Introducción al Deep Learning

Ignacio Haeussler

5 de enero de 2018

1. Abstract

El presente informe expone los principales conceptos involucrados en el desarrollo de redes neuronales profundas, y analiza los métodos específicos utilizados en el reconocimiento de imágenes por medio de redes neuronales convolucionales (CNNs). El análisis se realiza en torno a experimentos asociados a la técnica de Dropout, la cantidad de capas convolucionales utilizadas, comparaciones con perceptrones multicapa (MLP) y la técnica de Augmentation. En base a ellos, se obtienen resultados que señalan la importancia de incorporar invarianzas en la representación, del uso de técnicas para evitar el overfitting, y de la consideración de las relaciones entre los parámetros elegidos, como los pesos y las conexiones, y variables de desempeño, como el tiempo requerido y el gradiente percibido por capa.

2. Introducción

El Deep Learning es un área del Machine Learning introducida con el objetivo de orientar la disciplina hacia una de sus metas originales: la Inteligencia Artificial. Es parte de un conjunto de métodos basados en aprender representaciones de datos, a diferencia del desarrollo de algoritmos con tareas específicas. Ha sido aplicado a los campos de visión computacional, reconocimiento de audio y de voz, procesamiento de lenguaje natural, filtrado en redes sociales, bioinformática y traducción automática, con resultados comparables o superiores a seres humanos expertos.

El objetivo de este informe es presentar el análisis introductorio realizado sobre distintas componentes esenciales en el desarrollo de una red neuronal convolucional, un tipo de red neuronal utilizada en Deep Learning, específicamente en el campo de visión computacional. Para ello se

utilizó la famosa base de datos CIFAR-10 de imágenes de 32x32 pixeles con 10 clases distintas: aviones, autos, pájaros, gatos, ciervos, perros, ranas, caballos, barcos y camiones. Además, se utilizó la biblioteca de código abierto Tensorflow para procesar datos y desarrollar las redes neuronales convolucionales.

3. Marco Teórico

3.1. Redes convolucionales

Una red neuronal convolucional es una red neuronal que se caracteriza por la forma particular en que aplica cuatro operaciones principales: convolución, aplicación de no linealidad, submuestreo y clasificación, las que son detalladas a continuación.

Una capa convolucional posee un ancho, una altura y una profundidad. Cada nivel de profundidad determina una subcapa, llamada también mapa de características. Por ejemplo, la imagen (capa inicial) poseería comúnmente 3 mapas de características, una por cada componente de color (RGB). El ancho y la altura determinan la cantidad de unidades (neuronas, o pixel en caso de capa inicial) por subcapa, mientras que cada mapa de características define un conjunto de pesos, llamado kernel o filtro. Esto último se debe a que tal conjunto de pesos realiza una operación de filtrado (de extracción de una característica) sobre un volumen de entrada (definido por un subconjunto de mapas de características, proveniente de la capa anterior). El filtrado se realiza aplicando el kernel por cada unidad sobre un volumen pequeño y conexo, existiendo comúnmente un traslape entre los datos filtrados de unidades adyacentes. Las unidades de una subcapa filtrarían en conjunto la totalidad de un subconjunto de subcapas (mapas de características) de la capa previa, aplicando la fórmula de convolución o de correlación cruzada:

$$K * I = \sum_u \sum_v K[u,v] \cdot I[i-u, j-v]; \qquad K \otimes I = \sum_u \sum_v K[u,v] \cdot I[i+u, j+v]$$

Convolución y Correlación cruzada en 2 dimensiones: K el kernel (filtro) a aplicar, e I es la imagen o subconjunto de mapas de características a procesar

Posteriormente, un sesgo es sumado, generando un efecto de umbral: el sesgo tiende a ser negativo, por lo que si la convolución es pequeña, la unidad "no se disparará" o su señal será muy baja, dependiendo de la función de activación utilizada. Esta última sera aplicada inmediatamente después

de la adición del sesgo, con lo que la fórmula utilizada por cada unidad en una capa convolucional será:

$$conv(K, I)_{ij} = \sigma(b + \sum_u \sum_v \sum_w K[u, v, w] \cdot I[i - u, j - v, k - w])$$

Fórmula aplicada por cada unidad en una capa convolucional. Notar que la salida i,j contendrá no solo la convolución en i,j de la imágen o subconjunto de mapas de características, sino que también su profundidad (en k).

