UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FíSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

EVALUACIÓN DE LA INCORPORACIÓN DE CARACTERÍSTICAS EXTRAÍDAS POR MÉTODOS CONVENCIONALES EN REDES NEURONALES CONVOLUCIONALES PARA EL PROBLEMA DE CLASIFICACIÓN DE CIFAR10

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

CRISTÓBAL FRANCISCO QUEZADA SUBIABRE

PROFESOR GUÍA:
CLAUDIO PÉREZ FLORES

MIEMBROS DE LA COMISIÓN:
ANDRÉS CABA RUTTE
PABLO ESTÉVEZ VALENCIA

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERA CIVIL ELÉCTRICA
POR: CRISTÓBAL FRANCISCO QUEZADA SUBIABRE
FECHA: 2020
PROF. GUÍA: CLAUDIO PÉREZ FLORES

EVALUACIÓN DE LA INCORPORACIÓN DE CARACTERÍSTICAS EXTRAÍDAS POR MÉTODOS CONVENCIONALES EN REDES NEURONALES CONVOLUCIONALES PARA EL PROBLEMA DE CLASIFICACIÓN DE CIFAR10

En este trabajo se presenta una metodología para la incorporación de características *hand-crafted* en redes neuronales convolucionales, específicamente características Gabor. Estas características se incorporan a las características generadas por la red y además son sintonizadas de acuerdo a la tarea a resolver.

Se modifican 2 arquitecturas, las redes DenseNet y ResNet. Se utilizan versiones pequeñas de estas redes, con 20 capas de profundidad (DenseNet20 y ResNet20) y se evalúa su desempeño en la tarea de clasificación del conjunto de datos CIFAR10. Se realizan experimentos utilizando un conjunto de datos normal, con *data augmentation* y *cutout*. Las características son agregadas de manera tal que no modifiquen la estructura de la red, manteniendo el número de mapas de salidas y dimensiones en cada una de la capas convolucionales. Luego, la mejor configuración es utilizada en versiones más profundas de las redes DenseNet y ResNet, con 100 y 110 capas, respectivamente.

Se logra aumentar el desempeño utilizando esta metodología en una variante de estas redes, aumentando de 84.96 % a 85,83 %.

Agradecimientos

Agradezco a mis padres por el apoyo durante todo este tiempo. Gracias por ser un pilar fundamental durante todas mis etapas de estudio y vida, por apoyarme en cada uno de los proyecto que intenté emprender. Y por sobre todo por brindarme un hogar donde prima el amor y la felicidad.

Agradezco a mi padre por su apoyo y motivación constante. Una historia que me gustaría rememorar y que forja el inicio en mi camino en la matemáticas que luego me llevó a terminar estudiando ingeniería. Es cuándo estaba en la enseñanza media y le pedí a mi papá que por favor me comprara un libro de matemáticas. Sin mucho entender de que se trataba, recorrió todo San Diego hasta encontró ese libro, Álgebra Superior de Hall and Knight. Y así, otros mucho ejemplos, donde siempre intento que tuviera todo lo que necesitaba.

Agradezco a mi madre por inculcar en mí valores que creo que me han permitido ser una buena persona y enfrentar mi etapa universitaria sin mayores problemas. Unos de ellos es el valor de comprometerse con lo que haces y nunca faltar a tu palabra.

Durante mi etapa universitaria son muchos las personas que pude conocer, con varias de las cuáles he forjado una linda amistad. Recuerdo con mucho cariño todos los grupos que formé con Javier Witto y Mike Henderson, personas espectaculares y muy inteligentes. También, aquellas múltiples veces que pude trabajar con otro par de amigos, Bastián Mendoza y Francisco Nicolai. No me gustaría dejar fuera a mis amigos Iñaki Cubillos y Fernando Montecinos.

También, no me gustaría dejar pasar una mención a mi primo no sanguíneo Vicente Quezada. A quién conocí el primer día de Universidad y compartimos amor por el mismo deporte, el atletismo. En donde junto al gran Emilio Molina entrenamos arduamente en los comienzo de nuestra etapa universitaria.

Al final de mi etapa universitaria pude ir de intercambio a Portugal, donde conocí personas espectaculares, con las que espero mantener contacto pese a la enorme distancia.

Agradezco al profesor Claudio Peréz por su preocupación y rigurosidad al desarrollar esta memoria.

Finalmente, se agradece al proyecto FONDECYT 11191610 que tiene dentro de sus objetivos la investigación realizada en esta memoria de título. El proyecto financia la investigación desarrollada dentro del Departamento de Ingeniería Eléctrica de la Universidad de Chile.

Tabla de Contenido

1. Introducción **1**
1.1. Objetivos . . . 2
1.1.1. Objetivos generales . . . 2
1.1.2. Objetivos específicos . . . 2
1.2. Estructura de la memoria . . . 2

2. Marco Teórico **4**
2.1. Métodos de extracción de características . . . 4
2.1.1. Métodos clásicos . . . 4
2.1.1.1. Filtrado . . . 5
2.1.2. Métodos basados en aprendizaje profundo . . . 7
2.1.2.1. Capas convolucionales . . . 8
2.1.2.2. Capa de pooling . . . 10
2.1.2.3. Capa *fully connected* . . . 11
2.1.2.4. Regularización . . . 11
2.1.2.5. Entrenamiento red neuronal convolucional . . . 11
2.2. Redes neuronales convolucionales . . . 13
2.2.1. Deep residual Networks (ResNet) . . . 13
2.2.2. Densely connected convolutional networks (DenseNet) . . . 15
2.3. Trabajos relacionados . . . 18
2.3.1. Estrategias para combinar características handcrafted con características CNN . . . 18
2.3.2. Desempeño de redes neuronales convolucionales en CIFAR10 . . . 19
2.4. Bases de datos . . . 21
2.4.1. CIFAR10 . . . 21

3. Metodología **22**
3.1. Incorporación de características . . . 23
3.1.1. Sintonización características . . . 23
3.1.1.1. Padding replicado . . . 24
3.1.1.2. Sintonización parámetros con PSO . . . 24
3.1.2. Incorporación de características a la red . . . 25
3.1.2.1. Sintonización de filtros en la red . . . 26
3.1.2.2. Filtros convencionales . . . 26
3.2. Redes neuronales convolucionales . . . 27
3.2.1. DenseNet . . . 27

3.2.2. ResNet . . . 30
3.3. Conjunto de datos . . . 32
3.3.1. CIFAR10 . . . 32
3.3.2. Data augmetation . . . 33
3.4. Entrenamiento . . . 34

4. Resultados **35**
4.1. Extracción de características . . . 35
4.1.1. Filtros Gabor . . . 35
4.2. DenseNet . . . 36
4.2.1. Determinar factor de compresión . . . 36
4.2.2. Incorporación características . . . 36
4.2.3. DenseNet100 . . . 37
4.3. ResNet . . . 38
4.3.1. Incorporación de características . . . 38
4.3.2. ResNet110 . . . 38

5. Discusión **40**
5.1. DenseNet . . . 40
5.1.1. DenseNet20 . . . 40
5.1.2. DenseNet100 . . . 41
5.2. ResNet . . . 41
5.2.1. ResNet20 . . . 41
5.2.2. ResNet 110 . . . 42
5.3. Comparación DenseNet y ResNet . . . 42

6. Conclusión **43**

Bibliografía **43**

Anexo A. Tiempos de entrenamiento **46**
A.1. DenseNet . . . 46
A.2. ResNet . . . 47

Capítulo 1

Introducción

En las etapas tempranas del procesamiento de imágenes, las características *handcrafted* como características Haar-like, HOG, LBP y SIFT fueron muy usadas en la mayoría de problemas de clasificación. Estas características, describen las imágenes a partir de diferentes aspectos, como borde y texturas, pero tienen una capacidad de descripción limitada. Hace algunos años, aparecieron en el campo de la visión por computador las redes neuronales convolucionales (CNN de sus siglas en inglés), las cuáles aprenden las características a partir de grandes bases de datos y han logrado un desempeño increíble en diferentes problemas de visión computacional, excediendo por mucho el desempeño logrado por las características *handcrafted*. Pero hay un problema, las características aprendidas por las CNN son difíciles de interpretar y entender, por lo que no se sabe que aspecto están representado [1].

Dado el proceso de entrenamiento de las CNN, éstas aprenden características automáticamente. Controlar la composición de las características de una CNN es también un problema difícil. Algunos investigadores buscan entender la interpretabilidad de las características aprendidas por la red [2], y otros estudian la relación entre las características *handcrafted* y las características CNN. En [3] se muestra que las características *handcrafted* pueden proveer información complementaria a las características de alto nivel extraídas por una CNN para mejorar el desempeño en la tarea de detectar regiones más sobresalientes de una imagen. En [4] se compara SIFT y las CNNs, encontrando que comparten características comunes. En [1] se exploran diferentes métodos de fusión de características *handcrafted* los cuáles consisten en: (a) Combinar directamente las características *handcrafted* con las características extraídas por una CNN y entrenar el mismo clasificador de la CNN a partir de esta combinación (b) Combinar las características handcrafted codificadas y realizar un entrenamiento conjunto a partir de esta combinación. Finalmente en [5] se reemplazan en las primeras capas de una CNN algunos kernels de filtros por filtros Gabor con el objetivos de reducir el set de características a ser aprendidas y aumentar la velocidad de entrenamiento de la red, lográndose mejorar en el desempeño marginales (0.1 %).

En esta memoria se propone una metodología de trabajo para incorporar características extraídas con métodos clásicos en redes neuronales convolucionales para resolver el problema de clasificación de CIFAR10. El objetivo general es evaluar si es posible mejorar el desempeño de redes neuronales convolucionales incorporando características *handcrafted* en sus estruc-

tura, específicamente en redes que se logran resultados estado del arte en este problema de clasificación, como lo son DenseNet y ResNet.

El método consiste en incorporar a la estructura de redes neuronales convolucionales características extraídas con filtros Gabor desde las imágenes del conjunto de datos. Estas características generan un bloque de características que es incorporado a la red, sin cambiar las dimensiones de los mapas de característica de la red original.

Para el evaluar el método si utiliza el conjunto de datos CIFAR10.

Dado que las características *handcrafted* fueron diseñadas para describir el contenido de una imagen a partir de aspectos específicos, estas podrían proveer información complementaria a la CNNs en las tareas de clasificación, lo que permitiría lograr un mejor desempeño.

1.1. Objetivos

1.1.1. Objetivos generales

El objetivo general de esta memoria es evaluar con una metodología si es posible mejorar el desempeño de CNNs en la tarea de clasificación de CIFAR10 incorporando características *handcracted* en su estructura.

1.1.2. Objetivos específicos

Los objetivos específicos de esta memoria son:

- Evaluar si las características *handcrafted* extraídas con filtros Gabor permiten mejorar el desempeño en la tarea de clasificación CIFAR10 al ser incorporadas a las redes neuronales ResNet y DenseNet.
- Evaluar los efectos para las redes DenseNet y ResNet de utilizar simultáneamente en el entrenamiento de las CNN *data augmentation* e incorporar características *handcrafted*.

1.2. Estructura de la memoria

El presente informe se divide en en 5 capítulos y se estructura de la siguiente manera.

En el Capítulo 1 se detalla la motivación para realizar el proyecto y se establecen los alcances, objetivos y resultados esperados.

En el capítulo 2 se presenta el marco teórico. El cual incluye una sección para la descripción de la estructura y entrenamiento de redes neuronales. Además, se presenta una segunda

sección donde se presentan algunos de los métodos de extracción de características utilizando métodos clásicos y algunas de las redes utilizadas en los métodos basados en aprendizaje profundo. Además,en la sección 2.3.1 se entregan los antecedentes que existen relativos a la incorporación de características de métodos clásicos en la estructura de las redes neuronales convolucionales.

En el capítulo 3 se describe la metodología. Esta contiene las redes neuronales que se utilizaron y como se incorporaron las características.

En el capítulo 4 se muestran los resultados de cada una de las variaciones y diferentes metodologías de entrenamientos en las redes neuronales convolucionales.

Finalmente, en el capítulo 5 se presenta la discusión de los resultados y en capítulo 6 se presentan las conclusiones del trabajo.

Capítulo 2

Marco Teórico

A continuación, se presenta el marco teórico y los antecedentes relacionados a la metodología a desarrollar. Dado que se propone modificar la estructura de la red convolucional, en la sección 2.1.2 se muestra la estructura general de las redes convolucionales. También, se presentan los métodos de extracción de características, divididos en dos aristas: métodos clásicos y métodos de aprendizaje profundo basados en redes neuronales. Además, se describe en detalle la arquitectura de las redes a utilizar: ResNet y DenseNet.

2.1. Métodos de extracción de características

2.1.1. Métodos clásicos

Los problemas de reconocimiento de imágenes son desafiantes, como por ejemplo los de reconocimiento de rostros, debido a que se debe lidiar con muchos factores que dificultan que los algoritmos logren un alto desempeño, como: iluminación, escalas, poses, y oclusión. En [6], se describe la metodología utilizada para la extracción de características de imágenes, la cuál consiste en 4 puntos principales: filtrado , codificado, agrupación espacial (*spatial pooling*) y representación holística. En la figura 2.1 se presenta un diagrama de la metodología.

