Un Estudio sobre el Preprocesamiento para Redes Neuronales Profundas y Aplicación sobre Reconocimiento de Dígitos Manuscritos

Daniel Peralta[1], Andrés Herrera-Poyatos[1], y Francisco Herrera[1]

Departamento de Ciencias de la Computación e Inteligencia Artificial, Universidad de Granada, 18071 Granada, España
dperalta@decsai.ugr.es, andreshp9@gmail.com, herrera@decsai.ugr.es

Resumen Los recientes avances en el entrenamiento de redes neuronales profundas las señalan como modelos muy eficaces para el aprendizaje de patrones complejos. Tareas como el reconocimiento de dígitos manuscritos pueden ser llevadas a cabo de forma automática con una alta precisión. Sin embargo, es fundamental un adecuado preprocesamiento de los datos para optimizar los resultados de tales redes. En este trabajo se hace un análisis de diferentes técnicas de preprocesamiento de imágenes de dígitos manuscritos, y se evalúa su impacto en el rendimiento de varias arquitecturas de redes neuronales profundas. Los experimentos sobre el conocido conjunto de datos MNIST revelan la potencia de las redes convolutivas para el reconocimiento de dígitos. Asimismo, diferentes preprocesamientos afectan de forma distinta a diferentes arquitecturas de red, permitiendo mejorar su tasa de acierto.

Palabras clave: Clasificación · Deep learning · Preprocesamiento · Dígitos manuscritos · Aumento de datos

Agradecimientos D. Peralta tiene una beca FPU del Ministerio de Educación y Ciencia (FPU12/04902).

1. Introducción

La clasificación, en el ámbito del aprendizaje automático, consiste en extraer conocimiento de un conjunto de instancias etiquetadas para aprender a inferir la etiqueta dada una instancia [4]. Un aspecto fundamental de un buen clasificador es su capacidad de generalización, para clasificar correctamente instancias nuevas que no están en el conjunto de entrenamiento.

Las redes neuronales profundas (*Deep Neural Networks*, DNN) han acaparado mucha atención en los últimos años debido a su capacidad y flexibilidad para el aprendizaje de patrones complejos [12]. Se han aplicado con éxito sobre diversos problemas, como la clasificación de imágenes [9,10], el reconocimiento de dígitos [11] y el reconocimiento del habla [7,18], entre otros.

Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial, pp. 867-876

Una red neuronal parte de un vector, matriz o tensor de entrada y aplica una serie de transformaciones, estructuradas en capas de neuronas, hasta obtener una determinada salida [14]. Los avances en tecnología de procesamiento GPU (*Graphics Processor Unit*), así como la aparición de conjuntos de datos de gran tamaño, han permitido el entrenamiento de DNN con múltiples capas de neuronas [12], capaces de aprender patrones cada vez más complejos.

El preprocesamiento de datos [5] es parte fundamental de cualquier proceso de aprendizaje automático. En algunos casos, busca corregir deficiencias en los datos que puedan dañar el aprendizaje, como omisiones, ruido y valores extremos [6]. En otros casos, se persigue adaptar los datos al modelo que se pretende entrenar para simplificar u optimizar el proceso.

La elevada capacidad de abstracción de las DNN implica que pueden trabajar más fácilmente con datos originales de alta dimensión que no serían viables para otros algoritmos, reduciendo la necesidad de una preparación manual de los datos con conocimiento experto. Sin embargo, un preprocesamiento adecuado sigue siendo importante para obtener resultados de calidad [3].

Una de las técnicas de preprocesamiento más utilizadas con DNN es el aumento de datos vía transformación de los datos originales [9,11,19]. Consiste en replicar las instancias del conjunto de entrenamiento introduciendo diversos tipos de transformaciones (traslaciones, rotaciones, simetrías, etc.). El aumento de datos permite entrenar modelos menos sensibles a ruido y sobreentrenamiento.

