Iván Fernández Navarro
NIA: 100383564

Carlos García Corral
NIA: 100383380

Grupo 84

Práctica 2 - Parte 2:

Problema de clasificación con Redes Convolucionales y Perceptrón Multicapa

Redes de Neuronas Artificiales

ÍNDICE:

1. Introducción 2

2. Realización de experimentos 2

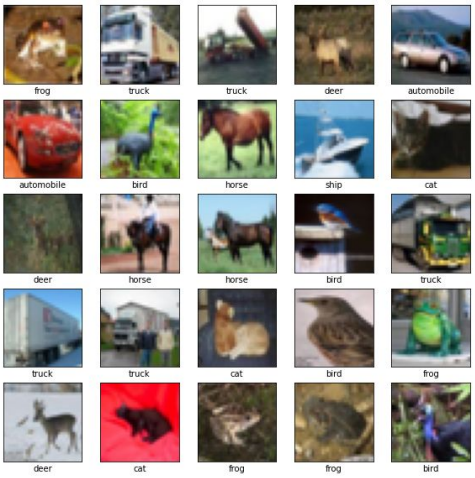
3. Modelo Perceptrón multicapa 3
   3.1. Experimentación realizada 3
      3.1.1. Mejor experimento 4

4. Modelo de Redes Convolucionales (CNN) 4
   4.1. Experimentación realizada 4
      4.1.1. Primer experimento 4
      4.1.2. Segundo experimento 5
      4.1.3. Tercer experimento 5
      4.1.4. Cuarto experimento 5
      4.1.5. Quinto experimento 5
   4.2. Análisis de los resultados 6

5. Conclusiones 6

1. Introducción

En esta práctica comprobaremos la eficacia de dos modelos de redes de neuronas no lineales distintas: el Perceptrón Multicapa y las redes Convolucionales. Para este propósito utilizaremos el conjunto de datos CIFAR10, el cual contiene 60000 imágenes divididas en 10 clases diferentes, 6000 imágenes por clase.



[1] Ejemplo de 25 imágenes del conjunto de datos con sus respectivas clases.

Para comprobar la eficacia de nuestros modelos de perceptrón multicapa hemos realizado 5 experimentos en los que modificaremos el número de capas ocultas y el número de neuronas de cada capa oculta.

Para comprobar la eficacia de nuestras redes convolucionales hemos realizado 6 experimentos en los que vamos modificando los siguientes aspectos:

- Número de capas convolucionales
- Número de filtros en las capas de convolución
- Tamaño del Kernel en las capas de convolución
- Dropout
- Número de ciclos (épocas) de aprendizaje si se estima conveniente

2. Realización de experimentos

Debido al gran coste computacional que supone la realización de algunos de estos experimentos, principalmente los que incluyen redes convolucionales, hemos optado por realizarlos en el entorno de colaboración Google Colab, en el que se pueden crear cuadernos de Jupyter para ejecutar scripts de Python.
Para la ejecución de los experimentos hemos creado dos funciones:

- **create_custom_pm()** con los parámetros:
  - n_hidden_layers (int)
  - hidden_layer_neurons (int[])
  - hidden_layer_function (string[])
- **create_custom_cnn()** con los parámetros:
  - n_convolution_layers (int)
  - n_filters (int[])
  - size_filters ((int,int)[])
  - convolutional_layer_function (string[])
  - strides (int[])
  - padding (int[])
  - dropout (float)

- n_pooling_layers (int)
- n_fully_connected_layers (int)
- fully_connected_layer_neurons (int[])
- fully_connected_function (string[])

Además, hemos creado un array de experimentos en el que almacenamos objetos de este estilo, con la configuración de cada experimento:

Ejemplo de experimento PM

```
      'type': 'pm',
      'settings': {
 'n_hidden_layers': 2,
 'hidden_layer_neurons': [30, 15],
 'hidden_layer_function': ['relu', 'tanh']
      },
      'learning_rate': 1e-3,
      'name': 'PM_1',
      'n_epochs': 20
```

Ejemplo de experimento CNN

```
     'type': 'cnn',
     'settings': {
'n_convolution_layers': 3,
'n_filters': [16, 8, 4],
'size_filters': [(3,3)]*3,
'convolutional_layer_function': ['relu']*3,
'strides': [1]*3,
'padding': ['valid']*3,
'dropout': 0.75,
'n_pooling_layers': 1,
'n_fully_connected_layers': 2,
'fully_connected_layer_neurons': [64,16],
'fully_connected_function': ['relu']*2
     },
     'learning_rate': 1e-3,
     'name': 'CNN_2',
     'n_epochs': 20
```

Para realizar los experimentos simplemente iteramos sobre el array y por cada experimento creamos el modelo, lo entrenamos y guardamos los resultados en su propia carpeta.

