Reporte 12:
Red Neuronal

Jorge Torres

16 de mayo de 2022

Capítulo 1

Entrenamiento para Números

1.1. Objetivo

Esta actividad es una demostración básica de aprendizaje a máquina: se reconocerán dígitos de imágenes pequeñas en blanco y negro con una red neuronal. El objetivo consiste en estudiar de manera sistemática el desempeño de la red neuronal en términos de su puntaje F (F-score en inglés) para los diez dígitos en función de las tres probabilidades asignadas a la generación de los dígitos (n, g, b), variando a las tres en un experimento factorial adecuado.

1.2. Desarrollo

El desarrollo de la actividad está basado en el código implementado por E. Schaeffer, con algunas modificaciones para variar las probabilidades de generación de los dígitos y realizar el análisis estadístico [1]. La implementación completa se puede obtener del repositorio en GitHub de J. Torres [3].

Las funciones `binario` y `decimal` del código 1.1 convierten un número decimal en binario y uno binario en decimal, respectivamente.

Código 1.1: Conversiones Binario y Decimal

```
binario <- function(d, l) {
  b <-  rep(FALSE, l)
  while (l > 0 | d > 0) {
    b[l] <- (d %% 2 == 1)
    l <- l - 1
    d <- bitwShiftR(d, 1)
  }
  return(b)
}

decimal <- function(bits, l) {
  valor <- 0
  for (pos in 1:l) {
    valor <- valor + 2^(l - pos) * bits[pos]
  }
  return(valor)
}

df = data.frame()
```

Las líneas del código 1.2 establecen las listas de probabilidades de que los bits del mapa de bits creado sean de color negro, gris o blanco, respectivamente. Este mapa de bits se utilizará para entrenar y probar la red neuronal.

Código 1.2: Probabilidades de Negro, Gris y Blanco

```
probn = c(0.98,0.65,0.35)
probg = c(0.95,0.75,0.55)
probb = c(0.45,0.10,0.005)
```

En el código 1.3 se inician las iteraciones entre las listas de probabilidades y se realizan 12 repeticiones para cada iteración. Se toman los valores de un archivo de texto y las probabilidades respectivas para generar un mapa de bits con una serie de dígitos de manera aleatoria, como se aprecia en la figura 1.1.

