ISSN 1870-4069

Preprocesamiento de bases de datos de imágenes para mejorar el rendimiento de redes neuronales convolucionales

Fidel López Saca, Andrés Ferreyra Ramírez, Carlos Avilés Cruz, Juan Villegas Cortez, Arturo Zúñiga López, Eduardo Rodrigez Martinez

Universidad Autónoma Metropolitana, Azcapotzalco, Ciudad de México, México

fidelosmcc@gmail.com, {fra,caviles,juanvc,azl,erm}@azc.uam.mx

Resumen. En los últimos años las redes neuronales convolucionales han sido muy populares en el procesamiento de datos a gran escala y muchos trabajos han demostrado que son herramientas muy prometedoras en muchos campos, en especial, en la clasificación de imágenes. Teóricamente, las características de las redes convolucionales pueden mejorar cada vez más con el aumento de las capas de la red; sin embargo, más capas pueden aumentar drásticamente el costo computacional. Además de las características de la red, el tamaño y forma en que se construyen los conjuntos de entrenamiento y prueba, también es un aspecto importante a considerar para mejorar el rendimiento de la red. En este trabajo, proponemos un planteamiento para mejorar el rendimiento de una red neuronal convolucional mediante la división y formación apropiada de los conjuntos de entrenamiento y prueba.

Palabras clave: Redes neuronales convolucionales, tensorflow, reconocimiento de imágenes, procesamiento digital de imágenes, aprendizaje profundo.

Image Data Set Preprocessing for Improving the Performance of Convolutional Neural Networks

Abstract. Recently, convolutional neural networks have been risen to popularity in tasks such as big data preprocessing and computer vision. Many works have shown that they are a promesing tool in several applications such as digital image classification. Theoretically, a convolutional neural network performance can be increased as the number of layers in its architecture increases, however, more layers can also increase its computational cost. Besides, the network topology, the size and making of the training and testing sets also have a big impact in the network performance. In this work we proposed a strategy to improve the performance of a convolutional neural network based on the appropiate division of the training and testing sets.

Keywords: Convolutional neural networks, tensorflow, image recognition, digital image processing, deep learning.

1. Introducción

Separar los conjuntos de datos en conjuntos de entrenamiento y prueba forma parte importante para el entrenamiento y la evaluación de los modelos de redes neuronales convolucionales (RNC) [12, 19]. Normalmente al dividir el conjunto de datos, la mayoría de los datos se utiliza para formar el conjunto de entrenamiento y una parte menor se emplea para el conjunto de prueba. Los datos son muestreados de forma aleatoria para asegurar que los conjuntos sean representativos de todo el conjunto de datos y para eliminar el sesgo de selección, lo que puede minimizar los efectos de las diferencias entre los datos y comprender mejor las características de la RNC.

Tradicionalmente se selecciona el 70 % de los datos de origen para formar el conjunto de entrenamiento y el 30 % para el conjunto de prueba [3, 8]. Sin embargo, aún está abierta la pregunta: ¿cuál es la relación ideal para mejorar el rendimiento de una RNC? Esta relación genera un conflicto, con conjuntos de entrenamiento inmensamente grandes se mejora la capacidad de aprendizaje de la RNC, y con conjuntos de prueba grandes se generan intervalos de confianza más precisos.

Además, cuando el número de ejemplos de entrenamiento es infinitamente grande e imparcial, los parámetros de la red convergen a uno de los mínimos locales de la función de pérdida empírica a ser minimizada, y cuando este número es finito, la función de pérdida real es diferente a la función de pérdida empírica. En consecuencia, ya que los ejemplos de entrenamiento son parciales, los parámetros de la red convergen a una solución sesgada. Esto se conoce como *sobre-ajuste* o *sobre-entrenamiento* porque los valores de los parámetros se ajustan demasiado bien a la especialidad de los ejemplos de entrenamiento sesgados y nos son óptimos en el sentido de minimizar el error de generalización dado por la función de pérdida [1].

