	3	6	0	0	993	5	10
8	7	0	3	13	3	2	2	6	932	6
9	2	3	1	6	11	4	0	6	6	970

(b) Matrice de confusion

FIGURE 2 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 8 orientations et 50 neurones

## 4 pixels, 8 orientations, 100 neurones



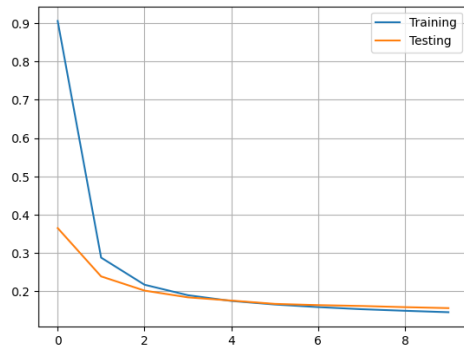
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	969	0	3	0	1	2	2	1	1	1
1	0	1120	3	1	1	0	2	3	5	0
2	6	4	996	4	2	0	2	8	10	0
3	0	0	2	982	0	11	0	4	10	1
4	3	4	4	1	954	0	1	3	2	10
5	3	0	2	13	0	855	6	0	7	6
6	10	1	1	0	4	2	937	0	3	0
7	0	9	10	3	5	1	0	988	3	9
8	7	2	5	11	1	2	2	3	933	8
9	2	4	0	9	9	6	0	13	6	960

(b) Matrice de confusion

FIGURE 3 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 8 orientations et 100 neurones

## 4 pixels, 4 orientations, 9 neurones



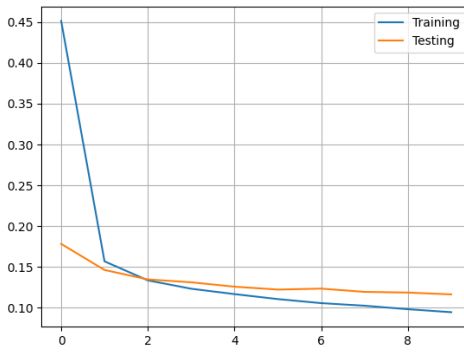
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	955	2	4	0	2	5	6	2	2	2
1	0	1121	4	1	4	0	3	0	2	0
2	5	9	987	4	2	0	4	11	10	0
3	0	2	4	964	1	13	0	6	15	5
4	3	7	5	0	939	0	2	3	3	20
5	1	0	3	18	1	841	10	0	16	2
6	6	1	0	2	7	6	932	0	4	0
7	1	11	16	2	13	0	0	961	2	22
8	8	1	9	13	4	5	3	6	912	13
9	7	5	0	10	17	1	1	12	5	951

(b) Matrice de confusion

FIGURE 4 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 4 orientations et 9 neurones

## 4 pixels, 4 orientations, 50 neurones



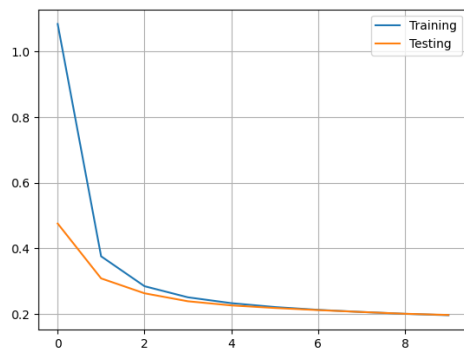
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	965	1	2	0	0	3	5	2	1	1
1	1	1118	4	1	1	0	3	2	5	0
2	6	7	987	7	0	0	4	11	8	2
3	0	0	2	977	0	11	0	4	13	3
4	4	2	1	0	945	0	1	2	2	25
5	1	1	0	17	1	859	5	1	6	1
6	5	1	2	1	3	4	937	0	5	0
7	1	6	4	4	5	0	0	987	5	16
8	13	2	3	15	2	8	4	4	909	14
9	4	4	0	10	6	2	2	9	4	968

(b) Matrice de confusion

FIGURE 5 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 4 orientations et 50 neurones