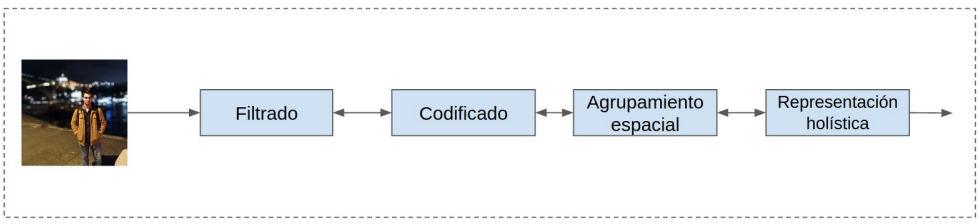


Figura 2.1: Metodología para extracción de características de rostros.(1)Filtrado, (2)Codificado, (3)Agrupamiento espacial y (4) Representación holística.

En [6], se describe la metodología con los siguientes etapas:

1. Filtrado: Para generar características robustas de una imagen, es útil convolucionar la imagen con un filtro específico, ya sea utilizando filtros convencionales, o filtros aprendidos desde conjuntos de datos. El filtrado puede ser multinivel para formar un conjunto de filtros en cascada [6].
2. Codificado: La salida de la etapa de codificado puede ser un histograma o un vector de características. Muchos trabajos codifican características locales solo utilizando histogramas, en cambio otros utilizan técnicas más complejas como diccionarios. Los diccionarios pueden ser generados por *hard clustering* o *soft clustering* [6].
3. Agrupamiento espacial(*spatial pooling*): Puede ser visto como un modo para comprimir el vector codificado y lograr una representación espacial de la imagen para formar una representación holística final. Hay dos métodos clásicos de agrupamiento: El primero es, agrupamiento promedio, que conserva la respuesta promedio. El segundo es, agrupamiento máximo, que conserva la respuesta máxima. La división en bloques puede ser vista como una característica del agrupamiento. Luego, características extraídas de diferentes bloques son unidas.[6]

   La división en bloques es beneficiosa, ya que las características de una imagen está basada en características ubicadas en diferentes áreas(Por ejemplo, en un problema de reconocimiento de rostros: ojos, nariz y boca). Sin la apropiada división, las características locales extraídas de diferentes partes se mezclan y no se permite la correcta discriminación [6].
4. Representación holística: Se trata sobre reducción de dimensionalidad y fusión de características, para generar una representación final de las características. La concatenación de características basadas en bloque puede producir vector características de muy alta dimensionalidad y por lo tanto se requiere de apropiada reducción de dimensionalidad. Además, para combinar múltiples características, se debe recurrir a técnicas de fusión de características [6].

2.1.1.1. Filtrado

Existen numerosos métodos de filtrado, pero en la literatura se identifican principalmente los siguientes:

1. **Filtros Gabor**: Los filtros Gabor son ampliamente usados en procesamiento de imágenes ya que capturan características locales correspondientes a frecuencia espacial y localización espacial. La definición de un kernel Gabor es: $\phi_{\mu,\nu} = \frac{k_{\mu,\nu}^2}{\sigma^2} exp(\frac{k_{\mu,\nu}^2 z^2}{2\sigma^2})[exp(\mathrm{i}k_{\mu,\nu}z) - exp(-\frac{\sigma^2}{2})]$, donde μ y ν son orientación y escala de los kernels Gabor respectivamente, y $z = (x, y)$ define la orientación espacial. Vector de onda es definido como $k_{\mu,\nu} = k_v \mathrm{e}^{\mathrm{i}\phi_m \mu}$ donde $k_v = k_{max}/f^u$ y $\phi_\mu = \pi\mu/8$ con k_{max} la máxima frecuencia y f el factor de espaciamiento entre kernels en el dominio de la frecuencia. A menudo se utiliza $\nu \in \{0, 1, 2, 3, 4\}$ y $\mu \in \{0, 1, 2, 3, 4, 5, 6, 7$, resultando 40 ondas Gabor de 5 escalas y 8 orientaciones, con $\sigma = 2\pi$, $k_{max} = \pi/2$ y $f = \sqrt{2}$. Una representación de Gabor es obtenida convolucionando la imagen de entrada con un conjunto de filtros Gabor de varias escalas y orientaciones. El filtrado con filtros Gabor es computacionalmente costoso y la salida

es de alta dimensionalidad por lo que un método de reducción de dimensionalidad es esencial [7].

Algunas prácticas son: concatenar las magnitudes de todas las orientaciones y escalas en una imagen de características Gabor y aplicar reductores de dimensionalidad con *Enhanced Fisher Linear Discriminat Classifier* [8]; seleccionar las características más discriminativas con random forest [9]; construir una imagen Gabor para cada kernel Gabor y luego aplicar *clustering* a cada imagen para construir un *codebook* para codificar patrones por medio de histograma[10].

2. **Patrones binarios locales (LBP por sus siglas en inglés)**: El proceso computacional de LBP es simple y eficiente: Cada píxel de la imagen es etiquetado por medio de una comparación con los píxeles vecinos, esta comparación es representada por un número binario. Se puede utilizar la notación (P,R) para los parámetros de LBP. Estos parámetros denotan el muestreo de P muestras en un círculo de radio R. El operador LBP usa un kernel de $(2R+1)\times(2R+1)$ para resumir la estructura local de la imagen. En un centro dato, (x_c, y_c), se toman los $(2R+1)\times(2R+1)$ píxeles vecinos que rodean al píxel central. Sin embargo, R es usualmente asignado como 1, que resulta en 8 píxeles vecinos. Si el valor del píxel central es mayor que el valor del píxel vecino, se marca ese píxel con un "1", de otro modo se marca con un "0". Luego, se obtiene 8 valores 0 o 1 que forma un número binario que representa un **patrón de textura local**- Finalmente, este número binario representado como un número decimal, multiplicando cada cada dígito del número binario por una potencia de 2 y sumando [6].

   Dada una imagen I y denotando i_c como el nivel de gris del píxel c de la imagen I, el operador LBP es un píxel es definido:

$$LBP_{(P.R)} = \sum_{p=0}^{P-1} s(\mathrm{i}_p - \mathrm{i}_c)2^p \tag{2.1}$$

   con $s(x) = 1$ si $x \leq 0$ 0 de otro modo.

   Para cada imagen, una histograma es computado utilizado los valores obtenidos con el operado LBP en cada píxel. Utilizando un operador LBP con 8 píxeles vecinos, el histograma debería tener $2^8 = 256$ bins, el rango del histograma es de 0 a 255. Una propiedad importante del operador LBP es la invarianza a **al cambio de fotometría monotónica** (*invariance to monotonic photometric change*) causado por las variaciones de iluminación [6].

   Aunque LBP puede ser directamente usado, hay métodos para lograr características más robustas basadas en LBP. Un ejemplo es fusión de características: celdas de imágenes son generalmente usados y los histogramas de estas celdas son concatenados [6].

3. **Histogramas de gradientes orientados (HOG):** El descriptor HOG ha mostrado ser robusto para obtener representaciones de características locales. La idea esencial del descriptor es capturar o codificar la apariencia y forma local del objeto como la distribución de la intensidad local de los gradientes [11].

   El algoritmo sigue los siguientes pasos:

   (a) Para reducir la varianza en la iluminación en las diferentes imágenes, se realiza una normalización en escala de grises, para que todas las imágenes tengan el mismo rango de intensidad.

(b) El mismo filtro [-1,0,1] es usado para calcular el gradiente horizontal $G_x(x, y)$ y el gradiente vertical $G_y(x, y)$ de cada píxel.

(c) La magnitud m y orientación θ de los gradientes $G_x(x, y)$ y $G_y(x, y)$ son computadas para cada píxel (x, y) dentro de una ventana de detección siguiendo las siguientes ecuaciones:

$$H(x, y) = \sqrt{\mathrm{i}(x, y)} \tag{2.2}$$

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y) \tag{2.3}$$

$$G_y(x, y) = H(x, y + 1) - H(x, y - 1) \tag{2.4}$$

donde $\mathrm{i}(x, y)$ es la luminancia en escala de grises y $H(x, y)$ significa la luminancia normalizada.

(d) Computar el valor de la norma (G) y la orientación (α) en cada píxel, con las siguientes ecuaciones:

$$G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \tag{2.5}$$

$$\alpha(x, y) = tan^{-1}(G_y(x, y)G_x(x, y)) \tag{2.6}$$

(e) Dividir la imagen de entrada en celdas de igual dimensión y agrupar estas celdas en bloques. Antes de calcular la característica HOG, la magnitud del gradiente es normalizado dentro del bloque.

(f) Luego de la normalización, la proporción de superposición es de 0.5 por ancho de block.

(g) Interpolación trilineal es usado por cada celda para las características de bajo nivel.

2.1.2. Métodos basados en aprendizaje profundo

Los métodos de aprendizaje profundo, como la redes neuronales convolucionales (CNN, por sus siglas en inglés), usan una cascada de múltiples capas para extracción y transformación de características. Estas capas están compuestas de filtros los cuáles son aprendidos desde los conjuntos de datos y permiten extraer características que presentan diferentes niveles de abstracción. La estructura que la red neuronal convolucional presenta es jerárquica, en donde las características que se presentan en las capas inferiores son de bajo nivel (texturas, bordes) y las que se presentan en las capas superiores presentan son más complejas y propias de la tarea a resolver (nariz respingada, ojos grandes, etc.). Estas características muestran fuerte invarianza a la pose e iluminación [12].

Una red neuronal convolucional esta compuesta por capas. Cada capa tiene el objetivo de transformar una entrada 3D a una salida 3D utilizando funciones diferenciables. Se utilizan principalmente 3 tipos de capas para construir una CNN: **capas convolucionales**, **capas de pooling** y **capas totalmente conectadas**. Todas estas capas son apiladas para formar una CNN [13].

En las redes neuronales convolucionales, la primera capas presenta extractores de características similares a los filtros Gabor. La segunda capa, aprende características de texturas

más complejas. Las características de la tercera capa son más complejas, y algunas estructuras simples comienzan a aparecer. En el caso de un problema de reconocimiento de rostros, pueden ser por ejemplo: nariz respingada y grandes ojos. En la cuarta capa, la salida de la red es suficiente para explicar cierto atributo facial [12].

Generalmente las estructuras utilizadas como modelos de red neuronal en el reconocimiento de rostro y otras aplicaciones, son aquellas que tiene un excelente desempeño en el desafío ImageNet, las típicas son: AlexNet, VGGNet, GoogleNet, ResNet y SENet. Estas estructuras son usadas directamente o con pequeñas variaciones [12].

2.1.2.1. Capas convolucionales

Las capas convolucionales son la unidad básica de las redes neuronales convolucionales. Estas capas convolucionales estan formadas por neuronas dispuesta en un arreglo 3D (ancho,alto, profundidad), esta neuronas estas conectadas a una pequeña región en la capa anterior, esta conexión esta denotada por pesos (W), que determinan un conjunto de filtros aprendibles. Entonces, cada neurona está asociada a un filtro y este filtro esta focalizado espacialmente en un pequeña región y se extiende por todo el volumen de esa región. [13]

En la figura 2.2 se muestra una capa convolucional (azul) y las neuronas de una misma columna, las cuales se enfocan en la misma región espacial del volumen de entrada (32x32x3). La región en el volumen de entrada en la que se concentran pueden ser por ejemplo de 5x5 píxeles, y debido a la profundidad de 3 canales el filtro determinado por las conexiones de cada neurona es de 5x5x3. Luego, con cada uno de estos filtros se realiza una convolución sobre la entrada 3D, lo cuál produce un **mapa de activación** de 2 dimensiones que entrega la respuesta de cada uno de los filtros en cada posición espacial. La cantidad de mapas de activación depende la profundidad de la capa convolucional [13].

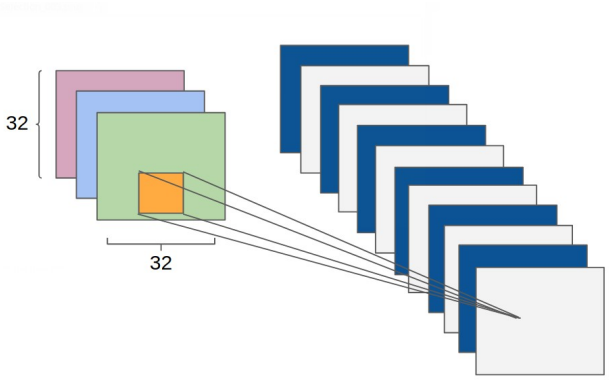


Figura 2.2: Neuronas en una capa convolucional y conexión con capa anterior.

Intuitivamente, la red aprenderá filtros que se activen cuando están en presencia de algún tipo de características visuales tales como bordes de alguna orientación o manchas de algún color [13].

La disposición espacial de la salida de una capa convolucional esta determinada por 3 parámetros: **profundidad**, ***stride*** y ***zero-padding*** [13].

1. Profundidad: Corresponde al número de filtros que serán usados.
2. Stride: Se refiere a la separación con la cuál se aplica cada filtro. Cuándo el *stride* es 1 entonces el movimiento del filtro es de 1 pixel a la vez, como se muestra en la figura 2.3. Cuando el *stride* es 2 el filtro se salta 2 píxeles cada vez para ser aplicado. A mayor **stride** menor es la salida de **mapa de activación** de salida [13].

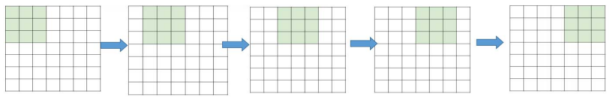


Figura 2.3: Stride de 1, el filtro se desliza un píxel a la vez.