Elegir una relación apropiada para generar los conjuntos de entrenamiento y prueba puede no ser suficiente para obtener mejores rendimientos en RNC. Muchas de las bases de datos que se utilizan para aplicaciones de clasificación de imágenes contienen categorías con un número de imágenes desigual, por lo que es importante explorar la conveniencia de estandarizar el número de ejemplos por clase (a la clase con el menor número de ejemplos) o utilizar el conjunto completo para generar los conjuntos de entrenamiento y prueba.

En las aplicaciones de clasificación de imágenes con RNC, se utilizan bases de datos con miles de millones de ejemplos, por lo que, la carga de los datos a la red se convierte en un problema a considerar. Para cargar cada imagen a la red es conveniente generar archivos que contengan la información de las imágenes. Archivos en donde cada imagen es etiquetada automáticamente en función del nombre de la clase a la que pertenece, lo que permite almacenar datos de imágenes de gran tamaño, incluidos datos que no caben en memoria, y leer lotes de imágenes de manera eficiente durante el entrenamiento de la red.

En este trabajo utilizamos diferentes bases de datos de referencia para entrenar tres RNC muy conocidas en la literatura, AlexNet [7], GoogleNet [17] y ResNet [4], para probar si estandarizar una base de datos realmente incrementa el porcentaje de rendimiento con las redes neuronales convolucionales, como en otros problemas de clasificación.

Desarrollamos y aplicamos un toolbox para generar los conjuntos de datos de entrenamiento y prueba, los cuales son representados como archivos que contienen la información de las imágenes.

El resto de este documento está organizado de la siguiente manera. En la sección 2 se revisan las redes convolucionales utilizadas. La sección 3 introduce los conjuntos de datos empleados para los experimentos. La sección 4 introduce el toolbox utilizado para generar los conjuntos de entrenamiento y prueba. La sección 5 muestra los experimentos y resultados. Finalmente, la sección 6 discute las conclusiones y el trabajo futuro.

2. Redes neuronales convolucionales

A continuación se proporciona una breve descripción de las redes que utilizamos en nuestra propuesta:

- **AlexNet**: tiene una profundidad de 8 capas, 5 capas para extracción de características y 3 capas totalmente conectadas. Entre las capas utiliza la función ReLU, la cual reduce el tiempo de aprendizaje, y presenta diferentes variaciones [9]. Esta red implementa la operación de convolución entre la imagen de entrada y un filtro de 11×11 en la primera capa, con el objetivo de extraer diferentes características. Las capas de convolución tienen entre 96 y 384 filtros. La red incluyó la normalización local, pooling para extraer los valores más representativos, y softmax para realizar la clasificación de $1,000$ clases. Para evitar el sobre entrenamiento, AlexNet implementa el Dropout [14] que principalmente deshabilita un nodo temporalmente así como sus entradas y salidas. La red alcanza la tasa de error de conjunto de pruebas de top-5 de 15,3 % con el conjunto de datos ImageNet [15].
- **GoogleNet**: ganadora del concurso ILSVRC2014 en su etapa de clasificación, dejando en segundo lugar a VGGNet [13]. Propone una nueva arquitectura con 22 capas de profundidad, en donde la principal contribución es un módulo llamado Inception que aproxima una CNN dispersa, cuyo resultado es el uso de aproximadamente 12 veces menos parámetros que AlexNet. Los bloques de convolución contienen filtros de tamaño 1×1, 3×3 y 5×5. En conjunto, GoogleNet contiene aproximadamente 100 capas y alcanza una taza de error sobre el conjunto de pruebas de top-5 de 6,67 % usando la base de datos ImageNet.
- **ResNet**: es una red de 152 capas, tiene menos complejidad que GoogleNet pero con mejor precisión con un error de top-5 de 3.57 % en el conjunto de pruebas de ImageNet, tiene diseños con 50 y 101 capas pero la que dio mejor resultado fue la de 152. También tiene bloques internos con filtros entre 64 y 512.

Las redes neuronales convolucionales utilizan variantes del gradiente descendente para el aprendizaje; como son el gradiente descendente estocástico [11], gradiente descendente estocástico con momentos [16], estimación adaptiva del momento (ADAM, adaptive moment estimation) [6] , entre otros [11].