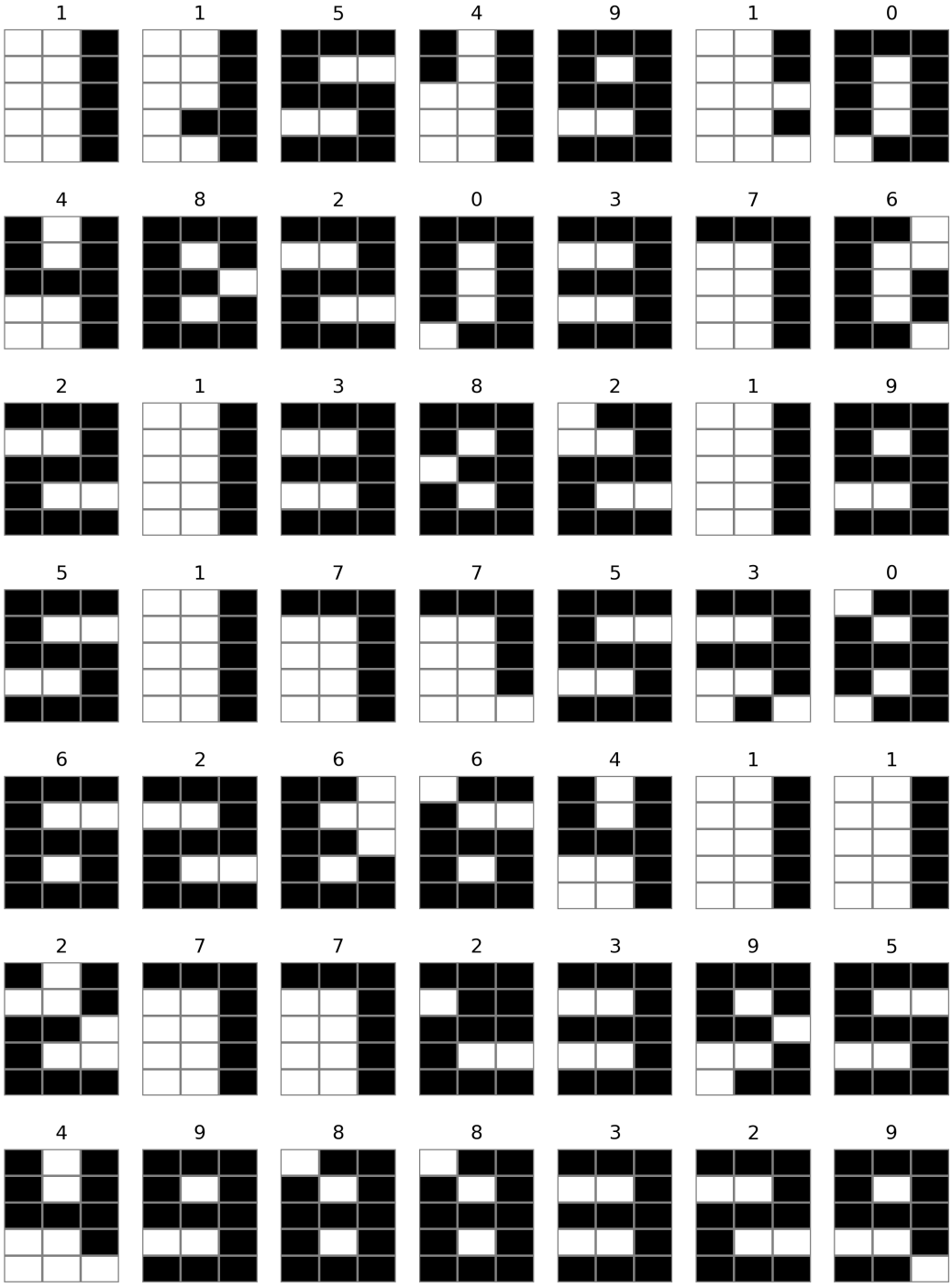

Figura 1.1: Serie de dígitos generados aleatoriamente, tomando las probabilidades del código 1.2.

Código 1.3: Generación del Mapa de Bits de los Dígitos

```
for (ne in probn){
  for (g in probg){
    for (b in probb){
      for (replica in 1:12){
        modelos <- read.csv("digits.txt", sep=" ", header=FALSE, stringsAsFactors=F)
        modelos[modelos=='n'] <- ne
        modelos[modelos=='g'] <- g
        modelos[modelos=='b'] <- b

        r <- 5
        c <- 3
        dim <- r * c

        tope <- 9
        digitos <- 0:tope
        k <- length(digitos)
        contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
        rownames(contadores) <- 0:tope
        colnames(contadores) <- c(0:tope, NA)
```

En el código 1.4 se establecen la tasa de aprendizaje y el factor por el cual irá disminuyendo dicha tasa en la etapa de entrenamiento. También se establece la cantidad de perceptrones necesarios para la red neuronal. La etapa de entrenamiento consiste en tomar dígitos enteros del 0 al 9 de manera aleatoria y comparar el resultado del código 1.3 con el valor real por medio de los perceptrones. En este caso se realizan 5000 pasos de entrenamiento y se ajustan los perceptrones dependiendo de la validez del resultado.