3. Zero-padding: Uno de los problemas de la aplicación de la convolución es la perdida de información que puede existir en los bordes de la imagen. Una manera eficiente de resolver este problema, es usar ***zero-padding***. Además, permite controlar el tamaño de la salida. Por ejemplo en la figura 2.3 el tamaño de la entrada es N=7, y del filtro es F=3 y de stride S = 1, entonces la salida será de 5x5, ya que no se puede aplicar el filtro en os costados. Sin embargo, agregando **zero-padding** de P = 1, como se muestra en la figura la salida será de 7x7, que es exactamente el tamaño de la entrada original [13].

Figura 2.4: Ejemplo de zero-padding. Se agregan ceros alrededor de la imagen para realizar el filtrado.

La fórmula para determinar el mapa de salida es:

$$O = 1 + \frac{N + 2P - F}{S} \tag{2.7}$$

Luego de la capa convolucional se utiliza una función de activación la cuál siendo seleccionada apropiadamente permite acelerar y mejorar el proceso de entrenamiento [14].

En la figura 2.5 se muestran las funciones de activación comunes. Sin embargo, la función ReLU (por sus siglas en inglés Rectified Linear Unit) ha sido usado más a menudo hasta ahora por las siguientes razones:

1. ReLU es una función simple, de rápido computo de la función y de su gradiente [13].

$$ReLU(x) = max(0, x) \tag{2.8}$$

$$\frac{ReLU}{x}(x) = \{1 \quad si \quad x > 0; 0 \quad si \quad no\} \tag{2.9}$$

2. La funciones saturadas como la sigmoid y tanh causan problemas en el **back propagation**. Debido a que como las CNN son profundas, la señal del gradiente se comienza a desvanecer entre las capas, este problema es conocida como "gradiente desvaneciente". Esto se sucede porque el gradiente de aquellas funciones es muy cercano a cero, en cualquier zona cercana al centro. En cambio, ReLU tiene un gradiente constante cuándo la entrada es positiva [13].

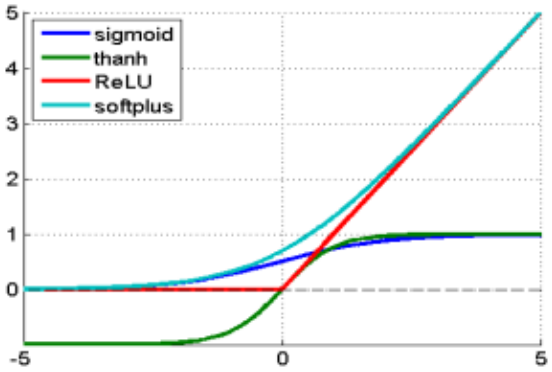


Figura 2.5: Funciones de activación.

2.1.2.2. Capa de pooling

El objetivo principal de la capa de *pooling* es hacer ***down-sampling*** para reducir la complejidad en las capas siguiente. En el procesamiento de imagen, puede ser considerado como reducir la resolución. Uno de los métodos más comunes de *pooling* es max-pooling. Este método divide la imagen en sub-regiones rectangulares y retorna el valor máximo de cada región. Uno de los más usados es max-pooling de 2x2 [13]. En en la figura 2.6 se muestra la operación de max-pooling..

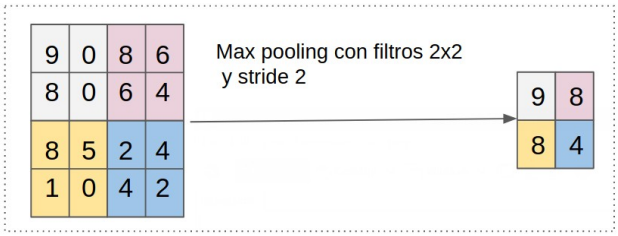


Figura 2.6: Max-pooling 2x2. Ejemplo de pooling máximo con filtros de tamaño 2x2 y stride 2.

Se debe tomar en cuenta que *down-sampling* no conserva la posición de la información. Entonces, debería ser aplicado solo cuándo la presencia de información es importante [13].

2.1.2.3. Capa *fully connected*

Cada neurona de la capa *fully-connected* está totalmente conectada a cada una de las neuronas de la capa previa y en la capa posterior. Generalmente esta capa es dispuesta luego de una capa de *pooling* y es utilizada para construir un **clasificador**. El clasificador puede ser construido con múltiples capas *fully-connected* y con una capa de salida que depende de la tarea para la que se utilizará la CNN. Si es para clasificación, se utiliza una capa de salida igual al número de clases del problema de clasificación [13].

Uno de los clasificadores más utilizados es el clasificador *softmax* que tiene de salida probabilidades de pertenencia a cada clase, sumando 1 [14].

2.1.2.4. Regularización

Uno de los principales problemas en el entrenamiento de redes profundas es el sobre entrenamiento. La regularización es la técnica empleada para sobrellevar este problema. Dropout es una de las principales técnicas de regularización. En este método la activación de algunas neuronas de la capa *fully connected* son apagadas, es decir, su salida es cero durante una etapa del entrenamiento [14].

2.1.2.5. Entrenamiento red neuronal convolucional

Entrenar una red neuronal convolucional con muchas capas es un proceso tedioso debido a la naturaleza no-convexa de su función de costos y los requerimientos computacionales necesarios para entrenar la red si los conjuntos de datos son muy grandes [14]. Raramente las CNN son entrenadas desde cero con inicialización aleatoria. La convergencia en el entrenamiento puede ser lograda con la adecuada inicialización [15].

La técnica de **trasferencia de aprendizaje** puede ser implementada para tener una inicialización significativa que ayude a disminuir los tiempos de entrenamiento y convergencia. En algunos casos, solamente el clasificador o algunas de las últimas capas necesitan ser entrenadas nuevamente, lo que conlleva una reducción del tiempo de entrenamiento [14].

Con una cuidadosa selección de parámetros como tasa de aprendizaje, número de iteraciones, número de muestras de entrenamiento y de prueba, tamaño del *batch* de entrenamiento, etc. es posible lograr un buen modelo. La red es es entrenada utilizando ***backpropagation*** y **gradiente descendente** para la actualización de los pesos de la red [14].

El entrenamiento con **backpropagation** es afectado por un factor clave: el gradiente desvaneciente [16]. Este problema hace difícil de entrenar las capas inferiores de la red de una red

multicapa, dado que la tasa de error (utilizada para la actualización de pesos) decae convergiendo a cero o diverge exponencialmente. Este problema puede ser manejado seleccionando apropiadamente una función de activación como la sigmoid o tanh. Pero, se ha demostrado que con la función de activación ReLU es más efectiva para redes neuronales convolucionales [17].

2.2. Redes neuronales convolucionales

2.2.1. Deep residual Networks (ResNet)

No es fácil entrenar redes neuronales profundas debido a los problemas con gradiente desvaneciente en las capas superficiales durante el *back propagation*. Dado que *back propagation* está basado en la regla de la cadena, hay una tendencia del gradiente a disminuir a medida que se avanza hacia las capas mas superficiales. Esto se debe a la multiplicación de números pequeños, especialmente por los pequeños valores absolutos de los errores y parámetros [18].

Para aliviar el proceso de degradación de gradiente en [19] se introduce el concepto de aprendizaje residual profundo.

En la figura 2.7 se muestra una combinación entre un bloque típico de una CNN y bloque residual de la red ResNet. La idea de ResNet es que para prevenir que el gradiente sea degradado, se permite que la información fluya a través de una conexión de atajo (*shorcut connection*) para alcanzar las capas superficiales.

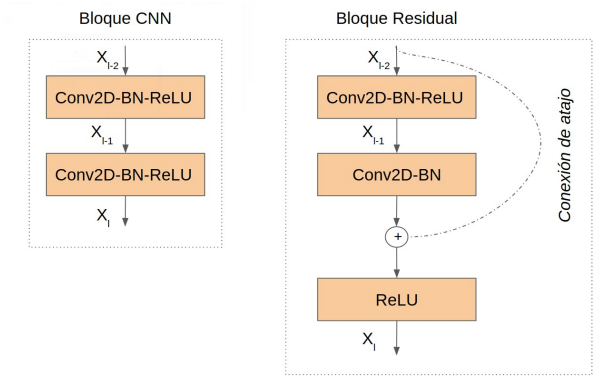


Figura 2.7: Comparación entre un bloque típico de una CNN y un bloque de ResNet. Se introduce una conexión de atajo.

En la figura 2.7 los mapas de características son representados como x, para la red CNN y ResNet. En específico, los mapas de características de salida de la capa l son x_l. La operaciones en la capa CNN son **Conv2D-Batch Normalization (BN)-ReLU**.

El conjunto de operaciones a las que se someten los mapas de características de una CNN se definen como H() = Conv2D-Batch Normalization (BN)-ReLU, entonces:

$$x_{l-1} = H(x_{l-2}) \tag{2.10}$$

$$x_l = H(x_{l-1}) \tag{2.11}$$

Los mapas de características de la capa $l-2$ son transformados a x_l por H(). El mismo conjunto de operaciones es aplicado para transformar x_{l-1} en x_l.

En cambio para ResNet, la conexión de atajo mostrada en la figura 2.7 es caracterizada por las siguientes operaciones entre los bloques convolucionales:

$$x_{l-1} = H(x_{l-2}) \tag{2.12}$$

$$x_l = ReLU(F(x_{l-1}) + x_{l-2}) \tag{2.13}$$

$F(x_{l-1})$ representa las operaciones Conv2D-BN sobre los mapas de características de x_{l-1}, lo que es conocido como mapeo residual. La conexión de atajo no agrega parámetros extras o complejidad computacional extra.

Para realizar la operación $F(x_{l-1}) + x_{l-2}$ ambos términos deben tener las mismas dimensiones. Si las dimensiones son diferentes, se realiza una proyección lineal de x_{l-2} para coincidir con las dimensiones de $F(x_{l-1})$.En [19], la proyección lineal para el caso en el que el mapa de características es de la mitad de tamaño, se utiliza la operación Conv2D con kernel de dimensiones 1×1 y strides=2.

En la figura 2.8, se muestra la arquitectura de ResNet con 3 bloques residuales para CIFAR10.

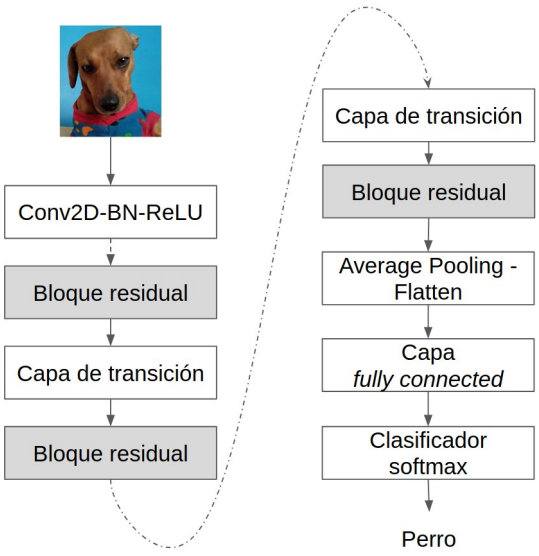


Figura 2.8: ResNet para el problema de clasificación CIFAR10, con 3 bloques residuales .

El modelo presentado anteriormente en conocido como ResNet v1. En [20] se presenta un versión mejorada de ResNet conocida como ResNet v2. La mejora recae principalmente en que cambia la composición de los bloques residuales.

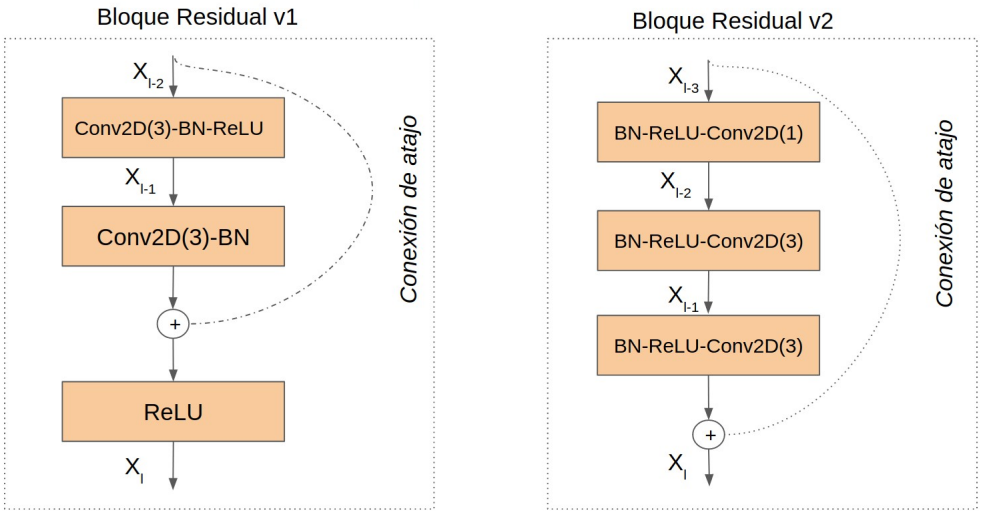


Figura 2.9: Comparación entre bloques residuales de Resnet v1 y Resnet v2 .

