Una red neuronal con una neurona

Juan I. Perotti,[*] Benjamín Marcolongo,[**] and Martín Abrudsky[***]
Instituto de Física Enrique Gaviola (IFEG-CONICET),
Ciudad Universitaria, 5000 Córdoba, Argentina and
Facultad de Matemática, Astronomía, Física y Computación,
Universidad Nacional de Córdoba, Ciudad Universitaria, 5000 Córdoba, Argentina
(Dated: 12 de febrero de 2025)

En este trabajo, bla bla bla...

I. INTRODUCCIÓN

Las redes neuronales... bla bla bla [1].

II. TEORÍA

Los Autoencoders bla bla... ReLU... Dropout... Convolución... El Error Cuadrático Medio... Adam...

III. DATOS

FashionMNIST consiste en un conjunto de 70.000 imágenes de 28x28 píxeles en escalas de grises ... etiquetadas en 10 categorías ... Los experimentos involucran un conjunto de entrenamiento de 60.000 imágenes elegidas al azar y un conjunto de 10.000 imágenes de validación con las imágenes restantes. Las imágenes se normalizan...

IV. MODELOS

La arquitectura del modelo 1 es la siguiente:

- La capa 1...
    1. Bla bla
    2. Blu blu
    3. Ble ble
- Cha cha
- Che che

V. RESULTADOS

La tabla I resume la lista de experimentos realizados con el fin de explorar sistemáticamente la variación de distintos hiperparámetros del modelo y del proceos de optimización.

En la fig. 1, vemos bla bla ...

VI. DISCUSIÓN

La comparación de ... bla bla bla

VII. CONCLUSIONES

Concluyendo ...

VIII. AGRADECIMIENTOS

JIP, BM y MA agradecen el finaciamiento y el apoyo institucional de CONICET, SeCyT y la UNC.

[*] juan.perotti@unc.edu.ar
[**] benjaminmarcolongo@unc.edu.ar
[***] martin.abrudsky@unc.edu.ar

[1] J. A. Hertz, A. S. Krogh, and R. G. Palmer, *Introduction To The Theory Of Neural Computation* (Taylor & Francis Group, Boca Raton London New York, 1999).

Apéndice A: Modelos

1. Modelo 1

El código bla bla bla...

```
## Definimos el primer modelo
class MultiLayerPerceptron(nn.Module):
    def __init__(self):
        super().__init__()
```

Experimento nº	n_1	n_2	p	Optimizador	Batch-size	learning-rate	Objetivo
1	128	64	0.2	Adam	256	10^{-3}	Experimento base.
2	64,256	64	0.2	Adam	256	10^{-3}	Variar n_1.
3	128	32,128	0.2	Adam	256	10^{-3}	Variar n_2.
4	128	64	0.0,0.5	Adam	256	10^{-3}	Variar p.
5	128	64	0.2	SGD	256	10^{-3}	Cambiar el optimizador.
6	128	64	0.2	Adam	128,512	10^{-3}	Variar el batch-size.
7	128	64	0.2	Adam	256	$10^{-2}, 10^{-4}$	Variar el learning-rate.

Tabla I. Lista de experimentos en donde se varían distintos hiperparámetros de un experimento base. En todos los casos se entrenaron los modelos durante 30 épocas, partiendo de pesos sinápticos (o parámetros) inicialmente elegidos al azar.

