

Isabella Salgado 201730418  
Johan Hernández 201729696  
Juan Pablo Naranjo 201730006

## Tarea 8

### Punto 1

a)

El primer paso a seguir fue dividir los datos entregados en clases. Para hacer esto, primero se obtuvieron solo los landmarks asociados a la emoción felicidad y se guardaron en un subconjunto separado. Este subconjunto de solo landmarks de la emoción felicidad correspondió a los datos de clase 1.

Para obtener el subconjunto de datos de clase 0, se tomó el pool de datos que quedó después de extraer todas las emociones de felicidad, y de este se extrajo un subconjunto de datos del mismo tamaño del subconjunto de los datos de clase 1, con la diferencia de que en este caso se extrajeron datos del resto de emociones (neutral, triste, miedoso, furioso y disgusto) de manera equitativa por emoción, hasta tener un tamaño del subconjunto de datos igual al de la clase 1. Este nuevo subconjunto correspondió a los datos de clase 0.

El objetivo del clasificador creado en este punto es que logre diferenciar emociones de felicidad (clase 1) del resto de emociones (clase 0) a partir de los landmarks que se tenían disponibles.

b)

Teniendo los datos de clase 1 y clase 0 debidamente separados, se procedió a separar **cada clase** en subconjuntos de entrenamiento (80 %) y de validación (20 %). Es decir, al final se obtuvo un conjunto de datos de entrenamiento para la clase 1 y uno para la clase 0, y un conjunto de datos de validación para la clase 1 y otro para la clase 0 (en total 2 subconjuntos de entrenamiento y 2 subconjuntos de validación). Se tomó la precaución de que los datos de ambas clases estuvieran equitativamente distribuidos entre los subconjuntos de entrenamiento y validación.

c)

Se procedió a calcular la media de Procrustes de los datos, solo usando los datos de entrenamiento de cada clase. Es decir, para calcular esta media de Procrustes, se creó un nuevo conjunto de datos de entrenamiento global, que combinó las dos particiones de entrenamiento creadas en el literal anterior.

En la figura 1 se muestran todos los datos de entrenamiento.

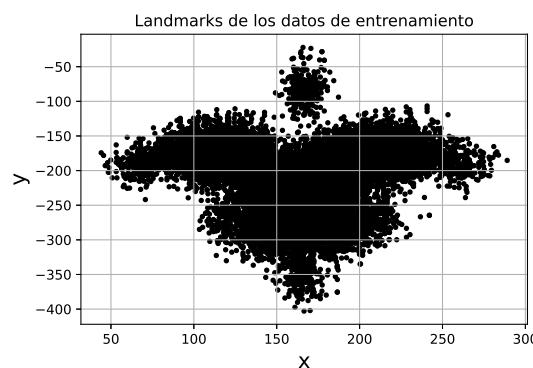


Figura 1: Landmarks de los datos de entrenamiento

Para calcular la media de Procrustes, primero se procedió a removerles su media, así obteniendo el resultado mostrado en la figura 2

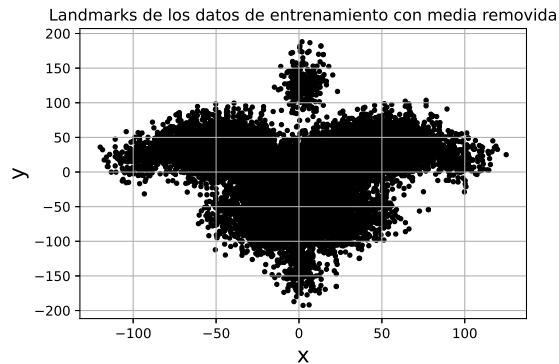


Figura 2: Landmarks de los datos de entrenamiento con media removida

Finalmente, realizando los cálculos necesarios, el resultado obtenido al calcular la media de Procrustes se muestra en la figura 3.

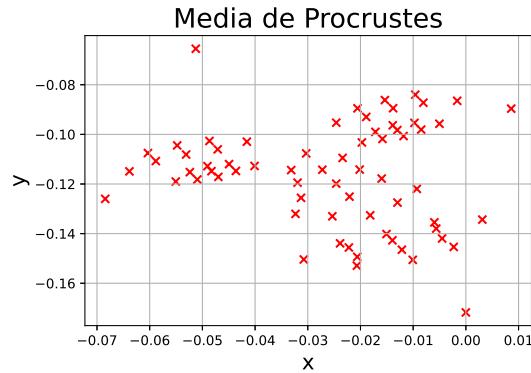


Figura 3: Media de Procrustes

Teniendo la media de Procrustes, se procedió a alinear los landmarks de los datos de entrenamiento y los datos de validación a dicha media. Los landmarks de los datos de entrenamiento alineados a la media de Procrustes se pueden apreciar en la figura 4, mientras que los datos de validación alineados a la media de Procrustes se pueden observar en la figura 5.