## 7 pixels, 8 orientations, 9 neurones

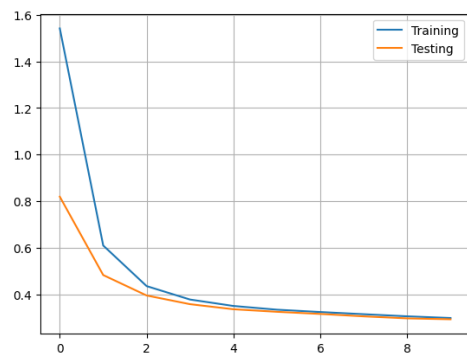


(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	950	4	6	0	0	0	7	1	7	5
1	1	1115	3	5	2	0	5	1	3	0
2	9	8	958	22	9	1	1	10	13	1
3	2	0	9	952	4	16	0	4	20	3
4	0	9	4	0	929	0	2	5	5	28
5	2	0	1	23	0	827	11	0	25	3
6	15	2	0	1	9	11	914	0	5	1
7	0	3	20	3	13	0	0	925	5	59
8	12	5	4	16	7	19	7	5	883	16
9	7	2	2	13	18	3	0	24	8	932

(b) Matrice de confusion

FIGURE 6 – Modèle utilisant les caractéristiques HOG avec 7 pixels par cellule, 8 orientations et 9 neurones

**7 pixels, 4 orientations, 9 neurones**

(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	919	2	3	5	0	5	25	1	15	5
1	0	1100	8	1	9	0	9	3	5	0
2	7	4	943	25	11	2	5	18	16	1
3	2	1	27	913	4	13	1	6	38	5
4	0	8	7	1	907	0	4	16	7	32
5	1	0	0	49	3	778	17	0	40	4
6	16	0	1	1	15	16	895	0	14	0
7	1	3	37	3	18	0	0	885	9	72
8	6	8	12	29	5	16	8	4	867	19
9	2	3	4	15	19	7	0	43	17	899

(b) Matrice de confusion

FIGURE 7 – Modèle utilisant les caractéristiques HOG avec 4 pixels par cellule, 8 orientations et 9 neurones

**Commentaires**

Pour nous, les deux derniers systèmes sont les meilleurs. Par rapport aux autres, ils semblent offrir les meilleures performances et ne présentent pas de signes de surapprentissage (“overfitting”). Les modèles dotés d’un grand nombre de neurones, tels que ceux avec 50 et 100 neurones dans la couche cachée, montrent clairement des signes de surapprentissage. En outre, lorsque la taille des cellules est de 4x4 pixels, les résultats sont meilleurs avec 4 orientations. En revanche, pour les cellules de 7x7 pixels, une configuration à 8 orientations semble plus performante.



# Reconnaissance de chiffres par réseau neuronal convolutif

# Reconnaissance des chiffres par réseau neuronal convolutionnel

## Réseau neuronal convolutionnel profond

- Époques : 50
- Taille du lot : 128
- Couches cachées :
  - Convolution 2D : 5x5
  - MaxPooling 2D : taille du pool : 2x2
  - Convolution 2D : 5x5
  - MaxPooling 2D : taille du pool : 2x2
  - Convolution 2D : 3x3
  - MaxPooling 2D : 2x2
  - Couche Flatten
  - Dense (25 neurones), fonction d'activation : ReLU

Sortie : 10 neurones, fonction d'activation : softmax

Complexité du modèle :

Layer (type)	Output Shape	Param #
10 (InputLayer)	[(None, 28, 28, 1)]	0
11 (Conv2D)	(None, 28, 28, 9)	234
11_mp (MaxPooling2D)	(None, 14, 14, 9)	0
12 (Conv2D)	(None, 14, 14, 9)	2034
12_mp (MaxPooling2D)	(None, 7, 7, 9)	0
13 (Conv2D)	(None, 7, 7, 16)	1312
13_mp (MaxPooling2D)	(None, 3, 3, 16)	0
flat (Flatten)	(None, 144)	0
14 (Dense)	(None, 25)	3625
15 (Dense)	(None, 10)	260