En este trabajo se analizan distintas técnicas de preprocesamiento y aumento de imágenes de dígitos manuscritos. En concreto, analizamos técnicas basadas en centrado, traslación, rotación y deformación elástica. El objetivo es evaluar cómo afectan estas transformaciones a los resultados de distintas arquitecturas de redes neuronales profundas, aplicadas sobre el ampliamente utilizado dataset MNIST [13], que contiene 70.000 imágenes de caracteres organizadas en 10 clases.

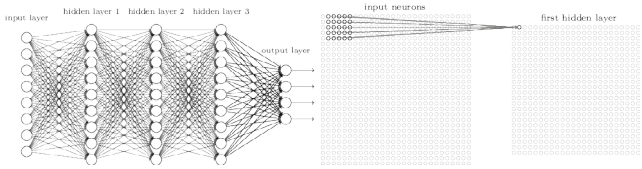
El documento se estructura como sigue. Primero, la Sección 2 expone las características generales de *Deep Learning* para clasificación. La Sección 3 describe las técnicas de preprocesamiento de datos y las DNN aplicadas sobre el problema de reconocimiento de dígitos en este trabajo. La Sección 4 describe los experimentos realizados y presenta un análisis de sus resultados. Finalmente, la Sección 5 cierra el documento, exponiendo las conclusiones del estudio realizado.

2. *Deep Learning*

Una red neuronal se compone de una serie de capas de neuronas, cada una de las cuales calcula una suma ponderada de la salida de las neuronas de la capa anterior y aplica una función de activación. Las DNN son redes neuronales con varias capas ocultas. En general, cada capa consigue extraer un cierto grado de abstracción, de modo que una DNN permite aprender patrones más complejos y genéricos [14]. Existen distintos tipos de capas de neuronas para las DNN [11,14]:

- Capas totalmente conectadas: cada neurona está conectada mediante pesos a todas las neuronas de la capa anterior (Figura 1a).

- Capas convolutivas: cada neurona está conectada a un área de neuronas de la capa anterior (Figura 1b). Los pesos se comparten entre todas las neuronas de una misma capa, reduciendo el espacio de búsqueda del aprendizaje.
- Capas de *pooling*: suelen situarse tras una capa convolutiva. Al igual que en éstas, cada neurona está conectada a un área de la capa anterior, y devuelve el máximo o la media de esos valores.



(a) DNN totalmente conectada (b) Capa convolutiva

Figura 1. Tipos de capas de neuronas [14]

En la práctica, se denomina redes neuronales convolutivas (*Convolutional Neural Networks*, CNN) a las que incluyen combinaciones de los tres tipos de capas. Estas redes están muy adaptadas al cómputo de imágenes y estructuras que incluyen algún tipo de relación espacial, como demuestran los resultados obtenidos en diversas competiciones [9,17].

Cuando una red se utiliza como clasificador para un problema con clases $c_1, ..., c_m$, la capa de salida contiene una neurona por clase, formando un vector $\mathbf{a} = (a_1, ..., a_m)$. Para convertir estos valores a probabilidades se utiliza la función SoftMax (Ecuación 1), donde $\text{SoftMax}(a_i)$ es la probabilidad de que la entrada pertenezca a la clase c_i. Por tanto, para cada instancia se busca que todas las neuronas de salida tengan valores cercanos a cero, excepto la de la clase correcta, que debería ser próxima a uno.

$$\text{SoftMax}(a_i) = \frac{e^{a_i}}{\sum_{j=1}^{m} e^{a_j}} \ , \ i = 1, ..., m \tag{1}$$

El entrenamiento de una red consiste en optimizar los pesos de cada neurona para que la salida sea la deseada para cada entrada. Por tanto, el espacio de búsqueda tiene tantas dimensiones como el número total de pesos. El algoritmo de referencia es la propagación hacia atrás con gradiente descendente (GD) [16]. En cada iteración, se calculan las salidas de la red para todas las instancias de entrenamiento, y se utiliza el error respecto a la salida deseada para calcular su derivada respecto a los pesos. Éstos se modifican en la dirección marcada por esa derivada. El proceso es iterativo hasta que el aprendizaje converge.