Cabe señalar que el lugar en que la convolución debe comenzar a aplicarse no está hasta aquí bien definido, pues ocurre que al aplicar un filtro sobre un pixel del borde de la imagen (quedando el pixel en el 'centro' del filtro), algunos pesos podrían quedar sin elementos correspondientes con los que multiplicarse. Aquí es donde entra el concepto de 'padding', el que consiste en agregar elementos al borde de las imágenes o mapas de características para poder aplicar el filtro a cada elemento. Un valor de padding comúnmente utilizado es el cero, lo que es llamado 'zero-padding'. Sin embargo, también es posible no aplicar el filtro a todos los elementos (que no ocurra que todos queden alguna vez en el 'centro' del filtro), es decir, no utilizar padding.

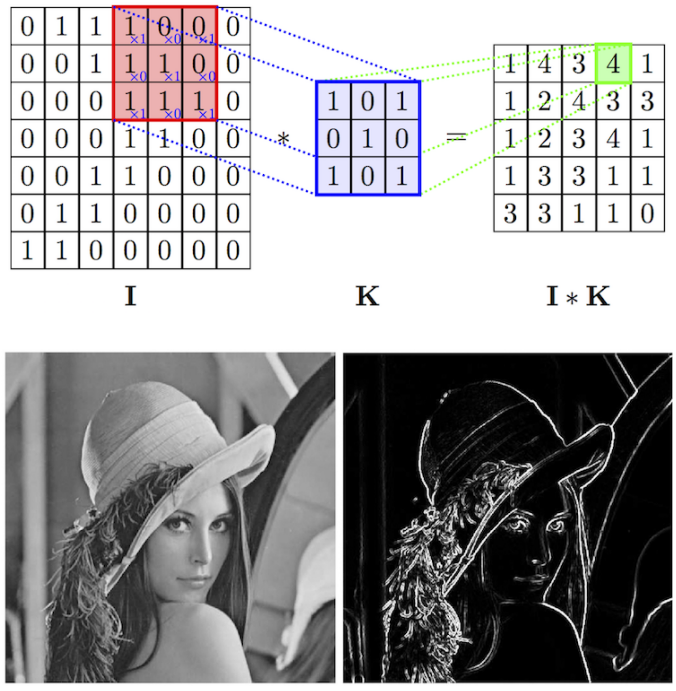

Figura 1: Convolución (obtenida desde [1])

Las capas convolucionales son capaces mediante esta estructura de (simultáneamente) extraer características locales y distintas de un flujo único o combinado de datos de entrada, establecer invariantes al desplazamiento (utilizando el mismo filtro en cada unidad de un mapa de características particular), y limitar de manera razonable la cantidad de parámetros utilizados (y de ese modo, la cantidad de cálculos a realizar). En la Figura 1, se puede observar cómo sobre la imagen se aplican un filtro, generando un mapa de características (nueva imagen).

Luego de cada capa de convolución es utilizada una capa de submuestreo tradicionalmente sin traslape. Es decir, el filtro utilizado se mueve una cantidad de espacios igual a su tamaño (en la dirección del movimiento) después de ser utilizado.

