Escola Tècnica Superior d'Enginyeria ETSE-UV

Grado en Ciencia de Datos

VNIVERSITAT DE VALÈNCIA

Trabajo Fin de Grado

Mejora de Modelos de Clasificación mediante la Reutilización de Información de Capas de Normalización en Redes Neuronales

Autor:
Ander Góngora Allué

Tutores:
Valero Laparra Pérez-Muelas
Pablo Hernández Cámara

VNIVERSITAT DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria ETSE-UV

Trabajo Fin de Grado

Mejora de Modelos de Clasificación mediante la Reutilización de Información de Capas de Normalización en Redes Neuronales

Autor: Ander Góngora Allué

Tutores: Valero Laparra Pérez-Muelas
Pablo Hernández Cámara

Declaración de autoría:

Yo, Ander Góngora Allué, declaro la autoría del Trabajo Fin de Grado titulado "Mejora de Modelos de Clasificación mediante la Reutilización de Información de Capas de Normalización en Redes Neuronales" y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

Valencia, 20 de julio de 2024

Fdo: Ander Góngora Allué

Resumen:

El objetivo principal de este TFG es investigar el impacto de la reutilización de información de las capas de normalización en el rendimiento de modelos de clasificación basados en redes neuronales. Específicamente, se analizarán dos modelos (VGG16 y PerceptNet) y se compararán seis variantes: los modelos sin normalización, con Batch Normalization (BN), con BN reutilizando media y desviación estándar, con Instance Normalization (IN), con IN reutilizando media y desviación estándar, con Generalized Divisive Normalization (GDN), con GDN reutilizando el denominador de normalización. El estudio se enfocará en la clasificación de imágenes del dataset CIFAR100.

- Objetivo 1: Implementar y entrenar los modelos VGG16 y PerceptNet en la tarea de clasificación en CIFAR100.
- Objetivo 2: Modificar los modelos para incluir capas de BN, IN y GDN, así como sus variantes mejoradas (reutilización de media y desviación estandar), y comparar su rendimiento.
- Objetivo 3: Analizar si la reutilización de parámetros de normalización mejora el aprendizaje y la precisión de clasificación.

Agradecimientos:

Quisiera expresar mi más sincero agradecimiento a todas las personas que me han apoyado y han ayudado a que haya realizado este Trabajo de Fin de Grado.
En primer lugar, a mi familia, por el apoyo constante que me han dado y su comprensión.Su confianza y motivación ha sido un pilar esencial para mí.
A mis amigos, por estar a mi lado, por sus palabras de ánimo y por todas las veces que me han ofrecido ayuda.
Por último, quiero agradecer a mis tutores de TFG, Valero Laparra Pérez-Muelas y Pablo Hernández Cámara. En especial a Pablo, que me ha ayudado en incontables ocasiones y cuya orientación ha sido crucial para realizar el trabajo.

Índice general

Capítulo 1

Introducción

1.1. Introducción

Las redes neuronales son fundamentales en la Ciencia de Datos gracias a su capacidad de encontrar relaciones complejas y no lineales en los datos. Se utilizan en una gran cantidad de tareas, ya sea de predicción o clasificación. Contienen estructuras abstractas y complicadas, gracias a las que pueden capturar patrones y características de alto nivel que algunos modelos más simples no son capaces. Las redes neuronales profundas han ayudado en la Ciencia de Datos aportando nuevas capacidades de aprender y adaptarse a diferentes datos. Asimismo, son una herramienta muy poderosa mediante la que se pueden abordar problemas complejos y desarrollar soluciones innovadoras.

En las redes neuronales, se suelen utilizar ciertas capas de normalización. Estas capas ayudan a que el modelo sea más estable y no varíe tanto según los datos de entrada, es decir, que sea más capaz de generalizar. El objetivo es regularizar y escalar los datos de cada capa en un rango determinado. Así, puede ayudar a evitar problemas de saturación de funciones de activación evitando que se ralentice el aprendizaje y obtener un bajo rendimiento del modelo. Existen diferentes capas de normalización como Normalización por Lotes (BN [1]), Normalización por Instancias (IN [2]), Normalización Divisiva Generalizada (GDN [3]), etc.

Lo que sucede con estas capas de normalización es que, al escalar los datos a un rango, se pierde parte de la información que contenían los datos. Por ejemplo, con una normalización básica como restarle la media y dividirlo entre la desviación estándar, se pierde el valor de la media y la desviación estándar que contenían los datos. Por ello, es que ha surgido la idea de este estudio.