3. Modelo Perceptrón Multicapa

3.1. Experimentación realizada

Hemos realizado 5 experimentos con distintos modelos del perceptrón multicapa para comprobar qué hiperparametros se adaptan mejor al conjunto de datos. Por razones de simplicidad se mostrará en la memoria el mejor experimento, puesto que el perceptrón multicapa ya se ha estudiado con profundidad en la parte 1 de la práctica, y así se dará más espacio a las redes convolucionales.

3.1.1. Mejor experimento

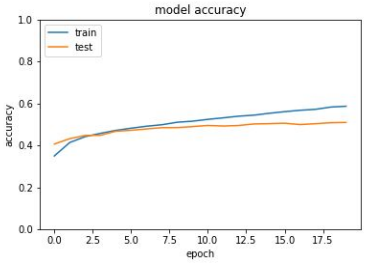
3.1.1.1. Modelo

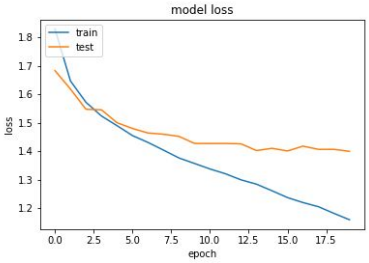
3 capas ocultas con 200, 100 y 50 nodos respectivamente,

función de activación sigmoidal y 20 épocas de entrenamiento.

3.1.1.2. Resultados

En las figuras se puede observar un loss final de entrenamiento de 1.16, un loss final de testing de 1.42, un porcentaje final de aciertos de entrenamiento de 0.59 y un porcentaje final de aciertos de testing de 0.51.
Este mejor experimento comparándolo con la parte 1 de la práctica donde también se usaba el perceptrón multicapa es muy pobre. Por lo que podemos concluir que el perceptrón multicapa no es un modelo muy fiable cuando se usan tantas imágenes y existen tantas clases de predicción.





4. Modelo de Redes Convolucionales (CNN)

4.1. Experimentación realizada

Hemos realizado 5 experimentos con distintas configuraciones de redes convolucionales para comprobar qué hiperparametros se adaptan mejor al conjunto de datos.

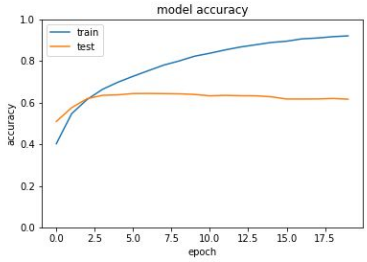
4.1.1. Primer experimento

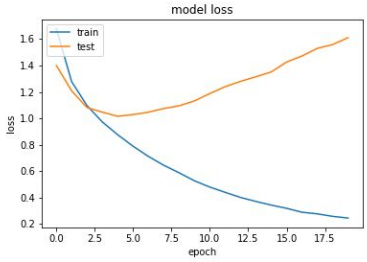
4.1.1.1. Modelo

3 capas convolucionales, filtros de 3x3 y 16 filtros por capa.

4.1.1.2. Resultados

Un claro caso de overfitting





4.1.2. Segundo experimento

4.1.2.1. Modelo

3 capas convolucionales, filtros de 3x3, 16, 8 y 6 filtros respectivamente.

4.1.2.2. Resultados

Menos overfitting, una mejor accuracy

4.1.3. Tercer experimento

4.1.3.1. Modelo

10 capas convolucionales, 32 filtros y 8 filtros según la capa, filtros de 5x5 y 3x3.