Código 1.4: Etapa de Entrenamiento de los Perceptrones

```
        tasa <- 0.15
        tranqui <- 0.99
        n <- floor(log(k-1, 2)) + 1
        neuronas <- matrix(runif(n * dim), nrow=n, ncol=dim)

        for (t in 1:5000) {
          d <- sample(0:tope, 1)
          pixeles <- runif(dim) < modelos[d + 1,]
          correcto <- binario(d, n)
          for (i in 1:n) {
            w <- neuronas[i,]
            deseada <- correcto[i]
            resultado <- sum(w * pixeles) >= 0
            if (deseada != resultado) {
              ajuste <- tasa * (deseada - resultado)
              tasa <- tranqui * tasa
              neuronas[i,] <- w + ajuste * pixeles
            }
          }
        }
```

La etapa de prueba consiste en tomar los perceptrones obtenidos de la etapa de entrenamiento y aplicarlos directamente al mapa de bits. De esta forma, la red neuronal determinará a qué valor pertenece cada dígito generado. Se hace un conteo de todos los resultados obtenidos, el cual se utiliza para su posterior análisis. Esto se implementa en el código 1.5.

Código 1.5: Etapa de Prueba y Obtención de Resultados

```
        for (t in 1:300) {
          d <- sample(0:tope, 1)
          pixeles <- runif(dim) < modelos[d + 1,]
          correcto <- binario(d, n)
          salida <- rep(FALSE, n)
          for (i in 1:n) {
            w <- neuronas[i,]
            deseada <- correcto[i]
            resultado <- sum(w * pixeles) >= 0
            salida[i] <- resultado
          }
          r <- min(decimal(salida, n), k)
          contadores[d+1, r+1] <- contadores[d+1, r+1] + 1
        }
```

Para obtener el puntaje F y realizar las pruebas estadísticas, se utiliza la ecuación 1.1, obtenida de la lectura en pruebas estadísticas de Shmueli [2], y se aplica para cada grupo de probabilidades por medio del código 1.6.

$$F = 2\frac{(precision)(recall)}{precision + recall} \tag{1.1}$$

Código 1.6: Obtención del Puntaje F

```
        precision = diag(contadores) / colSums(contadores[,1:10])
        recall = diag(contadores) / rowSums(contadores)
        fscore = (2 * precision * recall) / (precision + recall)
        result = c(ne, g, b, replica, fscore)
        df = rbind(df, result)
      }
    }
  }
}
```

1.3. Resultados

Las distribuciones de los puntajes f (F-score) para las 12 repeticiones de cada grupo de probabilidades se muestran en la figura 1.2. Se puede observar cómo aumenta el puntaje para los grupos en los que la probabilidad de bits negros es mayor y la probabilidad de bits blancos es menor. Esto es de esperarse, pues en general se crearían imágenes de los dígitos más nítidas y con menos ruido.

Los puntajes obtenidos no parecen seguir una distribución normal, por lo que se ha realizado una prueba de tipo `Shapiro-Wilk` para comprobar dicho supuesto, mientras que se realiza una prueba de `Levene` para determinar la homogeneidad de varianza entre los grupos. Los resultados se muestran en el cuadro 1.1. Los "outliers" se refieren a la cantidad de valores atípicos en los grupos. Se puede observar que, para la mayoría de grupos, el valor p obtenido es menor a 0,05, por lo que no tienen una distribución normal.

Debido a esta distribución, se ha optado por realizar una prueba del tipo `Kruskall-Wallis` para determinar si existe una diferencia significativa en el puntaje F dependiente de las probabilidades iniciales. Los resultados se muestran en el cuadro 1.2. Se obtiene un valor p mucho menor a 0,05, por lo que se puede saber que la diferencia en el puntaje F es significativa al variar las probabilidades.