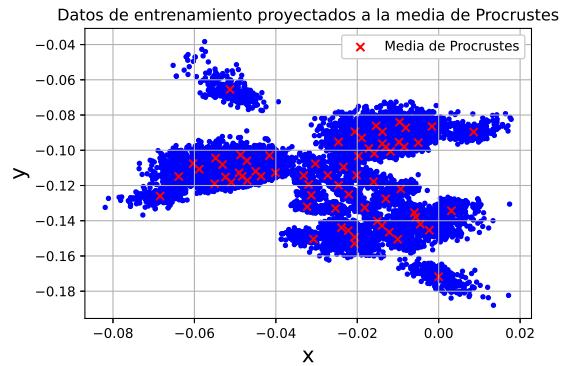


Figura 4: Landmarks de los datos de entrenamiento proyectados a la media de Procrustes

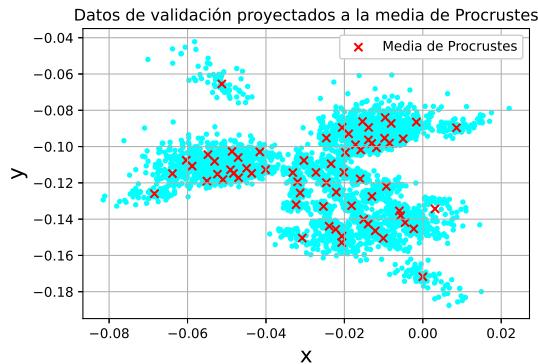


Figura 5: Landmarks de los datos de validación proyectados a la media de Procrustes

d)

Usando los landmarks alineados a la media de Procrustes de ambas particiones (entrenamiento y validación), se procedió a crear un vector de tamaño  $2 \times 67 = 134 \times 1$  (el tamaño de cada cara es de  $67 \times 1$ ), el cual se organizó de tal forma que las características asociadas a la componente horizontal de cada landmark se ubicaran en las posiciones pares del vector, y las características asociadas a la componente vertical de ese mismo landmark se ubicaran en las posiciones impares del vector. Este vector se creó para cada cara (expresión facial) dentro de los datos de entrenamiento y de validación. Para tener un orden establecido, se creó una matriz con estos vectores para los datos de entrenamiento y otra para los datos de validación de cada clase. Para más claridad, el vector para cada expresión quedó organizado de la siguiente forma,

$$v = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ x_{67} \\ y_{67} \end{bmatrix}$$

en donde  $x_i, y_i$  corresponden respectivamente a las componentes horizontales y verticales de cada landmark, y cada uno de estos vectores conforman las columnas de la matriz creada para cada conjunto de entrenamiento y validación global.

e)

En este punto, se encontraron los índices de los landmarks asociados a las características asociadas a: ojo izquierdo, ojo derecho, ceja izquierda, ceja derecha, boca, nariz, y contorno de la cara dentro de cada vector  $v$  creado en el literal anterior. A continuación se presentan las listas de índices para cada parte de la cara, teniendo en cuenta que la programación se realizó en Python (índices empiezan en cero):

- Ojo izquierdo: 18, 20, 22, 24, 26, 28, 30, 32, 34
- Ojo derecho: 0, 2, 4, 6, 8, 10, 12, 14, 16
- Ceja izquierda: 52, 54, 56, 58, 60, 62, 64, 66
- Ceja derecha: 36, 38, 40, 42, 44, 46, 48, 50
- Boca: 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124
- Nariz: 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96
- Contorno de la cara: 126, 128, 130, 132

f)

Se procedió a crear la función clasificadora a partir de regresión logística, conocida como la función de entropía cruzada para los datos de entrenamiento. Se creó un vector  $t$  del mismo tamaño de cada vector de características que tuvo un 1 en las posiciones  $t_i$  donde se encontraban los datos de clase 1 y 0 en las posiciones  $t_i$  donde se encontraban los datos de clase 0. Teniendo este vector, y sabiendo que

$$y_i = \frac{1}{1 + e^{w^T x_i}} \quad (1)$$

se definió la función de entropía cruzada como:

$$-\sum_{i=1}^m [t_i \ln(y_i) + (1 - t_i) \ln(1 - y_i)] + \lambda \|w\|_2^2 \quad (2)$$