Como se muestra en la figura 2.9, ahora el bloque residual considera para una entrada x_{l-3} las operaciones consecutivas BN-ReLU-Con2D(1), BN-ReLU-Con2D(3), BN-ReLU-Con2D(1) y las posterior restroalimentación a través de una conexión de atajo de x_{l-3}.

2.2.2. Densely connected convolutional networks (DenseNet)

DenseNet ataca el problema de gradiente desvaneciente utilizando un enfoque diferente. En vez de usar conexiones de atajo, todos los mapas de características previos se utilizan como entrada de la siguiente capa, formando un bloque Denso como en el que se muestra en la figura 2.10 [21].

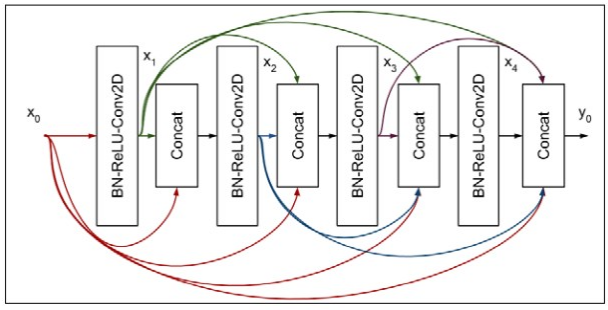


Figura 2.10: Bloque Denso de 4 capas. La entrada de cada capa está conformada por todos los mapas de características previos

La entrada de la capa l es la concatenación de todos los mapas de características previos. Se define la operación $H(x)$ como BN-ReLU-Conv2D, entonces, la salida de la capa l es:

$$x_l = H(x_0, x_1, x_2, ..., x_{l-1}) \tag{2.14}$$

Conv2D usa un kernel de tamaño 3. El número de mapas de características generados por capa es llamado tasa de crecimiento, k. Normalmente, $k = 12$, pero en [21] también utilizan $k = 24$. Por lo tanto, si el número de mapas de características x_0 es k_0, entonces el número total de mapas de características al final de un bloque Denso de 4 capas es $4 \times k + k_0$ [18].

En [21] cada bloque Denso es precedido por la operación BN-ReLU-Conv2D, a medida que el número de mapas de características duplica la tasa de crecimiento, $k_0 = 2 \times k$. Se utiliza un kernel de tamaño 3. En la capa de salida, DenseNet sugiere que se utilice un capa de pooling antes de una capa fully connected y el clasificador softmax(). Si data augmentation no es usado, un capa de dropout debe seguir a la capa convolucional de cada bloque Denso [18].

Para prevenir que el número de mapas de características se incremente hasta el punto de ser computacionalmente ineficiente, se introducen las capas cuellos de botella (*Bottleneck*) como se muestra en la figura 2.11. La idea es que después de cada concatenación; una convolución de 1×1 con un tamaño de filtro igual a $4k$ es aplicada. Esta técnica de reducción de dimensionalidad previene que el número de mapas de características a ser procesado por la capa Conv2D(3) crezca rápidamente [18].

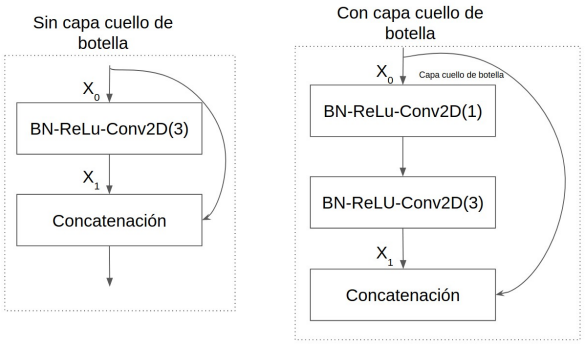


Figura 2.11: Una capa en un bloque Denso de DenseNet, con y sin la capa cuello de botella BN-ReLU-Conv2D(1).

La capa cuello de botella modifica la la capa DenseNet realizando las operaciones BN-ReLU-Conv2D(1)-BN-ReLU-Conv2D(3), en vez de solo realizar el conjunto de operaciones BN-ReLU-Conv2D(3) [18].

Para resolver el problema de discordancia en las dimensiones de los mapas de características, DenseNet divide una red neuronal profunda en múltiples bloques Densos que son unidos por capas de transición, como se muestra en la figura 2.12. Dentro de cada bloque Denso, las dimensiones de los mapas de características se mantienen constantes [18].

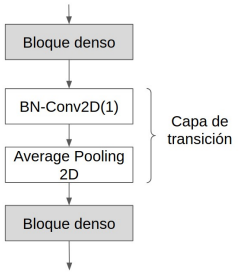


Figura 2.12: Capa de transición entre dos bloques Densos.

El rol de la capa de transición es transicionar de un mapa de características de mayor tamaño a uno de menor tamaño entre dos bloques Densos. La reducción de tamaño en usualmente a la mitad. Esto se logra por medio de una capa de *average pooling*.

Sin embargo, antes de que los mapas de características sean pasados por la capa de average pooling, el número es reducido por un cierto factor de compresión, $0 < \theta < 1$, usando Conv2D(1). En [21] se utiliza $\theta = 0{,}5$. Por ejemplo, si la salida de la última concatenación del bloque Denso previo es (64,64,512), entonces, luego de la operación Conv2D(1) las nuevas dimensiones del mapas de características serán (64,64,256) [18].

Cuando la **cuello de botella** y la **capa de compresión** son utilizadas, se denomina a la red DenseNet-BC [19].

2.3. Trabajos relacionados

2.3.1. Estrategias para combinar características handcrafted con características CNN

En [1], se muestran diferentes estrategias para complementar con características extraídas con métodos convencionales (características *handcrafted*) las características extraídas por una red neuronal convolucional. Estas estrategias son evaluadas sobre el problema de clasificación de CIFAR10.

1. Combinación de características *handcrafted*: Se describe el contenido de la imagen utilizando diferentes propiedades como color, forma, textura y características locales. Se eligen hitogramas RGB, LAB y HSV como características de color, transformada HOG como característica de forma, histogramas LBP como características de texturas, y BoW (*Bag of Words*) basados en dende-SIFT como características locales. Estas características son fusionadas directamente concatenándolas pero en diferentes combinaciones para encontrar la combinación con mejor desempeño. Después de eso, las características fusionadas son conectadas a un clasificador de red neuronal de 2 capas ocultas *fully conected* [1].

   Se determina que la característica que otorga el mejor desempeño individual es la transformada HOG que alcanza un 52,71 %. Pero, la combinación de las 4 características es la que logra el mejor desempeño de todas la combinaciones alcanzando un 61,18 %[1].

2. Combinación directa de características de una CNN y características *hand-crafted*: Se utiliza la red convolucional CaffeNet (tiene 3 capas convolucionales, 3 capas de pooling y 2 capas *fully-connected*), las cuál es entrenada sobre CIFAR10. Luego, se extrae un vector de características desde la última capa de pooling de la red convolucional y luego es fusionado con las características *hand-crafted*. Este nuevo vector es utilizado como entrada a las capa *fully-connected* del clasificador de la red CaffeNet y se vuelve a entrenar [1].

   El desempeño de clasificación utilizando solo CaffeNet es de 74,83 %. Luego, se evalúa el desempeño agregando las características anteriores en diferentes combinaciones.El mejor desempeño se logra agregando las 4 características aumentando el desempeño en 1,68 %. El autor concluye que de acuerdo a estos resultados, se puede aprender que las características extraídas con la CNN son suficientes para describir una imagen en muchos aspectos, pero aún hay espacio para mejorar [1].

3. Codificación y entrenamiento conjunto: Muchas características *handcrafted* no son vectores normalizados y tienen diferentes formas, por lo que la combinación no es directa, ellas deben ser codificadas antes. Se eligen 4 características representativas: Canny edge, LBP feature map, Gabor Filter response map y dense-SIFT. Para cada una de estas características es realizada una codificación con capas convolucionales para formar un bloque de características. Este bloque luego en añadido a la arquitectura de CaffeNet como se representa en la figura 2.13. La estructura es entrenada junta completamente. Al combinar las 4 características se logra un desempeño de 78,47 %. Entonces, la codificación provee de información complementaria y la estrategia de entrenamiento

conjunto hace que estas características se fusionen mejor [1].

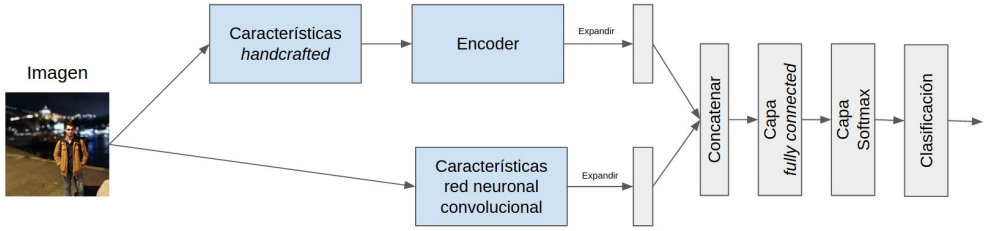


Figura 2.13: Combinación de características *hand-crafted* con CNN.

En [22] se modifica la estructura de una red neuronal convolucional (con dos capas convolucionales) ajustando los parámetros de la primera y/o segunda capa con filtros Gabor. Los filtros de la primera y/o segunda capa pueden ser entrenables o no entrenables. Ajustar los filtros de la red convolucional con filtros Gabor no presenta mejoras en el desempeño (degradación $0 - 3\,\%$), pero si presenta menores tiempo de entrenamiento (1,4×) y menor cantidad de almacenamiento (2,23×).

2.3.2. Desempeño de redes neuronales convolucionales en CIFAR10

En la tabla 2.1 se presenta el *accuracy* en el conjunto de prueba para diferentes redes sobre el conjunto de datos CIFAR10. El símbolo "+" indica estándar data augmentation, que considera traslación y/o reflejo.

Tabla 2.1: *accuracy* (%) en CIFAR10. k denota la tasa de crecimiento de la red. "+" indica estándar data augmentation (traslación y/o reflejo). Todos los resultados de las DenseNets son obtenidos utilizando dropout. Las DenseNets logran mayor *accuracy* usando menos parámetros que las ResNet. Sin data augmentation, DenseNet tiene un mejor desempeño, por un largo margen [21].

Método	Profundidad	Parámetros	C10	C10+
Network in network	-	-	89.59	91.19
All-CNN	-	-	90.92	92.75
Deeply Supervised Net	-	-	90.31	92.03
Highway Network	-	-	-	92.28
FractalNet	21	38.6M	89.82	94.78
with Droput/Drop-path	21	38.6M	92.67	95.4
Resnet	110	1.7M	-	93.39
Resnet [Deep Networks with Stochastic Depth]	110	1.7 M	86.37	93.59
ResNet with Stochastic Depth	110	1.7M	88.34	94.77
	1202	10.2M	-	95.09
Wide ResNet	16	11.0M	-	95.19
	28	36.5M	-	95.83
with Dropout	164	1.7M	88.74	94.54
ResNet (pre-activation)	1001	10.2M	89.44	95.38
DenseNet(k=12)	40	1.0M	93.00	94.76
DenseNet(k=12)	100	7.0M	94.23	95.90
DenseNet(k=24)	100	27.2M	94.17	96.26
DenseNet-BC(k=12)	100	0.8M	94.08	95.49
DenseNet-BC(k=24)	250	15.3M	94.81	96.38
DenseNet-BC(k=40)	190	25.6M	-	96.54

2.4. Bases de datos

2.4.1. CIFAR10

CIFAR10 es un conjunto de datos de 10 clases. El conjunto de datos es una colección de pequeñas imágenes de 32×32 en RGB, de objetos en el mundo real. Estos objetos son: aviones, autos, aves, gatos, ciervos, perros,ranas, caballos, barcos y camiones.

En la figura 2.14 se muestra un ejemplo de muestras de CIFAR10. En el conjunto de datos, hay 50.000 imágenes de entrenamiento y 10.000 imágenes de prueba.

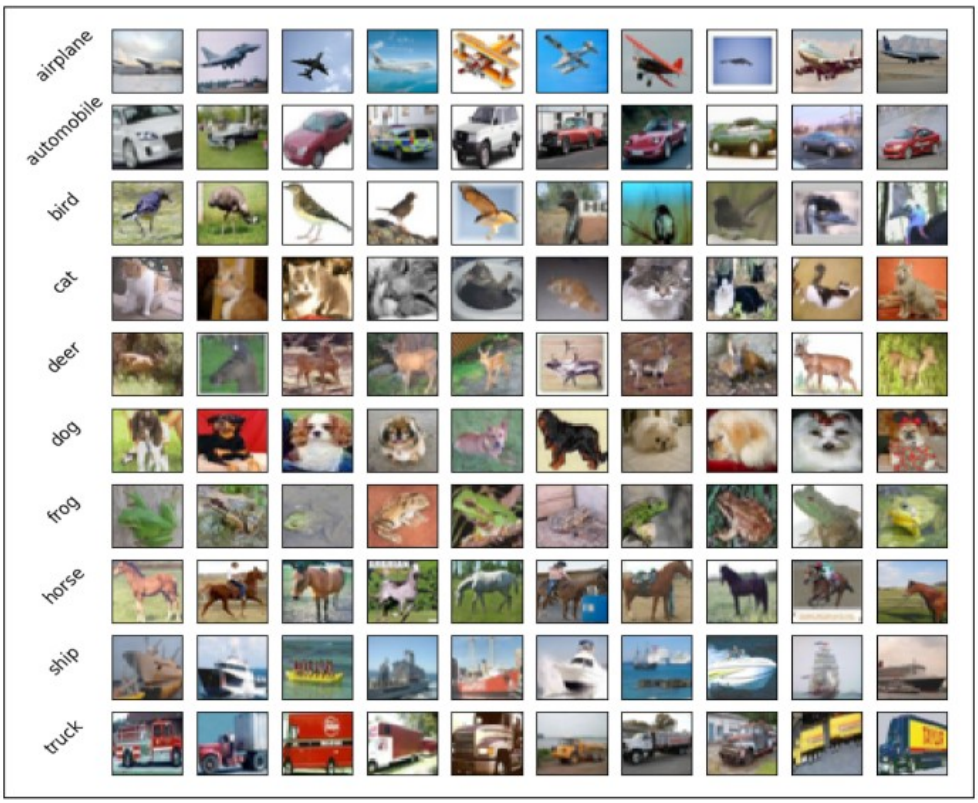


Figura 2.14: Ejemplo de las clases en CIFAR10.