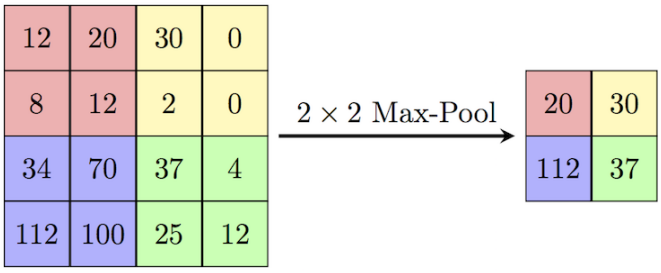

Figura 2: Submuestreo mediante max-pooling en 2 dimensiones (obtenida desde [1])

De este modo se reduce a una fracción la cantidad de datos que fluyen desde la capa de convolución a la siguiente, reduciendo así la resolución de la imagen procesada, evitando así problemas de dependencia espacial (de posición) con respecto a los objetos o símbolos a identificar. Al reducir de este modo la cantidad de los datos que fluyen desde capas iniciales a capas superiores, se refuerzan las decisiones basada en representaciones de alto nivel por sobre las realizadas en base a datos en bruto, alcanzando una mayor independencia de variaciones insustanciales en ellos.

3.2. Overfitting

El overfitting consiste en el sobreajuste de un modelo de machine learning a los ejemplos utilizados en el entrenamiento, el que perjudica el rendimiento de los modelos a otros ejemplos (como los utilizados en el testing del modelo). A esto se le llama mala generalización (capacidad de responder adecuadamente a nuevos ejemplos a partir de otros anteriores). Con respecto al Deep Learning, el overfitting tiende a ser un problema importante debido a que el tamaño de las redes (y por lo

tanto, la cantidad de neuronas) utilizadas deja mucho espacio al desarrollo de inadecuados criterios de decisión basados en datos de bajo nivel, de alta dependencia del conjunto de ejemplos utilizado. Esto ocurre en CNNs a pesar de la utilización de capas de convolución y de submuestreo (que lidian en parte con este problema). Por ello es que recurre a mecanismos adicionales para evitar la coadaptación de las neuronas a patrones de bajo nivel y de alta variabilidad y estimular las decisiones basadas en criterios de alto nivel, adecuados a una mejor generalización.

Uno de estos métodos es el Dropout, el que consiste en, durante el entrenamiento, eliminar con cierta probabilidad la salida de una unidad, obligando a las neuronas a desarrollar una menor codependencia. Si durante el entrenamiento, la salida de es mantenida con una probabilidad p, durante el testing, cada peso se multiplica por p, de modo que la salida esperada sea igual a la esperada en el entrenamiento.

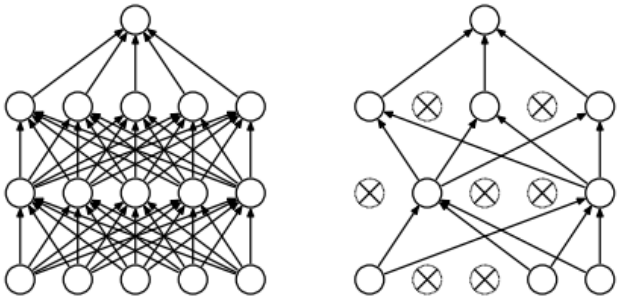

Figura 3: Ejemplo dropout (obtenida desde [3])

Por otro lado, logra aproximar eficientemente el resultado promedio de un número exponencial de redes más delgadas (utilizando en el testing pesos más pequeños), obteniendo una mejora sustancial en la generalización. Sin embargo, el entrenamiento tiende entre 2 y 3 veces más extenso.

3.3. Vanishing Gradient

El gradiente para una capa está dado por:

$$\Sigma'(s^l)(w^{l+1})^T\Sigma'(s^{l+1})(w^{l+2})^T... \ \Sigma'(s^L)\nabla_z C$$

Donde Σ' es matriz diagonal de funciones de activación derivadas σ', s^l los productos punto en la capa l (de L capas), w^l la matriz de pesos para la capa l, y $\nabla_z C$ el gradiente de la función de costo C con respecto a la salida z.

Cuando la función de activación es sigmoidal, la atenuación generada por sus derivadas sucesivas es exponencial en la cantidad de capas atravesadas por el algoritmo de backpropagation. Esto se debe a que la derivada de las funciones sigmoidales logística y de tangente hiperbólica tienen la siguiente forma.

