ISSN 1870-4069

# Entrenamiento dinámico de redes convolucionales profundas para clasificación de imágenes

Oscar Frausto-Pérez, Alfonso Rojas-Domínguez,
Manuel Ornelas-Rodríguez, Héctor Puga, Martín Carpio

Instituto Tecnológico de León,
León, Guanajuato, México
m12241437@itleon.edu.mx, alfonso.rojas@gmail.com,
mornelas67@yahoo.com.mx, pugahector@yahoo.com,
juanmartin.carpio@itleon.edu.mx

**Resumen.** En años recientes, el Aprendizaje Profundo se ha posicionado como uno de los paradigmas más exitosos para la solución de problemas de reconocimiento de patrones, tales como clasificación de imágenes. En particular, las CNNs (Convolutional Neural Networks) constituyen uno de los modelos más utilizados para esta tarea. Una CNN típica incorpora cientos de miles de parámetros que deben ser ajustados iterativamente, en lo que se conoce como el entrenamiento de la red, mediante el uso de grandes bancos de imágenes de referencia. Este entrenamiento representa un alto costo de recursos y tiempo computacional incluso cuando se cuenta con GPUs para realizar el procesamiento. En este trabajo se propone un esquema alternativo para el manejo de los datos de entrenamiento para CNNs, que consiste en un muestreo selectivo-adaptativo de los datos. Mediante experimentos con la base de datos CIFAR10 para clasificación de imágenes, se demuestra que este esquema consigue disminuir el tiempo de entrenamiento sin sacrificar significativamente el desempeño de las redes.

**Palabras clave:** redes neuronales convolucionales, clasificación de imágenes, tiempo de entrenamiento.

# Dynamic Training of Deep Convolutional Networks for Image Classification

**Abstract.** In recent years, Deep Learning has positioned itself as one of the most successful paradigms for the solution of pattern recognition problems such as image classification. Particularly, CNNs (Convolutional Neural Networks) are one of the most used models for this task. A typical CNN incorporates hundreds of thousands of parameters that must be adjusted in what is known as the training of the network, through the use of large reference training sets. This training represents a high cost in computational time and resources even with the availability of GPUs to do the processing. In this work, an alternative scheme for management of the training data for CNNs is proposed which consists in the selective-adaptive sampling of the data. Through experiments performed on the CIFAR10 dataset for image classification, it is shown that the proposed scheme succeeds in decreasing the training time without significantly sacrificing the performance of the networks.

**Keywords:** convolutional neural networks, image classification, training time.

## 1. Introducción

Desde 2006, el aprendizaje estructurado profundo, o más comúnmente llamado aprendizaje profundo o aprendizaje jerárquico, se ha convertido en una nueva área de investigación en aprendizaje máquina [1]. Durante los últimos años las técnicas desarrolladas a partir de la investigación de aprendizaje profundo han impactado sobre una amplia gama de trabajos de procesamiento de información y en particular de imágenes, dentro de los ámbitos tradicionales y nuevos, incluidos los aspectos clave de aprendizaje automático e inteligencia artificial [2]. Específicamente, las redes neuronales convolucionales (CNN por sus siglas en inglés), han mostrado un desempeño que se ubica entre los mejores en la actualidad para las tareas de clasificación y detección de objetos en imágenes digitales [1] y también para detección de objetos en video y en tiempo real, donde dos algoritmos muy conocidos debido a su velocidad de procesamiento han destacado, estos son Yolo [3] y SSD [4].

Las CNNs tienen sus orígenes en el trabajo de Yann LeCun [5] y aunque su capacidad estuvo limitada por varios años, la utilización de GPUs para su implementación ha originado un despunte muy importante en el estado del arte a partir del 2011. Otra técnica que se utiliza con frecuencia para mejorar el porcentaje de clasificación de las CNNs es el *aumento de datos* (data augmentation en inglés). En términos de clasificación de imágenes o detección de objetos, la aumentación consiste en aplicar transformaciones a las imágenes de entrenamiento, dichas transformaciones pueden ser: rotación, espejo, recorte, ruido Gaussiano, entre otras.

El entrenamiento de una CNN requiere mucho tiempo y es computacionalmente costoso pues utiliza grandes conjuntos de imágenes de entrenamiento (típicamente se requieren varias decenas de miles de imágenes). Esto dificulta el empleo de otras técnicas que buscan mejorar el desempeño de CNNs o lograr el diseño automático de redes; por ejemplo, los trabajos [6] y [7] proponen llevar a cabo la optimización de la arquitectura de CNNs para las tareas de clasificación de imágenes (ver Fig. 1), mediante el uso de algoritmos evolutivos.

Esto es debido a que inicialmente la cantidad de capas convolucionales, así como la cantidad de mapas de activación por capa (ligados a la cantidad de filtros convolucionales), no se encuentran definidas previamente, sino que deben ajustarse según el problema por resolver, así como otras especificaciones del usuario, sumando al hecho de que para investigadores con poca experiencia es difícil diseñar y proponer nuevas arquitecturas que resulten exitosas.