El entrenamiento de una DNN es más costoso computacionalmente, por el mayor espacio de búsqueda y la mayor complejidad del cómputo del GD. El gradiente descendente estocástico (*Stochastic Gradient Descent*, SGD) palia este problema utilizando un subconjunto de las instancias de entrenamiento (denominado *batch*) en cada iteración, de modo que el cálculo del error queda sesgado respecto al óptimo pero a cambio es mucho más rápido. Cada iteración sobre el conjunto de entrenamiento completo, denominada *epoch*, requiere de múltiples iteraciones sobre los pequeños *batches*.

Esta técnica suele combinarse con operaciones complementarias para mejorar las propiedades de convergencia y generalización de la red:

- Momento [15]: la dirección para modificar los pesos es una combinación lineal de la dirección previa de búsqueda y la obtenida con el *batch* actual.
- *Dropout* [20]: consiste en asignar un valor cero a un subconjunto aleatorio de los valores de una determinada capa. Favorece la aparición de caminos redundantes en la red, reduciendo el sobreajuste.
- Normalización: se añade a la función de error la suma de los pesos (o de sus cuadrados), para mejorar la robustez del aprendizaje.

3. Preprocesamiento de dígitos manuscritos

En este trabajo se aplican distintos métodos de preprocesamiento de datos sobre el dataset público de dígitos manuscritos MNIST [13] (Sección 3.1). Como ya se ha indicado, el objetivo es evaluar cómo afectan estas transformaciones (Sección 3.2) a los resultados obtenidos por distintos tipos de redes neuronales profundas (Sección 3.3), para sentar las bases de trabajos futuros en el ámbito.

3.1. Base de datos MNIST

Los experimentos realizados para este trabajo utilizan la base de datos de dígitos manuscritos MNIST [13]. Contiene 70.000 imágenes de dígitos (28×28 píxeles), de las cuales 60.000 se usan para entrenamiento y 10.000 para test. Los dígitos de entrenamiento y test fueron escritos por personas diferentes para evitar correlaciones. Las imágenes se distribuyen de forma equitativa entre 10 clases, una por cada dígito del 0 al 9. Se pueden ver listados completos y actualizados de los resultados obtenidos para la base de datos en [13,2].

3.2. Preprocesamiento y aumento de dígitos

Se han aplicado los siguientes métodos de preprocesamiento, algunos ya estudiados en diferentes propuestas del ámbito [11,14], dando la mayoría de ellos lugar a un aumento en el número de datos:

- Rotación: al tratarse de mapas de píxeles y no de imágenes vectoriales, es necesario aplicar interpolaciones tras la rotación, lo que añade variabilidad a las imágenes resultantes.

- Desplazamiento: cada imagen se desplaza un número de píxeles.
- Deformación elástica [19]: se calcula aleatoriamente para cada píxel la dirección de la deformación en el plano, y se aplica una convolución con *kernel* gaussiano sobre tales direcciones.
- Centrado: proponemos un método de centrado de los dígitos. Primero, se eliminan de los bordes de cada imagen las filas y columnas en blanco, tras lo cual pueden quedar imágenes de diferente tamaño. Luego, se calcula el máximo de filas ($r_{\max}$) y columnas ($c_{\max}$) de las imágenes recortadas. Todas las imágenes se redimensionan mediante un escalado (independiente para cada eje) a un tamaño ($r_{\max} \times c_{\max}$). Este preprocesamiento elimina las partes de la imagen que no aportan información, al tiempo que el escalado introduce una cierta deformación.
  Cuando el centrado se combina con otras transformaciones, se aplica en último lugar para obtener imágenes finales centradas.

La Figura 2 muestra una de las imágenes de MNIST, junto con una imagen por cada una de las cinco transformaciones descritas. La Figura 3 muestra el valor promedio de las imágenes de la base de datos original y la obtenida tras aplicar el algoritmo de centrado a todas las imágenes.

(a) Original (b) Rotación (c) Desplazamiento (d) Deformación elástica (e) Centrado

Figura 2. Instancia de MNIST antes y después de las transformaciones.