3. Conjuntos de datos

En este trabajo utilizamos 3 bases de datos diferentes, que tienen como características principales: categorías con un número de imágenes desigual e imágenes con tamaños diferentes. Estas se describen brevemente a continuación:

- **Oliva y Torralba** [10] : Este conjunto de datos consta de 2,688 imágenes a color, pertenecientes a 8 categorías o clases; las imágenes fueron obtenidas de diferentes fuentes: bases de datos comerciales, sitios web y cámaras digitales.
- **ImageNetDogs** [5] : Este conjunto de datos consta de 20,580 imágenes a color, pertenecientes a 120 clases o razas de perros de todo el mundo; las imágenes fueron obtenidas de la base de datos ImageNet [15]. El tamaño por cada imagen es variable.
- **Caltech 256** [2] : Este conjunto de datos consta de 30,607 imágenes a color pertenecientes a 257 categorías, el número mínimo de imágenes en cualquier categoría es de 80.

En la Tabla 1 se muestran las características de las bases de datos.

Tabla 1. Bases de datos utilizadas y sus características.

Conjunto de datos	**Clases**	**Dimensiones**			**No. imágenes**		
		Ancho	**Alto**	**Prof**	**Total**	**Min x clase**	**Max x clase**
Oliva & Torralba	8	256	256	3	2,688	260	410
ImageNetDogs	120	$200-500$	$150-500$	3	20,580	148	252
Caltech 256	257	300	200	3	30,607	80	827

4. Toolbox para generar conjuntos de entrenamiento y prueba

Para generar los conjuntos de entrenamiento y prueba en bases de datos de gran escala, como ImageNet [15], es recomendable generar archivos de tipo *tfrecord* [18]. Archivos que son utilizados para guardar una gran cantidad de imágenes, que pueden ser de diferentes tamaños, codificadas en arreglos multidimensionales.

Para generar los archivos *tfrecord*, se desarrollo un toolbox, que permite crear los archivos de entrenamiento y prueba de una manera fácil y rápida; además de permitir la generación de parámetros que ayudan al entrenamiento de la red. El toolbox se ejecuta directamente desde la linea de comandos con `python`

`tfpyToolbox.py`, ver Figura 1. Cuando se visualiza la ventana podemos seleccionar la ubicación de las imágenes ya clasificadas, la ruta donde se guardarán los archivos, si se requiere redimensionar la imagen o recortar, el tamaño de la imagen, y el tamaño de los conjuntos de imágenes.

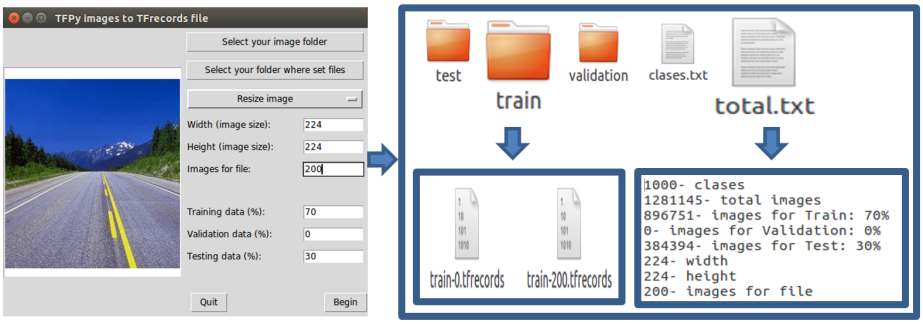

Fig. 1. TFpyToolbox.

La ventaja de este toolbox es que podemos generar conjuntos de entrenamiento y prueba, preparados para ser utilizados en cualquier RNC. El toolbox genera diferentes carpetas con archivos de tipo *tfrecord* y archivos de texto que ayuda en el entrenamiento como son: la descripción de todas las clases, tamaño de la imagen, totales por conjunto de datos. El toolbox separa la base de datos de entrenamiento para utilizarla con diferentes modelos y diferentes parámetros sin necesidad de volver a generar los conjuntos de datos, esta separación se hace por cada clasificación, debido a que las clasificaciones tienen diferentes cantidades de imágenes.