Capítulo 3

Metodología

En la figura 3.1 se muestra la metodología propuesta para la incorporación de características en redes neuronales convolucionales, la cuál consiste en los siguientes pasos:

1. Selección de extractor de características: Se utiliza un banco de 32 filtros Gabor de 4 amplitudes diferentes y 8 orientaciones.
2. Sintonización de características *handcrafted*: Los parámetros del extractor de características (λ,σ,γ) deben ser ajustados de acuerdo al problema para el cual se quieren utilizar, en este caso, para el problema de clasificación de CIFAR10.
3. Incorporación de características a la red: Las características *handcrafted* deben agregarse a la red de manera que no afecten la estructura de la red. Para los experimentos se utilizan las redes DenseNet y ResNet.
4. Evaluar: Se evalúa el desempeño de las redes neuronales convolucionales. Dada la naturaleza estocástica presente en el entrenamiento de redes neuronales, se realizan 3 entrenamientos de cada red y se comparan los desempeños de las redes utilizando el test estadísticos t-test.

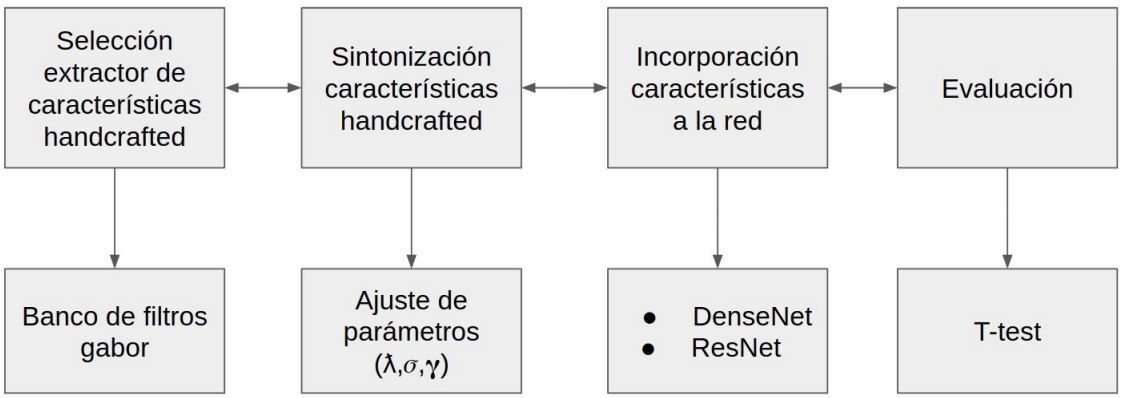


Figura 3.1: Metodología para la incorporación de características *hancrafted* en redes neuronales convolucionales.

3.1. Incorporación de características

3.1.1. Sintonización características

Como extractor de características *handcrafted* se utiliza un banco de filtros Gabor compuesto por 32 filtros. El banco de filtros Gabor esta compuesto por 4 grupos de filtros (F_1, F_2, F_3 y F_4). Los filtros de un mismo grupo tienen el mismo tamaño de filtro y los mismos parámetros de la función Gabor (λ,σ,γ), a excepción de la orientación θ, que varía de acuerdo al número de filtros que se requiere por grupo. Si se requieren m filtros por grupo, $\theta = k \cdot \frac{\pi}{m}$ con $k = 1...m$. Para este trabajo se utilizan **8 orientaciones**.

Los grupos de filtros en un banco de filtros Gabor, tienen amplitudes consecutivas λ,$\sqrt{2}\lambda$,$2\sqrt{2}\lambda$, y $3\sqrt{2}\lambda$, para los grupos de filtros F_1, F_2, F_3 y F_4, respectivamente . Además, el tamaño de los filtros son números impares consecutivos.

El banco de filtros Gabor debe ser sintonizado de acuerdo a problema a resolver, en este caso, el problema de clasificación de imágenes de CIFAR10. Para determinar los parámetros de los filtros Gabor se realiza el proceso representado en la figura 3.2. El cuál consiste en:

1. Transformar cada imagen de entrada a escala de grises.

2. Agregar **padding replicado** a la imagen, para obtener mapas de características de las misma dimensiones al aplicar los filtros.

3. Entrenar conjuntamente una red neuronal MLP, que utilice como entrada las características generadas por el banco de filtros Gabor(al ser convolucionado con la imagen de entrada) y sintoniza los parámetros de los filtros con PSO.

Figura 3.2: Proceso de selección filtros Gabor.

3.1.1.1. Padding replicado

Los filtros Gabor deben ser convolucionados con imágenes de una dimensión, es por ello que las imágenes del conjunto de datos son transformadas a escala de grises. Además, para obtener mapas de características de igual dimensión a partir de cada uno de los filtros se aplica un *padding* llamado replicado (*replicate*), mostrado en la Figura 3.3, el cuál repite el borde de las imágenes.

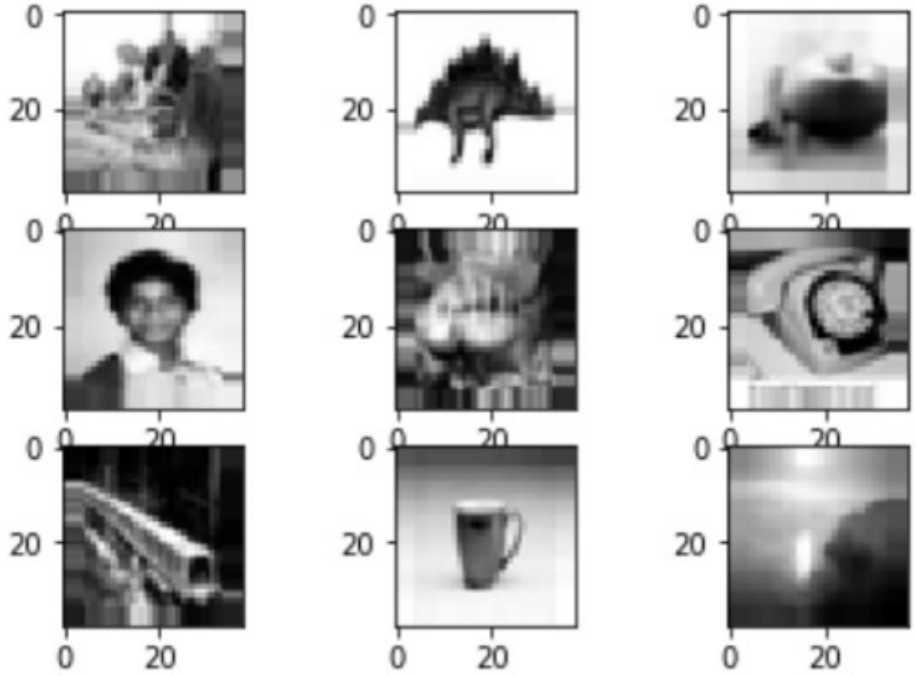


Figura 3.3: Ejemplos CIFAR100 utlizando padding replicado en escala de grises y normalizado.

Los mapas de características de salida deben tener dimensión 32×32. Para calcular el tamaño del *padding* se debe utilizar la fórmula $O = 1 + \frac{N+2P-F}{S}$. Se requiere que $O = 32$, por que el tamaño del padding con $S = 1$ es $P = \frac{F-1}{2}$, donde F es el tamaño del filtro.

3.1.1.2. Sintonización parámetros con PSO

Para determinar una inicialización para los parámetros del banco de filtros Gabor, se entrena una un clasificador MLP de 2 capas ocultas que utiliza como entrada solo las características obtenidas por medio del banco de filtros Gabor con parámetros λ, σ y γ a determinar.

En la red se realiza un entrenamiento conjunto, en donde los pesos del clasificador son obtenidos con un optimizador (Adam, SGD, etc.) y los parámetros de los filtros son ajustados con PSO.

La red se entrena durante 100 épocas, en cada época se ajustan los pesos del clasificador con *backpropagation* y cada 10 épocas se ajustan los parámetros de los filtros Gabor en el conjunto de entrenamiento.

Se utiliza PSO con restricciones y como función objetivo la entropía cruzada, se establecen restricciones inferiores y superiores para σ,γ y λ. Las restricciones son las siguientes:

1. restricciones inferiores: $[\gamma_{\mathrm{i}},\sigma_{\mathrm{i}},\lambda_{\mathrm{i}}]=[0.01,0.01,1]$
2. restricciones superiores: $[\gamma_s,\sigma_s,\lambda_s]=[2,10,25]$

El algoritmo de PSO está configurado con un máximo de 200 iteraciones y 300 partículas. El valor de convergencia es $\exp -8$.

Se seleccionan como valores iniciales de los parámetros de los filtros Gabor, aquellos que presenten el mejor desempeño en el conjunto de validación luego de las 100 épocas de entrenamiento.

3.1.2. Incorporación de características a la red

Para realizar experimentos con el objetivo de determinar la mejor configuración para la incorporación de características *handcrafted* a las redes DenseNet y ResNet, se utilizan versiones pequeñas de estas redes, con 20 capas de profundidad, denominadas DenseNet20 y ResNet20, respectivamente.

En la figura 3.4 se observa el conjunto de experimentos que se realizará para evaluar el desempeño de la incorporación de características. Los cuáles consiste en los siguientes:

1. Red pequeña: Se entrena una versión pequeña de la red a evaluar. Se realizan entrenamientos utilizando el conjunto de datos normal, con data *augmentation* (método que nos permite aumentar el conjunto de datos realizando perturbaciones y modificaciones a las imágenes originales) y con *cutout* (método de data augmentation que permite aumentar el número de datos removiendo secciones aleatorias de las imágenes) .

2. Red pequeña con características handcrafted: Por un lado, se incorporan las características *handcrafted* sintonizadas, las cuáles deben ser ajustadas en el proceso de entrenamiento de la red. Y por otro lado, en vez de agregar características *handcrafted*, se cambian los filtros Gabor por filtros convencionales. Luego, cada una de las variantes se entrena con data *augmentation* y *cutout*.

3. Red grande: Se entrena una versión grande de la red a evaluar. Como red grande se utiliza DenseNet con 100 capas (DenseNet100) y ResNet con 110 capas (ResNet110).

4. Red grande con características *handcrafted*: Se entrena la red grande solo incorporando los filtros Gabor y utilizando un conjunto de datos normal.

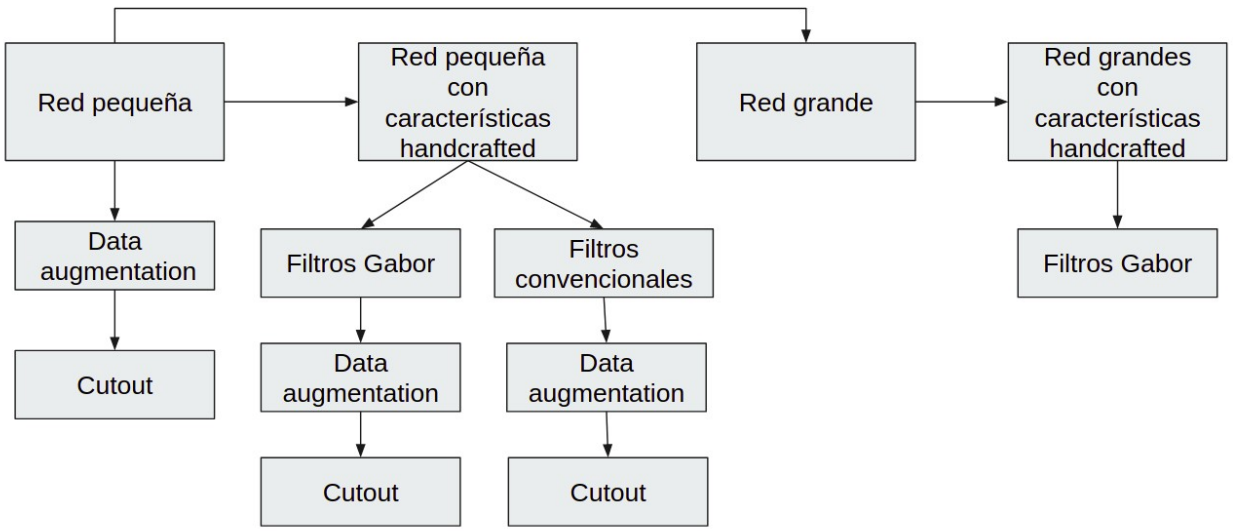


Figura 3.4: Conjunto de experimentos para evaluar la incorporación de características en redes neuronales convolucionales.