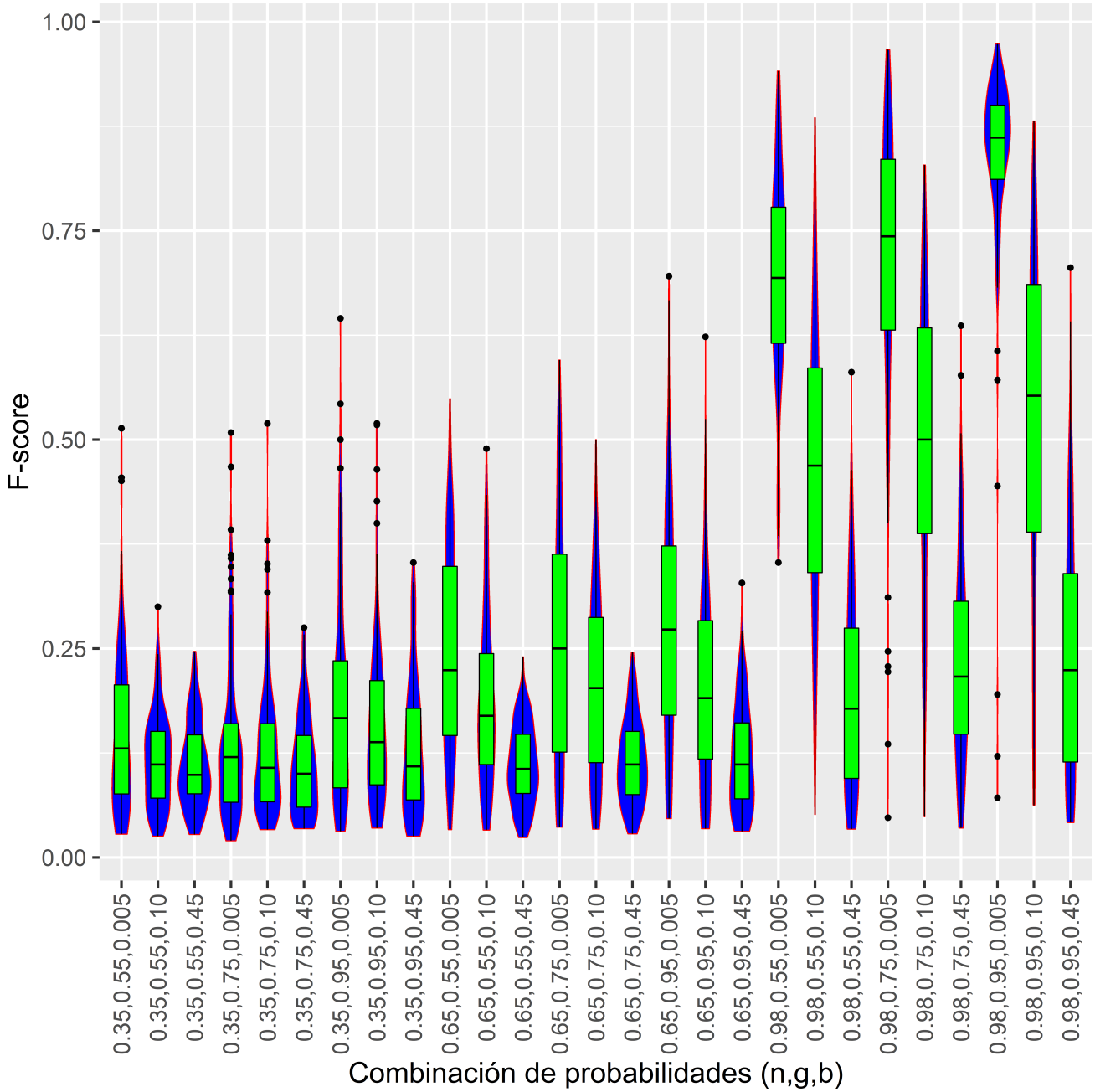

Figura 1.2: Puntajes F obtenidos para todos los grupos de probabilidades.

Cuadro 1.1: Resultados de las pruebas de normalidad y homogeneidad de varianza.