Para realizar la clasificación se procedió a encontrar el vector  $w \in \mathbb{R}^{m+1}$  (contando el término de bias  $w_0$ ) que minimiza la función de entropía cruzada, en donde  $\lambda > 0$  es un parámetro de regularización que además minimiza la magnitud de cada componente del vector  $w$  y  $m$  es el tamaño del conjunto de datos. Esta minimización se realizó usando la función *minimize* de *scipy.optimize* en Python.

Una vez encontrado este vector  $w$ , se utilizó la regla MAP para realizar la clasificación entre clase 1 y clase 0. Esta regla de clasificación dicta que si

$$w^T x_i > 0 \quad (3)$$

entonces el dato  $x_i$  pertenece a la clase 1, mientras que si

$$w^T x_i < 0 \quad (4)$$

entonces el dato  $x_i$  pertenece a la clase 0.

g)

Para terminar de ajustar el clasificador, se realizaron pruebas para diferentes valores de  $\lambda$  y mirar con qué valor se obtenía el menor error de clasificación posible. Se corrió el clasificador esta vez con los datos de validación (globales), y se obtuvieron los resultados mostrados en la figura 6. Hay que tener en cuenta de que los errores obtenidos cada vez que se corre el script varían, pues cada vez que se corre, los landmarks seleccionados para la clase 0 se vuelven a asignar de manera aleatoria, cambiando levemente los errores obtenidos en este punto de la clasificación.

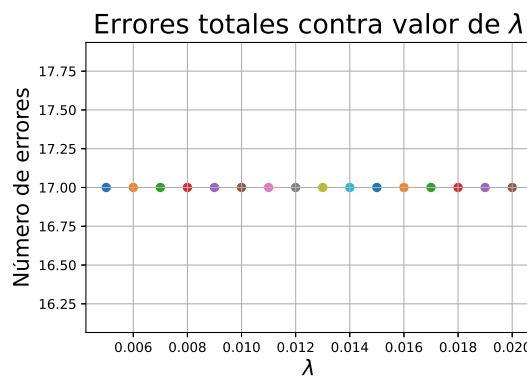


Figura 6: Errores totales contra valor de  $\lambda$

Como se puede notar, en este caso de prueba particular, la cantidad de errores se mantuvo constante para más de 10 valores de  $\lambda$ . Por lo tanto, se puede utilizar cualquiera de estos para evaluar el clasificador. En este caso, se seleccionó un valor  $\lambda = 0,019$ .

**h)**

Usando el valor de  $\lambda$  elegido en el literal anterior, se procedió a evaluar el clasificador, obteniendo los resultados mostrados a continuación.

- Número de clasificaciones correctas: 55
- Porcentaje de clasificaciones correctas (datos de validación): 80.88 %
- Número total de errores: 13
- Porcentaje de error (datos de validación): 19.12 %
- Errores de clasificación de la clase 1: 0
- Errores de clasificación de la clase 0: 13
- Porcentaje de error Clase 1:  $0/34 = 0.0\%$
- Porcentaje de error Clase 0:  $13/34 = 38.24\%$

Tabla 1: Matriz de confusión

	Clase 1	Clase 0
Clase 1	34	0
Clase 0	13	21

Se puede notar que el porcentaje de clasificaciones correctas es un valor relativamente alto, mientras que las clasificaciones incorrectas fueron bajas, demostrando un funcionamiento adecuado del clasificador. De la matriz de confusión, se puede notar que no se cometieron errores al clasificar los datos de clase 1, mientras que para diferenciar los datos de clase 0 se presentaron más problemas. Sin embargo, la cantidad de clasificaciones correctas fue mayor al número de clasificaciones incorrectas. Los errores cometidos se pueden deber al hecho de que los landmarks de expresiones faciales como enojado o disgusto de ciertas instancias particulares pueden ser muy similares a los landmarks de la expresión facial de felicidad, generando confusión para el clasificador.