Se quiere observar si, en vez de omitir o eliminar esa información al normalizar los datos, si se reutiliza en la red neuronal podría mejorar los resultados. Por eso, se ha pensado en utilizar algunos parámetros de las capas de normalización en las últimas capas de las redes neuronales.

1.2. Motivación

Las redes neuronales están avanzando rápidamente, sin embargo, a medida que las arquitecturas de las redes neuronales son más complejas, también son más difíciles los

desafíos que surgen ya sea en su entrenamiento o en su rendimiento. Para mejorar la estabilidad y velocidad, una técnica muy empleada es la normalización mediante diferentes capas de normalización.

La motivación principal de este trabajo es investigar cómo la información eliminada en las capas de normalización puede ser reutilizada en las capas finales de la red neuronal para mejorar el rendimiento del modelo. La hipótesis parte de que al reintroducir esta información, se puede obtener un modelo más preciso y robusto sin dejar de lado los beneficios que ofrecen las técnicas de normalización.

Importancia del problema

- **Mejora del rendimiento de los modelos**: La información perdida durante la normalización puede contener características relevantes que si se reincorporan, podrían mejorar la precisión y generalización del modelo. La búsqueda de mejora de redes neuronales es un desafío al que podría contribuir.

- **Optimización del entrenamiento**: Como se siguen utilizando las capas de normalización, el modelo aprovechará las ventajas que ofrecen. Además, la reutilización de información eliminada podría contribuir a la optimización del entrenamiento, así, las redes neuronales podrían converger más rápido.

- **Contribución al conocimiento científico**: Investigar entorno a técnicas para mitigar la pérdida de información en las capas de normalización puede abrir nuevos estudios relacionados con nuevas arquitecturas de redes neuronales.

1.3. Objetivos

Este trabajo de fin de grado, se centra en la mejora de redes neuronales que utilizan capas de normalización. En una gran cantidad de arquitecturas de redes neuronales, se emplean capas de normalización. En estas capas, tras normalizar los datos, parte de la información se elimina. Asimismo, si esta información que se desecha en estas capas de normalización se utiliza como información adicional en capas futuras, podría mejorar los resultados que obtiene la red neuronal. Gracias a esta aproximación, esa información que se eliminaba tras la normalización, se vuelve a utilizar en capas posteriores.

Para cumplir con el objetivo, se realizarán diferentes metas:

Analizar el impacto de la normalización

Evaluar diferentes técnicas de normalización y observar si, mediante la utilización de estas capas, el rendimiento de las redes neuronales mejora o convergen más rápido.

Desarrollar técnicas para reutilizar la información

Proponer métodos para capturar y reintroducir la información eliminada en las capas de normalización en las capas finales del modelo.

Implementar y evaluar modelos mejorados

Implementar la reutilización de la información en los modelo y comparar los resultados con los modelos sin la implementación mediante métricas.

Los resultados esperados son demostrar que, a través de la reutilización de la información, los modelos ofrecen un rendimiento superior.

1.4. Organización de la memoria

La memoria comienza con la explicación de las arquitecturas que se utilizarán (VGG16 y PerceptNet) y de las capas de normalización de las que se analizará la reutilización de la información (Batch Normalization, Instance Normalization y Generative Divisive Normalization).

Posteriormente, se comentarán los modelos que se utilizarán en el estudio, una breve explicación de su arquitectura junto a su funcionamiento y cómo se ha realizado la reutilización de la información.

Después, se presentarán los resultados obtenidos tras el entrenamiento de los modelos junto a sus métricas (loss, precision, accuracy, AUC, recall). En esta sección, también se compararán los modelos con la reutilización de la información de las capas de normalización y sin ella, así, se podrá observar si esta técnica es capaz de aportar mejoras a los modelos.

Finalmente, se realiza una conclusión final del trabajo realizado cerrando el trabajo.

Como información extra se encuentra un apéndice donde se dan más detalles sobre el desarrollo de los gráficos de las arquitecturas de las redes vistas en la memoria.

Capítulo 2

Estado del arte

2.1. Análisis de aplicaciones similares

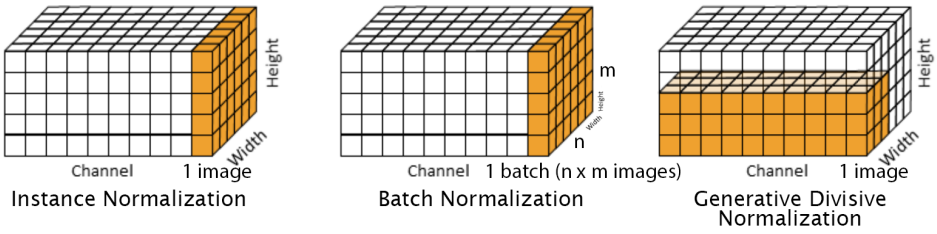

Figura 2.1: Las diferentes normalizaciones que se analizarán. Imagen modificada, extraída de [4].