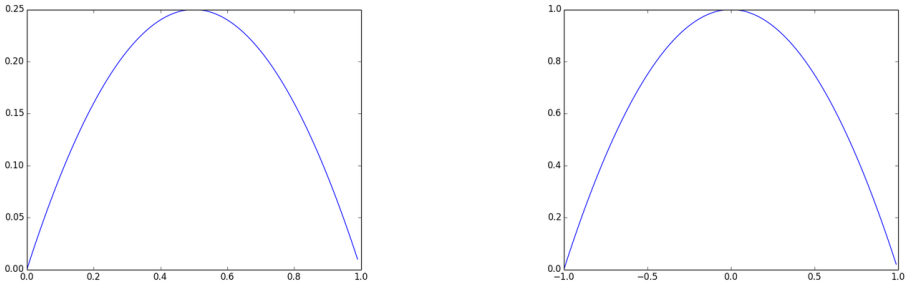

Figura 4: Derivadas función logística y tangente hiperbólica (obtenida desde [4])

Como se puede apreciar su valor está en (0,1), de modo que generan un efecto de atenuación sobre la magnitud del gradiente. Sin embargo la función de activación lineal rectificada (ReLu) no posee este problema:

$$ReLu(s) = max(s, 0)$$

Función de activación lineal rectificada (ReLu)

Es fácil ver que la derivada de esta función es 1 para $s > 0$ y 0 para $s < 0$, por lo que no genera atenuación exponencial al ser utilizada en capas sucesivas. Aun así, los pesos igualmente podrían generar un desvanecimiento al estar entre -1 y 1, lo que en general es cierto a menos que haya un overfitting considerable [5], por lo que igualmente tenderá a haber un desvanecimiento a pesar de que éste sea considerablemente menor.

4. Descripción de experimentos

Cuatro experimentos serán realizados relacionados con técnicas de reducción del overfitting, el tipo y cantidad de capas y el contenido de los ejemplos utilizados en el entrenamiento de una red neuronal profunda. Como ya se ha mencionado en la introducción, se utilizará la base de datos CIFAR-10, con imágenes de 10 clases distintas por lo que se hará énfasis en las CNNs, dado que estas han resultado ser considerablemente exitosas en el reconocimiento de imágenes.

Adicionalmente, ciertos decisiones e hiperparámetros han sido elegidos. En primer lugar, CIFAR-10 se compone de 60 mil imágenes, de las cuales 50 mil son dedicadas a entrenamiento y 10 mil a testing. En nuestro caso, hemos seleccionado 5 mil de las 50 mil de entrenamiento para validación. Por otro lado, se ha utilizado un tamaño de minibatch igual a 64, zero-padding (ver marco teórico), algoritmo de optimización Adam, función de costo de entropía cruzada, función de activación ReLu en las capas intermedias y Softmax en la última capa.

