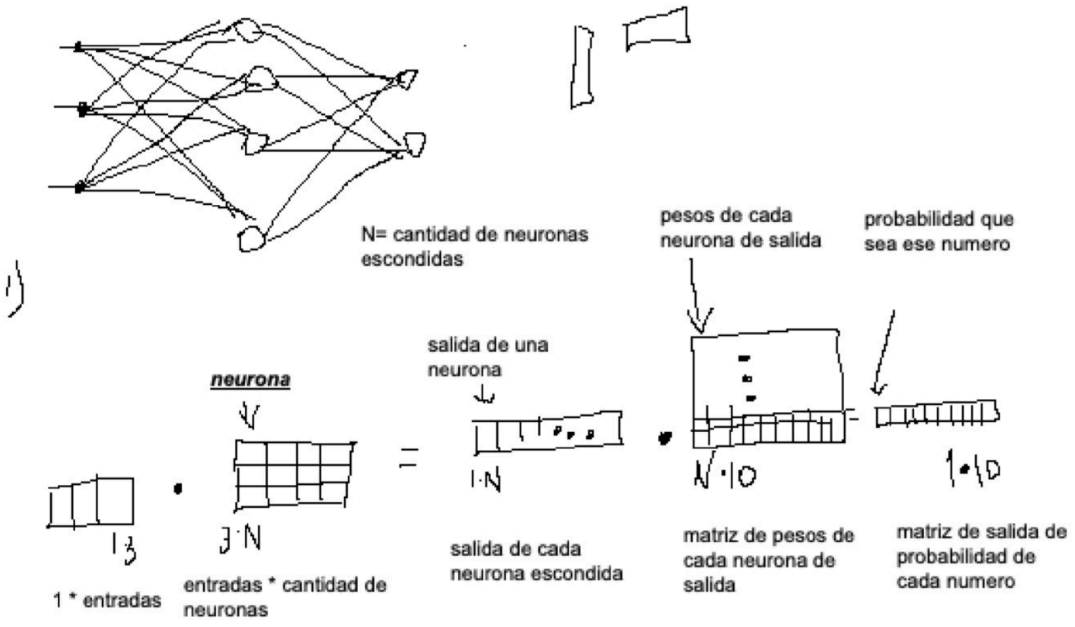
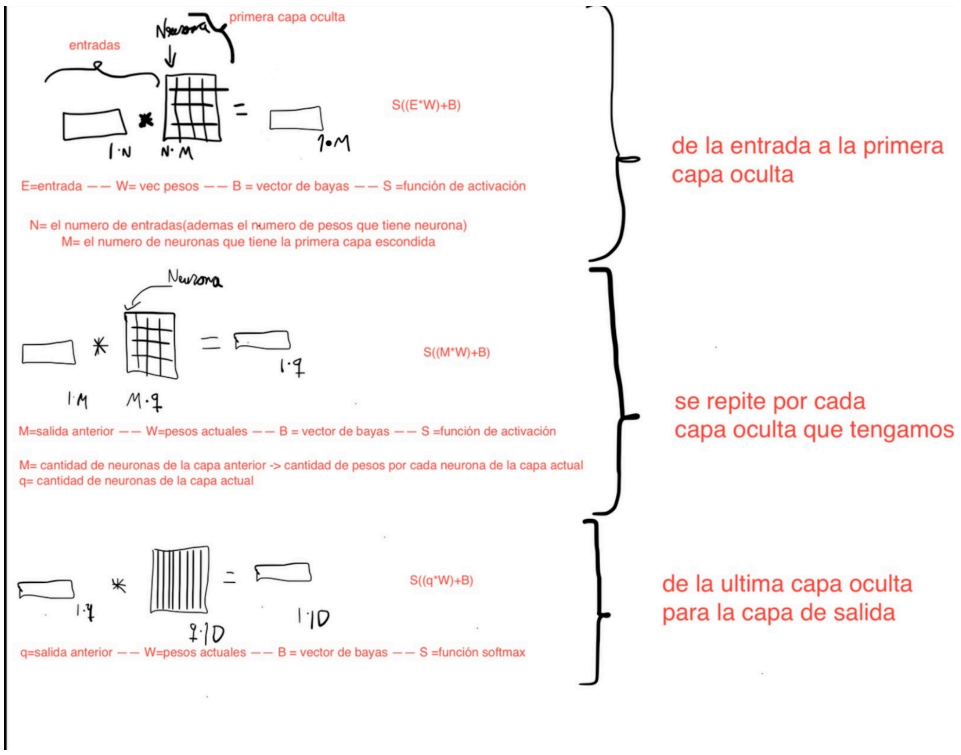
Al inicio de la implementación solo se podía ingresar el número de neuronas por capa, pero queda confirmado que mientras mas sea la cantidad de capas las ultimas son las que “aprenden mas” ademas tambien queda mucho mejor entendido como es que se “dividen” el conocimiento en las diferentes capas y perceptrones