2.1.1. Batch Normalization

La capa Batch Normalization (BN) [1] ha sido utilizada en muchas arquitecturas de redes neuronales profundas. Mediante el Batch Normlaization, se normalizan los datos a lo largo de cada lote. Gracias a eso, se puede utilizar tasas de aprendizajes muy superiores y no es tan importante la inicialización. Actúa también como un regularizador que puede llegar a eliminar la necesidad de utilizar Dropout. El Batch Normalization lo que hace es lo siguiente para cada x en los mini-batch[1]:

Primero, se calcula la media y la varianza para el batch (m es el número de muestras del batch):

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{(media del batch)}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad \text{(varianza del batch)}$$

[1]Fórmula extraída del algoritmo de Batch Normalization [1].

Después, se normaliza cada muestra dentro del miso batch:

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

A pesar de los beneficios que ofrece el BN, puede eliminar información relevante en los datos de entrada. Como se puede observar en la fórmula, se resta la media del Batch μ_B y se divide entre la raíz cuadrada de la desviación estándar del Batch σ_B^2 más un poco de ruido ϵ (actúa como regularización y ayuda a reducir el overfitting). Tanto μ_B como σ_B^2 son los parámetros de interés en este proyecto, ya que van a ser los que, en capas posteriores, se vuelvan a introducir y serán procesados.

2.1.2. Instance Normalization

El método Instance normalization [2], suele utilizarse en la generación de imágenes y en la transferencia de estilo, aunque se puede utilizar en diferentes campos. En este caso, se normaliza imagen a imagen, a diferencia de utilizar Batch Normalization que normalizada todas las imágenes de cada lote a la vez[2]:

Se inicia calculando la media y la varianza para cada instancia (solo una entrada):

$$\mu_n = \frac{1}{HW}\sum_{w=1}^{W}\sum_{h=1}^{H} x_{n,w,h} \quad \text{(media de una muestra)}$$

$$\sigma_n^2 = \frac{1}{HW}\sum_{w=1}^{W}\sum_{h=1}^{H}(x_{n,w,h} - \mu_n)^2 \quad \text{(varianza de una muestra)}$$

Donde H y W son la altura y el ancho de la entrada y $x_{n,w,h}$ es el valor del píxel en la posición (w, h) de la n-ésima entrada.

Posteriormente, se normalizan las entradas:

$$\hat{x}_{n,w,h} = \frac{x_{n,w,h} - \mu_n}{\sqrt{\sigma_n^2 + \epsilon}}$$

En este caso, la información de interés que será reutilizada en las capas posteriores serán las medias y varianzas de cada muestra (μ_n y σ_n^2).

[2]Fórmula simplificada de Instance Normalization [2].

2.1.3. Generalized Divisive Normalization

Las Generalized Divisive Normalization (GDN) [3], son una transformación paramétrica no lineal utilizada para modelar la densidad de datos de imágenes. En estas capas de normalización, se emplea la siguiente fórmula[3]:

$$y_i = \frac{z_i}{(\beta_i + \sum_j \gamma_{ij} |z_j|^{\alpha_{ij}})^{\epsilon_i}}$$

donde:

$$z = Hx$$

- y_i: Es la i-ésima salida de la transformación GDN.
- z_i: Es la i-ésima entrada, obtenida mediante $z = Hx$, donde H es una matriz de transformación lineal y x es el vector de entrada.
- β_i: Es un parámetro positivo que actúa como un sesgo o desplazamiento.
- γ_{ij}: Es un peso que determina la contribución de $|z_j|^{\alpha_{ij}}$ al i-ésimo denominador.
- α_{ij}: Es un exponente que controla la no linealidad de la entrada z_j en la normalización.
- ϵ_i: Es otro exponente que ajusta la escala de la normalización.

La transformación GDN es una generalización de la normalización divisiva clásica, dando como resultado una forma de normalización local. Asimismo, se trata de una normalización local ya que ajusta cada respuesta en función de un conjunto cercano de respuestas. Esto es diferente a la normalización global porque no se aplica la misma medida de normalización a todos los datos. Gracias a ello, es capaz de ajustar cada píxel o característica en función de sus vecinos cercanos en vez de utilizar una media y desviación estándar calculada sobre toda la imagen.