Figura 3. Promedio de las imágenes de MNIST por clase, antes y después del centrado

Las transformaciones descritas hasta ahora se pueden combinar para aumentar aún más el volumen y la variabilidad de los datos para el entrenamiento de la red. La Tabla 1 muestra todas las combinaciones de transformaciones consideradas en este estudio, junto con el número de instancias del conjunto de entrenamiento en cada caso. Nótese que cada dataset también contiene las imágenes sin transformaciones, sean centradas o no.

Tabla 1. Combinaciones de transformaciones consideradas en este estudio

Combinaciones	**Instancias de entrenamiento**
Original	60.000
Centrado	60.000
Elástico	300.000
Desplazamiento	300.000
Rotación	300.000
Elástico-centrado	300.000
Rotación-centrado	300.000
Desplazamiento-elástico	1.500.000
Desplazamiento-rotación	1.500.000
Rotación-elástico	1.500.000
Rotación-elástico-centrado	1.500.000

3.3. Redes neuronales profundas consideradas

Para evaluar el comportamiento de los preprocesamientos descritos se han implementado dos redes neuronales diferentes, utilizando la entropía cruzada (*Cross Entropy*) como función de pérdida y capas de salida de 10 neuronas con función de activación SoftMax.

Por una parte, se ha diseñado una red neuronal completamente conectada (*Fully Connected DNN*, FCDNN), con 3 capas ocultas de 1024, 1024 y 2048 neuronas respectivamente, ReLUs como función de activación, *dropout* y normalización L1. Para el entrenamiento de la res se ha aplicado el algoritmo de gradiente descendiente tradicional.

Tabla 2. Topología de la red LeNet

Capa	**Tamaño**	***Stride***	**Activación**
conv1	$5 \times 5 \times 20$	1	–
pool1	2×2	2	–
conv2	$5 \times 5 \times 50$	1	–
pool2	2×2	2	–
fc1	500	–	ReLU
fc2	10	–	SoftMax

Por otra parte, se ha usado la red convolutiva LeNet [11] con 2 capas convolutivas (seguidas de 2 capas de *max pooling*) y 2 capas totalmente conectadas (Tabla 2), todo ello entrenado con SGD. Para los datasets de imágenes centradas se ha añadido a la capa *conv1* un *padding* de 3 píxeles, y se ha aumentado el tamaño de la máscara a 7 píxeles.

4. Experimentos y resultados

Esta sección describe el entorno en el cual se han realizado los experimentos (Sección 4.1) y presenta un análisis de los resultados (Sección 4.2).

4.1. Configuración

La red convolutiva LeNet ha sido implementada en el *software* Caffe [8]. Por su parte, para la red FCDNN se ha utilizado H2O [1] en R. La Tabla 3 muestra los parámetros utilizados para el preprocesamiento de los datos y para el entrenamiento de las redes neuronales descritas en la Sección 3.3.

Tabla 3. Parámetros de los algoritmos

Algoritmo	**Parámetro**	**Valor**
LeNet (SGD)	Número de iteraciones	10.000 / 50.000
	Tamaño de *batch*	64 / 256
	Tasa de aprendizaje	$lr_0(1+\gamma * iter)^{-b}$
	Tasa de aprendizaje inicial (lr_0)	0,01
	γ	0,0001
	b	0,75
	Momento	0,9
	Coeficiente de regularización L2	0,0005
FCDNN (GD)	Epochs	100
	Tasa de aprendizaje	ADADELTA [21]
	Tasa de aprendizaje inicial	0,005
	Coeficiente de regularización L1	10^{-5}
Transformación elástica	Desviación típica	6
	Número de transformaciones	4
Desplazamiento	Magnitud	± 3 píxeles
	Dirección	Vertical y horizontal
Rotación	Ángulos	± 8 y ± 16 grados
Centrado	Tamaño final	20×20 píxeles

4.2. Resultados

Se han llevado a cabo dos estudios experimentales: uno inicial con los parámetros de la Tabla 3, y un segundo estudio ampliando el número de iteraciones y el tamaño de *batch* para la red LeNet, para así aumentar el número de *epochs*, recorriendo varias veces el dataset completo incluso en las combinaciones con mayor número de instancias.