De los archivos *tfrecord* se leen las imágenes y las clases separándolos por lotes y de manera aleatoria, *Tensorflow* [1] tiene la función *shuffle_batch* para hacerlo, esta función regresa dos lotes, uno de imágenes y otro de clases en nuestro caso será un lote de 32. Cuando se obtienen las imágenes se pueden visualizar algunas para verificar que están de forma correcta y poder continuar con el proceso. En la Figura 2 se muestra de manera general, el proceso de generación y lectura de los archivos.

5. Experimentos y resultados

5.1. Parámetros de entrenamiento de las RNC

Las RNC fueron entrenadas utilizando ADAM [6] con un tamaño de lote de $\beta = 32$ imágenes y un decaimiento de pesos (factor de regularización) de $\lambda = 0{,}0005$. Los pesos iniciales en cada una de las capas fueron inicializados con

[1] TenserFlow. *https://www.tensorflow.org*, (Enero 2017) .

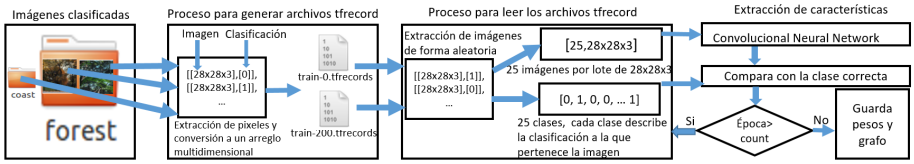

Fig. 2. Escritura y lectura de archivos tfrecord.

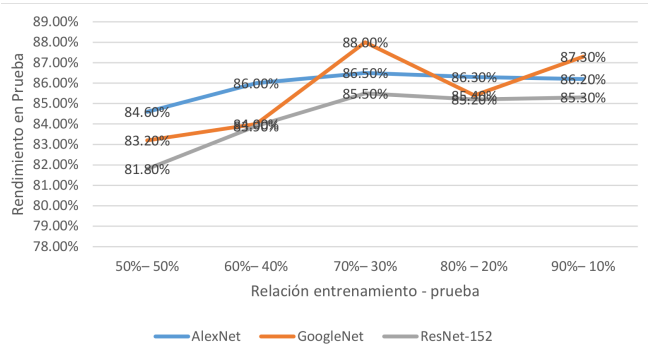

Fig. 3. Porcentajes de éxito en entrenamiento y prueba para las distintas redes y el conjunto de datos de Oliva & Torralba.

una distribución gaussiana con una media de 0 y una desviación estándar de 0,01. Los umbrales de activación en cada una de las capas fueron inicializados a cero. Iniciamos con una tasa de aprendizaje de $\mu = 0{,}001$ la cual se disminuyó en un factor de 10 después de cada 25 épocas, para tener cambios de aprendizaje más específicos en 100 épocas de entrenamiento. Las redes fueron entrenadas en una GPU NVIDIA GeForce GTX TITAN X, con 12 GB de memoria RAM y 3076 núcleos, con sistema operativo Linux Ubuntu 16.04, linux kernel 4.12, Python 2.7, Tensorflow 1.12, NVIDIA CUDA® 8.0, NVIDIA cuDNN v5.1.

5.2. Relación entrenamiento/prueba

Para evaluar la relación de división de los conjunto de entrenamiento y prueba, elegimos la base de datos Oliva & Torralba utilizando un método de rejilla, variamos la relación desde $50-50$ hasta $90-10$ para el porcentaje del conjunto de entrenamiento y prueba respectivamente. Las redes fueron entrenadas desde cero y en cada prueba se registro el rendimiento de la red.

La redes obtuvieron el rendimiento más alto utilizando una relación 70/30 para entrenamiento y prueba, ver Figura 3.

5.3. Estandarización de conjuntos de entrenamiento

Para evaluar el efecto de la estandarización o no estandarización de las bases de datos en el rendimiento de las RNC, planteamos las siguientes pruebas:

1. En la primer prueba, estandarizamos el número de ejemplos por clase de las bases de datos, a la clase con el menor número de imágenes; como se puede ver en la Tabla 2.
2. En la segunda prueba, utilizamos las bases de datos completas, en la Tabla 3 se muestran las características de cada conjunto de datos.