Outliers	47
Normalidad por grupo	0,35/0,55/0,005: $p = 1{,}72 \times 10^{-6}$ 0,35/0,55/0,10: $p = 1{,}81 \times 10^{-4}$ 0,35/0,55/0,45: $p = 1{,}06 \times 10^{-4}$ 0,35/0,75/0,005: $p = 1{,}62 \times 10^{-7}$ 0,35/0,75/0,10: $p = 2{,}05 \times 10^{-6}$ 0,35/0,75/0,45: $p = 2{,}43 \times 10^{-4}$ 0,35/0,95/0,005: $p = 2{,}35 \times 10^{-6}$ 0,35/0,95/0,10: $p = 4{,}48 \times 10^{-6}$ 0,35/0,95/0,45: $p = 2{,}02 \times 10^{-4}$ 0,65/0,55/0,005: $p = 5{,}30 \times 10^{-5}$ 0,65/0,55/0,10: $p = 2{,}37 \times 10^{-2}$ 0,65/0,55/0,45: $p = 2{,}31 \times 10^{-2}$ 0,65/0,75/0,005: $p = 3{,}42 \times 10^{-3}$ 0,65/0,75/0,10: $p = 3{,}93 \times 10^{-4}$ 0,65/0,75/0,45: $p = 1{,}16 \times 10^{-5}$ 0,65/0,95/0,005: $p = 6{,}09 \times 10^{-3}$ 0,65/0,95/0,10: $p = 8{,}46 \times 10^{-4}$ 0,65/0,95/0,45: $p = 7{,}73 \times 10^{-5}$ 0,98/0,55/0,005: $p = 6{,}68 \times 10^{-4}$ 0,98/0,55/0,10: $p = 2{,}58 \times 10^{-1}$ 0,98/0,55/0,45: $p = 4{,}47 \times 10^{-4}$ 0,98/0,75/0,005: $p = 8{,}90 \times 10^{-6}$ 0,98/0,75/0,10: $p = 1{,}25 \times 10^{-1}$ 0,98/0,75/0,45: $p = 3{,}07 \times 10^{-3}$ 0,98/0,95/0,005: $p = 8{,}79 \times 10^{-15}$ 0,98/0,95/0,10: $p = 2{,}38 \times 10^{-6}$ 0,98/0,95/0,45: $p = 1{,}34 \times 10^{-3}$
Homogeneidad de varianza	$p = 3{,}43 \times 10^{-22}$

Cuadro 1.2: Resultados al aplicar la prueba estadística `Kruskal Wallis`.

Chi cuadrada	Valor de p
1603,7	$2{,}2 \times 10^{-16}$

1.4. Conclusiones

Como se puede observar por la gráfica presentada y los resultados de los análisis estadísticos, es razonable concluir que no solamente existe una deferencia significativa en el puntaje F al variar las probabilidades iniciales, sino que esta diferencia es mucha más apreciable al tener mayores probabilidades de dibujar bits negros y menores probabilidades de dibujar un bit blanco. Esto es debido a que, con tales probabilidades, e obtienen imágenes menos ruidosas, por lo que la red neuronal puede realizar mejores identificaciones.

Capítulo 2

Entrenamiento para Números y Símbolos

2.1. Objetivo

El objetivo del primer reto consiste en extender y entrenar la red neuronal para que reconozca, además, por lo menos doce símbolos ASCII adicionales, aumentando la resolución de las imágenes a 5×7 del original de 3×5.

2.2. Desarrollo

El desarrollo para el primer reto es muy similar al implementado en el capítulo 1. La diferencia está en la cantidad de símbolos que se utilizan y, por ende, la cantidad de perceptrones que la red neuronal necesita para las fases de entrenamiento y prueba. En el código 2.1 se listan las probabilidades (n, g, b) utilizadas, así como la cantidad de dígitos que ahora debe reconocer la red neuronal.

Código 2.1: Nuevos Parámetros de Operación

```
modelos <- read.csv("digits2.txt", sep=" ", header=FALSE, stringsAsFactors=F)
modelos[modelos=='n'] <- 0.99
modelos[modelos=='g'] <- 0.92
modelos[modelos=='b'] <- 0.002

r <- 7
c <- 5
dim <- r * c

n <- 56
w <- ceiling(sqrt(n))
h <- ceiling(n / w)

tope <- 21
digitos <- 0:tope
k <- length(digitos)
contadores <- matrix(rep(0, k*(k+1)), nrow=k, ncol=(k+1))
rownames(contadores) <- 0:tope
colnames(contadores) <- c(0:tope, NA)

n <- floor(log(k-1, 2)) + 1
neuronas <- matrix(runif(n * dim), nrow=n, ncol=dim)
```

Utilizando estos datos, el mapa de bits cuyos símbolos ahora debe reconocer la red neuronal se observa en la figura 2.1.