En el caso de las GDN, lo interesante para reutilizarlo en las capas posteriores será el denominador, es decir, $(\beta_i + \sum_j \gamma_{ij} |z_j|^{\alpha_{ij}})^{\epsilon_i}$. La imagen se divide píxel a píxel (z_i) entre el denominador. Al dividir la imagen, esta información se pierde en el proceso. Por ello, es interesante reutilizar dicha información en las últimas capas de la red neuronal.

[3]Fórmula extraída de Density Modeling of Images Using a Generalized Normalization Transformation (*3.1 Proposed Generalized Divisive Normalization (GDN) Transform)* [3].

2.2. Tecnologías

Se han analizado 2 arquitecturas de redes neuronales diferentes. Ambas arquitecturas tienen como finalidad clasificar imágenes. Se ha utilizado el conjunto de datos CIFAR-100 [5], conjunto que cuenta con 60.000 imágenes con una dimensión de 32x32 píxeles y a color (32x32x3). Se ha dividido en 3 conjuntos diferentes:

- Train: 40.000 imágenes
- Validation: 10.000 imágenes
- Test: 10.000 imágenes

Asimismo, se han entrenado las redes neuronales con el conjunto "train" y se han validado con el conjunto "validation". Posteriormente, se han analizado todos los modelos en el conjunto test para realizar una comparación exhaustiva de los resultados.

Principalmente se han utilizado las arquitecturas VGG16 y PerceptNet. Estos modelos han sido modificados ya sea añadiendo o cambiando capas de normalización como reintroduciendo la información de las capas de normalización en las últimas etapas de los modelos. Se explicará en detalle en otras secciones.

2.2.1. VGG16

La arquitectura VGG16 [6] consta de 16 capas principales, de ahí su nombre. Estas capas incluyen 13 capas convolucionales y 3 capas densas (fully connected). Esta red es una red de clasificación que se utilizó para la clasificación de imágenes originalmente con ImageNet. Para este experimento, se ha utilizado una versión más reducida explicada más adelante 3.1.1.

Arquitectura

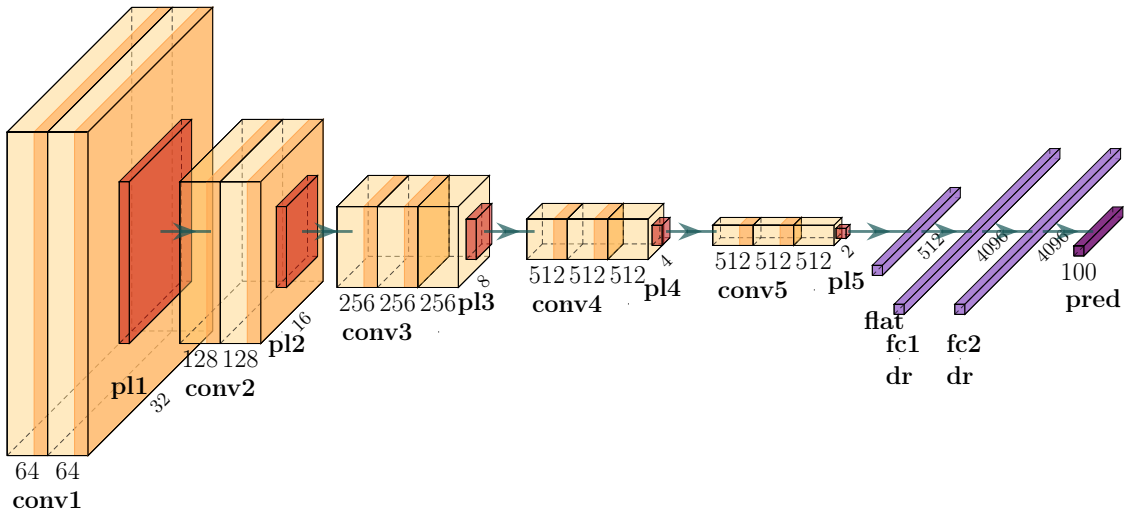

Figura 2.2: Estructura de la red neuronal VGG16 [6].

La red sigue un diseño simple, utiliza únicamente convoluciones de 3×3 píxeles con padding para mantener las dimensiones espaciales, seguidas de capas de pooling 2×2 con stride de 2. Gracias al pooling, el tamaño de la imagen se va disminuyendo a la mitad y gana en profundidad, se duplica. Parte de 64 de profundidad y se va duplicando 2.2.

Capas Convolucionales

Las capas convolucionales empleadas tienen un filtro de (3×3). Se organizan en bloques y cada bloque es seguido por una capa de pooling. Los primeros dos bloques tienen 2 capas convolucionales cada uno, mientras que los tres bloques siguientes tienen 3 capas convolucionales cada uno.

Capas de Pooling

Después de cada bloque de convoluciones, se aplica una capa de max pooling con un filtro (2×2) para reducir la dimensionalidad espacial.