4.1.3.2. Resultados

Muy poco overfitting, mejora a lo largo de los epochs constante y estabilizandose.

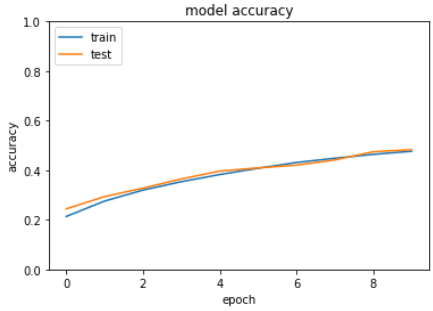
4.1.4. Cuarto experimento

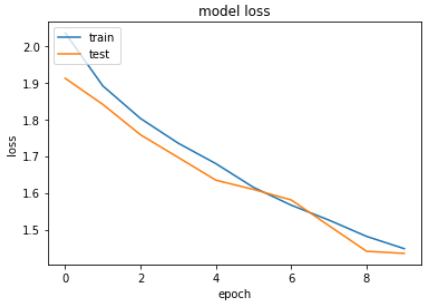
4.1.4.1. Modelo

15 capas convolucionales, 32 filtros y 8 filtros según la capa, filtros de 5x5 y 3x3.

4.1.4.2. Resultados

Muy poco overfitting, mejora a lo largo de los epochs constante y estabilizandose.





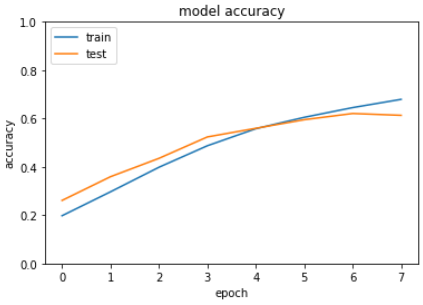
4.1.5. Quinto experimento

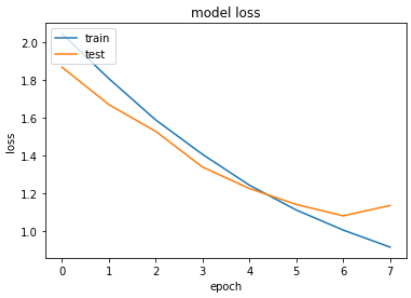
4.1.5.1. Modelo

5 capas convolucionales, 8,16,24,32 y 64 filtros respectivamente, filtros de 2x2, sin dropout.

4.1.5.2. Resultados

Un poco de overfitting, mejora a lo largo de los epochs constante y estabilizandose.





4.2. Análisis de resultados

Experimento	Accuracy final		Loss final	
	Training	**Test**	**Training**	**Test**
1	0.9230	0.6359	0.2347	1.5218
2	0.8785	0.5795	0.3551	1.6665
3	0.6013	0.5733	1.1226	1.2111
4	1.4027	1.4380	0.4898	0.4852
5	0.7634	0.6466	0.6772	1.0869

Después de realizar todos los experimentos nuestra mejor configuración ha sido cuando hemos ido aumentando por cada capa el número de filtros y los filtros eran pequeños. (quinto experimento)

Matriz de confusión

694	27	70	39	11	7	3	14	99	36
15	808	1	16	1	7	4	11	45	92
86	5	414	156	128	82	51	48	12	18
21	18	62	407	55	290	57	72	12	6
26	5	90	99	517	47	58	146	6	6
13	9	29	201	30	605	14	90	5	4
7	1	29	127	54	24	735	15	4	4
9	7	13	43	33	99	6	767	2	21
94	61	5	25	1	4	3	7	753	47
20	111	2	14	7	2	2	36	40	766

5. Conclusiones

Si comparamos los dos mejores experimentos podremos concluir que el modelo de redes convolucionales es mucho más apto tanto en error como en porcentaje de aciertos para este tipo de clasificación con imágenes que el modelo del perceptrón multicapa.

Finalmente vemos la gran utilidad de las redes convolucionales ya que como reconocimiento de patrones y la habilidad para clasificar imágenes en base a ellos, podemos llegar a aplicaciones mucho más complejas.

Práctica 2 - Parte 2:

Problema de clasificación con Redes Convolucionales y Perceptrón Multicapa