Con respecto a las capas utilizadas en los distintos experimentos, se mantiene la misma capa de inicial para todos ellos, teniendo un tamaño igual a 3072, debido a las dimensiones de las imágenes utilizadas (32x32x3, el último tres debido a las tres componentes de color RGB). Sin embargo, se tienen distintas configuraciones con respecto a las demás capas. En el primer experimento, se realizaron 15 épocas durante el entrenamiento para los dos casos considerados: 0.5 y 1.0 de probabilidad de mantención de una neurona. En éste, se mantienen dos capas convolucionales, cada una con su respectiva capa de submuestreo (max-pooling en particular) y dos capas finales FC (totalmente conectadas). La primera capa convolucional utiliza 32 filtros 5x5x3, cada uno con sus sesgos asociados ((5x5x3+1)x32=2432 parámetros, entrada de 5x5x3x32x32x32=2457600 en bruto y 32x32x3=3072 distintas y 32x32x32=32768 salidas distintas, una por unidad), generando 32 mapas de características de 32x32 valores. Luego su capa de max-pooling utiliza matrices 2x2, dividiendo a la mitad el alto y ancho de los mapas de características (entrada de 32x32x32=32768 sin traslape y salida de 16x16x32=8192 distintas, una por unidad de submuestreo). La segunda por otro lado utiliza 32 filtros de 5x5x32 ((5x5x32+1)x32=25632 parámetros, vs entrada de 5x5x32x16x16x32=6553600 en bruto y 16x16x32=8192 distintas y 32x16x16=8192 salidas/unidades distintas), generando otros 32 mapas de características de 16x16, cada uno basado en los 32 mapas de características de 16x16 de entrada combinados. Luego su capa de max-pooling utilizando matrices de 2x2 se encarga nuevamente de dividir el ancho y el alto por 2, generando 32 mapas de características de 8x8 (16x16x32=8192 entradas y 8x8x32 salidas/unidades distintas). Finalmente se tienen dos capas totalmente conectadas. La primera con 256 unidades, conecta cada una de ellas con cada una de los valores de los 32 mapas de características entrantes, utilizando para ello (32x8x8+1)x256=524544 parámetros (vs entrada de 32x8x8x256=524288 en bruto y 2048 distintas con solo 256 salidas/unidades distintas). La segunda conecta estas 256 con cada una de las 10, utilizando (256+1)x10=2570 parámetros (con 2560 entradas), y generando las 10 salidas correspondiente a cada una de las clases posibles. Con esto se tiene un total de 555178 parámetros, teniendo el mayor porcentaje claramente en la primera capa totalmente conectada (94.5 %), permitiéndonos reconocer que el principal trabajo de

clasificación ocurre en este lugar y que el trabajo realizado por las capas convolucionales estaría principalmente asociado a la extracción de características de alto nivel y descarte de datos de menor relevancia. En contraste con esto, la primera y segunda capas convolucionales suman un total de conexiones de entrada de 6553600+2457600=9011200, de un total de la red de 9579008, es decir, un 94.2 % de las conexiones por lo que la cantidad de parámetros utilizados en cierto modo es una ilusión del trabajo de extracción de información realizado. También es evidencia de que la extracción de características clave no requiere una gran cantidad de parámetros, pero (en el caso de uso de capas convolucionales) sí de conexiones.

Posteriormente, en el segundo experimento se verifica el efecto de la variación del número de capas convolucionales. En él se prueban dos configuraciones adicionales, y se comparan con la ya utilizada. Para cada una de las tres configuraciones se realizan 10 épocas durante el entrenamiento. En primer lugar, se observa el efecto de utilizar solo una capa convolucional (y su capa de max-pooling). En tal situación, la salida de la primera (y única) capa de max-pooling pasa directamente a la primera capa FC, no habiéndose reducido el tamaño de los mapas de características solo hasta 16x16, de modo que la cantidad de parámetros utilizados por la primera capa FC aumenta a (32x16x16+1)x256=2097408 (vs las mismas 16x16x32=8192 entradas distintas que recibía la segunda capa convolucional), manteniéndose el tamaño de su salida igual a 256 y la forma y salida de la segunda capa FC. Con ello se tendría un total de 2097408+2432+2570=2102410 parámetros. En el segundo caso la tercera capa convolucional recibiría una igual cantidad de mapas de características y también generaría 32 otros, de modo que se tendrían 32x(5x5x32+1)=25632 parámetros vs 8x8x32=2048 entradas y 32x8x8=2048 salidas. La capa de max-pooling en este caso reduciría el ancho y el alto a 4x4, de modo que la capa primera capa FC tendría (32x4x4+1)x256=131328 parámetros, una entrada de 4x4x32=512 y una salida de 256. La red tendría entonces un total de 131328+25632+25632+2432=185024 parámetros, menos de 10 veces menos parámetros que la red previamente considerada.