Aunque dicha estrategia de optimización de la arquitectura de CNN mediante un algoritmo evolutivo ha resultado exitosa, su complejidad computacional es muy elevada y puede ser prohibitiva para muchos investigadores y practicantes que no cuentan con una infraestructura de cómputo suficientemente potente. Por ejemplo, en [7] se menciona que el tiempo requerido para la evolución de una CNN utilizando una variante de algoritmo genético y aplicada al conjunto de datos CIFAR10[1] fue de 35 días de cómputo en una GPU; mientras, en [8] se reporta que el método allí descrito consume

[1] http://www.cs.toronto.edu/kriz/cifar.html

2,750 días de cómputo en una GPU para evolucionar una red para el mismo problema de CIFAR10 (para más datos de sistemas del estado del arte, ver [7]).

Es por ello que en este trabajo se propone una metodología para el uso inteligente de las imágenes que son utilizadas para entrenamiento de las redes con finalidad de reducir el tiempo de entrenamiento facilitando así el uso otras técnicas de mayor costo computacional. Nuestra propuesta está inspirada en algoritmos de optimización metaheurísticos o de estimación de distribución, pero en vez de realizar un muestreo de las soluciones (en este caso arquitecturas de red), se aplica una selección y un muestreo dirigido sobre los datos de entrenamiento. A esta propuesta le hemos llamado Entrenamiento Dinámico y se describe en detalle en la Sección 3.

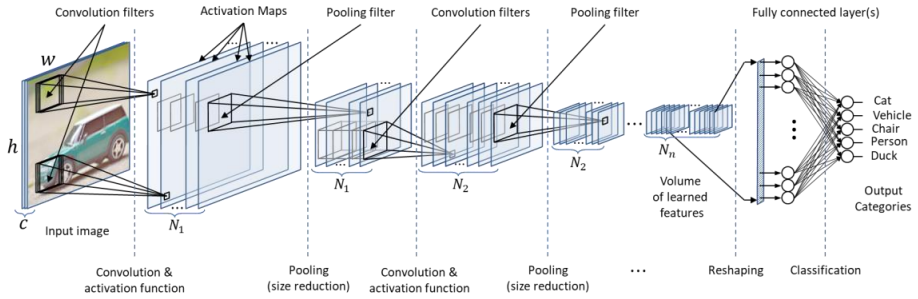



**Fig. 1.** Estructura de una Red Neuronal Convolucional para Clasificación de Imágenes.

## 2. Antecedentes

El gran costo computacional y las grandes cantidades de memoria requeridas por las CNNs es una constante en todas las tareas que esta técnica intenta resolver, ya sea clasificación de imágenes, detección de objetos o segmentación semántica. En esta última tarea, los autores de [9] proponen un esquema de muestreo adaptativo que utiliza mapas de error a-posteriori, generados a lo largo del entrenamiento, para enfocar el muestreo en regiones difíciles, lo cual resulta en un mejor aprendizaje.

En el trabajo [10] se propone un método para acelerar el entrenamiento de una red convolucional usando muestreo selectivo aplicado a la detección de hemorragias de retina sobre imágenes a color. En ese trabajo, los autores incrementan dinámicamente la probabilidad de que un muestreo mal clasificado sea seleccionado en la siguiente iteración de entrenamiento. Esto lo llevan a cabo asignando un peso dinámico a cada pixel de una imagen perteneciente al grupo de entrenamiento negativo (refiriéndose a los peores clasificados) y de esta manera indicando su nivel de *informatividad*. Después de cada época de entrenamiento, el peso de cada pixel de entrenamiento negativo es actualizado. Debido al tipo de imágenes que utiliza y a la tarea que pretende resolver, se observa que su propuesta no trata de manera global la imagen, sino que es más específica, afectando solo a ciertos puntos o pixeles en una imagen dada.

En cambio, la tarea seleccionada en el presente trabajo, sobre la cual se aplicará el método propuesto es la de clasificación de imágenes sobre el conjunto de datos CIFAR10. En esta tarea se busca clasificar de manera correcta cada imagen en un conjunto compuesto por 50,000 imágenes de entrenamiento y 10,000 imágenes de

prueba (también llamada validación) distribuidas en 10 categorías. La Fig. 2 presenta imágenes pertenecientes al repositorio CIFAR10. La elección de este conjunto de datos se realizó bajo criterios como que se trata de un conjunto de imágenes relativamente pequeño y bien conocido sobre el que se han reportado numerosos resultados exitosos en la literatura. En otras palabras, seleccionamos CIFAR10 porque se trata de un problema de diagnóstico, no con el objetivo de superar el desempeño de redes profundas existentes, sino con el de mostrar la efectividad de nuestra propuesta en lo que corresponde a lograr un entrenamiento más eficiente de dichas redes.