Estudio con 10.000 iteraciones La Tabla 4 muestra los resultados de FCDNN y LeNet con los distintos preprocesamientos. Se han aplicado 100 epochs de entrenamiento para FCDNN. Para LeNet, que utiliza SGD para el entrenamiento, se han separado en columnas distintas la mejor tasa de acierto en test obtenida a lo largo del entrenamiento de la red, y la tasa de acierto en la última iteración. Una diferencia alta entre ambos valores indicaría un sobreentrenamiento de la red. El mejor valor para cada columna se ha resaltado en negrita.

Se hace patente que, en general, la red convolutiva LeNet se adapta mejor al procesamiento de imágenes que FCDNN, pese a utilizar un número de *epochs* y una carga computacional menores. Es muy destacable que el mejor resultado

Tabla 4. Tasas de acierto en test con FCDNN y LeNet

	FCDNN	**LeNet (10.000 iter.)**			**LeNet (50.000 iter.)**		
Dataset entrenamiento	**Final**	**Mejor**	**Final**	**Epochs**	**Mejor**	**Final**	**Epochs**
Original	98.66 %	99.09 %	99.09 %	10.67	99.34 %	99.13 %	213.33
Centrado	98.73 %	98.96 %	98.86 %	10.67	99.10 %	99.07 %	213.33
Elástico	98.97 %	99.11 %	99.04 %	2.13	99.45 %	99.37 %	42.67
Desplazamiento	98.27 %	99.12 %	98.91 %	2.13	99.34 %	99.23 %	42.67
Rotación	98.84 %	98.99 %	98.97 %	2.13	99.35 %	99.27 %	42.67
Elástico-centrado	**99.16 %**	99.13 %	99.13 %	2.13	99.39 %	99.31 %	42.67
Rotación-centrado	98.93 %	99.01 %	98.94 %	2.13	99.29 %	99.20 %	42.67
Desplazamiento-elástico	99.07 %	98.65 %	98.17 %	0.43	99.50 %	99.31 %	8.53
Desplazamiento-rotación	98.94 %	98.55 %	97.59 %	0.43	99.37 %	98.77 %	8.53
Rotación-elástico	98.89 %	**99.33 %**	**99.33 %**	0.43	**99.54 %**	99.45 %	8.53
Rotación-elástico-centrado	99.14 %	99.27 %	99.24 %	0.43	99.52 %	**99.46 %**	8.53

con LeNet se ha obtenido con 0,43 epochs; es decir, el entrenamiento solamente ha utilizado un 43 % de las instancias del dataset, cada una de las cuales se ha utilizado solamente una vez. Este hecho destaca la potencia de una adecuada técnica de preprocesamiento.

La tabla muestra que cada topología de red proporciona su mejor resultado con un preprocesamiento distinto. Mientras que la red FCDNN se adapta bien a las huellas centradas, debido a que conllevan un espacio de búsqueda menor, y es capaz de sacar partido de las transformaciones elásticas. Sin embargo, LeNet no obtiene beneficio del centrado de las imágenes dado que las capas convolutivas son capaces de extraer ese conocimiento por sí solas; por tanto, el mejor rendimiento se obtiene con combinaciones de las otras transformaciones, como rotaciones y deformaciones elásticas.

Estudio con mayor número de *epochs* Para completar el estudio, se han repetido los experimentos de LeNet, aumentando el tamaño de *batch* a 256 y el número de iteraciones a 50.000. De esta forma, el número de *epochs* se acerca a los 100 utilizados con FCDNN. Como muestra la Tabla 4. Además, la diferencia entre el mejor acierto y el acierto final para LeNet con la base de datos original señala un cierto sobreajuste de la red, problema que no aparece con los conjuntos de datos preprocesados.

La Figura 4 muestra la mejor tasa de acierto con LeNet para cada uno de los preprocesamientos, tanto para 10.000 como para 50.000 evaluaciones. Se refleja el importante aumento de la tasa de acierto, especialmente con los datasets más grandes. Las transformaciones que han visto su tasa más aumentada son los desplazamientos. Aunque la diferencia entre las tasas de acierto de los distintos preprocesamientos se ha reducido respecto a los valores de la Tabla 4, la combinación de rotaciones y deformación elástica sigue proporcionando los mejores resultados, demostrando que un preprocesamiento adecuado facilita en gran medida el entrenamiento de las redes neuronales profundas y permite tanto obtener resultados mejores tras un largo entrenamiento, como una convergencia temprana a zonas prometedoras del espacio de búsqueda.