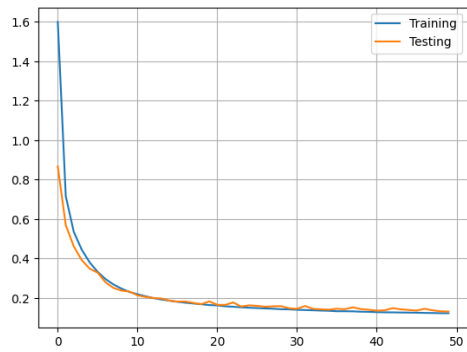
Total params: 7,465

Trainable params: 7,465

Non-trainable params: 0

## Expérimentation

### 25 neurones



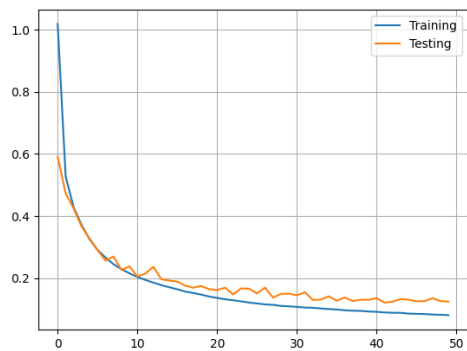
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	950	0	4	0	2	1	17	2	4	0
1	0	1110	1	6	3	0	1	4	10	0
2	4	2	1002	3	2	0	3	10	6	0
3	1	0	6	980	0	4	0	6	13	0
4	0	0	3	0	955	0	9	2	2	11
5	1	0	1	20	1	842	5	2	17	3
6	6	1	2	0	2	4	939	0	4	0
7	1	3	18	4	1	1	0	979	5	16
8	6	0	5	2	4	7	5	3	936	6
9	4	1	0	4	15	6	1	11	13	954

(b) Matrice de confusion

FIGURE 8 – Modèle utilisant 25 neurones CNN dans la partie feed forward

### 200 neurones



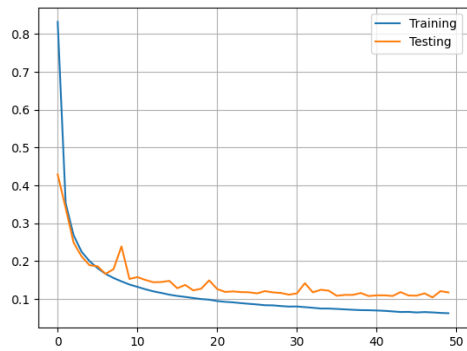
(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	961	0	5	0	2	3	5	1	3	0
1	1	1118	4	1	0	1	5	1	4	0
2	2	0	1014	7	1	0	1	5	2	0
3	0	0	7	985	1	6	0	6	5	0
4	0	1	2	3	964	0	5	0	2	5
5	2	0	3	13	2	867	2	0	3	0
6	3	1	3	0	4	4	939	0	4	0
7	0	3	11	3	1	1	0	1065	2	2
8	16	0	12	14	15	11	2	5	893	6
9	3	5	1	9	30	8	0	8	7	938

(b) Matrice de confusion

FIGURE 9 – Modèle utilisant 200 neurones CNN dans la partie feed forward

### 500 neurones



(a) Graphique d'erreur

	0	1	2	3	4	5	6	7	8	9
0	966	2	0	1	0	2	3	1	5	0
1	0	1120	8	0	0	0	4	0	3	0
2	5	0	1013	4	0	1	0	2	6	1
3	1	0	13	979	0	13	0	4	0	0
4	0	0	5	0	968	0	4	0	1	4
5	6	0	1	8	0	874	1	0	1	1
6	9	1	3	0	2	4	935	0	4	0
7	0	2	22	6	0	0	0	990	6	2
8	17	1	10	3	1	4	4	6	926	2
9	4	1	7	4	11	7	0	12	12	951

(b) Matrice de confusion

FIGURE 10 – Modèle utilisant 500 neurones CNN dans la partie feed forward

# Questions générales

# Conclusion