**i)**

En la figura 7 se puede apreciar una gráfica de barras que representa los índices dentro de cada vector de características de cada expresión facial dentro de los datos de validación que tuvieron más relevancia al momento de realizar la clasificación con el vector  $w$  óptimo encontrado en el literal anterior, recordando que las componentes de este vector representan los pesos o importancia de la característica en el índice  $j$  al momento de realizar la clasificación.

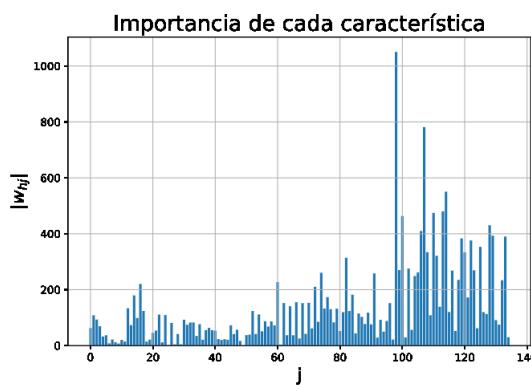


Figura 7: Importancia de las características

A continuación se presentan nuevamente los índices correspondientes a cada característica dentro de una expresión facial particular:

- Ojo izquierdo: 18, 20, 22, 24, 26, 28, 30, 32, 34
- Ojo derecho: 0, 2, 4, 6, 8, 10, 12, 14, 16
- Ceja izquierda: 52, 54, 56, 58, 60, 62, 64, 66
- Ceja derecha: 36, 38, 40, 42, 44, 46, 48, 50
- Boca: 98, 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124
- Nariz: 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96
- Contorno de la cara: 126, 128, 130, 132

Es claro que los índices de mayor importancia para decidir si la expresión facial es clase 1 o clase 0 son los índices relacionados con la boca, pues en la gráfica se puede notar una magnitud de los parámetros del hiperplano de clasificación (vector  $w$ ) mucho más alta alrededor de estos índices en comparación con el resto. Los siguientes índices más importantes corresponden a los de la ceja izquierda y la nariz. Los índices menos importantes corresponden a los de los ojos.

## Punto 2

a)

Para la solución de este literal se tomaron todas las imágenes del *dataset* provisto y por medio de una función anónima de *Matlab* se filtraron con una matriz de variables de tamaño 3x3. Finalmente, se creó la función de activación mostrada en la ecuación 5, donde  $w$  es un vector de 197 posiciones que almacena los pesos de la red a optimizar y  $v$  es la imagen filtrada anteriormente, pero, ahora en forma de un vector de tamaño 196.

$$\hat{y} = \frac{1}{1 + \exp\left(w_0 + \sum_{j=1}^{196} v_j w_j\right)} \quad (5)$$

b)

Con el fin de resolver el problema de optimización del error cuadrático solicitado, se creó un único vector a optimizar de tamaño 206x1 llamado  $p$ , donde las primeras 9 posiciones corresponden a los coeficientes  $h$  del filtro y las restantes 197 posiciones corresponden a los pesos  $w$ .

El problema de optimización a resolver se puede apreciar en la figura 6.

$$E(p) = \sum_{i=1}^{200} (y_i - g(I_i; p))^2 + \lambda \|p^T\|_2^2 \quad (6)$$

Donde,  $p$  es el vector a optimizar,  $g(I_i; p)$  corresponde a la función de activación mostrada en la ecuación 5,  $I_i$  corresponde a cada imagen que se está evaluando y  $\lambda$  corresponde a un número real mayor a cero que se usa para regularizar la función de error mostrada en la figura 6, con el fin de que al momento de optimizar el vector  $p$  esta optimización no entregue valores muy elevados dentro de este vector.

c)

Con el fin de encontrar un buen valor de  $\lambda$  se realizaron 150 pruebas con distintos valores de  $\lambda$ , en la figura 8 se pueden observar como cambian los errores de acuerdo al valor de  $\lambda$ .