En el tercer experimento se compara la red convolucional con un perceptrón multicapa (MLP, red neuronal convencional). Para ello, se utiliza un MLP con 1, 2 y 3 capas totalmente conectadas (más la capa de entrada que se mantiene). En el caso de uso de una capa FC, esta tuvo 10 unidades, cada una conectada a cada uno de de los 32x32x3=3072 pixeles en la capa de entrada, sumando (32x32x3+1)x10=30730 parámetros (30720 conexiones/entradas en bruto) y

las 10 salidas por cada clase. En el caso de dos FCs, la primera utilizó 100 unidades, teniendo las mismas 3072 entradas distintas, pero (32x32x3+1)x100=307300 parámetros (307200 conexiones/entradas en bruto), y modificando las entradas de la última capa a 100 y su número de parámetros a (100+1)x10=1010 (1000 conexiones/entradas en bruto), sumando un total de 308310 parámetros (308200 conexiones totales). Por último, en el caso de tres FCs, la primera de ellas utilizó 250 unidades, 3072 entradas, (32x32x3+1)x250=768250 parámetros (768000 conexiones/entradas en bruto) y 250 salidas. La segunda las mismas 100 unidades, 250 entradas distintas (307200 conexiones), (250+1)x100=25100 parámetros y 100 salidas. La última, nuevamente 100 entradas, (100+1)x10=1010 parámetros (1000 conexiones) y 10 salidas. Un total de 76825+25100+1010=794360 parámetros (768000+307200+1000=1076200 conexiones totales). El último experimento consistió en utilizar una mayor cantidad de imágenes, agregando a las iniciales reflexiones horizontales y pequeñas variaciones de color, sin modificar las capas de la red neuronal utilizada en el primer experimento y realizando 15 épocas.

5. Resultados

5.1. Dropout

	Tiempo (min)	Training (exactitud)	Testing (exactitud)
$p = 0.5$	35.8	90.6 %	72.0 %
$p = 1.0$	36.1	100 %	69.2 %

Tabla 1: Resultados Dropout

5.2. Capas convolucionales

	Tiempo (min)	Training (exactitud)	Testing (exactitud)
1 Conv.	11.1	67.2 %	64.8 %
2 Conv.	20.3	84.4 %	71.0 %
3 Conv.	22.9	85.9 %	72.5 %

Tabla 2: Resultados capas convolucionales

5.3. MLP

	Tiempo (min)	Training (exactitud)	Testing (exactitud)
2 Capas	3.1	43.8 %	35.1 %
3 Capas	8.2	40.6 %	26.7 %
4 Capas	20.4	79.7 %	49.9 %

Tabla 3: Resultados entrenamiento MLP

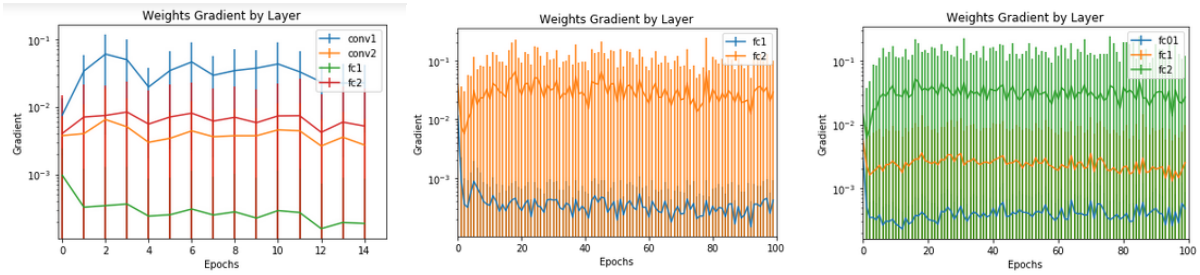

Figura 5: Gradiente en pesos por época para CNN de 2 convoluciones y MLP de dos 2 y 3 FCs

	conv1	conv2	fc01	fc1	fc2
2-conv	$13.3x10^{-3}$	$12.5x10^{-4}$	-	$19.1x10^{-7}$	$39.0x10^{-5}$
2-mlp	-	-	-	$32.5x10^{-7}$	$10.0x10^{-4}$
3-mlp	-	-	$13.0x10^{-7}$	$39.8x10^{-6}$	$10.0x10^{-4}$

Tabla 4: (N° de parámetros)$^{-1}$ en red de 2 capas convolucionales y en mlp de 2 y 3 capas FC

