Entrenamiento de CNN asistido por Algoritmos Evolutivos

Miguel Langone - Posgrado en Computación
mlangone13@gmail.com - 4.651.185-1

***Abstract*—Este documento detalla el uso de algoritmos evolutivos como alternativa para el entrenamiento de redes neuronales convolucionales. Asimismo se presenta una mutación basada en gradiente por descendiente estocástico aplicada en un problema de clasificación de imágenes.**

***Index Terms*—Algoritmo, Gen, Genoma, Pipeline, Cruzamiento, Mutación, Red neuronal, Backpropagation, Entrenamiento.**

I. Introducción al problema

HOY en día uno de los recursos más utilizados para el procesamiento de imágenes son las redes neuronales convolucionales. Esto se debe mayormente a la gran capacidad de generalización para distintos tipos de problemas como detección de objetos, segmentación y clasificación de imágenes, entre otros. El entrenamiento de las mismas se lleva a cabo de una manera muy similar a las redes tradicionales aplicando el algoritmo de Backpropagation como método de optimización sobre una función de costo (loss function).

En este informe veremos una alternativa a dicho método de optimización la cual consiste en aplicar técnicas de algoritmos evolutivos para encontrar la mejor configuración de pesos (w_i) en un tiempo razonable y típicamente menor que el del algoritmo basado en Backpropagation. Además se propone una nueva técnica de mutación llamada Mutación SGD basada en aplicar uno o más pasos del descendiente por gradiente estocástico bajo determinadas condiciones.

A. ¿Como funciona Backpropagation?

El algoritmo *Backpropagation* se utiliza para aprender los pesos de una red multicapa de tipo *feed-forward*. Dada una cantidad arbitraria de salidas, se puede considerar la siguiente medida de error sobre el conjunto de entrenamiento:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in S} (t_{kd} - o_{kd})^2 \tag{1}$$

donde:

- S es el conjunto de unidades (neuronas) de salida.
- D es el conjunto de ejemplos de entrenamiento.
- t_{kd} es el valor objetivo para el ejemplo d y para la salida k.
- o_{kd} es el valor devuelto por una unidad, aplicado al ejemplo d, para la salida k.

Backpropagation busca en todo el espacio de hipótesis definido los posibles valores para los pesos en cada una de las unidades de la red. La idea es mediante el uso del algoritmo del descendiente por gradiente estocástico modificar el vector de pesos en la dirección que produce el mayor decremento del error hasta llegar al mínimo posible.

B. ¿Qué limitaciones tiene Backpropagation?

Al utilizar el descendiente por gradiente estocastico en busca de minimizar la función de error antes descrita, el algoritmo Backpropagation tiene el problema en caer en mínimos locales y no asegurar el mínimo global de la función objetivo. Además es necesario que las funciones de activación que se utilizan en cada capa sean derivables para que el algoritmo funcione correctamente.

C. Propuesta

Dada una red para clasificar imágenes de colores de tamaño 32X32 en 10 clases distintas(CIFAR10[1]) con 4 capas convolucionales y 2 capas completamente conectadas, se busca encontrar los pesos W(ij) que maximicen la accuracy para dicha tarea.

La propuesta consiste en sustituir la optimización basada en Backpropagation por un algoritmo evolutivo capaz de obtener resultados similares en un tiempo razonable con menor o igual capacidad de cómputo.

La comparación se realizará contra la misma red convolucional entrenada tradicionalmente en un ambiente de software y hardware idéntico.

II. Diseño de la solución

En esta sección se detallan todos los aspectos del diseño de la solución del problema, desde las herramientas utilizadas hasta las decisiones de diseño tomadas.

A. Biblioteca utilizada

Se utilizará la biblioteca DEAP[3] desarrollada en Python para el entrenamiento del algoritmo genético. Esto se debe a que permite una integración muy facil con Keras[4] para la utilización y evaluación de redes neuronales. Todo el pipeline se desarrollo en Python 3.6.

B. Representación de la solución

La solución del problema esta dada por un array de tamaño P=1250858, donde P es la cantidad de parámetros entrenables de todas las capas convolucionales y capas densas. Cada Gen es de tipo float de 32 bits.

La solución tiene la forma:

$$ind(i) = \left|a_{11} \quad a_{12} \quad a_{13} \quad ... \quad a_{i1} \quad a_{i2} \quad a_{i3} \quad .... \quad a_{ij}\right|$$

donde:

$$lenght(ind(i)) = P$$

Cada individuo se recompone en la arquitectura de red de la figura 1.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
activation_1 (Activation)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 30, 30, 32)	9248
activation_2 (Activation)	(None, 30, 30, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 15, 15, 32)	0
dropout_1 (Dropout)	(None, 15, 15, 32)	0
conv2d_3 (Conv2D)	(None, 15, 15, 64)	18496
activation_3 (Activation)	(None, 15, 15, 64)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	36928
activation_4 (Activation)	(None, 13, 13, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 64)	0
dropout_2 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 512)	1180160
activation_5 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
activation_6 (Activation)	(None, 10)	0

Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0

Fig. 1. Arquitectura de red convolucional.

C. Función de fitness

La función de fitness a maximizar es la accuracy de la red CIFAR10 utilizando los pesos del individuo i:

$$f(ind(i)) = accuracy(CIFAR10(ind(i)))$$

siendo la función de accuracy de la siguiente manera:

$$accuracy = \frac{tp + tn}{tp + fp + tn + fn}$$

siendo

- **tp**: Cantidad de verdaderos positivos
- **tn**: Cantidad de verdaderos negativos
- **fp**: Cantidad de falsos positivos
- **fn**: Cantidad de falsos negativos

D. Estrategia de inicialización de individuos

La inicialización de los pesos de la red neuronal se realiza utilizando la inicialización de Xavier[7].

La misma consiste en elegir números aleatorios entre:

$$+ - \frac{\sqrt{6}}{\sqrt{n_i + n_{i+1}}}$$

siendo n_i la cantidad de entradas de la capa y n_{i+1} la cantidad de salidas de la misma.

Luego de realizar esa selección se transforma la arquitectura de red en un vector de tamaño 1250858, siendo así en uno de los individuos del algoritmo evolutivo.

E. Estrategia Evolutiva

La técnica utilizada realizar el algoritmo evolutivo fue (μ, λ) con μ la cantidad de padres que se eligen, siendo éste un parámetro configurable (**POP**), y λ = 50 (fijo) la cantidad de hijos que se crean en cada generación de dicho algoritmo.

F. Operadores evolutivos

A continuación se detallan los operadores evolutivos utilizados para la realización del algoritmo evolutivo en su totalidad.