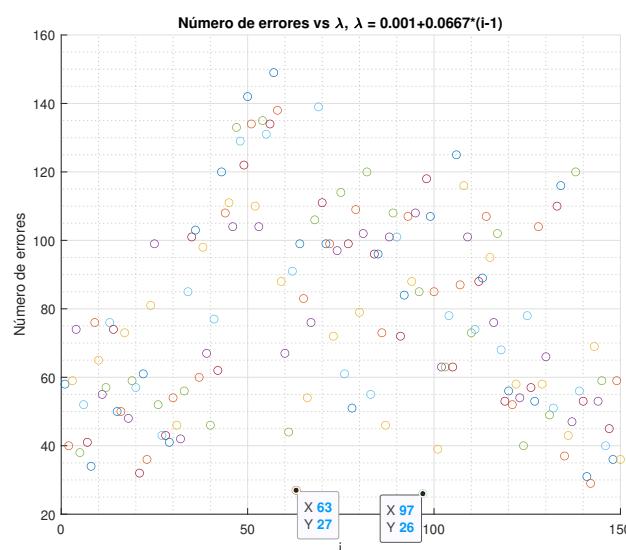


Figura 8: Número de errores de  $\lambda$ ,  $\lambda = 0.001 + 0.0667*(i-1)$

Antes de analizar la figura 8 cabe mencionar que el valor de  $\lambda$  corresponde a la presentada en la ecuación 7, con  $i = [1, 150]$ .

$$\lambda = 0,001 + 0,0667 \cdot (i - 1) \quad (7)$$

Tal como se puede apreciar en la figura 8 los errores cambian bastante entre un valor de  $\lambda$  a otro, esto quiere decir que la función objetivo es muy sensible a cambios relativamente pequeños (0.0667) de  $\lambda$ . El mejor valor  $\lambda$  se consiguió cuando  $i = 97$ , es decir  $\lambda = 6,4042$ , con el cual se obtuvieron 26 errores.

d)

Es importante aclarar que para este punto se realizó un nuevo entrenamiento de la red usando el mejor valor de  $\lambda$  obtenido en el literal anterior de este taller, pero, se cambiaron las condiciones de parada de la función *fminsearch* de *Matlab*, se ajustaron el número máximo de funciones evaluadas a 80000 y el número máximo de iteraciones a 100000. Con esto, se logró reducir el número de errores a 11, en la tabla 2 se puede observar la matriz de confusión asociada a este entrenamiento.

Tabla 2: Matriz de confusión - Punto 2

Clase	Número 2	Número 9
Número 2	94	6
Número 9	5	95

En las figuras 9 y 10 se pueden observar algunos ejemplos de imágenes que el algoritmo clasificó erróneamente y correctamente para el número 2, respectivamente.

Ejemplos de imágenes en las que falló el algoritmo - Número 2

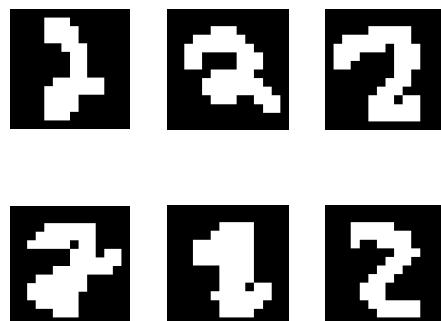


Figura 9: Ejemplos de imágenes en las que falló el algoritmo - Número 2

Ejemplos de imágenes en las que acertó el algoritmo - Número 2



Figura 10: Ejemplos de imágenes en las que acertó el algoritmo - Número 2

En las figuras 11 y 12 se pueden observar algunos ejemplos de imágenes que el algoritmo clasificó erróneamente y correctamente para el número 9, respectivamente.

Ejemplos de imágenes en las que falló el algoritmo - Número 9

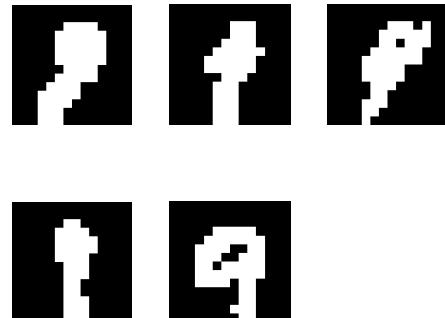


Figura 11: Ejemplos de imágenes en las que falló el algoritmo - Número 9

Ejemplo de imágenes en las que acertó el algoritmo - Número 9



Figura 12: Ejemplos de imágenes en las que acertó el algoritmo - Número 9

En la figura 13 se puede observar el filtro resultante de 3x3 como columnas en 3D.

Grafico de barras en 3D del filtro resultante

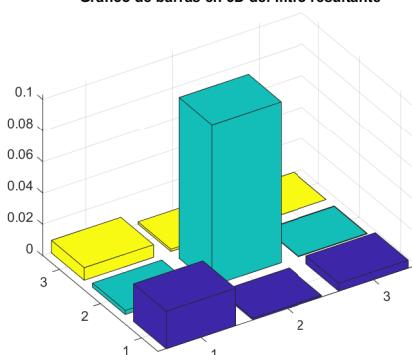


Figura 13: Gráfico de barras en 3D del filtro resultante

En las figuras 14 y 15 se puede apreciar algunos ejemplos de imágenes que el algoritmo clasificó correctamente después de ser filtradas con los coeficientes presentados en la figura 13.