Para evaluar el rendimiento de las RNC, las redes se entrenaron desde cero, con cada una de las bases de datos. Para las pruebas se utilizó el método de retención y cada base de datos fue dividida en conjuntos de entrenamiento y prueba. Los conjuntos de entrenamiento fueron formados con el 70 % de las imágenes de cada base de datos y el 30 % restante se utilizó para formar los conjuntos de prueba; la selección de las imágenes se realizó de manera aleatoria. Para ajustar las RNC a cada uno de los conjunto de entrenamiento, es necesario igualar el número de neuronas de salida al número de clases de cada conjunto.

Los resultados de la prueba 1 se muestran en la Tabla 4, mientras que los resultados de la prueba 2 se muestran en la Tabla 5. Como se puede apreciar en los resultados, para las tres bases de datos y los tres tipos de redes consideradas, los mejores resultados en cuanto a rendimiento, se obtienen cuando se utilizan bases de datos completas. En general GoogleNet tuvo mejores resultados, pero es interesante analizar los dos experimentos. Al realizar la primera prueba con Caltech 256 se dejaron más de 10, 000 imágenes fuera del conjunto de datos, al realizar la segunda prueba con la base de datos completa se incrementó el rendimiento para las tres redes. ResNet 152 al tener más capas necesita más cantidad de imágenes para tener mejor rendimiento, al tener mayor profundidad implica mayor cantidad de tiempo en entrenamiento, necesitando 4 veces más que GoogleNet, con las bases de datos utilizadas.

Teóricamente, la prueba 1 intenta quitar el sesgo hacia la clase con el mayor número de ejemplos. De la Tabla 6 se puede ver que el porcentaje de reconocimiento de la clase con mayor número de ejemplos en Caltech 256, *Clutter*, disminuye con la estandarización, pero esto sucede por que se le quitan ejemplos de entrenamiento a la CNN. Por el contrario, el porcentaje de éxito para la mayoría de las clases con menos ejemplos aumenta – las clases *golden-gate-bridge, harpsichord, scorpion-101, sunflower-101, top-hat* son las que menos ejemplos presentan en Caltech256. Lo que confirma que si se esta disminuyendo el sesgo hacia la clase *Clutter*. Sin embargo, llama la atención que el porcentaje de éxito para la clase con etiqueta *top-hat* disminuye, por lo que se procedió a calcular el porcentaje de mejora promedio para cada una de las clases. Como se observa en la Tabla 7, la prueba 1 incrementa el porcentaje de éxito de 106 clases, mientras que disminuye el de 143 clases. Esta es la razón por la que se obtiene un porcentaje de reconocimiento mejor para la prueba 2.

Como complemento y para mejor análisis de datos, para visualizar las clases que más se confunden, se generaron matrices de confusión por cada prueba, como por ejemplo la Tabla 8 con la prueba de la red GoogleNet y el conjunto de datos Oliva & Torralba con la base de datos completa. Se puede ver que en la diagonal se obtiene la suma de 712 lo que nos da un 88 % de éxito, y que la clase *OpenCountry* es el que más se confunde con la clase *Coast*, el que tiene

Tabla 2. Bases de datos separadas en entrenamiento y pruebas, obteniendo la clase que contiene la mínima cantidad de imágenes, así se podrá entrenar con la misma cantidad de imágenes por clase.

Conjunto de datos	**Clases**	**Cantidad de imágenes**		
		Entrenamiento	**Prueba**	**Total**
Oliva & Torralba	8	1, 456	624	2, 080
ImageNetDogs	120	12, 360	5, 400	17, 760
Caltech 256	257	14, 392	6, 168	20, 560

Tabla 3. Bases de datos separadas en entrenamiento y pruebas, utilizando el 70 % de imágenes por clase para entrenamiento y el 30 % para pruebas.