1) Operadores de selección: La estrategia de selección elegida fue selección por torneo con un tamaño 3 fijo. No se detalla más sobre este operador dado que no fue probado exhaustivamente como si lo fueron los demás operadores.

2) Operadores de cruzamiento: Luego de aplicar los operadores de selección se aplica un operador de cruzamiento de un punto con la probabilidad **CROSS_PROB** la cual será un parámetro a determinar en la sección III.

3) Operadores de mutación: Como operador de mutación se presenta un nuevo tipo de mutación especifico llamado *Mutación SGD* el cual se detalla en II-G. La probabilidad de realizar dicha mutación esta dada por **MUT_PROB** la cual será otro parámetro más a determinar en la sección III.

G. Mutación SGD

Llamamos mutación SGD a la aplicación del descendiente por gradiente estocástico sobre un individuo y un subconjunto de elementos del entrenamiento (en este caso un subconjunto de imágenes). En pocas palabras, la mutación SGD es un entrenamiento parcial y acotado a un par de épocas con la finalidad de aproximar al individuo a un mínimo local.

Esta mutación tiene 3 parámetros configurables los cuales son:

- SAMPLE_RATIO: porcentaje del conjunto de entrenamiento elegido de manera aleatoria para realizar el entrenamiento (entre 0.0 y 1.0).
- BATCH_SIZE: tamaño del batch para ajustar durante el entrenamiento parcial. Se elije 64 elementos de batch.
- N_ITER: cantidad de épocas las cuales se aplica el algoritmo de SGD.

Como se puede ver, si se utiliza un SAMPLE_RATIO muy alto y un N_ITER también alto, la mutación de un individuo se decanta en el algoritmo de Backpropagation. Además, el uso de un N_ITER muy alto puede producir el fenómeno conocido como *overfitting* debido a la perdida de generalización del algoritmo.

Dicho esto es que se elije SAMPLE_RATIO = 0.05 y N_ITER = 3.

H. Hardware utilizado

En cuanto al hardware utilizado para el entrenamiento, se utilizó el ClusterUY [2] el cual permite ejecutar distribuido utilizando varias CPUs de alta capacidad de procesamiento.

- **Procesadores** Se utilizaron 30 procesadores Xeon Gold 6138 con 3GB de memoria RAM cada uno.
- **Almacenamiento** El almacenamiento provisto por ClusterUY es de 300GB de disco de estado solido.
- **Sistema Operativo** El sistema operativo es Linux CentOS 7

III. Configuración de parámetros

En esta sección se detallan las pruebas realizadas para buscar la mejor configuración de parámetros del algoritmo evolutivo. Los parámetros que se van a probar junto con sus valores posibles son los siguientes:

Tipos de algoritmos			
# Alg	POP	MUT PROB	CROSS PROB
1	30	0.05	0.5
2	30	0.05	0.8
3	30	0.1	0.5
4	30	0.1	0.8
5	50	0.05	0.5
6	50	0.05	0.8
7	50	0.1	0.5
8	50	0.1	0.8

TABLE I
DISTINTAS CONFIGURACIONES DE ALGORITMOS SEGÚN POBLACIÓN, PROBABILIDAD DE CRUZAMIENTO Y PROBABILIDAD DE MUTACIÓN.

A. Test de normalidad sobre las muestras

En todos los casos se aplicó el test de Anderson - Darling[8] para comprobar la normalidad de los datos obtenidos y así poder compararlos. Esto se realizó con la biblioteca de Python, SciPy [6], con el paquete *stats.anderson* que provee el test directamente y retorna si la distribución es o no normal.

Las tablas 2 3 y muestran los resultados de 30 ejecuciones para cada algoritmo en 3 instancias distintas. Para acortar el tiempo insumido se realizaron 20 generaciones (pasos de evolución) de cada combinación y cada instancia.

		Instancia	Instancia 1			Instancia 2			Instancia 3		
			Max Acc	Mean Acc	STD Dev Acc	Max Acc	Mean Acc	STD Dev Acc	Max Acc	Mean Acc	STD Dev Acc
#POP	Mut Prob	Cross Prob									
30	0.05	0.5	0.4710	0.441591	0.022300	0.46350	0.415400	0.027207	0.50425	0.416908	0.027535
		0.8	0.4794	0.431781	0.023874	0.45525	0.412758	0.025097	0.47250	0.414067	0.024606
	0.1	0.5	0.5248	0.481669	0.024796	0.50225	0.459350	0.019168	0.51425	0.460742	0.028091
		0.8	0.5342	0.470868	0.026259	0.48125	0.444333	0.017767	0.50050	0.450358	0.019624
50	0.05	0.5	0.5018	0.444756	0.023529	0.47650	0.421708	0.023854	0.47850	0.415117	0.025236
		0.8	0.4908	0.443623	0.030307	0.47575	0.412842	0.026163	0.47375	0.409283	0.025215
	0.1	0.5	0.5108	0.478201	0.016102	0.49100	0.455683	0.016560	0.48475	0.451908	0.019722
		0.8	0.5042	0.463756	0.019866	0.48700	0.443692	0.026341	0.48700	0.446500	0.019980

Fig. 2. Resultados estadísticos de cada algoritmo probado en 3 instancias distintas para la función de fitness. De izquierda a derecha se tienen el máximo fitness (accuracy), fitness medio, y la desviación estándar de dicho algoritmo para esa instancia.

		Instancia	Instancia 1			Instancia 2			Instancia 3		
			Max Elapsed Time	Mean Elapsed Time	STD Dev Elapsed Time	Max Elapsed Time	Mean Elapsed Time	STD Dev Elapsed Time	Max Elapsed Time	Mean Elapsed Time	STD Dev Elapsed Time
#POP	Mut Prob	Cross Prob									
30	0.05	0.5	585.430803	521.745615	24.437289	466.933791	422.403172	19.607599	452.778627	420.785072	13.812575
		0.8	746.316652	706.364144	24.789834	623.731419	568.553354	20.758944	613.912909	569.912134	20.594343
	0.1	0.5	768.695339	689.171115	30.563393	608.147165	557.944349	23.160279	617.549176	557.684428	27.499869
		0.8	976.926695	903.185832	32.816633	745.428003	705.847191	23.098694	761.439846	708.900775	26.899257
50	0.05	0.5	563.900180	539.820142	15.989484	459.459568	427.614431	16.678255	469.287759	432.997367	18.076713
		0.8	761.460937	734.047248	20.275110	614.341956	586.707867	18.949213	628.322354	588.960352	15.789166
	0.1	0.5	754.845568	694.054892	38.624649	652.158874	582.845869	28.153603	641.246135	580.468143	27.768426
		0.8	966.439411	905.739424	26.348486	770.798395	725.155535	21.082239	769.009404	723.767458	23.059995

Fig. 3. Resultados estadísticos para cada algoritmo probado en 3 instancias distintas para el tiempo insumido de las ejecuciones (en segundos).