Capas Densas

Después de las capas convolucionales y de pooling, la red tiene 3 capas densas. Las dos primeras capas tienen 4096 neuronas cada una y la última capa tiene 100 neuronas (correspondientes al número de clases que tiene CIFAR-100), con una capa softmax para la clasificación.

Funciones de activación

Todas las capas convolucionales y densas utilizan la función de activación ReLU (Rectified Linear Unit), que ayuda a introducir no linealidades en la red y acelera la convergencia del entrenamiento. La única capa que cambia es la última, que utiliza la función de activación “softmax”.

2.2.2. PerceptNet

La arquitectura PerceptNet [7] es una red diseñada para la normalización de características y para mejorar la representación de imágenes.

Arquitectura

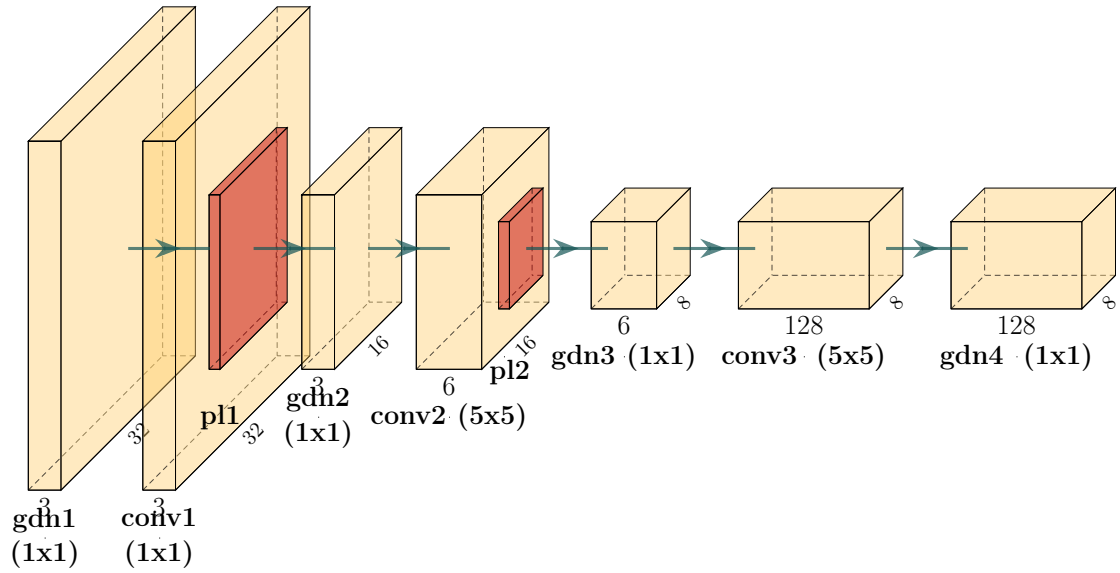

Figura 2.3: Estructura de la red neuronal PerceptNet [7].

La red sigue un diseño específico utilizando la transformación de normalización generalizada (GDN) y capas convolucionales de varios tamaños para procesar y normalizar las características de entrada. La arquitectura se muestra en la figura 2.3.

Capas GDN

La capa GDN (Generalized Divisive Normalization) [3] es una capa que normaliza las características de entrada. Se utilizan exactamente 4 capas GDN con un filtro de 1×1. Estas capas son esenciales para ajustar la distribución de las características de entrada, así, se normalizan adecuadamente los datos para las siguientes etapas.

Capas Convolucionales

PerceptNet utiliza varias capas convolucionales con diferentes tamaños de kernel. En particular, usa kernels de 1×1 y 5×5 para extraer características en diferentes escalas:

- Convolución 1×1 con 3 filtros
- Convolución 5×5 con 6 filtros
- Convolución 5×5 con 128 filtros

Estas capas convolucionales permiten a PerceptNet capturar los patrones en las imágenes, logrando una representación de las características visuales. Las convoluciones 1×1 son útiles para la transformación lineal de las características y las convoluciones 5×5 son efectivas para capturar estructuras más complejas.

Capas adicionales

La arquitectura PerceptNet es muy útil para la extracción de características. Sin embargo, para clasificación, es necesario añadir alguna capa extra:

1. **Capa de Pooling**: Se añade una capa de Global Average Pooling para reducir la dimensionalidad y así procesar la información en las capas posteriores. Esta capa toma el promedio de todas las características en cada mapa de características reduciendo significativamente la cantidad de datos que se procesan en las siguientes capas.

2. **Capa Flatten**: Se aplanan las características para que puedan ser procesadas en la siguiente capa.