Conjunto de datos	**Clases**	**Cantidad de imágenes**		
		Entrenamiento	**Prueba**	**Total**
Oliva & Torralba	8	1, 879	809	2, 688
ImageNetDogs	120	14, 358	6, 222	20, 580
Caltech 256	257	21, 314	9, 293	30, 607

Tabla 4. Resultados misma cantidad de imágenes por clase.

CNN	**Oliva & Torralba**			**ImageNetDogs**			**Caltech 256**		
	Minutos	**Top-1**	**Top-5**	**Minutos**	**Top-1**	**Top-5**	**Minutos**	**Top-1**	**Top-5**
AlexNet	22	85,4 %	99,7 %	125	30,6 %	60,2 %	146	43,7 %	64,5 %
GoogleNet	26	84,8 %	99,8 %	179	29,6 %	61,9 %	213	42,7 %	65,5 %
ResNet 152	105	84,3 %	99,4 %	769	13,5 %	38,1 %	887	23,2 %	43,3 %

Tabla 5. Resultados utilizando las bases de datos completas.

CNN	**Oliva & Torralba**			**ImageNetDogs**			**Caltech 256**		
	Minutos	**Top-1**	**Top-5**	**Minutos**	**Top-1**	**Top-5**	**Minutos**	**Top-1**	**Top-5**
AlexNet	23	86,5 %	99,5 %	122	33,4 %	64,4 %	147	49,9 %	70,1 %
GoogleNet	27	88,0 %	100 %	175	30,5 %	64,5 %	215	50,8 %	71,8 %
ResNet 152	106	85,5 %	99,5 %	760	14,7 %	41,0 %	890	32,9 %	54,5 %

Tabla 6. Comparación entre la prueba uno y la prueba dos con las clases con mayor cantidad de imágenes y menor cantidad de imágenes del conjunto de datos Caltech 256, con las clases $C1, C2, ..., C6$ que corresponden a *Clutter, golden-gate-bridge, harpsichord, scorpion-101, sunflower-101, top-hat.*

Clase	**Imágenes**	**Prueba 1**	**Prueba 2**
C1	827	0,25 %	0,61 %
C2	80	0,58 %	0,54 %
C3	80	0,75 %	0,41 %
C4	80	0,54 %	0,41 %
C5	80	0,83 %	0,79 %
C6	80	0,37 %	0,54 %

Tabla 7. Porcentaje de mejora promedio para cada una de las clases estandarizadas en Caltech-256.

	Núm. clases	**Porcentaje**
Clases con mayor rendimiento	106	0,045
Clases con menor rendimiento	143	0,065
Clases sin cambiar	8	0,0

Tabla 8. Matriz de confusión con las clases $C1, C2, ..., C8$ que corresponden a *Opencountry, Coast, Forest, Highway, Inside_city, Mountain, Street, Tallbuilding*, que contiene el conjunto de datos de Oliva & Torralba, tiene un éxito en pruebas de 88 % con 712 imágenes correctas en la predicción de la red GoogleNet, también se puede ver que la clase que más se confunde es *Opencountry* con la clase *Coast*.

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**Total**
C1	106	11	0	3	0	3	0	0	123
C2	3	96	1	2	0	6	0	0	108
C3	3	0	89	0	0	7	0	0	99
C4	7	9	0	55	1	1	4	1	78
C5	0	1	0	0	89	0	2	1	93
C6	3	2	2	0	0	106	0	0	113
C7	0	0	1	2	6	2	76	1	88
C8	1	3	0	0	5	3	0	95	107
Total	123	122	93	62	101	128	82	98	809

menor éxito es *Highway* con 70 %. En la Figura 4 se muestran las predicciones obtenidas con algunas de las imágenes utilizadas.

Fig. 4. Prueba de la red AlexNet, la latra R significa que es la clasificación real, la letra P significa que es la predicción de la red.