En las figuras 4, 5 y 6 se muestran los resultados de las medias y las desviaciones estándar para cada algoritmo en cada instancia de prueba.

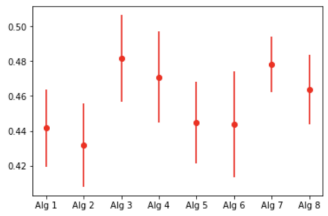

Fig. 4. Resultados de media y desviación estándar para la instancia 1 para cada algoritmo.

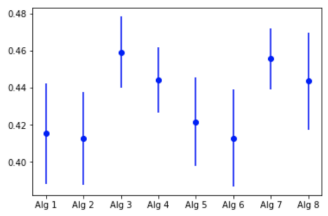

Fig. 5. Resultados de media y desviación estándar para la instancia 2 para cada algoritmo.

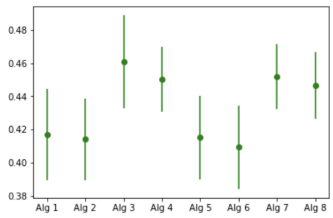

Fig. 6. Resultados de media y desviación estándar para la instancia 3 para cada algoritmo.

Luego de aplicar el test de normalidad de Anderson-Darling se comprueba que los valores de fitness siguen una distribución normal y se procede a comparar los algoritmos para determinar el mejor. Para esto se realiza un test de hipótesis de ANOVA (Analysis of Variance) para determinar si las medias de los conjuntos son similares o no.

B. Elección del mejor algoritmo

Una vez determinado que los valores de los mejores fitness siguen una distribución normal, es que podemos realizar un

test de hipótesis con la finalidad de determinar si los valores obtenidos provienen de una media similar o no.

Para ello se utiliza un test de hipótesis de una vía (1-way) ANOVA para descartar o confirmar si los valores de cada algoritmo provienen de una media similar o no. Por simplicidad se utiliza la biblioteca SciPy[6] de Python que provee el test antes mencionado [5] y realiza todos los cálculos necesarios.

Para ello se planten las siguientes hipótesis:

- Hipótesis nula: Los valores provienen de la misma media.
- Hipótesis alternativa: Los valores no provienen de la misma media.

Se utiliza $\alpha = 5\%$, buscando así tener un 95% de confidencia para asegurar que la hipótesis nula no sea rechazada. Por tanto, si el p_valor para el test de ANOVA es menor que el nivel de significancia (0.05) entonces rechazamos la hipótesis nula, asegurando que las medias de los grupos difieren.

Se realizan todas las combinaciones entre los distintos algoritmos para (uno vs uno) y se obtienen los resultados de los p_valores de la tabla 7.

	Alg1	Alg2	Alg3	Alg4	Alg5	Alg6	Alg7	Alg8
Alg1	1	2.24717e-08	0.111291	2.54474e-05	0.601053	1.53353e-09	0.772219	0.000183794
Alg2	2.24717e-08	1	1.3008e-10	0.112718	2.74316e-07	0.530117	2.40887e-06	0.00358646
Alg3	0.111291	1.3008e-10	1	1.77104e-07	0.0415278	4.47558e-12	0.103748	7.59027e-07
Alg4	2.54474e-05	0.112718	1.77104e-07	1	0.00018903	0.204929	0.000548638	0.249549
Alg5	0.601053	2.74316e-07	0.0415278	0.00018903	1	4.06977e-08	0.874206	0.00154833
Alg6	1.53353e-09	0.530117	4.47558e-12	0.204929	4.06977e-08	1	1.17876e-06	0.00352707
Alg7	0.772219	2.40887e-06	0.103748	0.000548638	0.874206	1.17876e-06	1	0.00406708
Alg8	0.000183794	0.00358646	7.59027e-07	0.249549	0.00154833	0.00352707	0.00406708	1

Fig. 7. Resultados del p_valor para los test de hipótesis ANOVA para todos las las configuraciones de algoritmos. En rojo se encuentran aquellos algoritmos los cuales no podemos asegurar que las medias difieran.

El algoritmo con mejor fitness promedio y mejor fitness total es el número 3 el cual, según los test de hipótesis realizados, difiere su media a todos los algoritmos exceptuando el algoritmo 1 y el 7.

De todas formas, se entiende que el algoritmo 3 fue superior en generar al los demás algoritmos propuestos y proviene de una muestra con media distinta al resto.

Por esto el algoritmo seleccionado es el número 3 con la configuración:

- Cantidad de individuos: 30
- Probabilidad de mutación: 0.1
- Probabilidad de cruzamiento: 0.5

Este algoritmo será evaluado contra el entrenamiento utilizando Backpropagation en las siguientes secciones.

IV. Evaluación de algoritmo final

Una vez seleccionado la configuración de parámetros la cual dio mejores resultados es que se pasa a la fase final de evaluación. En esta etapa se busca comparar el típico entrenamiento basado en Backpropagation contra el algoritmo evolutivo recientemente elaborado.

La evaluación consiste en dos etapas, la primera busca evaluar cual algoritmo llega a las mejores soluciones y la segunda etapa busca determinar cual llega más rápido a su propia mejor solución.

Como se explicó anteriormente, ambos algoritmos se probaron utilizando el ClusterUY con 30 núcleos de procesamiento en paralelo y con 3GB de memoria RAM cada uno.

En cuanto al algoritmo utilizando Backpropagation únicamente, se entrenó durante 25 épocas obteniendo el Benchmark de la figura 8 para superar con el algoritmo evolutivo.

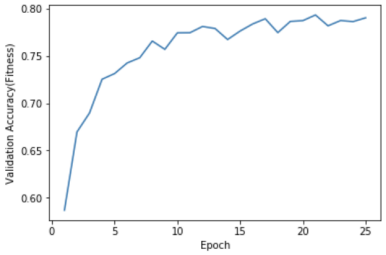

Fig. 8. Evolución de Accuracy de validación durante el entrenamiento de 25 épocas utilizando Backpropagation.

La demora promedio es de 1600 segundos (aprox. 27 minutos) para realizar 25 épocas de entrenamiento obteniendo como mejor accuracy de validación 79%.