**Fig. 2.** Imágenes pertenecientes al repositorio CIFAR 10, cada columna muestra una selección de imágenes correspondientes a cada etiqueta.

## 3. Propuesta

En esta sección se describe cada uno de los pasos de nuestra propuesta. Ésta funciona bajo la suposición de que utilizando un muestreo selectivo-adaptativo de imágenes en la etapa de entrenamiento se puede disminuir el tiempo del mismo sin disminuir de manera significativa el desempeño de la red. Los parámetros utilizados por nuestro método son los valores de **selección** ($\alpha$), el **porcentaje de remuestreo** ($\beta$) y el **umbral de rendimiento** ($\delta$).

1. El primer paso es crear un subconjunto $A$ del conjunto de imágenes de entrenamiento original, con base en un valor de **selección** $\alpha$. Una consecuencia inmediata de esta selección es la creación del subconjunto $B$, formado por las imágenes que no se hayan seleccionado. En otras palabras, se realiza una partición del conjunto de imágenes original. Cabe mencionar que el desempeño de la red siempre se evaluará con base en un conjunto de validación que es independiente de cualquier conjunto

utilizado en el entrenamiento. Es decir, este último conjunto nunca se utiliza para el ajuste de los pesos de la red. La Fig. 3 muestra de manera gráfica este paso.

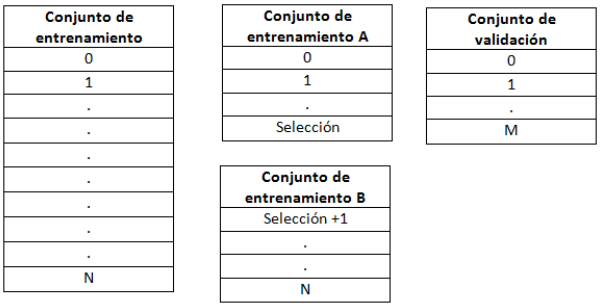
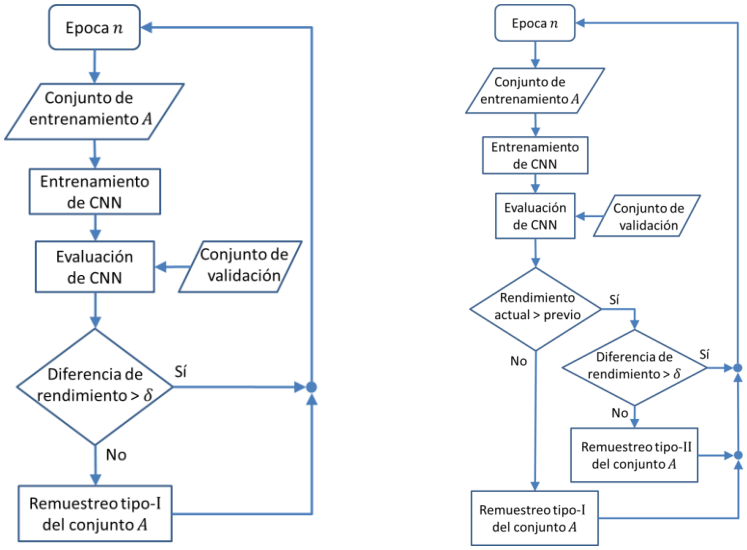



**Fig. 3.** Partición del conjunto de entrenamiento. Nótese que existe siempre un conjunto de imágenes independiente, usado para validación.




a) b)

**Fig. 4.** Diagrama de flujo general del esquema de entrenamiento propuesto.

2. El siguiente paso se realiza dentro del proceso iterativo de entrenamiento de las redes. Cada iteración o 'época' se denota como $n$ y una red se entrena durante un conjunto de iteraciones $n \in [1, 2, \ldots, N]$. Por cada época se entrena la CNN con el conjunto $A$, para después ser evaluada con el conjunto de validación y encontrar un valor de desempeño de la red (porcentaje correcto en la clasificación de imágenes).
3. El tercer paso es calcular la diferencia entre dos valores de desempeño de iteraciones consecutivas para guiar la decisión de remuestrear o no el conjunto de

entrenamiento. En nuestro diseño base (Fig. 4-a) si la mejora del desempeño es menor al umbral $\delta$ el conjunto $A$ es remuestreado según la Ec. (1).