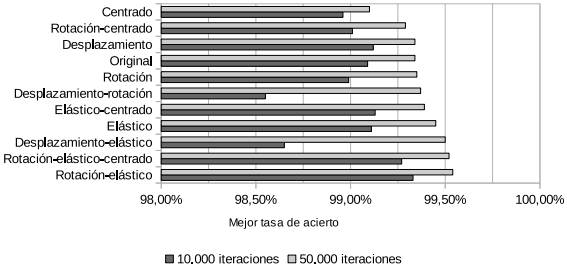


Figura 4. Mejores tasas de acierto con LeNet

5. Comentarios finales

En este trabajo, se ha comparado la influencia de distintas técnicas de preprocesamiento y aumento de datos sobre varias arquitecturas de redes neuronales profundas. Partiendo del problema de reconocimiento de dígitos manuscritos, se han aplicado las técnicas mencionadas y combinaciones de ellas sobre el conjunto de imágenes MNIST, y se han evaluado los resultados obtenidos con una red profunda totalmente conectada (FCDNN), y la red convolutiva LeNet. Asimismo, se han utilizado el gradiente descendiente y el gradiente descendiente estocástico para el entrenamiento de las redes, y se han evaluado los resultados con diferentes números de evaluaciones.

Los resultados obtenidos dejan patente que las combinaciones de métodos de procesamiento pueden mejorar el rendimiento de LeNet y de FCDNN, independientemente del número de iteraciones en el entrenamiento. Además, afectan de forma distinta a cada arquitecturas de red: la propuesta de centrado de imágenes facilita el entrenamiento de la red FCDNN, mientras que combinaciones más complejas funcionan mejor con LeNet.

Referencias

1. Arora, A., Candel, A., Lanford, J., Ledell, E., Parmar, V.: Deep Learning with H2O. H2O.ai (2015)
2. Benenson, R.: Classification datasets results, http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html
3. Cireşan, D., Meier, U., Masci, J., Schmidhuber, J.: Multi-column deep neural network for traffic sign classification. Neural Networks 32, 333–338 (2012)
4. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification. John Wiley & Sons (2012)
5. García, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer, New York, 1st edn. (2015)

6. García, S., Luengo, J., Herrera, F.: Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. Knowledge-Based Syst. 98, 1–29 (2016)
7. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. IEEE Signal Process. Mag. 29(6), 82–97 (2012)
8. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional Architecture for Fast Feature Embedding. CoRR abs/1408.5 (2014)
9. Krizhevsky, A., Sulskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Adv. Neural Inf. Process. Syst. pp. 1097–1105 (2012)
10. Le, Q.V.: Building high-level features using large scale unsupervised learning. In: 2013 IEEE Int. Conf. Acoust. Speech Signal Process. pp. 8595–8598 (2013)
11. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE 86(11), 2278–2324 (1998)
12. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)
13. LeCun, Y., Cortes, C., Burges, C.J.: The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/
14. Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015)
15. Qian, N.: On the momentum term in gradient descent learning algorithms. Neural Networks 12(1), 145–151 (1999)
16. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
17. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. 115(3), 211–252 (2015)
18. Sainath, T.N., Mohamed, A.r., Kingsbury, B., Ramabhadran, B.: Deep convolutional neural networks for LVCSR. In: 2013 IEEE Int. Conf. Acoust. Speech Signal Process. pp. 8614–8618 (2013)
19. Simard, P., Steinkraus, D., Platt, J.: Best practices for convolutional neural networks applied to visual document analysis. In: Seventh Int. Conf. Doc. Anal. Recognition, 2003. Proceedings. vol. 1, pp. 958–963. IEEE Comput. Soc (2003)
20. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15(1), 1929–1958 (2014)
21. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method. CoRR abs/1212.5 (2012)