Figura 2.1: Mapa de bits a reconocer con la inclusión de los nuevos símbolos.

2.3. Resultados

Lo que se muestra en el código 2.2 es la matriz de resultados que arroja el programa con los datos establecidos. Las filas representan el valor real de cada símbolo (21 en total), mientras que las columnas representan el valor que la red neuronal interpreta. De esta forma, cada relación fila-columna representa la cantidad de veces que la red neuronal reconoce el símbolo de la fila (real) como el símbolo de la columna (valor interpretado). Un ejemplo bastante extremo sería la intepretación del valor 0; se puede observar cómo, de las 14 pruebas que hizo la red neuronal para este valor, únicamente 3 de ellas fueron correctas, mientras que 11 de ellas las interpreta como un valor 1.

Código 2.2: Matriz de Interpretación

```
   0  1  2 3 4  5  6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 <NA>
0  3 11  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    0
1  0 11  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    0
2  0  0 12 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    0
3  3  0  7 1 0  0  1 1 0 0  0  0  0  0  1  0  0  0  0  0  0  0    0
4  0  0  0 0 0  1  0 0 0 0  0  0  7  2  0  0  0  0  0  0  0  0    0
5  0  0  0 0 0  0  0 2 0 0  0  0  1  1  1 11  0  0  0  0  0  0    2
6  7  0  1 0 0  0  1 0 2 0  0  0  0  0  0  0  0  0  0  0  0  0    0
7  0  0  1 0 0  0 11 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    0
8  7  2  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    0
9  0  2  0 0 0 15  0 0 0 0  0  0  1  1  0  0  0  0  0  0  0  0    0
10 0  0  1 0 0  0  0 0 2 0 11  0  0  0  0  0  0  0  0  0  0  0    1
11 0  0  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  3  0  0  0   10
12 0  0  0 0 0  0  0 0 0 0  0  0 15  0  0  0  0  0  0  0  0  0    2
13 0  0  0 0 0  1  0 0 0 0  0  0  0  6  0  0  0  0  0  0  0  1    1
14 0  0  0 0 0  0  0 0 1 0  1  0  0  0 17  2  0  0  0  0  0  0    0
15 0  0  0 0 0  0  0 0 3 0  0  0  0  0  0  7  0  0  0  0  2  0    1
16 2  0  0 0 0  0  0 0 0 0  0  0  0  0  0  0  3  7  0  0  1  3    0
17 3 11  0 0 0  0  0 0 1 1  0  0  0  0  0  0  1  0  0  0  0  0    4
18 0  0  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0 12  0  0  0    0
19 0  0  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0 13  0  0    0
20 0  0  0 0 0  0  0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0    9
21 0  8  0 0 0  0  0 0 0 1  0  0  0  0  0  0  0  1  0  0  0  0    1
```

2.4. Conclusiones

En esta ocasión, se puede observar que la red neuronal presenta algunos problemas al intentar reconocer símbolos que son más complejos. Sin embargo, los resultados aún son en su mayoría positivos al poder interpretar la mayoría de ellos como el símbolo que corresponde. Las imágenes no presentan demasiado ruido, pero es posible que se mejore la capacidad de reconocimiento de la red neuronal al incrementar la cantidad de perceptrones y acomodarlos en capas, donde los resultados de uno afecten la entrada de los que le siguen.

Bibliografía

[1] E. Schaeffer. Neuralnetwork, 2019. URL https://github.com/satuelisa/Simulation/tree/master/NeuralNetwork.

[2] Boaz Shmueli. Multi-class metrics made simple, part ii: the f1-score, 2019. URL https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1.

[3] J. Torres. P_12, 2022. URL https://github.com/FeroxDeitas/Simulacion-Nano/tree/main/Tareas/P_12.