En una variante (diagrama de flujo en la Fig. 4-b), primero se determina si el desempeño actual es mayor que el anterior. De no ser así, el conjunto $A$ se remuestrea de acuerdo a la Ec. (1), llamado remuestreo tipo-I, usando el conjunto actual. Si el desempeño actual es superior al anterior, se determina si la diferencia supera al umbral de desempeño $\delta$, de no ser así, el conjunto $A$ se remuestrea según la Ec. (2), llamado remuestreo tipo-II, basado en el mejor conjunto de entrenamiento $A^*$. El remuestreo tipo-I del conjunto $A$ consiste en reemplazar un porcentaje $\beta$ de sus elementos (en este caso imágenes), seleccionados de manera aleatoria, con elementos del conjunto $B$, también seleccionados de manera aleatoria. Esto puede definirse como la unión de $\alpha(1-\beta) = \alpha - \alpha\beta$ elementos seleccionados de $A$ bajo distribución uniforme, con $\alpha\beta$ elementos seleccionados del conjunto $B$. El resultado de esta operación será el nuevo conjunto $A$. La Ec. (1) describe este proceso:

$$A \leftarrow \{A(\mathcal{U}(\alpha - \alpha\beta))\} \cup \{B(\mathcal{U}(\alpha\beta))\}. \tag{1}$$

El remuestreo tipo-II es muy similar al remuestreo tipo-I, excepto que en vez de seleccionar elementos del conjunto $A$ actual, se seleccionan del mejor conjunto de entrenamiento $A^*$, es decir, del que ha resultado en el mejor desempeño hasta la iteración actual. Lógicamente, con este método el mejor conjunto coincide con el conjunto de entrenamiento de la iteración anterior. Este proceso se describe con la Ec. (2). En este caso el valor del porcentaje $\beta$ se rige por la Ec. (3), donde $\gamma$ sirve como un factor multiplicativo y su valor se establece cercano a 1.0, mientras que $n$ representa la época en la que se encuentra el entrenamiento y $\beta_0$ es el valor inicial de $\beta$. De esta manera se logra que el valor de $\beta$ decrezca de manera potencial:

$$A \leftarrow \{A^*(\mathcal{U}(\alpha - \alpha\beta))\} \cup \{B(\mathcal{U}(\alpha\beta))\}, \tag{2}$$

$$\beta \leftarrow (\gamma^n)\,\beta_0. \tag{3}$$

## 4. Diseño de experimentos

La implementación de nuestra propuesta se realizó utilizando el lenguaje Python junto con la librería Pytorch [11] para la creación de la CNN. Para probar el efecto de los diferentes parámetros ($\alpha, \beta$ y $\delta$) se realizó una serie de experimentos en los que se varían los valores dados a dichos parámetros, utilizando el diseño base en la Fig. 4-a.

Se experimentó con una selección de $\alpha \in \{15000, 20000, 25000\}$ imágenes para formar el conjunto $A$ y con valores de $\beta \in \{0.1, 0.2, \ldots, 0.7\}$ para la fracción o porcentaje de reemplazo y se estableció el valor de $\gamma$ en 0.97. El valor del umbral de rendimiento se mantuvo fijo en $\delta = 0.01$ debido a que esta tasa de aprendizaje es un valor típico observado durante el entrenamiento regular de la red. De la combinación de estos valores de parámetros, se generan 21 experimentos que se ejecutaron con un GPU GeForce GTX 1070 de Nvidia, y se reportan los tiempos en la sección de resultados de este dispositivo.

El pseudocódigo para el funcionamiento general del sistema se muestra en la Fig. 5, aquí se muestran las llamadas a los métodos TrainEvalCNN donde la red realiza su entrenamiento a partir del currentSetA y su evaluación con el conjunto de validación

validationSet y al método Resample donde se actualiza el conjunto de entrenamiento currentSetA. Los pseudocódigos de los métodos ya descriptos se muestran en la Fig. 6 y Fig. 7 respectivamente.

```
# Main program
For epoch in epochs do:
    currentAcc ← TrainEvalCNN(currentSetA)
    If abs(currentAcc − priorAcc) < δ do:
        currentSetA ← Resample(currentSetA, β)
    end If
end For
priorAcc ← currentAcc
```

**Fig. 5.** Pseudocódigo del programa principal.

```
# TrainEvalCNN
For [images, labels] in currentSetA do:
    predictions ← model(images)
    loss ← lossFunction(predictions, labels)
    UpdateCNN(loss, optimizer)
end For
currentAcc ← EvalCNN(validationSet)
```

**Fig. 6.** Pseudocódigo para el entrenamiento y validación de la CNN.

```
# Resample
currentSetB ← setDifference (setTotal, currentSetA)
SetA' ← randomPermutation(currentSetA)
SetB' ← randomPermutation(currentSetB)
currentSetA ← concatenate( SetA' [0: α(1 − β)]), SetB' [0: αβ] )
```

**Fig. 7.** Pseudocódigo de remuestro del conjunto A.

La arquitectura CNN utilizada en nuestro estudio está basada en la descrita en [12], donde también se utiliza el conjunto de imágenes CIFAR10 para experimentación y se reportan buenos resultados. Dicha arquitectura es la expuesta en la Tabla 1. Los hiper-parámetros son: Tamaño de los lotes de entrenamiento: 512, tasa de aprendizaje inicial igual a 0.001, el optimizador elegido es Adam debido a que se compara de manera favorable con otros optimizadores [13]. La métrica utilizada para la tarea de clasificación de imágenes (también empleada por trabajos como [12]) es la siguiente:

$$Accuracy = \frac{\#\ correct\ predictions}{\#\ total\ predictions}. \quad (4)$$

