**Descripción red neuronal.**

La red neuronal creada fue basada en una red convolucional, donde se aplican filtros en vez de matrices de pesos, inicialmente se enfrentó un problema de dimensiones para estos filtros al querer adaptarlos a una red neuronal convolucional sin aplicar la convolución directa sobre la imagen. La forma en la que se procedió fue la siguiente:

1. Se transformó la imagen inicial en un vector como se plasma a continuación:

|  |  |  |  |
| --- | --- | --- | --- |
| 1,1 | 1,2 | 1,3 | 1,4 |
| 2,1 | 2,2 | 2,3 | 2,4 |
| 3,1 | 3,2 | 3,3 | 3,4 |
| 4,1 | 4,2 | 4,3 | 4,4 |

Imagen inicial.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1,1 | 1,2 | 1,3 | 1,4 | 2,1 | 2,2 | 2,3 | 2,4 | 3,1 | 3,2 | 3,3 | 3,4 | 4,1 | 4,2 | 4,3 | 4,4 |

Imagen transformada a vector

1. Tomemos las siguiente matriz de 3x3 como nuestro filtro:

|  |  |  |
| --- | --- | --- |
| A | B | C |
| D | E | F |
| G | H | I |

Al convolucionarla con la imagen inicial, la primera y segunda posición de nuestra imagen resultante, será (1,1)=[ A\*(1,1) + B\*(1,2) + C\*(1,3) + D\*(2,1) + E\*(2,2) + F\*(2,3) + G\*(3,1) + H\*(3,2) + I\*(3,3) ] y (2,2)=[ A\*(1,2) + B\*(1,3) + C\*(1,4) + D\*(2,2) + E\*(2,3) + F\*(2,4) + G\*(3,2) + H\*(3,3) + I\*(3,4)], a partir de estos resultados nos dimos cuenta que sería igual multiplicar la imagen transformada a vector por una matriz de 16\*4 como la siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| A |  |  |  |
| B | A |  |  |
| C | B |  |  |
|  | C |  |  |
| D |  | A |  |
| E | D | B | A |
| F | E | C | B |
|  | F |  | C |
| G |  | D |  |
| H | G | E | D |
| I | H | F | E |
|  | I |  | F |
|  |  | G |  |
|  |  | H | G |
|  |  | I | H |
|  |  |  | I |

Matriz de filtro (pesos) – Las posiciones vacías corresponden a valores de cero

Al multiplicar la imagen vector por esta matriz obtenemos la imagen convolucionada con lados de tamaño 1 x ((n-f)/stride + 1)\*\*2, n:= longitud del lado de la imagen inicial, f:= longitud del lado de la matriz de convolucion y stride:= tamaño del paso con el que movemos el filtro tanto horizontal como vertical. Lo siguiente muestra las dimensiones obtenidas:

1x16 \* 16x4 = 1x4

Luego es aplicado un reshape con filas de longitud ((n-f)/stride + 1) obtenemos una matriz de 2x2, nuestra imagen final.

1. A la matriz resultante (imagen finalmente convolucionada) se le aplica una función ReLu, definida como ReLu(matriz(i,j))=max(0,matriz(i,j)).

Los tres pasos anteriores pueden ser vistos de la siguiente forma:

.  
 .  
 .

La primera capa siendo una lista de nodos que corresponde a todos los pixeles de la imagen, las flechas indicando los pesos, que son organizados en una matriz de pesos descrita por la matriz de filtro (pesos) si no existe dicha flecha entre un nodo de la primera capa y la segunda es porque su peso es cero, y la función de activación para la segunda capa está dada por la función ReLu ya explicada.

El objetivo de un red convolucional es reducir la dimensionalidad aplicando filtros y capas de pooling, para esta última, la imagen es dividida en cuadriculas te tamaño 2x2 (en el caso de nuestra red) y max(cuadricula\_i) representa el pixel i en la imagen resultante.

Con lo ya expuesto, nuestra red consta de dos capas de convolución, una de maxpooling y una red final de clasificación como se muestra a continuación:

Matriz de pesos 1 (w1) Matriz de pesos 2 (w2) Matriz de pesos 3 (w3)

MAX-POOLING

128x128 nodos 126x126 nodos(a3) 124x124 nodos(a2) 62x62 nodos(a1) 1x2 nodos(a0)

Los filtros aplicados son de 3x3:

* Primer filtro: [0,7,0;9,9,1;1,0,0]
* Segundo filtro: [1,-1,2;-1,1,3;-1,4,1]

El max-pooling es de tamaño 2x2, reduce a la imagen que acaba de pasar por el segundo filtro en un **75%** de su tamaño original.

La capa de 62x62 nodos recibe la imagen del max-pooling para luego ser clasificada; la matriz de pesos 3 (w3) fue generada en un principio aleatoriamente.

La capa de 1x2 nodos es un vector binario indicando [0,1] para infectado y [1,0] para no infectado.

**Proceso de aprendizaje**

Para este proceso usamos backpropagation, gradiente en descenso, una taza de aprendizaje del 1% por iteración y una función de costo SUM((y-x)\*\*2). Para la activación de la última capa se hace a través de una función ReLu.

Para minimizar el error queremos derivar nuestra función de costo respecto al peso w3, que es nuestro parámetro en esta primera red de clasificación, queremos obtener dc/dw3.

Siendo C=SUM((y-x)\*\*2) nuestra función de costo, con x = a0 =ReLu(a1\*w3) , entonces tenemos que dc/dw3=(da\_0/dw3)\*( dc/da\_0), aplicando el método del gradiente en descenso, tenemos que w3=w3 – (lr)\*dc/dw3, siendo lr la taza de aprendizaje. Para propagar el error hacia atrás primero debemos pasar por el max-pooling, por lo que volveremos a la imagen dividida por cuadriculas en el forwardpropagation y solo mantendremos el máximo de cada cuadricula, los demás elementos de cada cuadricula serán cero (0), esto sería el “backpropagation” de la capa de max-pooling y desde este punto comenzaría la propagación del error de la red de clasfificación hasta la parte convolucional, lo obtenido anteriormete será nuestra nueva función de costo (C\_a2) en a2 derivada con respecto a a3 (llamaremos dicha derivada dC\_a2/da2), obtendriamos que w2=w2-lr\*(da3/dw1)\*(da2/da3)\*( dC\_a2/da2).

Finalmente necesitamos w1=w1-lr\*(dC\_a2/w1), siendo dC\_a2/w1= (da3/da)\*(dC\_a2/da3)

La función que controla el número de iteraciones hasta llegar al resultado deseado es C.

Luego de esto se probó con varias imágenes infectadas en el entrenamiento; en una de las iteraciones con una de las imágenes encontramos matrices de pesos w1, w2 y w3 que hacía converger al resultado deseado en menos de 10 e incluso 5 iteraciones, a partir de acá frenamos el enteramiento, guardamos las matrices y empezamos a hacer pruebas en pacientes infectados y no infectados y nos dimos cuenta que en pacientes infectados el error en la primera iteración al pasar por la fase de entrenamiento era inferior a 1, mientras que en no infectados era mayor a 1. El error anteriormente obtenido fue resultado de comparar la salida de la red con la esperada de una persona enferma ([0,1])

Realizamos pruebas en 959 imágenes basándonos en lo anterior para verificar si era posible clasificar sin un entrenamiento tan profundo, de las cuales 450 eran infectadas y 509 no infectadas, de las infectadas reconoció 317 correctamente y de las no infectadas reconoció 436 correctamente, de lo cual la red tuvo una certeza del **78.51%**