A. *Comparación de mejor solución*

En la sección III-B se realizaron test de hipótesis para comparar las muestras de cada algoritmo y determinar que efectivamente tienen medias y varianzas distintas para elegir la mejor combinación de parámetros.

Gracias a eso se determinó que la mejor configuración de parámetros, la cual será tratada de ahora en adelante, es:

- **#Población:** 30
- **Probabilidad de Mutación:** 0.1
- **Probabilidad de Cruzamiento:** 0.5

Se ejecutó el algoritmo con dichos parámetros 8 veces, y un total de 200 generaciones cada uno y se obtienen los resultados de la figura 9. Cada algoritmo demora en promedio un total de 4300 segundos (aprox. 71 minutos).

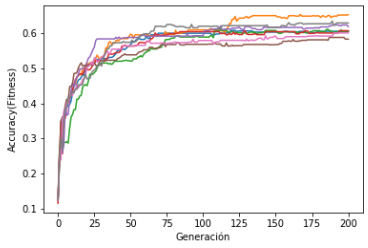

Fig. 9. Resultados para 8 ejecuciones del algoritmo evolutivo con la configuración encontrada.

Los resultados finales (tabla II) muestran que el algoritmo propuesto no fue superior al entrenamiento por Backpropagation en encontrar una mejor solución al cabo de varias generaciones.

Algoritmo	Mejor Accuracy	Accuracy Media	Tiempo insumido (seg)
Backpropagation	0.79	0.79	1600
AE(Mutación SGD)	0.65	0.62	4300

TABLE II
COMPARACIÓN DE RESULTADOS ENTRE EL ALGORITMO BACKPROPAGATION Y EL ALGORITMO EVOLUTIVO.

No obstante el sesgo introducido con la mutación SGD puede ser la clave para obtener mejores resultados, siendo que los parámetros intrínsecos a dicha mutación no fueron evaluados en este caso.

B. Velocidad de convergencia a mejor solución

Para analizar la velocidad de convergencia a la mejor solución debemos probar que tan rápido logramos llegar a la mejor solución encontrada, o en su defecto, a una solución casi similar. Para ello establecemos 4 percentiles los cuales vamos a evaluar, para encontrar en que generación logramos encontrar distintas soluciones.

- **Percentil 99**: En que iteración encontramos la solución con 99% de precisión en base a la mejor encontrada.
- **Percentil 95**: En que iteración encontramos la solución con 95% de precisión en base a la mejor encontrada.
- **Percentil 90**: En que iteración encontramos la solución con 90% de precisión en base a la mejor encontrada.
- **Percentil 70**: En que iteración encontramos la solución con 70% de precisión en base a la mejor encontrada.

En las tablas III y IV se muestran las generaciones (o épocas dependiendo el algoritmo) necesarias para lograr cada uno de los percentiles de desempeño para cada algoritmo.

Percentil	Accuracy esperada	Época	Tiempo insumido (seg)
100	0.79	25	1600
99	0.78	16	1024
95	0.75	7	448
90	0.71	3	192
70	0.55	1	64

TABLE III
ÉPOCAS NECESARIAS PARA ENCONTRAR EL ACCURACY DETERMINADO PARA CADA PERCENTIL USANDO BACKPROPAGATION.

Percentil	Accuracy esperada	Generación	Tiempo insumido (seg)
100	0.65	200	4300
99	0.64	120	2580
95	0.61	66	1419
90	0.58	26	559
70	0.45	9	193

TABLE IV
GENERACIONES NECESARIAS PARA ENCONTRAR EL ACCURACY DETERMINADO PARA CADA PERCENTIL USANDO EL AE CON MUTACIÓN SGD.

Las tablas muestran comportamientos similares de ambos algoritmos para encontrar las soluciones en el percentil 70 y 90. En los percentiles más altos el algoritmo evolutivo se estancó en un posible mínimo local y por eso es que el percentil 99 esta cerca del 60% de las generaciones realizadas (versus el percentil 99 de Backpropagation que esta en el 65% de las generaciones).

Luego de estos analisis no se puede comprobar una mejora en la velocidad de convergencia por parte del algoritmo evolutivo contra el entrenamiento por Backpropagation.

V. CONCLUSIONES Y TRABAJOS A FUTURO

En este informe se trabajó en una alternativa para entrenar redes neuronales convolucionales a partir de algoritmos evolutivos. Se planteo una alternativa de mutación basada en el descendiente por gradiente la cual agregar otro punto más de aleatoriedad a cada generación del algoritmo y realiza una perturbación (mutación) completa del individuo en un momento dado.

Algunas de las conclusiones que se pueden obtener luego de finalizado todos los experimentos es que realizar un entrenamiento completo de una red neuronal utilizando algoritmos evolutivos no queda descartado sino que se puede decir que es posible. Por otro lado, obtener un desempeño similar al encontrado por Backpropagation puede ser muy costo y requiere de una búsqueda exhaustiva por muchos parámetros configurables.

A. Trabajos a futuro

Como trabajos a futuro se plantean:

- **Búsqueda exhaustiva de parámetros:** Uno de los puntos más importantes a trabajar es realizar un análisis sobre los parámetros de la mutación SGD. La misma trae consigo 3 nuevos parámetros que pueden ser configurables para obtener un desempeño mucho mayor al logrado.
- **Expansión en una red mayor:** Se podría realizar el mismo procedimiento pero en una red mucho más grande como puede ser una VGG16 (con 16 capas convolucionales y 135 millones de pesos).

APPENDIX

A. Instructivo de uso

Para utilizar el algoritmo propuesto basta con ejecutar el comando:

```
>> python SGDEvolve.py
```

En caso de querer cambiar la configuración del algoritmo evolutivo, se debe entrar en el módulo y editar aquellos parámetros que se prefiera cambiar.