**Tabla 1.** Arquitectura CNN.

| Cantidad | Capa | Tamaño | Canales de entrada | Canales de salida | Función de activación |
|---|---|---|---|---|---|
| 1 | Imagen | 32 x 32 x 3 | - | - | - |

| Cantidad | Capa | Tamaño | Canales de entrada | Canales de salida | Función de activación |
|---|---|---|---|---|---|
| 1 | 3×3 conv. | 32 x 32 x 32 | 3 | 32 | ReLu |
| 2 | 3×3 conv. | 32 x 32 x 32 | 32 | 32 | ReLu |
| 1 | Max. Pooling | 16 x 16 x32 | 32 | 32 | - |
| 1 | 3×3 conv. | 16 x 16 x 64 | 32 | 64 | ReLu |
| 3 | 3×3 conv. | 16 x 16 x 64 | 64 | 64 | ReLu |
| 1 | Max. Pooling | 8 x 8 x 64 | 64 | 64 | - |
| 1 | 3×3 conv. | 4 x 4 x 128 | 64 | 128 | ReLu |
| 3 | 3×3 conv. | 4 x 4 x 128 | 128 | 128 | ReLu |
| 1 | Max Pooling | 2 x 2 x 128 | 128 | 128 | - |
| 3 | 3×3 conv. | 2 x 2 x128 | 128 | 128 | ReLu |
| 1 | Avg. Pooling | 1 x 1 x 128 | 128 | 128 | - |
| 1 | Fully Connected | 128 | 128 | 10 | Softmax |

## 5. Resultados

Dado el conjunto de parámetros descritos en la sección anterior, se llevó a cabo múltiples veces el entrenamiento de redes profundas CNN con la arquitectura reportada en la Tabla 1 y tomando una combinación de parámetros distinta cada vez, realizándose un total de 21 experimentos. Para cada uno de estos experimentos se registró el máximo porcentaje de clasificación obtenido por la red correspondiente. También se registró el tiempo que le llevó a la red ejecutar el entrenamiento completo, considerando un máximo de 200 épocas. Los resultados se reportan en la Tabla 2 y respectivamente. Puesto que todas las redes alcanzan el 100% de clasificación sobre su respectivo conjunto de entrenamiento, y puesto que dicho conjunto es dinámico, sólo se reporta el porcentaje de clasificación sobre el conjunto de validación, ya que este es realmente el indicativo del rendimiento de las redes.

**Tabla 2.** Clasificación (mostrada en %) obtenida con distintas combinaciones de parámetros.

| | **Fracción de reemplazo**, $\beta$ | | | | | | |
|---|---|---|---|---|---|---|---|
| **Selección**, $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 25,000 | 82.1 | 82.5 | 83.8 | 83.4 | 83.3 | 82.6 | **84.4** |
| 20,000 | 80.9 | 81.5 | 82.5 | 83.1 | 82.6 | 82.3 | 81.6 |
| 15,000 | 79.4 | 79.3 | 80.0 | 80.7 | 80.1 | 82.1 | 82.2 |

Como puede observarse en la Tabla 2, el porcentaje de clasificación más alto se logra utilizando la mayor cantidad de imágenes en la selección ($\alpha = 25{,}000$) y con la mayor fracción de reemplazo ($\beta = 0.7$). Este resultado (resaltado usando un tipo de fuente negrita) indica que el desempeño de la red se ve favorecido al disponer de una mayor cantidad de imágenes para su entrenamiento, pero también al incorporar la mayor diversidad en ese conjunto (lograda por medio del remuestreo).

Por otro lado, analizando la Tabla 3, puede observarse que entre mayor es la cantidad de imágenes seleccionadas, mayor es el tiempo requerido para el entrenamiento. Esta observación tiene sentido y es de hecho uno de los supuestos principales por los que se diseñó la presente propuesta. Lo que resulta más interesante es que el parámetro de la fracción de reemplazo prácticamente no tiene un efecto sobre el tiempo de

entrenamiento, puesto que se puede notar cómo los tiempos se mantienen aproximadamente constantes para todos los valores de la Tabla 3 en un mismo renglón.