3.1.2.1. Sintonización de filtros en la red

Los filtros Gabor tienen parámetros λ, σ y γ. Anteriormente, con una sintonización previa, se determinaron valores iniciales para los parámetros del banco de filtros Gabor λ_0, σ_0 y γ_0. Estos parámetros deben ser ajustados en el entrenamiento de la red, para realizar este ajuste, los parámetros de los filtros son modificados para ser considerados como pesos de la red y actualizado con el mismo optimizador de los pesos de la red.

3.1.2.2. Filtros convencionales

Los filtros convencionales son aquellos compuestos totalmente por pesos de la red. El banco de filtros Gabor es reemplazado por un banco de filtros convencionales.

El banco de filtros Gabor esta compuesto por 4 grupo de filtros Gabor, que tienen como dimensiones de filtros números impares consecutivos. Estos filtros son reemplazados por filtros convencionales de dimensiones $[F_1, F_1]$,...,$[F_4, F_4]$.Por lo tanto, el **bloque Gabor** es remplazado.

3.2. Redes neuronales convolucionales

3.2.1. DenseNet

Se utiliza la red Densenet y se incorpora el bloque de características, concatenando las características Gabor a la entrada de cada bloque denso, como se muestra en la figura 3.5. Para incorporase al segundo bloque denso se utiliza una capa de transición, para que coincidan las dimensiones. En la figura 3.6 se muestra en detalle como se incorporan las características en cada bloque denso.

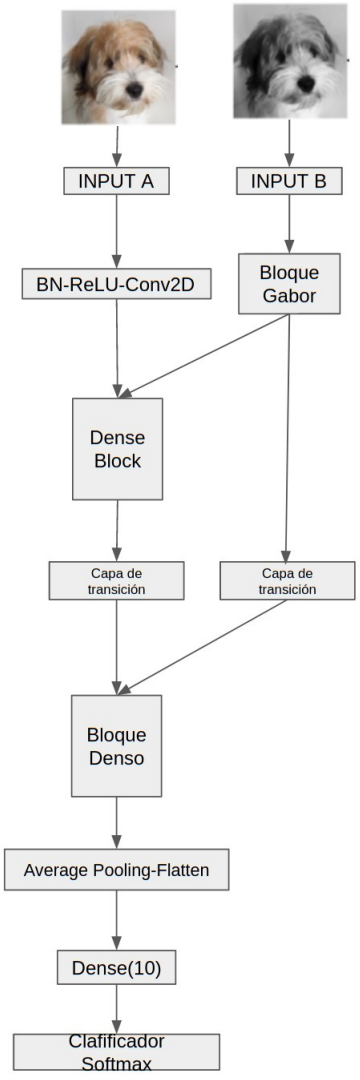


Figura 3.5: Densenet Gabor. Incorporación de características Gabor en DenseNet, las características Gabor son incorporadas al inicio de cada bloque denso.

En la figura 3.6 se muestra la incorporación de características en un bloque denso. El **bloque Gabor** esta compuesto por 32 mapas de características, estos mapas de características son obtenidos como se indica en la sección 3.1. El bloque de características Gabor que ingresa

al primer bloque denso tiene dimensiones [32,32,32].

El bloque Gabor es concatenado a lo mapas de características X_0, que es un bloque de características de dimensiones [32,32,24]. Este nuevo conjunto de características denominado X_1 tiene dimensiones [32,32,56] ingresa al bloque denso 1. En el bloque denso se realizan las operaciones BN-ReLU-Conv2D(1) y BN-ReLU-Conv2D(3) de manera consecutiva durante N veces, donde N es el número de cuellos de botellas dentro de un bloque denso.

El valor de N está determinado, por el número de bloques densos y la profundidad de la red. Entonces, $N = \frac{profundidad-4}{2 \times número \text{ de } bloques \text{ densos}}$

Finalmente, a la salida de un bloque denso tenemos un mapa de características de dimensiones $N \times k + k_0 + k_{Gabor}$. Con k la tasa de crecimiento de la red y k_0 la características iniciales que ingresan al bloque denso.

Luego, en la salida del bloque denso se aplica la capa de transición, que reduce la dimensión de los mapas de características a la mitad con la operación *pooling* promedio y luego hace un reducción de dimensionalidad en un factor de compresión θ. Obteniéndose a la salida de la capa de transición un bloque de características de $[16,16,\theta \cdot (N \cdot k + k_0 + k_{Gabor})]$.

Finalmente, se incorporan las características Gabor nuevamente a la salida de la capa de transición. Antes, se debe reducir la dimensión del bloque Gabor a la mitad, para que coincidan las dimensiones. Esto se realiza utilizando una capa de *pooling* promedio, resultando en un bloque de características Gabor de [16,16,32].

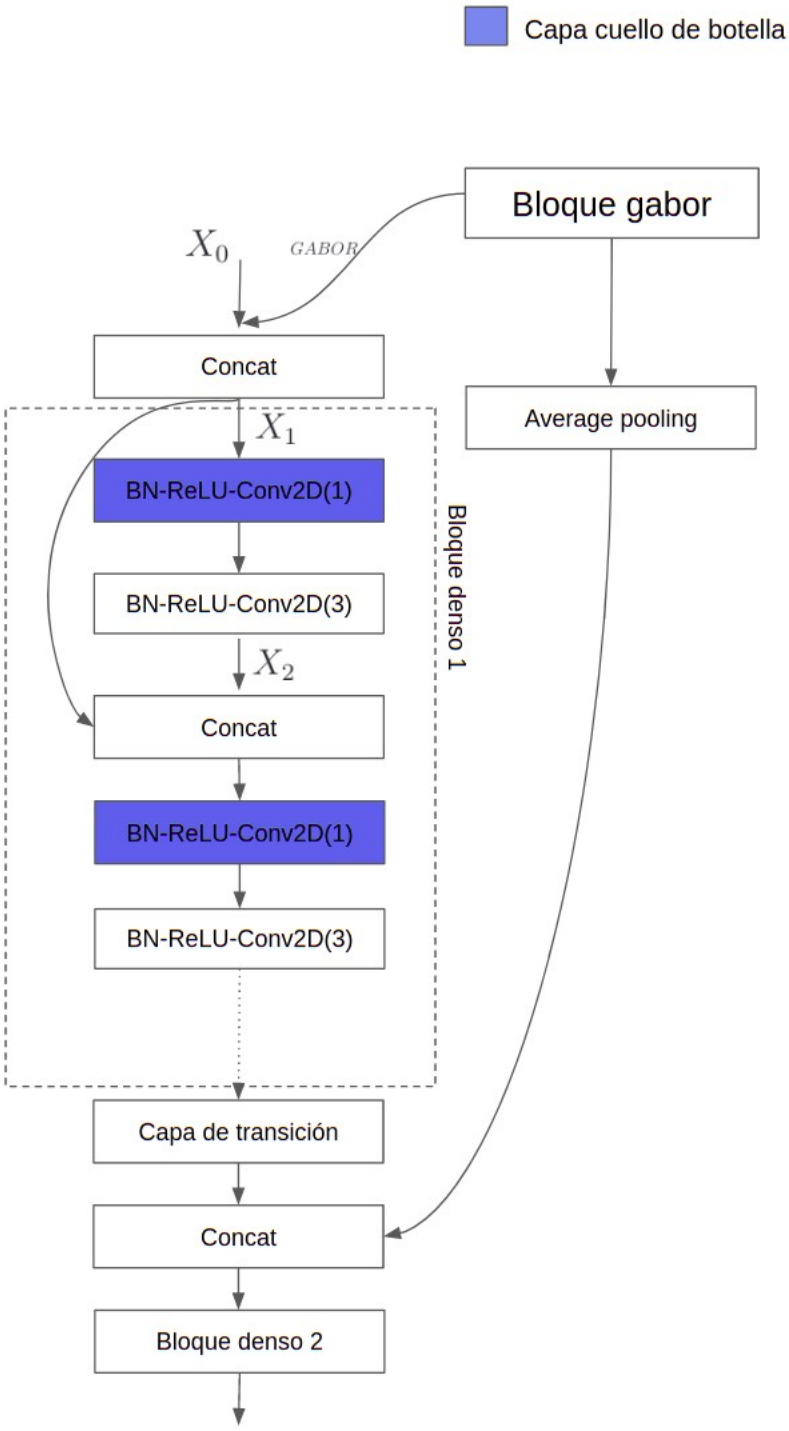


Figura 3.6: Incorporación de características en bloque Denso. Las características Gabor son concatenadas a la entrada del bloque denso y dentro del bloque denso estas son concatenadas a capa una de las capas convolucionales.

Para lo experimentos se utiliza como red pequeña DenseNet con 20 capas (que es denominada DenseNet20),con 2 bloques densos y 4 cuellos de botella en cada bloque. Y como red

grande se utiliza la red DenseNet con 100 capas y 3 bloques densos (DenseNet100).

Para ambas redes se utilizan tasa de crecimiento k=12.

Al incorporar los filtros Gabor en la red DenseNet, se debe determinar algunos parámetros en la arquitectura de la red, por ejemplo, el factor de compresión. Se evalúan los valores θ =1.0,0.7,0.5,0.3.

3.2.2. ResNet

Se utiliza la red ResNet v2 y se incorporan las características *handcrafted* a la entrada de cada una de las celdas residuales, como se muestra en la figura 3.7.

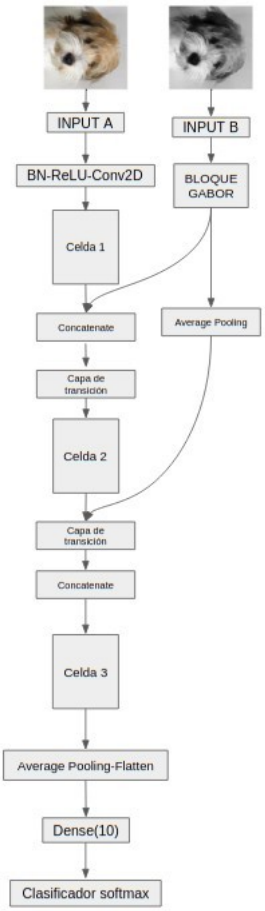


Figura 3.7: Resnet incorporación de características Gabor.

La red ResNet v2 apila una secuencia de operaciones (1 x 1)-(3 x 3)-(1 x 1) BN-ReLU-Conv2D también conocida como capas cuellos de botella. La primera conexión de atajo es la de la convolución 1x1.

La red ResNet v2 esta compuesta por 3 celdas las cuales van disminuyendo la dimensiones de sus mapas de características (largo y ancho) a medida se avanza en la red . Los mapas de

características de una misma celda tienen el mismo tamaño y el mismo número de filtros. En cada celda hay múltiples bloques residuales consecutivos y en cada bloque residual a la entrada se le aplican la secuencia de operaciones BN-ReLU-Conv2D(1), BN-ReLU-Conv2D(3), BN-ReLU-Conv2D(1) y además se realiza la conexión de atajo para generar la salida.

En el primer bloque residual de cada celda se aplica la primera convolución con *stride*=2, para reducir las dimensiones de los mapas de características a la mitad (*downsampled*). Además, a la conexión de atajo se le aplica la operación Conv2D(1) con *stride*=2, esta es la denominada capa de transición.

Los bloques de la red ResNet v2 mostrada en la figura, tienen la estructura que se muestra en la tabla 3.1.

Tabla 3.1: Dimensiones en cada celda de la arquitectura ResNet con 3 celdas. En cada celda los mapas de características tienen las mismas dimensiones y profundidad.

Bloque	Dimensiones	Número de filtros
Conv2D	[32,32]	16
Celda 1	[32,32]	64
Celda 2	[16,16]	128
Celda 3	[8,8]	256

En la figura 3.8 se muestra la **incorporación del bloque Gabor a la red ResNet**. Este bloque es concatenado a la salida de la celda 1. Luego, al bloque Gabor se le aplica la operación de *pooling* promedio para reducir la dimensión del mapa de características a la mitad y ser incorporados a la celda 2.

La primera capa convolucional de cada una de las celdas de una red ResNet realiza la convolución con *stride*=2, para reducir la dimensión de los mapas de características en la nueva etapa a la mitad.

El mapa de salida de la celda 1 tiene dimensiones [32,32,64] y este se le concatena el bloque Gabor de dimensiones [32,32,32], el resultado de esta operación es un mapa de características de [32,32,96], que denominaremos X_1. El cual ingresa al bloque residual 1, a esta entrada se le aplican las operaciones; BN-ReLU-Conv2D(1,stride=2) definida como $F(x)$; BN-ReLU-Conv2D(3) definida como $G(x)$; BN-ReLU-Conv2D(1) definida como $H(x)$. El número de filtros en cada una de las convoluciones de la celda 2 es 64.

La salida del bloque residual 1 es:

$$X_2 = H(G(F(X_1))) + P(X_1) \tag{3.1}$$

Donde $P(X_1)$ es la operación Conv2D(1,stride=2).

Para los experimentos se utiliza utiliza como red pequeña ResNet con 20 capas (que es denominada ResNet20) con 3 celdas residuales y 2 bloques residuales por celda. Como red

grande se utiliza ResNet con 110 capas (ResNet110) con 3 celdas residuales y 12 bloques residuales.