5.4. Augmentation

Tiempo (min)	Training (exactitud)	Testing (exactitud)
34.9	75.0 %	73.9 %

Tabla 5: Resultados augmentation

6. Análisis

6.1. Dropout

En la Tabla 1 se puede observar que el tiempo requerido por las redes es practicamente el mismo, mientras que la exactitud de entrenamiento de la red sin dropout ($p = 1.0$) es máxima, mientras que la con dropout, a pesar de poseer una mayor exactitud en el testing, es 10 % inferior. Esto se explica adecuadamente por una diferencia en el overfitting de las dos redes, en la que la red con dropout ($p = 0.5$) lograría enfrentarlo más adecuadamente. Por ello, desde ahora en adelante utilizaremos la red con dropout.

6.2. Capas

En la Tabla 2 se puede observar que ls red con 3 capas convolucionales logra desmepeñarse mejor tanto en el training (85.9 % vs 84.5 %) como en el testing (72.5 % vs 71.0 %), por lo que a pesar de que se demore un tanto más (22.9 mins vs 20.3 mis) que la de 2 convoluciones. Este resultado es interesante, pues esta última red posee (185024), aproximadamente 3 veces menos parámetros que los utilizados en la de dos convoluciones (555178). Probablemente esto se deba a que las características clave de alto nivel requerirían una resolución mucho menor a la utilizada por la red de dos convoluciones para ser correctamente expresadas, y que el descarte de datos de bajo nivel es importante y muy efectivo en la búsqueda de una clasificación exitosa. Teniendo esto en cuenta al observar los resultados de la red con una convolución, podemos entender por qué ella tenga un considerablemente peor rendimiento que las anteriores tanto en training como en testing (67.2 %vs 64.8 %), a pesar de que esta posea más de 10 veces los parámetros utilizados por la red de 2 convoluciones (2102410). Este resultado es contundente y revelador, por ello se tomará en cuenta como uno de los más valiosos del presente informe.

6.3. MLP

En la Tabla 3 se puede observar claramente los problemas de overfitting que presentan los tres MLP, al no poseer capas convolucionales que sean capaces de informar sobre las invarianzas existentes en el reconocimiento y clasificación de imágenes. Adicionalmente las exactitudes de testing se ven opacadas frente a cualquiera de las obtenidas al utilizar capas convolucionales. Esto resulta ser un nuevo resultado revelador y de mucha importancia, por lo que también será considerado como

uno de los más valiosos.

Por otro lado, los tiempos de los MLPs con 1 y 2 capas FC son considerablemente más bajos. Esto se logra explicar adecuadamente no en base a la cantidad de parámetros, sino que a la cantidad de conexiones utilizadas por las MLP frente a las redes convolucionales: Como se mostró en la descripción de este experimento, la primera MLP posee 30730 conexiones y la segunda 308200, 293 y 29 veces menos que las 9011200 requeridas por la red convolucional. Si ahora consideramos que en el caso de las MLPs fueron utilizadas 100 épocas (10 veces más que las utilizadas en por la segunda red convolucional en el segundo experimento), tiene sentido que esas MLPs tarden 3.1 y 8.2 minutos (6.5 y 2.5 veces menos que ella). Por último, la tercera (MLP de 3 FCs) tarda aproximadamente lo mismo (20.4 vs 20.3), utilizando 768000 conexiones realizando 10 veces más épocas que la CNN mencionada. Es claro que otras variables afectarían al tiempo requerido, pero teniendo en cuenta el tamaño y la complejidad de las redes estudiadas, la cantidad de conexiones resulta ser un estimador sorprendentemente acertado. Como curiosidad, utilizando arbitrariamente solo el rendimiento de la tercera MLP, se estima un costo de 1 minuto por época de una red de 3350000 conexiones en CIFAR-10 utilizando una CPU Intel Core i5 de séptima genaración.