**Tabla 3.** Tiempo de entrenamiento requerido con las distintas combinaciones de parámetros.

| **Selección**, | **Fracción de reemplazo**, $\beta$ | | | | | | |
|---|---|---|---|---|---|---|---|
| $\alpha$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 25,000 | 32m:28s | 32m:14s | 33m:14s | 36m:44s | 32m:25s | 32m:44s | 32m:38s |
| 20,000 | 27m:14s | 27m:15s | 27m:14s | 27m:15s | 27m:19s | 27m:20s | 27m:21s |
| 15,000 | 21m:22s | 21m:21s | 21m:26s | 21m:30s | 21m:24s | 21m:25s | 21m:32s |

Una vez que la combinación de parámetros que genera el mejor rendimiento ha sido determinada, es necesario comparar esos resultados contra las alternativas por defecto, es decir, contra redes entrenadas sin el esquema de entrenamiento dinámico. En las Figs. 8-10 se presentan las gráficas de clasificación como función de las épocas de entrenamiento. La Fig. 8 muestra las gráficas de entrenamiento y validación de una red entrenada con la totalidad de las imágenes disponibles. La Fig. 9 contiene las gráficas correspondientes a una red entrenada con una selección de $\alpha = 25{,}000$ sin utilizar entrenamiento dinámico. La Fig. 10 muestra las gráficas de una red entrenada usando $\alpha = 25{,}000$ y $\beta = 0.7$, es decir, entrenada bajo la mejor configuración de entrenamiento dinámico. Los resultados correspondientes se reportan en la Tabla 4.

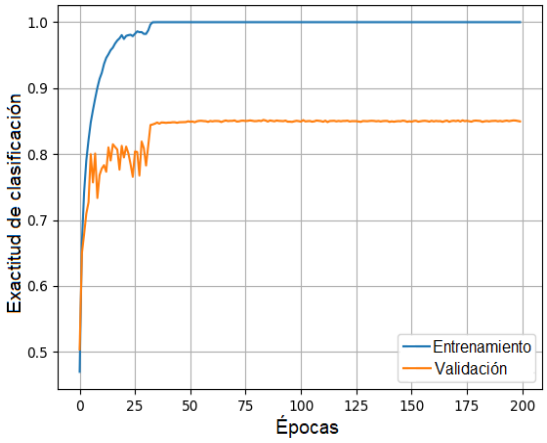



**Fig. 8.** Resultado de clasificación obtenido utilizando el conjunto completo (50,000 imágenes) de entrenamiento sin entrenamiento dinámico.

La principal observación sobre los resultados en la Tabla 4 es respecto de la exactitud de clasificación. Para la red entrenada con todo el conjunto de entrenamiento la exactitud (de validación) alcanzada es 85.17% y toma aprox. 50 min. completar el entrenamiento. Para la red que dispone solo de la mitad del conjunto de entrenamiento, el desempeño de clasificación es de 81.17% y se completa en 27 min. con 40 s. En otras palabras, al reducir el conjunto de entrenamiento en 50%, la exactitud de clasificación cae 4% pero se ahorra 45% de tiempo de cómputo (compárese Fig. 8 vs Fig. 9).

**Tabla 4.** Resumen de resultados comparativos, entrenamiento regular vs dinámico.

| Gráfica | Imágenes de entrenamiento | Entrenamiento dinámico | Exactitud de clasificación | Tiempo consumido |
|---|---|---|---|---|
| Fig. 8 | 100% | No | 85.17 % | 50m:25s |
| Fig. 9 | 50% | No | 81.17 % | 27m:40s |
| Fig. 10 | 50% | SÍ | 84.55 % | 33m:14s |

Ahora bien, al utilizar la mitad del conjunto de entrenamiento, pero introduciendo el entrenamiento dinámico, observamos que la red alcanza 84.55% de clasificación y se completa el entrenamiento en poco más de 33 min. Es decir, mediante el entrenamiento dinámico se puede ahorrar 34% de tiempo de cómputo (17 min. aproximadamente) y sólo se sacrifica un 0.62% en el desempeño de la red. Comparando la Fig. 8 vs la Fig. 10 puede observarse que las gráficas son muy parecidas entre sí.

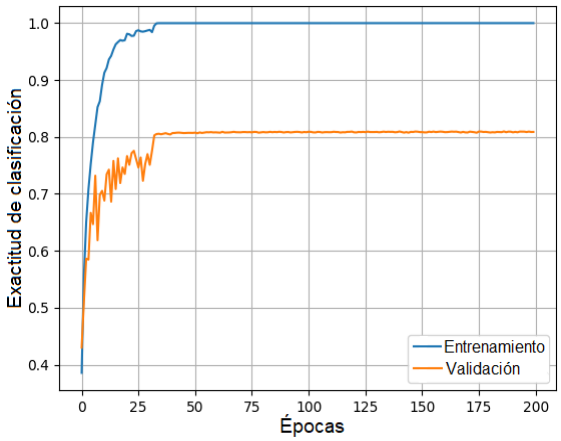



**Fig. 9.** Resultado de clasificación usando 25,000 imágenes, sin entrenamiento dinámico.

Finalmente, se repite el último experimento pero con la variante del diseño presentada en la Fig. 4-b y un número de épocas igual a 80 (puesto que en las Fig. 8 y Fig. 10 se observa que la red alcanza su potencial máximo en un número pequeño de épocas). El resultado se muestra en la Fig. 11, donde además el desempeño se grafica con elitismo; esto no afecta el resultado, sólo tiene el efecto estético de eliminar oscilaciones.