B. Tablas de configuración paramétrica

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	30	0.1	0.5	0.507400	675.263749	0.45700	557.262313	0.44300	562.914361
2	30	0.1	0.5	0.462600	722.205839	0.48575	555.006684	0.48575	597.996395
3	30	0.1	0.5	0.499600	664.324996	0.43775	518.122462	0.46600	561.795199
4	30	0.1	0.5	0.524800	713.949862	0.45200	562.101747	0.46925	617.549176
5	30	0.1	0.5	0.501000	688.342978	0.46375	556.877147	0.46625	552.946088
6	30	0.1	0.5	0.471200	627.808804	0.43900	598.580927	0.38550	499.893179
7	30	0.1	0.5	0.460200	652.655045	0.45275	556.644231	0.45900	567.521845
8	30	0.1	0.5	0.477400	693.092453	0.46175	566.010698	0.43175	504.758432
9	30	0.1	0.5	0.437000	680.697837	0.46275	573.670320	0.45100	550.500334
10	30	0.1	0.5	0.491400	667.554018	0.43675	521.951378	0.41950	580.539961
11	30	0.1	0.5	0.423000	676.376827	0.43750	550.639270	0.48675	532.267510
12	30	0.1	0.5	0.498600	718.682921	0.46850	608.147165	0.46275	549.191714
13	30	0.1	0.5	0.506800	678.551874	0.47575	539.833202	0.47750	616.529238
14	30	0.1	0.5	0.462200	637.230360	0.42500	564.439583	0.44125	564.592607
15	30	0.1	0.5	0.476800	695.455622	0.46500	536.505832	0.48750	567.219418
16	30	0.1	0.5	0.511800	720.464720	0.45200	521.467011	0.42125	521.770837
17	30	0.1	0.5	0.493000	672.590721	0.50150	521.198285	0.48925	594.381736
18	30	0.1	0.5	0.465000	736.610226	0.45075	547.995898	0.46525	563.255934
19	30	0.1	0.5	0.485600	768.695339	0.45375	565.849353	0.47800	545.154048
20	30	0.1	0.5	0.473400	687.968668	0.49025	567.097204	0.47625	546.312885
21	30	0.1	0.5	0.507521	683.835141	0.44450	579.211718	0.41350	540.553941
22	30	0.1	0.5	0.500997	711.385178	0.50225	605.458679	0.45550	550.399294
23	30	0.1	0.5	0.492950	728.286643	0.46250	555.463034	0.51425	528.795238
24	30	0.1	0.5	0.475422	700.066110	0.43500	536.729982	0.43550	573.997937
25	30	0.1	0.5	0.494176	674.991128	0.47850	585.240363	0.49725	579.494005
26	30	0.1	0.5	0.430979	723.246978	0.47625	536.520214	0.43675	533.918615
27	30	0.1	0.5	0.499475	671.307172	0.45400	555.379612	0.46725	564.915135
28	30	0.1	0.5	0.485254	686.869941	0.46875	573.128889	0.49075	565.020997
29	30	0.1	0.5	0.493166	644.586384	0.44575	549.423124	0.46150	531.561563
30	30	0.1	0.5	0.441325	672.035906	0.44375	572.374136	0.48725	564.785231

Fig. 10. Resultados de evaluación para el algoritmo 1.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	30	0.05	0.5	0.453000	499.632479	0.37825	407.955167	0.40650	403.587121
2	30	0.05	0.5	0.463200	482.215756	0.40525	424.743373	0.44500	431.097160
3	30	0.05	0.5	0.471000	492.200700	0.39725	419.574893	0.40425	423.986695
4	30	0.05	0.5	0.461800	558.294465	0.38600	427.279147	0.39250	419.801386
5	30	0.05	0.5	0.442000	567.283538	0.42275	427.621305	0.41700	408.112767
6	30	0.05	0.5	0.408000	500.270497	0.42900	426.539872	0.42100	423.541938
7	30	0.05	0.5	0.417800	507.376351	0.42775	445.065635	0.40250	452.778627
8	30	0.05	0.5	0.436000	537.740826	0.44575	416.313942	0.40100	413.828499
9	30	0.05	0.5	0.465000	509.467268	0.40725	407.094582	0.40300	420.499870
10	30	0.05	0.5	0.449200	515.873161	0.43750	406.081203	0.42725	432.628838
11	30	0.05	0.5	0.440400	481.981709	0.42225	426.876995	0.44350	422.023508
12	30	0.05	0.5	0.440200	513.649302	0.39550	466.933791	0.40450	402.367495
13	30	0.05	0.5	0.452000	524.120957	0.41025	386.467075	0.44825	421.779352
14	30	0.05	0.5	0.439600	585.430803	0.33600	400.024434	0.39200	400.367605
15	30	0.05	0.5	0.459600	527.168260	0.43300	416.671768	0.36475	405.544588
16	30	0.05	0.5	0.390400	518.083675	0.46350	437.027205	0.42775	439.538708
17	30	0.05	0.5	0.404800	521.805738	0.44650	405.878710	0.39975	425.569230
18	30	0.05	0.5	0.456800	533.412905	0.39675	421.498986	0.39075	433.322787
19	30	0.05	0.5	0.450400	508.476781	0.40575	414.469190	0.38025	422.981605
20	30	0.05	0.5	0.465200	512.299478	0.41400	436.736087	0.44225	394.225445
21	30	0.05	0.5	0.459301	519.953036	0.42750	451.396291	0.41975	428.937664
22	30	0.05	0.5	0.459735	527.340230	0.41325	455.771882	0.43075	435.511015
23	30	0.05	0.5	0.414879	504.771734	0.44650	423.634116	0.39525	434.058075
24	30	0.05	0.5	0.442779	571.846959	0.43200	390.291166	0.50425	419.086142
25	30	0.05	0.5	0.420819	534.368177	0.40425	420.545634	0.44450	420.240933
26	30	0.05	0.5	0.409090	530.374937	0.36400	398.716412	0.42200	425.482516
27	30	0.05	0.5	0.459454	510.527889	0.40200	413.198996	0.42450	444.001254
28	30	0.05	0.5	0.399813	511.282486	0.43325	418.239288	0.46150	409.878531
29	30	0.05	0.5	0.456104	500.008571	0.41675	413.700479	0.39525	401.526729
30	30	0.05	0.5	0.459365	545.109785	0.46225	465.747535	0.39575	407.246094