Por último, con respecto a los gradientes percibidos por las capas se puede observar en la Figura 5 que con respecto a las capas FCs, el gradiente disminuye a medida que nos alejamos de la capa de salida en todas las redes. Sin embargo, ello parece no ocurrir en el caso de redes convolucionales. La verdadera explicación surge cuando consideramos el inverso multiplicativo de la cantidad de parámetros utilizados en la Tabla 4. La relación es evidente, pues los inversos mayores se mapean a gradientes mayores, los inversos menores a gradientes menores y los similares a gradientes similares. La diferencia de ordenes de magnitud posiblemente se solucionaría mediante la multiplicación de una constante adecuada, pero al observar más detalladamente ello parece ser poco probable, debido a que existen saltos de ordenes de magnitud mayores en los inversos que en los gradientes. Esto podría deberse a un relación no lineal (cuadrática posiblemente) además de inversa en base a la cantidad de parámetros. Sin embargo, la relación descubierta entre parámetros y gradientes fue considerada valiosa, sobre todo debido a que se pone en evidencia debido al uso de la función de activación ReLu, pues de otro modo hubiera sido opacada en gran parte por el desvanecimiento del gradiente generado por una función de activación sigmoidal.

6.4. Augmentation

La exactitud en testing de la red entrenada con augmentation en Tabla 5 es superior a todas las obtenidas hasta ahora. Más aun, la diferencia entre la exactitud de training y de testing es muy baja. Esto pone en evidencia la efectividad de la técnica de augmentation en la disminución del overfitting.

7. Conclusiones

Los resultados principales que hemos obtenido son los que muestran la importancia de las capas convolucionales debido a su capacidad de aportar invarianzas en el reconocimiento de imágenes, manejando así efectivamente el overfitting. En particular, las diferencias con respecto a la cantidad de parámetros utilizados por la red con una capa convolucional y la con tres en el segundo experimento es de un orden de magnitud (2102410 vs 185024), sin embargo es la segunda la que obtiene un considerablemente mejor desempeño (64.8 % vs 72.5 % en testing). Luego, este resultado es reforzado por el obtenido en el tercer experimento con respecto a los MLPs, los que presentan overfittings muy grandes: 29.8 % de diferencia en el MPL de 3 capas FC. Esta última es la de mayor exactitud en testing, y posee una diferencia de 22.6 % con respecto a la red con 3 capas convolucionales (49.9 % vs 72.5 %), pero posee 4 veces más parámetros (794360 vs 185024).

En segundo lugar, están los resultados que nos muestran la capacidad de las técnicas de Dropout y de Augmentation para reducir aun más el overfitting. Primero el Dropout llevó a la red de dos capas convolucionales a desde 69.2 % a 72.0 % en testing, con un aumento del 2.8 %. Luego, Augmentation la llevó desde 72.0 % a 73.9 %, un aumento similar de 2.9 %.

Finalmente, están los resultados que nos muestran relaciones entre parámetros y variables de desempeño. Por un lado, se observó una relación aproximadamente lineal entre la cantidad de conexiones y el tiempo que tarda una red en ser entrenada, insinuando la posibilidad de que este sería uno, sino el más acertado parámetro estimador de tal variable de desempeño. Por el otro, se observó en los gráficos de los gradientes percibidos por capa dependían de manera sustancial de la cantidad de parámetros en tal capa, informando sobre la relevancia de mantener controlada la cantidad de parámetros por capa.

Todos estos resultados pueden ser hasta cierto punto utilizados en otros dominios del Deep Learning. En particular, está presente la incógnita sobre si sería posible adaptar la poderosa capacidad de las capas convolucionales de obtener características de alto nivel en un dominio distinto, como en el procesamiento de lenguaje natural.

8. Referencias

- Convolución [1]:
  https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html
- Max Pooling [2]:
  http://cs231n.github.io/convolutional-networks/
- Dropout [3]:
  https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/
- Derivada sigmoide [4]:
  https://www.quora.com/What-is-the-vanishing-gradient-problem
- Pequeños pesos [5]:
  https://papers.nips.cc/paper/563-a-simple-weight-decay-can-improve-generalization.pdf