Cuantitativamente, la variante de nuestra propuesta alcanza un porcentaje correcto de clasificación en validación del 84.85%, es decir, sólo 0.76% abajo del desempeño de la red entrenada con el conjunto de imágenes completo. Esta variante de entrenamiento dinámico requiere sólo un 45.8% del tiempo que toma entrenar la red con el conjunto de entrenamiento completo. En otras palabras, prácticamente se iguala el mejor desempeño de la red y se consigue un ahorro de más de la mitad del tiempo requerido para su entrenamiento. Para mostrar la robustez del método, la técnica se ejecutó por un total de 20 repeticiones, obteniendo un promedio de los valores máximos de clasificación de cada repetición de 85.20%. Fig. 12 presenta el promedio de cada experimento por época, con sus desviaciones estándar correspondientes.

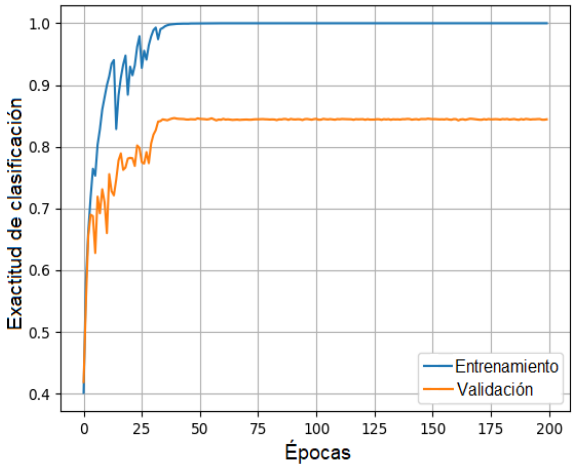



**Fig. 10.** Resultado de clasificación usando 25,000 imágenes y con entrenamiento dinámico.

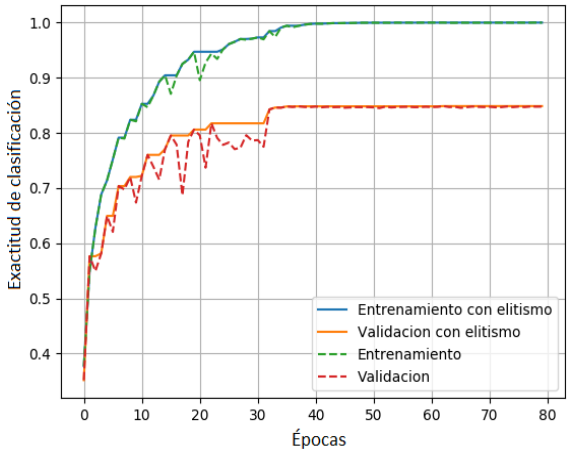



**Fig. 11.** Resultado de clasificación con 25,000 imágenes y variante de entrenamiento dinámico. Se grafican también las curvas de desempeño con elitismo.

Finalmente, para comprobar que el Entrenamiento Dinámico complementa otras técnicas y mantiene su relevancia en los resultados de clasificación, se realizó un experimento (con 20 réplicas) donde se aplica conjuntamente el entrenamiento dinámico y la técnica de aumento de datos. Esto se compara contra utilizar sólo el aumento de datos en un entrenamiento típico. Los resultados se reportan en la Tabla 5.

Como puede observarse, mediante el entrenamiento dinámico se logra ahorrar 46% de tiempo de cómputo y sólo se sacrifica 1.62% el desempeño de la red, comparado contra el entrenamiento tradicional. La Fig. 13 presenta el promedio de la exactitud alcanzada por época, como promedio sobre las 20 réplicas realizadas, así como su desviación estándar.

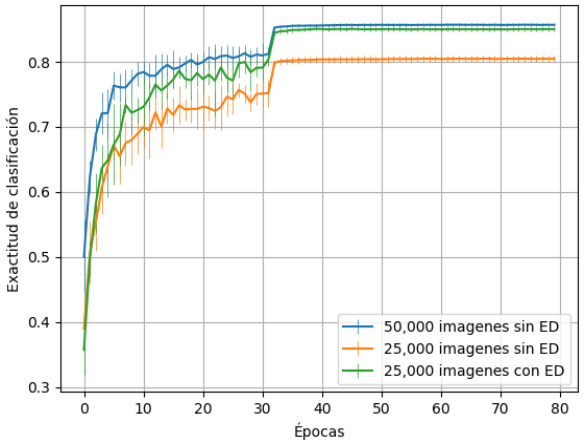



**Fig. 12**. Resultados de clasificación (validación). Se compara el entrenamiento típico vs Entrenamiento Dinámico (ED).