Fig. 11. Resultados de evaluación para el algoritmo 2.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	30	0.1	0.8	0.446800	858.013511	0.44450	702.217237	0.43825	666.697075
2	30	0.1	0.8	0.518000	910.299599	0.45425	745.428003	0.43875	697.562823
3	30	0.1	0.8	0.477400	893.655762	0.46050	692.885768	0.44075	705.323325
4	30	0.1	0.8	0.461600	836.324895	0.41475	708.978719	0.45525	761.439846
5	30	0.1	0.8	0.496800	905.068049	0.44775	714.991797	0.45575	740.525408
6	30	0.1	0.8	0.446800	893.995976	0.41175	727.621673	0.42175	657.721805
7	30	0.1	0.8	0.495800	932.540812	0.46700	687.189895	0.46825	753.211780
8	30	0.1	0.8	0.445600	871.174259	0.42750	694.929765	0.43525	705.409677
9	30	0.1	0.8	0.459800	902.771655	0.44225	648.648565	0.47900	743.746483
10	30	0.1	0.8	0.499200	938.237036	0.44200	735.953172	0.45625	712.248745
11	30	0.1	0.8	0.453600	890.545171	0.43650	706.145468	0.45425	660.201101
12	30	0.1	0.8	0.452400	885.281419	0.48125	713.024204	0.45100	707.181818
13	30	0.1	0.8	0.446400	913.512746	0.43450	697.822601	0.44100	671.896613
14	30	0.1	0.8	0.481800	937.306668	0.44225	729.778939	0.50050	735.860431
15	30	0.1	0.8	0.456200	902.895929	0.45000	744.083255	0.47475	719.508072
16	30	0.1	0.8	0.534200	976.926695	0.41875	677.013256	0.45425	727.611423
17	30	0.1	0.8	0.430000	843.697507	0.44350	727.817001	0.46525	731.258575
18	30	0.1	0.8	0.476400	887.296230	0.41000	725.198355	0.45600	718.776317
19	30	0.1	0.8	0.490200	883.520224	0.45975	732.054611	0.41950	745.442559
20	30	0.1	0.8	0.499000	888.967490	0.46225	688.109354	0.46475	700.252473
21	30	0.1	0.8	0.443870	947.727372	0.46825	671.132585	0.44700	701.672350
22	30	0.1	0.8	0.508208	927.418506	0.44075	702.717226	0.41900	682.401105
23	30	0.1	0.8	0.453690	862.502885	0.46975	721.601087	0.43625	708.061068
24	30	0.1	0.8	0.486669	937.401862	0.45625	669.572503	0.41225	702.364101
25	30	0.1	0.8	0.497105	897.725845	0.43850	716.946919	0.43525	710.091837
26	30	0.1	0.8	0.446911	891.323419	0.45075	711.726004	0.47525	687.435944
27	30	0.1	0.8	0.480541	913.319235	0.42625	681.714952	0.44125	740.516580
28	30	0.1	0.8	0.443471	935.628674	0.43025	717.999185	0.47400	691.063329
29	30	0.1	0.8	0.440999	959.230643	0.43525	683.956992	0.44200	694.490656
30	30	0.1	0.8	0.456570	871.258897	0.46300	698.156645	0.45800	687.049921

Fig. 12. Resultados de evaluación para el algoritmo 3.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	30	0.05	0.8	0.431000	730.108716	0.42200	553.850572	0.36925	569.877145
2	30	0.05	0.8	0.473000	717.417656	0.42075	559.359884	0.38800	558.639652
3	30	0.05	0.8	0.396400	674.775027	0.42575	540.623414	0.39925	600.817838
4	30	0.05	0.8	0.443400	746.316652	0.41350	535.048297	0.40200	563.932454
5	30	0.05	0.8	0.473800	740.567390	0.43775	576.662333	0.37800	536.304547
6	30	0.05	0.8	0.460400	710.934961	0.43250	562.940493	0.42300	556.521686
7	30	0.05	0.8	0.406400	687.042789	0.43525	585.712265	0.42475	541.191285
8	30	0.05	0.8	0.479400	724.810658	0.39350	601.791468	0.47250	575.721147
9	30	0.05	0.8	0.423600	672.901424	0.42175	545.015302	0.38000	547.109272
10	30	0.05	0.8	0.389400	677.844310	0.39975	541.797923	0.40825	539.626920
11	30	0.05	0.8	0.422200	731.035975	0.37950	585.433001	0.44500	601.112150
12	30	0.05	0.8	0.411400	715.861894	0.41625	567.576033	0.44000	576.833022
13	30	0.05	0.8	0.408400	706.633959	0.45525	570.906235	0.39200	591.842133
14	30	0.05	0.8	0.417200	732.177334	0.44200	587.548929	0.39250	587.382438
15	30	0.05	0.8	0.397600	685.930863	0.42475	571.023547	0.40225	548.751533
16	30	0.05	0.8	0.392600	698.611197	0.43150	555.566740	0.42025	536.297889
17	30	0.05	0.8	0.418000	708.526551	0.39650	573.195073	0.41125	560.409690
18	30	0.05	0.8	0.426800	714.859582	0.43125	596.814421	0.42100	595.869740
19	30	0.05	0.8	0.441400	643.572172	0.42350	573.261681	0.44475	613.912909
20	30	0.05	0.8	0.424400	684.414406	0.34600	549.354403	0.44100	585.010948
21	30	0.05	0.8	0.439897	719.803176	0.40800	567.027650	0.43325	592.580092
22	30	0.05	0.8	0.427389	729.781855	0.42675	572.630503	0.43300	572.331658
23	30	0.05	0.8	0.439065	702.093271	0.44200	577.433750	0.41500	572.571185
24	30	0.05	0.8	0.454200	706.932378	0.41125	578.075848	0.39025	589.686597
25	30	0.05	0.8	0.455612	680.655007	0.39175	597.228100	0.40925	570.556214
26	30	0.05	0.8	0.433020	699.861703	0.37275	547.098674	0.42350	563.090968
27	30	0.05	0.8	0.441552	663.248243	0.36700	543.736161	0.40300	578.905567
28	30	0.05	0.8	0.462608	740.472506	0.39175	574.949743	0.42175	563.022165
29	30	0.05	0.8	0.425959	725.055547	0.38550	541.206772	0.45700	547.770153
30	30	0.05	0.8	0.437341	718.677109	0.43700	623.731419	0.38100	559.685025