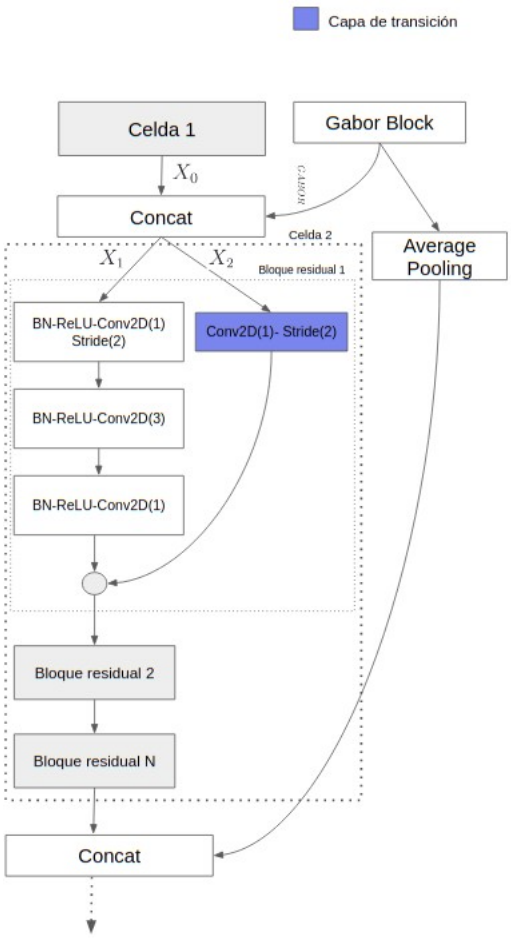


Figura 3.8: Incorporación de características Gabor en ResNet v2. Las características Gabor se incorporan a la salida de celda 1, donde el bloque Gabor es concatenado a X_0.

3.3. Conjunto de datos

3.3.1. CIFAR10

El conjunto de datos CIFAR10 (C10) consiste en imágenes de 10 clases de 32×32 píxeles. Los conjuntos de entrenamiento y prueba contienen 50.000 y 10.000 imágenes, respectivamente. Se conservan 5.000 imágenes de entrenamiento como conjunto de validación.

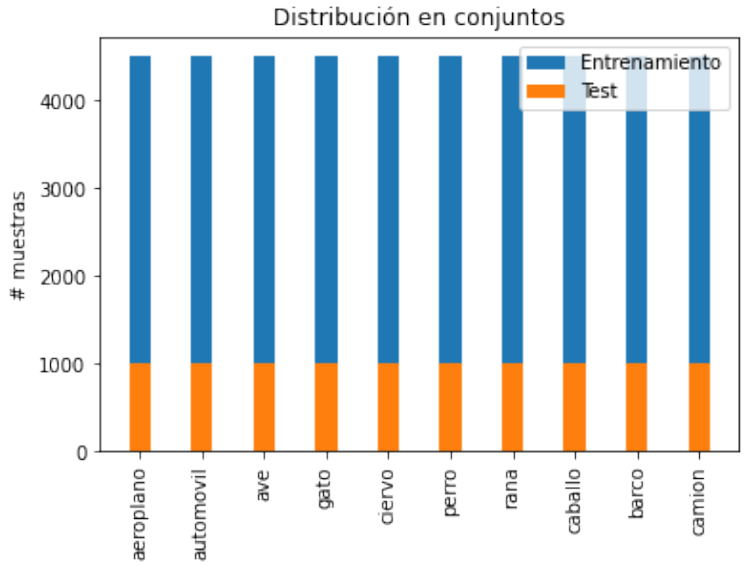


Figura 3.9: Distribución de cada una de las clases de CIFAR10 en los conjuntos de entrenamiento y prueba.

3.3.2. Data augmetation

Además, se utiliza data *augmentation* estándar (reflejo y traslación). Este data *augmentation* es denotado por un símbolo " + " al final del nombre del conjunto de datos (C10+), se muestra un ejemplo en la figura 3.10.

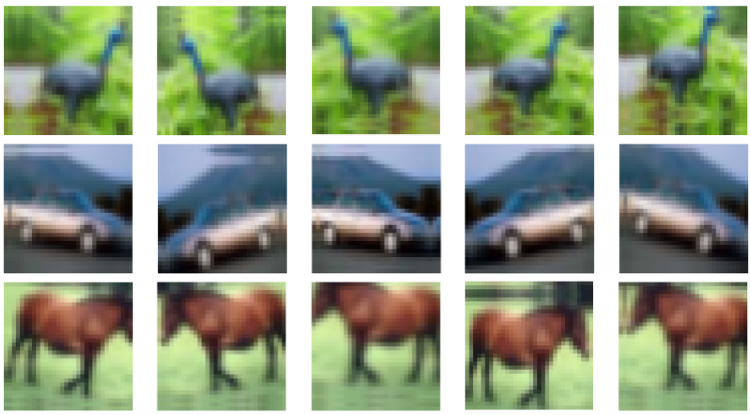
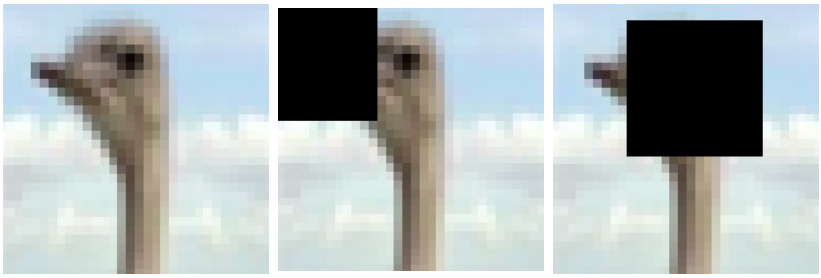


Figura 3.10: Ejemplo de técnica de reflejo para data augmentation simple en CIFAR10.

Por último al data *augmentation* estándar se le suma *cutout*, que consiste en remover secciones de la imagen de entrada, aumentando efectivamente el conjunto de datos con versiones parcialmente ocluidas del conjunto de datos existente como se muestra en la figura 3.11. En [23] se muestra que para la red ResNet el mayor desempeño se logra utilizando un área de oclusión de 16×16 píxeles. Este data *augmentation* con *cutout* es denotado por un símbolo " + +" al final del nombre del conjunto de datos (C10++).

Para preprocesamiento, se normalizan los datos usando la media del canal y la desviación estándar.

En la figura 3.9 se muestra la distribución de las 50.000 muestras de entrenamiento y de las 10.000 muestras de prueba.



(a) CIFAR10 (b) Cutout parcial (c) Cutout completo

Figura 3.11: Ejemplo cutout de 16x16 píxeles en CIFAR10.

3.4. Entrenamiento

Los entrenamientos se realizan utilizando inicialización normal He, un optimizador RMSprop y un batch de tamaño 32. La tasa de aprendizaje inicial es 10^{-3} y está ajustando para ser reducida después de las épocas 80, 120, 160 y 180, en un factor de 10^{-1}, 10^{-2}, 10^{-3} y $0{,}5 \times 10^{-3}$ respectivamente.

Capítulo 4

Resultados

4.1. Extracción de características

4.1.1. Filtros Gabor

Para determinar los parámetros y dimensiones para los filtros Gabor se realizan 3 experimentos, donde se modifica el tamaño del conjunto de filtros y se ajustan los parámetros con el entrenamiento conjunto de PSO y actualización de peso de la red. Los filtros Gabor son inicializados con σ =0.30,γ =0.5 y λ=10.

Tabla 4.1: Selección de tamaño de filtros Gabor y sintonización de parámetros. Se reporta el *accuracy* en el conjunto de prueba.

Nombre	Filtros	Parámetros	Accuracy prueba
Experimento 1	[5,5],[7,7],[9,9],[11,11]	σ: 0.3228, γ: 0.3511, λ: 24.6204	47.55
Experimento 2	[7,7],[9,9],[11,11],[13,13]	σ: 0.3722, γ: 0.4939, λ: 21.3390	45.34
Experimento 3	[9,9],[11,11],[12,12],[15,15]	σ: 0.2830, γ: 0.9086, λ: 13.4505	45.98

De acuerdo a los resultados mostrados en la tabla 4.1, el mejor desempeño se alcanza utilizando el conjunto de filtros de tamaño [5,5],[7,7],[9,9] y [11,11], con parámetros para los filtros Gabor de σ: 0.3228, γ: 0.3511, λ: 24.6204. **Estos parámetros se utilizan para obtener la características para cada una de las redes.**

4.2. DenseNet

4.2.1. Determinar factor de compresión

Se evalúa el desempeño de de la red incorporando características Gabor determinados a partir de los experimentos en 4.1 con diferentes factores de compresión en las capas de compresión utilizando la red DenseNet20. En la tabla 4.2 se reportan los resultados en el conjunto de prueba.

Tabla 4.2: *Accuracy* en el conjunto de prueba para diferentes factores de compresión θ en DenseNet20 con filtros Gabor.

θ	accuracy de prueba
1.0	85.70
0.6	85.86
0.5	85.96
0.3	85.50

4.2.2. Incorporación características

Se utiliza el factor de compresión que logra los mejores resultados, que es $\theta = 0{,}5$.

En la tabla 4.3 se muestra el *accuracy* en el conjunto de prueba para la estructuras de DenseNet20, DenseNet20 con filtros Gabor y DenseNet20 con filtros convencionales (En el apéndice A se muestran los tiempos de entrenamiento de cada una de estas redes). Además, se muestran el *accuracy* en el conjunto de prueba al utilizar data *augmentation* simple (C10+) y data *augmentation* con *cutout* (C10++).

Tabla 4.3: *Accuracy* en conjunto de prueba y número de parámetros para DenseNet20, DenseNet 20 con filtros Gabor y DenseNet20 con filtros con convencionales, utilizando los conjuntos de datos CIFAR10 (C10), CIFAR10 con data *augmentation* (C10+) y CIFAR10 con *cutout*(C10++). Se reporta el promedio±std de 3 ejecuciones.

Nombre	C10	C10+	C10++	Parámetros
DenseNet20	84.96±0.07	87.01±0.25	86.85±0.23	71.170
DenseNet20 filtros Gabor	85.83±0.10	87.34±0.17	87.03±0.07	87.253
DenseNet20 filtros convencionales	82.97±0.47	87.43±0.59	86.82±0.24	88.930

El mayor desempeño al utilizar el conjunto de datos normal (C10) se logra al incorporar los filtros Gabor a la red DenseNet20. En cambio, al utilizar data *augmentation* simple (C10+) se observa que el mayor desempeño se logra utilizando la red DenseNet con filtros convencionales. Al utilizar data *augmentation* con *cutout* se observa en la tabla 4.3 que el mejor desempeño se logra con DenseNet20 filtros Gabor.

Se utiliza el test estadístico **t test** para analizar si los resultados de las DenseNet con filtros Gabor son superiores a las DenseNet normal. Se utiliza un t-test con valor de significancia estadística de 0.05, con hipótesis nula que las medias son iguales. Se obtiene lo siguiente:

1. CIFAR10: Se obtiene un p valor 0.0002, por lo que se rechaza la hipótesis nula y las medias son diferentes. Obteniendo en DenseNet20 filtros Gabor mejor desempeño.
2. CIFAR10+ : Se obtiene un p valor 0.1317, por lo que no es posible rechazar la hipótesis nula.
3. CIFAR10++: Se obtiene un p valor de 0.2645, por lo que no es posible rechazar la hipótesis nula.

Por lo que solo utilizando CIFAR10 se logra obtener medias distintas.

4.2.3. DenseNet100

En la tabla 4.4 se reportan el *accuracy* del conjunto de prueba para 3 ejecuciones de DensNet100 y DenseNet100 con filtros Gabor. Además, se muestra el número de parámetros de cada una de las redes.

Tabla 4.4: *Accuracy* en conjunto de prueba y número de parámetros para DenseNet100 y DenseNet100 con filtros Gabor. Se reporta el promedio±std de 3 ejecuciones

Nombre	Accuracy test	Parámetros
DenseNet100	91.70±0.25	797,788
DenseNet100 filtros Gabor	91.24±0.35	900,301

De la tabla 4.4 se desprende que el desempeño de la red DenseNet100 es superior al obtenido por la red DenseNet100 con filtros Gabor. Se realiza un t-test para comparar las medias y se obtiene un p valor de 0.1376, por lo que no es posible rechazar la hipótesis nula.

4.3. ResNet

4.3.1. Incorporación de características

En la tabla 4.5 se muestra el *accuracy* en el conjunto de prueba para la estructuras de ResNet20, ResNet20 con filtros Gabor y ResNet20 con filtros convencionales (En el apéndice A se muestran los tiempos de entrenamiento de cada una de estas redes). Además, se muestran el *accuracy* en el conjunto de prueba al utilizar data augmentation simple (C10+) y data augmentation con cutout (C10++).

Tabla 4.5: *Accuracy* en conjunto de prueba y número de parámetros para ResNet20, ResNet 20 con filtros Gabor y ResNet20 con filtros con convecionales, utilizando los conjuntos de datos CIFAR10 (C10), CIFAR10 con data augmentation (C10+) y CIFAR10 con cutout(C10++). Se reporta el *accuracy* promedio±std de 3 ejecuciones.