**Tabla 5.** Resultados comparativos, entrenamiento dinámico y aumento de datos.

| Imágenes de entrenamiento | Aumento de datos | Entrenamiento dinámico | Exactitud de clasificación | Tiempo consumido |
|---|---|---|---|---|
| 100% | Sí | No | 90.65 % | 25m:36s |
| 50% | Sí | No | 86.95 % | 13m:40s |
| 50% | Sí | Sí | 89.03 % | 13m:50s |

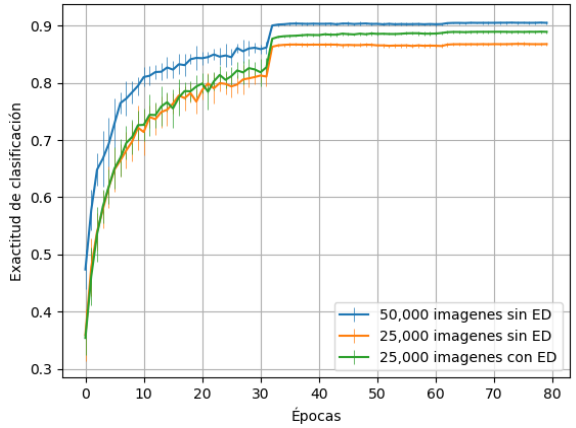



**Fig. 13.** Resultados de clasificación (validación) con aumento de datos. Se compara el entrenamiento típico vs Entrenamiento Dinámico (ED).

## 6. Conclusiones y trabajo a futuro

Se observó que mediante nuestra propuesta de entrenamiento dinámico, la cual consiste en un muestreo selectivo-adaptativo de los datos de entrenamiento, se puede disminuir en más de la mitad el tiempo de cómputo necesario para entrenar una CNN

aplicada a la tarea de clasificación de imágenes. Los resultados hasta el momento demuestran que este ahorro en el tiempo de entrenamiento se logra sin afectar negativamente el desempeño de la red. Además, el Entrenamiento Dinámico puede usarse junto con otras técnicas para mejora de desempeño, como Aumento de Datos, con ganancias de casi la mitad de tiempo de entrenamiento ahorrado.

Aunque aún se requiere más evidencia y argumentación, consideramos que la presente propuesta se podría extender y aplicarse en otros problemas de clasificación con diferentes conjuntos de datos, e incluso en otras tareas como detección de objetos o segmentación semántica. Es por ello que como trabajo futuro se contempla aplicar la metodología desarrollada en la tarea de detección de objetos utilizando el modelo SSD, con el objetivo de acelerar el entrenamiento de la red y que sea factible (en términos de costo computacional) emplear una técnica evolutiva para optimizar la arquitectura y los híper-parámetros de la misma.

**Agradecimientos.** Este trabajo se llevó a cabo gracias al apoyo del Consejo Nacional de Ciencia y Tecnología (CONACYT) de México, a través de los apoyos 703570 (O. Frausto) y CÁTEDRAS-2598 (A. Rojas). Parte de la experimentación se llevó a cabo en equipos adquiridos mediante el proyecto: "Evolución de Diseños Combinatorios para la Optimización de Problemas de Satisfacción de Restricciones", aprobado por TecNM en 2019, oficio M00.1/0379/2019.

## Referencias

1. Deng, L., Dong, Y.: Deep Learning : methods and applications. pp. 197–387 (2014)
2. Arel, I., Rose, D.C., Karnowski, T.P.: Deep machine learning - a new frontier in artificial intelligence research. IEEE Computational Intelligence Magazine 5(4), 13–18 (2010)
3. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-tiime object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Las Vegas (2016)
4. Liu, W. (et al.): SSD: Single Shot MultiBox Detector. In: European conference on computer vision, Amsterdam (2016)
5. LeCun, Y.B.L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to document recognition. In : Proceedings of the IEEE 86 (11), 2278–2324 (1998)
6. Garcia, A., Lara-Cabrera, R.: EvoDeep: a new Evolutionary approach for automatic Deep Neural Networks. Journal of Parallel and Distributed Computing 117, pp. 180–191 (2018)
7. Sun, Y., Xue, B., Zhang, M.: Automatically Designing CNN Architectures Using Genetic Algorithm for Image Classification. arXiv:1808.03818, 12 (2018)
8. Real, E. (et al.): Large-scale evolution of image classifiers. In: Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia (2017)
9. Berger, L., Hyde, E., Cardoso, M.J., Ourselin, S.: An adaptive sampling scheme to efficiently train fully convolutional networks for semantic segmentation. In: Annual Conference on Medical Image Understanding and Analysis, Southampton (2018)
10. van Grinsven, M.J., van Ginneken, B., Hoyng, C.B.: Fast convolutional neural network training using. In: IEEE transactions on medical imaging 35 (5), 1273–1284 (2016)
11. Paszke, A., Gross, S., Chintala, S.: Automatic differentiation in PyTorch (2017)

12. Truong, T.-D., Nguyen, V.-T., Tran, M.-T.: Lightweight Deep Convolutional Network for Tiny Object Recognition. In: International Conference on Pattern Recognition Applications and Methods, Madeira (2018)
13. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)