Fig. 13. Resultados de evaluación para el algoritmo 4.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	50	0.1	0.5	0.500400	656.546889	0.46350	563.636756	0.43775	569.252334
2	50	0.1	0.5	0.474200	713.363398	0.47100	610.360459	0.47525	594.165393
3	50	0.1	0.5	0.491000	693.944566	0.45675	625.429423	0.47175	600.720723
4	50	0.1	0.5	0.510800	746.909386	0.44800	524.329928	0.42925	554.316578
5	50	0.1	0.5	0.444800	725.941138	0.45500	566.317572	0.42825	540.843289
6	50	0.1	0.5	0.483600	754.845568	0.46300	585.564303	0.42700	610.956287
7	50	0.1	0.5	0.469800	720.794002	0.46300	575.584575	0.48475	634.631205
8	50	0.1	0.5	0.485400	725.171257	0.41550	549.809403	0.45525	576.346577
9	50	0.1	0.5	0.457400	727.619212	0.45575	602.101064	0.43950	600.463532
10	50	0.1	0.5	0.498600	683.059165	0.46650	586.945949	0.48200	580.735435
11	50	0.1	0.5	0.464200	651.146762	0.45050	611.466093	0.48075	576.609198
12	50	0.1	0.5	0.497000	683.193493	0.45150	587.487975	0.46300	597.134299
13	50	0.1	0.5	0.462000	738.934359	0.45850	556.719340	0.43475	588.336218
14	50	0.1	0.5	0.466400	726.374344	0.48475	599.135765	0.46800	602.358985
15	50	0.1	0.5	0.486000	732.160433	0.44525	548.932157	0.46425	562.988555
16	50	0.1	0.5	0.462800	725.915177	0.43850	599.669640	0.44875	566.526374
17	50	0.1	0.5	0.479400	602.421738	0.47075	593.306883	0.41175	627.369793
18	50	0.1	0.5	0.465400	670.304749	0.44275	652.158874	0.45300	580.726699
19	50	0.1	0.5	0.472600	703.543531	0.43375	626.819266	0.45625	607.504235
20	50	0.1	0.5	0.486600	651.080758	0.46875	608.218601	0.46125	568.680671
21	50	0.1	0.5	0.494480	709.945950	0.43350	553.956939	0.42425	641.246135
22	50	0.1	0.5	0.499456	627.263571	0.42975	595.766788	0.46500	577.920397
23	50	0.1	0.5	0.471941	677.390214	0.47100	581.374550	0.44075	523.424194
24	50	0.1	0.5	0.469884	700.200891	0.47300	556.199340	0.42925	592.223302
25	50	0.1	0.5	0.494995	736.586104	0.43900	578.587398	0.42000	564.406914
26	50	0.1	0.5	0.451544	655.420959	0.45575	604.301999	0.46100	545.692191
27	50	0.1	0.5	0.468808	698.261585	0.47000	546.477454	0.46700	554.326314
28	50	0.1	0.5	0.461251	700.936809	0.44150	561.127945	0.46675	568.479648
29	50	0.1	0.5	0.483188	631.930898	0.49100	580.719858	0.46400	536.146885
30	50	0.1	0.5	0.492087	650.439864	0.46300	552.863760	0.44675	569.511927

Fig. 14. Resultados de evaluación para el algoritmo 5.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	50	0.05	0.5	0.445600	563.900180	0.42150	449.678349	0.42650	413.407369
2	50	0.05	0.5	0.433000	550.718925	0.44725	437.757128	0.39325	438.110753
3	50	0.05	0.5	0.446000	558.552459	0.41325	428.274127	0.38250	428.971080
4	50	0.05	0.5	0.422000	543.318389	0.42125	414.120777	0.40775	430.846394
5	50	0.05	0.5	0.422600	544.076611	0.44575	444.562743	0.40000	434.079048
6	50	0.05	0.5	0.450000	529.113385	0.39375	457.026485	0.38900	428.268284
7	50	0.05	0.5	0.448400	551.218005	0.40600	449.144621	0.42200	400.589733
8	50	0.05	0.5	0.402200	510.666205	0.40075	415.822596	0.47850	449.433195
9	50	0.05	0.5	0.501800	513.820256	0.41625	425.009856	0.43650	445.970381
10	50	0.05	0.5	0.459000	491.044147	0.37350	429.318008	0.42775	421.446631
11	50	0.05	0.5	0.468400	536.474302	0.41375	430.123311	0.40425	468.150365
12	50	0.05	0.5	0.438400	515.175575	0.38750	400.709006	0.39775	451.382673
13	50	0.05	0.5	0.454600	557.969639	0.44725	404.867673	0.44100	450.049932
14	50	0.05	0.5	0.429600	534.571680	0.42550	434.987633	0.44325	431.077847
15	50	0.05	0.5	0.410600	539.208971	0.39075	402.532559	0.42200	433.900533
16	50	0.05	0.5	0.474200	532.926241	0.38450	414.729162	0.42000	469.287759
17	50	0.05	0.5	0.468200	563.458520	0.42225	408.531374	0.41050	425.471795
18	50	0.05	0.5	0.417400	543.267797	0.47650	441.342153	0.39475	439.146416
19	50	0.05	0.5	0.478000	554.922237	0.44000	407.922923	0.40525	444.504064
20	50	0.05	0.5	0.413800	543.528397	0.41300	409.118673	0.38050	415.976959
21	50	0.05	0.5	0.452116	546.461051	0.46475	459.459568	0.39675	425.110433
22	50	0.05	0.5	0.435547	536.957241	0.42050	449.134393	0.39925	399.603934
23	50	0.05	0.5	0.443133	528.625267	0.44475	433.673742	0.46650	438.205226
24	50	0.05	0.5	0.490595	550.113880	0.40125	435.154750	0.45775	462.852332
25	50	0.05	0.5	0.436128	536.887032	0.42025	420.868649	0.43875	424.659058
26	50	0.05	0.5	0.457853	552.317459	0.45875	437.841450	0.36875	405.472169
27	50	0.05	0.5	0.415387	548.500835	0.42000	408.848540	0.40075	416.623869
28	50	0.05	0.5	0.427629	546.041475	0.42800	441.143866	0.41650	420.636100
29	50	0.05	0.5	0.442353	534.209097	0.42275	419.181634	0.40475	420.692981
30	50	0.05	0.5	0.458150	536.559004	0.43000	417.547171	0.42075	455.993688