Nombre	C10	C10+	C10++	Parámetros
ResNet20	86.59±0.85	91.01±0.10	91.18±0.05	574.090
ResNet20 filtros Gabor	86.35±0.19	90.96±0.10	91.11±0.08	592.781
ResNet20 filtros convencionales	87.27±0.28	91.27±0.20	91.16±0.19	595.018

El mayor desempeño con el conjunto de datos CIFAR10 se obtiene con la red ResNet20 con filtros convencionales, superando por un amplio margen a las demás arquitecturas. Al utilizar data *augmentation* simple(C10+) el mayor desempeño también se logra con la red ResNet20 con filtros convencionales. Con data *augmentation* con *cutout* (C10++) el mayor desempeño se obtiene la red ResNet20. Entonces, para ningunas de las variantes del conjunto de datos el mejor resultados se logra con ResNet20 con filtros Gabor.

Para verificar que las medias de los experimentos son distintas entre ResNet20 y ResNet20 con filtros Gabor, se realiza un t-test con valor de significancia $p = 0{,}5$. Se obtiene lo siguiente:

1. CIFAR10: Se obtiene un p valor de 0.6581, por lo que no es posible rechazar la hipótesis nula.
2. CIFAR10+: Se obtiene un p valor de 0.5734, por lo que no es posible rechazar la hipótesis nula.
3. CIFAR10++: Se obtiene un p valor de 0.2681, por lo que no es posible rechazar la hipótesis nula.

De acuerdo al t-test, la incorporación de filtros Gabor no aumenta el desempeño de la red.

4.3.2. ResNet110

En la tabla 4.6 se reportan el *accuracy* de prueba para 3 ejecuciones de ResNet110 y ResNet110 con filtros Gabor. Además, se muestra el número de parámetros de cada una de las redes.

Tabla 4.6: *Accuracy* en el conjunto de prueba y número de parámetros para ResNet110 Y ResNet110 con filtros Gabor. Se reporta el *accuracy* promedio±std de 3 ejecuciones.

Nombre	Accuracy prueba	Parámetros
ResNet110	88.20±0.26	3,323,210
ResNet100 filtros Gabor	88.32±0.20	3,341,901

Se realiza un t-test para evaluar si las medias son diferentes y se obtiene un p valor de 0.5607, por lo que no es posible rechazar la hipótesis nula de que las medias son iguales.

Capítulo 5

Discusión

En este trabajo se presenta una modificación a la arquitectura de redes neuronales convolucionales que alcanzan resultados estado del arte, agregando información de características *handcrafted*, específicamente características que son obtenidas a partir de filtros Gabor, en diferentes etapas de la red para mejorar el desempeño en la tarea de clasificación de CIFAR10.

Se modifican 2 arquitecturas, las redes DenseNet y ResNet. Se utilizan versiones pequeñas de estas redes, con 20 capas de profundidad (DenseNet20 y ResNet20) y se evalúa su desempeño en la tarea de clasificación del conjunto de datos CIFAR10. Las características son agregadas de manera tal que no modifiquen la estructura de la red, manteniendo el número de mapas de salidas y dimensiones en cada una de la capas convolucionales. Luego, la mejor configuración es utilizada en versiones más profundas de las redes DenseNet y ResNet, con 100 y 110 capas, respectivamente.

5.1. DenseNet

5.1.1. DenseNet20

Al incorporar las características Gabor en la red DenseNet20 se observa de los resultados en la tabla 4.3 que el desempeño aumenta en comparación con la red DenseNet original de 84.96 % a 85.83 % por lo que agregar las características Gabor en esta arquitectura provee de información complementaria a la red. Además, se realiza un t-test que respalda el hecho de que las medias son diferentes estadísticamente.

Esta mejora en el desempeño en la tarea de clasificación se podría dar por el aumento en la cantidad de parámetros en la red, pasa de tener 71,170 a 87.253 parámetros. Para verificar que no se debe al aumento de parámetros, se cambia todos los filtros Gabor por filtros convencionales (pesos de la red). La red DenseNet con filtros convencionales tiene 88.930 parámetros, lo que significa un aumento del 1.92 % con respecto a la red con filtros Gabor, esta red logra un desempeño de 82.97 %, que es mucho menor al logrado por la red DenseNet

con filtros Gabor, incluso menor al logrado por la red original. Entonces, se concluye que la mejora en el desempeño se debe a que los filtros Gabor aportan información complementaria a la red y no al aumento de parámetros.

Al entrenar las red DenseNet20 con técnicas de data *augmentation* se observa que la brecha entre las arquitecturas propuestas disminuye, existiendo un traslape entre sus resultados y ya no es tan evidente la mejora en el desempeño. De hecho, al realizar un test estadístico para comparar las medias se obtiene p valores altos (mayores a 0.05), por lo que no hay diferencia estadística entre las medias.

Esto se justifica dado que con esta nueva forma de entrenamiento, con data *augmentation*, la red es capaz de aprender las características de bajo nivel que antes eran suministradas por los filtros Gabor.

5.1.2. DenseNet100

Se desarrolla un experimento para evaluar el desempeño de la red DenseNet utilizando 100 capas. La red DenseNet100 con filtros Gabor no logra superar el desempeño de la red DenseNet100, por lo que esta arquitectura al ser mucho más profunda logra aprender más características y dentro de ellas las suministradas por los filtros Gabor.

Que el desempeño para el problema de clasificación no aumente se justifica con el hecho que los filtros Gabor proveen de información redundante.

5.2. ResNet

5.2.1. ResNet20

En la red ResNet20, con una arquitectura compuesta de celdas residuales, las características se incorporan de manera tal que no modifiquen la estructura de la red. De los resultados se desprende, que la incorporación de características con filtros Gabor no permite un aumento en el desempeño.

En la red ResNet20, la nueva información agregada por los filtros Gabor puede ser redundante e incluso aumentar el ruido en la red. En cambio, cuando en vez de agregar filtros Gabor se agregan filtros convencionales (pesos de la red), se aumenta el desempeño con respecto a la red original. Esta mejora en el desempeño se justifica con el hecho de que agregar los filtros convencionales representa un aumento en el número de parámetros del 3.6 %, por lo que hay mayor número de parámetros que pueden ser ajustados durante la etapa de entrenamiento y además, la presencia de celdas residuales, logra evitar problemas como el gradiente desvaneciente, permitiendo ajustar correctamente estos nuevos parámetros, con el objetivo de alcanzar un mejor desempeño en la tarea de clasificación.

Al entrenar las red ResNet20 con técnicas de data augmentation se observa que no hay un aumento en el desempeño con respecto a la red normal al agregar filtros Gabor, y al agregar filtros convencionales. Esto puede ser por el hecho de que la arquitectura original ya puede aprender características suficientemente robustas y complementarias, entonces al agregar nuevos bloques con información, estos no son útiles.

5.2.2. ResNet 110

La ResNet110 logra obtener un mejor promedio en las ejecuciones que la red ResNet110 con filtros Gabor. Pero, el t-test indica que sus medias son iguales. Entonces, en esta arquitectura no se logra que los filtros Gabor provean de información que sea relevantes y/o no redundantes, tal que una red de esta profundidad no las pueda aprender por si solas.

5.3. Comparación DenseNet y ResNet

En DenseNet se logra que la incorporación de características proveniente de filtros Gabor ayuda a lograr un mejor desempeño para la tarea de clasificación de CIFAR10. Pero, esto no se logra en ResNet.

Cuando la red se entrena sin ninguna técnica de *data augmentation*, el mejor resultado se logra con DenseNet20 filtros Gabor, alcanzándose un 85,83 % promedio en el *accuray* en el conjunto de prueba.

En DenseNet las características Gabor son utilizadas en cada una de las capas convolucionales, debido a la arquitectura propia de los bloques densos. En cambio en ResNet las características Gabor solo aparecen en el primer bloque residual de cada una de las celdas. La retro-alimentación de características en DenseNet permite que la red aprenda a reusar las características para generar representaciones internas y reducir la redundancia de características.

Capítulo 6

Conclusión

En este trabajo se presenta un metodología para la incorporación de características *handcrafted* en redes neuronales convolucionales, específicamente características Gabor. Estas características se incorporan a las características generadas por la red y además son sintonizadas de acuerdo a las necesidades de la red, para el problema de clasificación CIFAR10.

A partir de los experimentos, se muestra que incorporar características *handcrafted* a redes neuronales convolucionales puede aportar para aumentar el desempeño en tareas de clasificación de imágenes. Como en el caso de la red DenseNet, en donde al utilizar una versión pequeña de esta red, la red DenseNet20, se logra mejorar el desempeño de 84,96 % a 85,83 % al incorporar características *handcradted* en zonas específicas de la red, para el problema de CIFAR10.

No en todas las variantes se logra mejorar el desempeño. Para ello se debe contar con una arquitectura que permita sacar provecho a las nuevas características incorporadas, reutilizándolas, como es en el caso de DenseNet, para que la red no aprenda durante el entrenamiento características que sean redundantes a las que se incorporan.

Al utilizar técnicas de generación de datos sintéticos como lo son data augmentation simple y data augmentation con *cutout*, no se logra un aumento del desempeño en comparación con la versión normal de las redes, al igual que al incorporar características en redes más profundas (DenseNet y ResNet).

Como trabajo futuro, se propone incluir mas métodos de extracción de características. Estos métodos de extracción de características deben conservar la posición espacial presentes en los mapas de caracterísiteristicas de las redes neuronales.

Bibliografía

[1] Z. Tianyu, M. Zhenjiang, and Z. Jianhu, "Combining cnn with hand-crafted features for image classification," in *2018 14th IEEE International Conference on Signal Processing (ICSP)*, pp. 554–557, Aug 2018.

[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," *CoRR*, vol. abs/1704.05796, 2017.

[3] G. Lee, Y. Tai, and J. Kim, "Deep saliency with encoded low level distance map and high level features," *CoRR*, vol. abs/1604.05495, 2016.

[4] L. Zheng, Y. Yang, and Q. Tian, "SIFT meets CNN: A decade survey of instance retrieval," *CoRR*, vol. abs/1608.01807, 2016.

[5] P. PROTIVASH, *Hybrid network of handcrafted and non-handcrafted features for computer vision classification taks*. PhD thesis, Department of computer science and engineering, The Pennsylvania state Univeristy, 2018.

[6] H. Wang, J. Hu, and W. Deng, "Face feature extraction: A complete review," *IEEE Access*, vol. 6, pp. 6001–6039, 2018.

[7] L. Wiskott, J.-M. Fellous, N. Krüger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," in *International Conference on Computer Analysis of Images and Patterns*, pp. 456–463, Springer, 1997.

[8] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image processing*, vol. 11, no. 4, pp. 467–476, 2002.

[9] V. Ghosal, P. Tikmani, and P. Gupta, "Face classification using gabor wavelets and random forest," in *2009 Canadian Conference on Computer and Robot Vision*, pp. 68–73, IEEE, 2009.

[10] S. Xie, S. Shan, X. Chen, X. Meng, and W. Gao, "Learned local gabor patterns for face representation and recognition," *Signal Processing*, vol. 89, no. 12, pp. 2333–2344, 2009.

[11] S. Zhang and X. Wang, "Human detection and object tracking based on histograms of oriented gradients," in *2013 Ninth International Conference on Natural Computation (ICNC)*, pp. 1349–1353, 2013.

[12] M. Wang and W. Deng, "Deep face recognition: A survey," *CoRR*, vol. abs/1804.06655, 2018.

[13] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, IEEE, 2017.

[14] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0588–0592, IEEE, 2017.

[15] D. Mishkin and J. Matas, "All you need is a good init," *CoRR*, vol. abs/1511.06422, 2016.

[16] J. F. Kolen and S. C. Kremer, *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*. IEEE, 2001.

[17] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *AISTATS*, 2011.

[18] R. Atienza, *Advanced deep learning with Keras: apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy tradients, and more*. Packt, 2018.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," *CoRR*, vol. abs/1603.05027, 2016.

[21] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.

[22] S. S. Sarwar, P. Panda, and K. Roy, "Gabor filter assisted energy efficient fast learning convolutional neural networks," *CoRR*, vol. abs/1705.04748, 2017.

[23] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.

Apéndice A

Tiempos de entrenamiento

Para el entrenamiento de las redes neuronales se utiliza el servicio gratuito **Google colab**, que permite acceso gratuito a capacidad de computo, en particular GPUs. En particular durante este trabajo se utilizaron las GPUs Tesla K80 y Tesla P100. Es importante indicar que esta GPUs son asignadas de acuerdo a disponibilidad y demanda.

A.1. DenseNet

Nombre	C10	C10+	C10++
DenseNet20	3:13:00	3:03:00	3:30:00
DenseNet20 filtros Gabor	3:33:00	5:00:00	3:30:00
DenseNet20 filtros convencionales	3:36:00	4:46:00	6:06:00
DenseNet100	16:23:00	-	-
DenseNet100 Gabor	18:46:00	-	-

Tabla A.1: Tiempos de entrenamientos aproximados HH:MM:SS de la red DenseNet utilizando conjunto de datos normal (C10), data *data augmentation* (C10+) y *cutout*(C10++).

A.2. ResNet

Nombre	C10	C10+	C10++
ResNet20	2:50:00	3:10:00	3:13:00
ResNet20 filtros Gabor	3:00:00	4:53:00	6:40:00
ResNet20 filtros convencionales	3:26:00	7:03:00	5:50:00
ResNet110	16:53:00	-	-
ResNet110 Gabor	15:00:00	-	-

Tabla A.2: Tiempos de entrenamientos aproximados HH:MM:SS de la red ResNet utilizando conjunto de datos normal (C10), data *data augmentation* (C10+) y *cutout*(C10++).