Ejemplos de imágenes en las que acertó el algoritmo - Filtradas - Número 2



Figura 14: Ejemplos de imágenes en las que acertó el algoritmo - Filtradas - Número 2

Ejemplos de imágenes en las que acertó el algoritmo - Filtradas - Número



Figura 15: Ejemplos de imágenes en las que acertó el algoritmo - Filtradas - Número 9

Analizando la figura 13 donde se pueden apreciar visualmente los coeficientes del filtro encontrado, se nota que el elemento más importante es el central, seguido de un valor de una esquina, esto es muy importante debido a que esta información indica que el filtro tenderá a realizar una detección de bordes hacia donde esté el elemento de la esquina que tiene mayor valor. Esto se puede confirmar al observar las figuras 14 y 15, en ellas es fácil notar que el filtro tiende a señalar los bordes de abajo a la derecha, como si hubiera una fuente de luz en ese lugar. Por esto, el tipo de operación que tiene este filtro es detección de los bordes inferiores derechos.

En la figura 16 se pueden observar los pesos obtenidos en una vista 3D, pero, en forma de imagen de tamaño 14x14, vista desde distintos ángulos. En la figura 17 también es posible ver estos mismos pesos, pero, en un formato 2D.

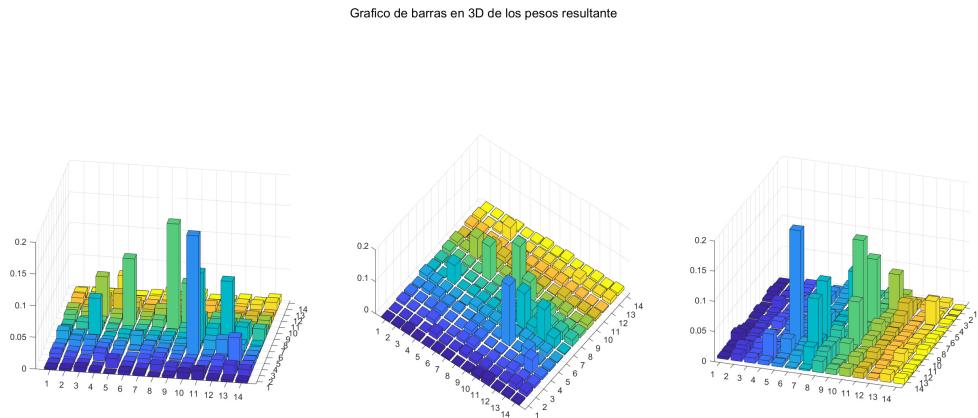


Figura 16: Gráfico de barras en 3D de los pesos resultante

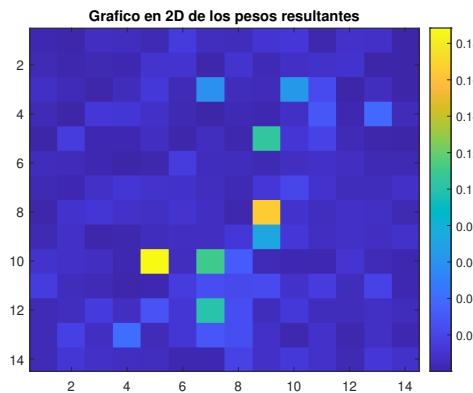


Figura 17: Gráfico en 2D de los pesos resultante

Tal como se puede apreciar en las figuras 16 y 17, es posible determinar que la región en la que más se enfoca la red neuronal para realizar la clasificación es la parte central, esto se puede observar más fácilmente en la figura 17, donde los valores de mayor tamaño están ubicados entre los valores del eje x, [4, 10], y, entre los valores del eje y, [7, 11]. Esto indica que cuando la red neuronal estaba evaluando una imagen se enfocaba en mirar como era la parte de abajo de esta, posiblemente para determinar si se encontraba con una línea aproximadamente vertical, número 9, o una línea curva y otra aproximadamente horizontal, número 2. En las figuras 14 y 15 es más fácil observar estas posibles diferencias.