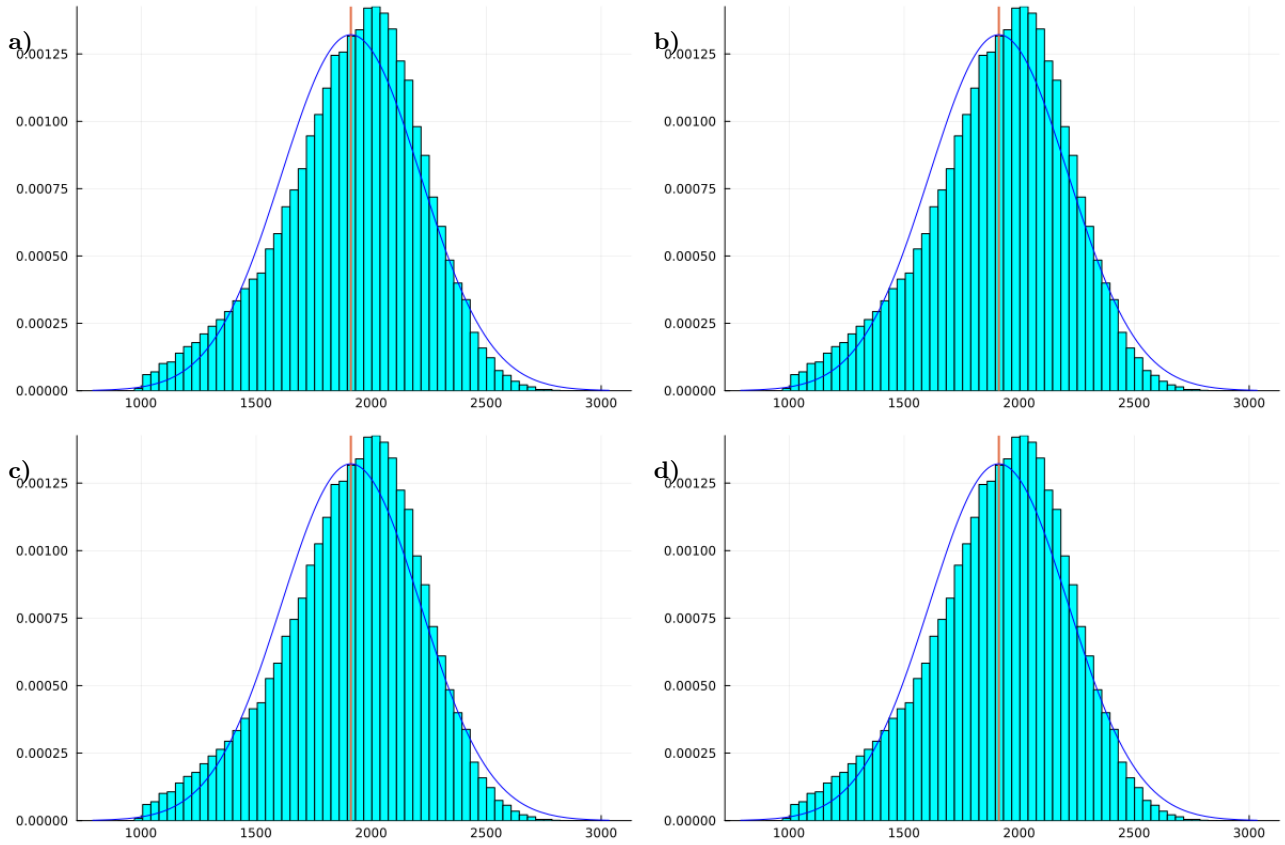

Figura 1. Un puñado de distribuciones Gaussianas. **a)** Una distribución Gaussiana. **b)** Otra distribución Gaussiana. **c)** Otra distribución Gaussiana más. **d)** Una distribución Gaussiana extra.

```
## "Achatamos" la matriz de 28x28 píxeles,
## transformandola en un vector de 784 elementos
self.flatten = nn.Flatten()

## Definimos el perceptrón multicapa con las
## siguientes capas:
##
## Entrada:        784 neuronas
## 1º capa oculta: 512 neuronas
## 2º capa oculta: 512 neuronas
```

```
        ## Salida:         10  neuronas
        ##
        ## Entre capa y capa, utilizamos función de
        ## activación ReLU
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 600),
            nn.ReLU(),
            nn.Linear(600, 120),
            nn.ReLU(),
            nn.Linear(120, 10),
            nn.ReLU()
        )

    def forward(self, x):
        x = self.flatten(x)
        x = self.linear_relu_stack(x)
        return x
```

2. Modelo 2

El código bla bla bla...

Apéndice B: Datos

Bla bla bla...