6. Conclusiones

Separar los datos en archivos para entrenamiento y prueba permite reutilizar las particiones para diferentes experimentos siempre con los mismo datos, disminuyendo el tiempo de entrenamiento. Dividir un conjunto de datos de imágenes en 70 % para entrenamiento y 30 % para pruebas mejora el rendimiento de las redes neuronales convolucionales. Estandarizar una base de datos a un número mínimo de imágenes (implica tener la misma cantidad de imágenes por clase) baja el rendimiento, debido a que se dejan de utilizar imágenes para el entrenamiento; sin embargo, sí se logra reducir el sesgo hacia la clase dominante.

Realizar la separación por clase (implica tener la base de datos completa), aumenta el porcentaje de éxito para los conjuntos de datos utilizados en este trabajo. A mayor cantidad de imágenes se obtienen mejores resultados para las redes con mayor profundidad, pero con mayor tiempo en entrenamiento, implica un aumento en el costo computacional. Sin embargo, cabe destacar que es posible que en bases más grandes se puede ajustar mejor y aumentar el rendimiento normalizando las clases. El entrenamiento de una RNC es la base para el aprendizaje, analizar los datos de entrenamiento y prueba puede ayudar a reducir los tiempos y a mejorar el aprendizaje. En este trabajo se ponen las bases para crear las propias bases de datos de imágenes, de entrenamiento y prueba utilizando archivos *tfrecord* para utilizarlas en redes neuronales convolucionales.

Como trabajo futuro se plantea experimentar con el pre-procesado de la imagen antes del ingreso a la red, modificando características como el enfoque, la claridad, recorte (*crop*), relleno (*padding*), entre otros, para simular la riqueza de imágenes. Implica mayor costo computacional, pero se espera un mejor rendimiento. Otra linea implica probar diferentes técnicas de muestreo para sobremuestrear las clases no dominantes sin llegar a producir sobre-entrenamiento.

Referencias

1. Amari, S.i., Murata, N., Muller, K.R., Finke, M., Yang, H.H.: Asymptotic statistical theory of overtraining and cross-validation. IEEE Transactions on Neural Networks 8(5), 985–996 (1997)
2. Caltech256: Caltech 256 dataset. www.vision.caltech.edu/ImageDatasets/ Caltech256 (Mayo 2016)
3. Friedman, J., Hastie, T., Tibshirani, R.: The elements of statistical learning, chap. 7, pp. 219–259. Springer series in statistics New York, 2nd edn. (2009)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015), http://arxiv.org/abs/1512.03385
5. Khosla, A., Jayadevaprakash, N., Yao, B., Fei-Fei, L.: Stanford Dogs Dataset. http://vision.stanford.edu/aditya86/ ImageNetDogs/ (Septiembre 2017)
6. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. arXiv:1412.6980v9 (2017)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

8. Lei, B., Xu, G., Feng, M., van der Heijden, F., Zou, Y., de Ridder, D., Tax, D.M.: Classification, parameter estimation and state estimation: an engineering approach using MATLAB, chap. 5, pp. 139–182. John Wiley & Sons, 2nd edn. (2017)
9. Mishkin, D., Sergievskiy, N., Matas, J.: Systematic evaluation of CNN advances on the imagenet. CoRR abs/1606.02228 (2016), http://arxiv.org/abs/1606.02228
10. Oliva, A.: Computational visual cognition laboratory. http://cvcl.mit.edu/ database.htm (Mayo 2016)
11. Ruder, S.: An overview of gradient descent optimization algorithms. CoRR abs/1609.04747 (2016), http://arxiv.org/abs/1609.04747
12. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach, p. 709. Malaysia; Pearson Education Limited, 3rd edn. (2009)
13. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR abs/1409.1556 (2014), http://arxiv.org/abs/1409.1556
14. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 1929–1958 (2014), http://jmlr.org/papers/v15/srivastava14a.html
15. Stanford Vision Lab: Imagenet. http://image-net.org/ (Octubre 2017)
16. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning (2013)
17. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. CoRR abs/1409.4842 (2014), http://arxiv.org/abs/1409.4842
18. Tensorflow: Importing data. https://www.tensorflow.org/programmers_guide/ datasets (Enero 2018)
19. Tibshirani, R., James, G., Witten, D., Hastie, T.: An introduction to statistical learning-with applications in R, p. 176. New York, NY: Springer (2013)