luego el maximo porcentaje de precisión fue al rededor de 40, al parecer si o si el “conocimeinto” se debe dividir en dinttios niveles, luego se cambio el modelo a esta manera



donde obviamente fue mucho mejor ya que se le permitía al programa tener mas capas, en la implementación por un motivo de facilidad se decidió poner como macros el numero de

neuronas por capa escondida, por que si no se tendría que poner a mano cada, igualmente en las distintas pruebas, tener tantas capas no era tan conveniente como que se “perdia” la información supongo que el calculo de la de las primeras capas era importante, pero casi irrelevante por la cantidad enorme de capas que nomas, entre tantas combinaciones una de las que salio mejor y se quedó fue con solo dos capas xd, pero con 128 neuronas por capa, igualmente supongo que funciono por que el conocimiento estaba concentrado y con todas la pruebas podría decir que no deberian haber mas de 5 capas por que se van volviendo irrelevantes las primeras y el numero de neuronas debe ser alto en este caso es de 128
Resultados:

```
Inicializando Red Neuronal...
Arquitectura: 784 -> 128 -> 128 -> 10
Epocas: 50, Mini-batch: 32, LR: 0.1

Cargando datos...
Datos de entrenamiento: 60000 muestras
Datos de prueba: 10000 muestras

Iniciando entrenamiento...
Epoca 1, Error promedio: 0.437496, Tiempo: 137s
Epoca 10, Error promedio: 0.172282, Tiempo: 1431s
Epoca 20, Error promedio: 0.065330, Tiempo: 2934s
Epoca 30, Error promedio: 0.051068, Tiempo: 4434s
Epoca 40, Error promedio: 0.042156, Tiempo: 5938s
Epoca 50, Error promedio: 0.035795, Tiempo: 7445s

== TESTING ==

Matriz de Confusion:
         0     1     2     3     4     5     6     7     8     9
 0:    967     0     2     1     0     3     4     2     1     0
 1:      0  1116     3     2     0     1     5     2     6     0
 2:      9     0   980     6     7     2     7     9     9     3
 3:      0     0    15   974     1     3     0     9     7     1
 4:      1     2     4     0   937     0    10     2     2    24
 5:      8     1     1    18     3   833    11     1    11     5
 6:     11     3     6     1     5     7   918     0     7     0
 7:      3    11    20     4     3     0     0   974     0    13
 8:      5     1     2     9     4     4     4     4   939     2
 9:      8     7     1    12    13     5     1     8     3   951

Precision total: 95.89%

Precision por clase:
Clase 0: 98.67%
Clase 1: 98.33%
Clase 2: 94.96%
Clase 3: 96.44%
Clase 4: 95.42%
Clase 5: 93.39%
Clase 6: 95.82%
Clase 7: 94.75%
Clase 8: 96.41%
Clase 9: 94.25%
Program ended with exit code: 0
```

vemos que tenemos un 95% de rescisión, creo que se pueda mejorar pero ahí nomas por ahora, vamos a ver que tal va con el transformer, además la paralización se centró más en las operaciones con matrices y la propagación de adelante y atrás más la actualización de los pesos