Fig. 15. Resultados de evaluación para el algoritmo 6.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	50	0.1	0.8	0.447400	897.008134	0.44475	732.073209	0.40100	699.999725
2	50	0.1	0.8	0.452200	919.448040	0.42475	717.507839	0.44950	742.001883
3	50	0.1	0.8	0.504200	966.439411	0.38675	682.966986	0.44100	741.561314
4	50	0.1	0.8	0.472200	938.654797	0.44275	749.693381	0.42925	714.310948
5	50	0.1	0.8	0.442400	860.851633	0.46725	716.427820	0.48450	720.338053
6	50	0.1	0.8	0.448800	913.008826	0.47825	739.024610	0.46350	738.347510
7	50	0.1	0.8	0.500400	945.217901	0.48250	770.798395	0.46775	720.949833
8	50	0.1	0.8	0.467600	886.118387	0.41350	703.259781	0.44725	683.535897
9	50	0.1	0.8	0.431800	888.022499	0.45400	737.778157	0.44850	704.794142
10	50	0.1	0.8	0.490600	874.569769	0.44400	714.194458	0.43050	717.541263
11	50	0.1	0.8	0.464000	880.218621	0.43875	731.180700	0.48700	738.663934
12	50	0.1	0.8	0.488800	942.534339	0.39775	711.106776	0.45050	715.668947
13	50	0.1	0.8	0.471600	885.794302	0.42300	722.030290	0.44700	699.476813
14	50	0.1	0.8	0.475800	933.067602	0.45375	722.596556	0.45625	734.836820
15	50	0.1	0.8	0.427000	889.312271	0.45300	737.285066	0.47725	736.139604
16	50	0.1	0.8	0.452200	931.232377	0.46775	734.679617	0.44025	690.945382
17	50	0.1	0.8	0.461600	896.143183	0.48375	727.756237	0.43500	761.421802
18	50	0.1	0.8	0.455200	933.515852	0.45025	718.764319	0.42800	732.396288
19	50	0.1	0.8	0.470200	878.131492	0.41675	748.720980	0.43375	708.670960
20	50	0.1	0.8	0.480200	913.251238	0.44400	720.088859	0.45225	736.867000
21	50	0.1	0.8	0.431220	887.257604	0.46325	766.096021	0.41525	769.009404
22	50	0.1	0.8	0.447460	909.086160	0.41100	706.623660	0.43800	767.835612
23	50	0.1	0.8	0.467039	891.985796	0.44875	719.093755	0.44450	678.897583
24	50	0.1	0.8	0.476593	881.352823	0.43975	670.163620	0.43175	734.268819
25	50	0.1	0.8	0.477766	877.162380	0.38550	695.000709	0.45575	702.520260
26	50	0.1	0.8	0.461191	910.343972	0.46790	720.293144	0.41100	732.227657
27	50	0.1	0.8	0.432803	961.186251	0.44825	723.717309	0.45975	753.990075
28	50	0.1	0.8	0.483000	902.515741	0.43750	733.391433	0.43650	708.867022
29	50	0.1	0.8	0.455557	889.374745	0.48700	737.811632	0.47100	700.570233
30	50	0.1	0.8	0.475850	919.373508	0.45200	744.530736	0.46150	726.967942

Fig. 16. Resultados de evaluación para el algoritmo 7.

			Instancia	Instancia 1		Instancia 2		Instancia 3	
				Best Acc	Elapsed Time	Best Acc	Elapsed Time	Best Acc	Elapsed Time
Gen	#POP	Mut Prob	Cross Prob						
1	50	0.05	0.8	0.426400	713.730044	0.38975	582.437956	0.41550	606.201102
2	50	0.05	0.8	0.410200	724.322896	0.44975	586.969057	0.42225	605.865867
3	50	0.05	0.8	0.446600	716.212012	0.42325	547.426536	0.41950	599.846940
4	50	0.05	0.8	0.423600	709.923130	0.41775	614.341956	0.41825	571.109444
5	50	0.05	0.8	0.469200	757.533391	0.38475	578.261882	0.38475	614.063778
6	50	0.05	0.8	0.460800	745.923811	0.43025	585.572833	0.42850	587.562714
7	50	0.05	0.8	0.448200	750.613795	0.47575	581.122153	0.38700	575.783711
8	50	0.05	0.8	0.439000	748.320037	0.43225	603.418851	0.38950	604.679362
9	50	0.05	0.8	0.434800	744.440473	0.39450	555.616188	0.42850	628.322354
10	50	0.05	0.8	0.423400	693.194394	0.39025	556.442685	0.47375	592.871908
11	50	0.05	0.8	0.366200	691.200945	0.37500	581.172687	0.41225	568.887476
12	50	0.05	0.8	0.475000	753.573826	0.44300	607.783692	0.44825	571.394470
13	50	0.05	0.8	0.455800	745.983276	0.44525	574.585160	0.44675	581.354863
14	50	0.05	0.8	0.465200	760.793800	0.38675	589.127830	0.39800	593.197742
15	50	0.05	0.8	0.467800	736.843965	0.41975	613.767661	0.40300	575.581501
16	50	0.05	0.8	0.416200	761.460937	0.40300	598.538642	0.36025	573.051808
17	50	0.05	0.8	0.467600	712.521600	0.43525	605.044981	0.37700	588.335638
18	50	0.05	0.8	0.425600	696.222787	0.43550	578.066242	0.39275	600.389496
19	50	0.05	0.8	0.428600	721.717731	0.44600	588.811854	0.42550	599.939604
20	50	0.05	0.8	0.396600	735.181774	0.41825	555.581041	0.40850	567.779847
21	50	0.05	0.8	0.486614	725.757999	0.40075	603.568250	0.39375	601.133809
22	50	0.05	0.8	0.479469	743.953284	0.42175	591.346288	0.36175	571.001795
23	50	0.05	0.8	0.475616	755.442616	0.39600	606.641316	0.41925	592.415797
24	50	0.05	0.8	0.490447	735.330098	0.38675	612.133671	0.41500	565.185240
25	50	0.05	0.8	0.448070	736.614889	0.42450	598.633150	0.38625	605.397149
26	50	0.05	0.8	0.386398	749.040902	0.37775	575.963294	0.42150	589.375103
27	50	0.05	0.8	0.447657	712.103164	0.40325	580.767788	0.44300	591.482881
28	50	0.05	0.8	0.438840	737.342500	0.37500	610.925375	0.38300	573.379159
29	50	0.05	0.8	0.458442	749.692038	0.42350	556.328728	0.41725	573.527637
30	50	0.05	0.8	0.413551	756.424928	0.37400	578.833948	0.40000	599.712376

Fig. 17. Resultados de evaluación para el algoritmo 8.

REFERENCES

[1] Cifar10 net. https://keras.io/examples/cifar10_cnn/. Accessed: 2019-12-01.

[2] Clusteruy. https://cluster.uy. Accessed: 2019-12-06.

[3] Deap. https://deap.readthedocs.io/en/master/. Accessed: 2019-11-15.

[4] Keras. https://keras.io/. Accessed: 2019-10-07.

[5] One-way anova, scipy. https://docs.scipy.org/doc/scipy/reference/ generated/scipy.stats.f_oneway.html. Accessed: 2019-12-01.

[6] Scipy. https://www.scipy.org/. Accessed: 2019-12-01.

[7] Weight initialization in neural networks: A journey from the basics to kaiming. https://towardsdatascience.com/weight-initialization-in-neural-networks-a-journey-from-the-basics-to-kaiming-954fb9b47c79. Accessed: 2019-10-06.

[8] M. A. Stephens. Edf statistics for goodness of fit and some comparisons, journal of the american statistical association, 1974. pp. 730-737.