3. **Capa Dense**: Se añade una capa densa que tiene el número de clases diferentes que hay en el conjunto de datos, en este caso, 100. Esta capa genera una salida que se interpreta como probabilidades de pertenencia a cada una de las clases.

Ejemplo de Uso

PerceptNet ha sido utilizado en la evaluación de la calidad perceptual de imágenes distorsionadas. En particular, se entrenó en el conjunto de datos TID2008 y se evaluó en varios conjuntos de datos adicionales como TID2013, CSIQ, LIVE y BAPPS. En estos experimentos, PerceptNet demostró ser capaz de generalizar bien a distorsiones que no fueron vistas durante la fase de entrenamiento, mostrando un rendimiento fuerte y consistente [7]. Este ejemplo resalta la capacidad de PerceptNet para mejorar la representación de imágenes y su potencial en aplicaciones de evaluación de calidad de imagen.

Capítulo 3

Modelos

En este capítulo, se presenta un análisis detallado de los diferentes modelos empleados en el experimento y las técnicas de reutilización de información para cada red neuronal.

El principal desafío de este trabajo reside en el desarrollo de la arquitectura de redes neuronales capaz de reutilizar la información de las capas de normalización. Para ello, en algún caso, se tuvo que crear unas capas personalizadas (para IN y GDN).

Para evaluar el impacto de las capas de normalización, se siguieron los siguientes pasos:

1. **Creación de modelos**: Se crearon las diferentes arquitecturas de los modelos VGG16 y PerceptNet. Las diferentes arquitecturas variaban entre no usar capas de normalización, utilizar capas de normalización (BN, IN y GDN) y reutilizar la información de las capas de normalización.

2. **Entrenamiento**: Tras entrenar un número suficiente de épocas, se aplicaron los modelos a un conjunto test.

3. **Métricas**: Se emplearon diferentes métricas para evaluar los resultados de los modelos (accuracy, loss, precision, recall y AUC).

4. **Resultados**: Interpretación de los resultados obtenidos.

3.1. Creación de modelos

Para la creación de modelos, se han realizado varias pruebas. Inicialmente, se realizó el estudio con las arquitecturas bases, es decir, sin reducir el tamaño de las arquitecturas, sin reducir el número de filtros o cualquier otra variación que pudiera afectar al modelo.

Tras realizar el análisis, no se obtuvieron unos resultados óptimos debido a la complejidad de las arquitecturas, por ello, se decidió modificar las estructuras de las redes neuronales que se emplearon. Para ello, se eliminaron algunas capas y se redujeron el número de filtros en alguna capa.

Posteriormente, se observó que algunos modelos no convergían correctamente, es decir, no conseguían aprender en el entrenamiento. Para evitar ese problema y mejorar los modelos , se utilizaron en varias ocasiones "Dropout" para solventarlo.

A continuación, se explica de una manera más detallada las nuevas arquitecturas que se desarrollaron para realizar la experimentación final.

En las figuras de los modelos, hay varias siglas referentes a las diferentes capas que se han utilizado:

- cnv: Convolutional
- pl: Max Pooling
- do: Dropout
- flat: Flatten
- pred: Fully Connected (ultima capa, para predecir)
- bn: Batch Normalization
- in: Instance Normalization
- gdn: Generalized divisive normalization
- Avg Pool: Global Average Pooling

3.1.1. Nuevas arquitecturas a partir de VGG16

Para la arquitectura VGG16 con el conjunto de datos CIFAR 100 tiene demasiados parámetros, lo que ocasiona que no llegue a entrenar correctamente. Por ello, se han añadido varias capas dropout, se ha reducido el número de filtros, se ha eliminado el bloque final de convoluciones y se han eliminado varias capas fully conneced:

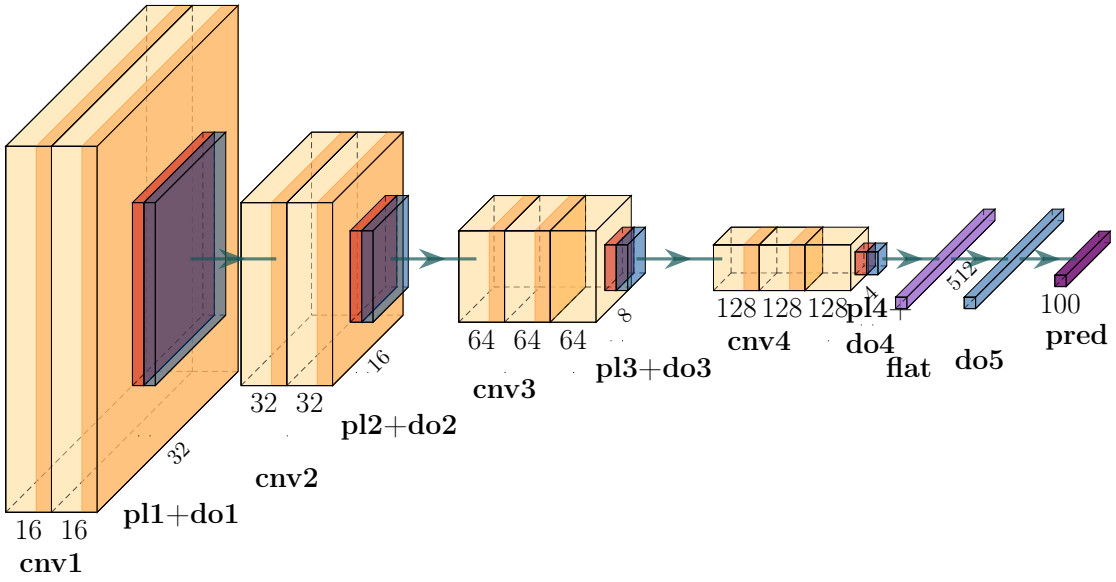

Figura 3.1: Estructura de la red neuronal VGG16 con normalización.

1. **Bloque 1:**
   - *a*) Conv2D: 16 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *b*) Conv2D: 16 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *c*) MaxPooling2D: tamaño de pool (2x2), strides (2x2).
   - *d*) Dropout: la proporción es 0.2 (rate).
2. **Bloque 2:**
   - *a*) Conv2D: 32 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *b*) Conv2D: 32 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *c*) MaxPooling2D: tamaño de pool (2x2), strides (2x2).
   - *d*) Dropout: la proporción es 0.2 (rate).
3. **Bloque 3:**
   - *a*) Conv2D: 64 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *b*) Conv2D: 64 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *c*) Conv2D: 64 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *d*) MaxPooling2D: tamaño de pool (2x2), strides (2x2).
   - *e*) Dropout: la proporción es 0.2 (rate).
4. **Bloque 4:**
   - *a*) Conv2D: 128 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *b*) Conv2D: 128 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *c*) Conv2D: 128 filtros, tamaño de kernel (3x3), activación ReLU, padding 'same'.
   - *d*) MaxPooling2D: tamaño de pool (2x2), strides (2x2).

e) Dropout: la proporción es 0.2 (rate).

5. **Capa de aplanado:**

   a) Flatten: convierte la matriz 2D en un vector.

   b) Dropout: la proporción es 0.2 (rate).

6. **Capa de clasificación:**

   a) Dense: número de unidades igual al número de clases (en este caso, 100), activación softmax.

Después, se han probado diferentes capas de normalización, siguiendo la estructura 3.2: Batch Normalization, Instance Normalization y Generalized Divisive Normalization.

VGG16 con Batch Normalization

Una vez obtenidos los resultados de entrenamiento del modelo VGG16 3.1.1, se han añadido capas de Batch Normalization. Estas capas de normalización se han añadido al final de cada bloque de convolución, antes de hacer el pooling:

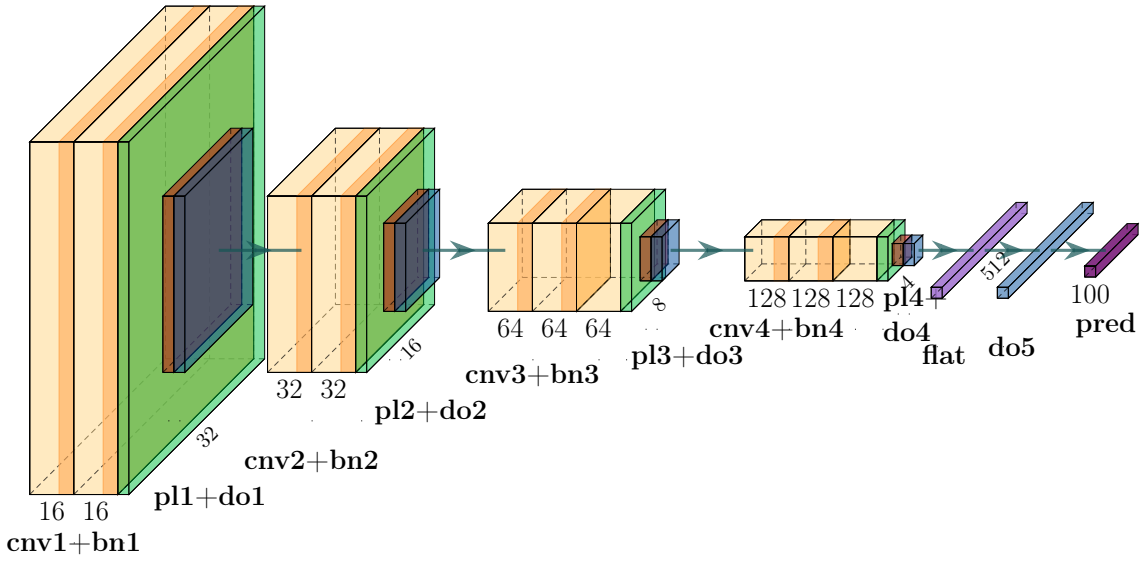

Figura 3.2: Estructura de la red neuronal VGG16 con Batch Normalization.

VGG16 con Batch Normalization y reutilización de información

Tras probar el modelo VGG16 con las capas de Batch Normalization, se ha continuado con la reutilización de la información de las capas de Batch Normalization antes de la última capa densa.

Para reutilizar la información, cuando se realiza el Batch Normalization, se calcula una media y una varianza de todo el batch. Posteriormente, se replica para cada imagen dentro del batch, la media y la varianza calculada, es decir, cada imagen dentro de un batch contará con su información (tras ser procesada al pasar por las capas anteriores) y

la media y la varianza de las diferentes capas de normalización que será igual en todas las imágenes dentro de un mismo batch:

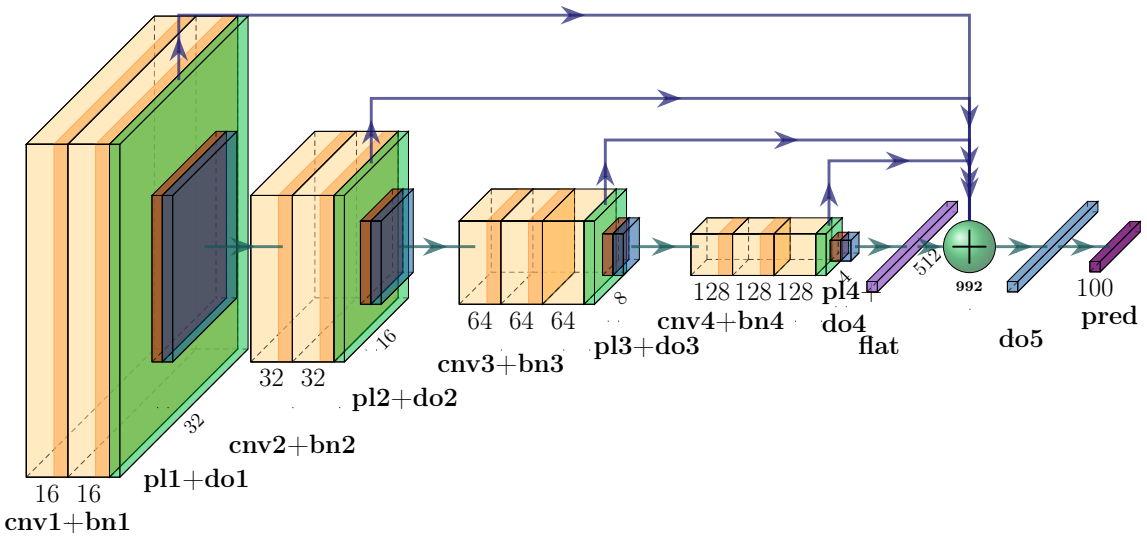

Figura 3.3: Estructura de la red neuronal VGG16 con Batch Normalization y reutilización.

Como se puede observar en la arquitectura 3.3, se realizan convoluciones en 4 bloques diferentes. Al final del cuarto bloque, se realiza un "flatten" (convierte la entrada multidimensional en un vector unidimensional) a los datos, dando como resultado un vector con 512 datos. Después, se le añade la información de las capas de normalización, es decir, la media y la varianza:

- block1_BN: 16 características.
- block2_BN: 32 características.
- block3_BN: 64 características.
- block4_BN: 128 características.

Para el bloque 1, se normaliza los datos con 16 características. Asimismo, de este bloque se utilizarán 16 datos de medias diferentes y 16 datos de varianzas diferentes en las capas futuras. Esto se realizará con los 4 bloques diferentes y varía el número de medias y varianzas que se utilizará en las capas finales, es decir, dependerá del número de características. En conclusión, por cada capa de normalización, se reutilizarán $n \cdot 2$, donde n es el número de características que se normalizan mediante BN.

Antes de la capa "Dense", hay 992 datos, los datos de la anterior capa flatten más el doble de la suma del número de características que se han normalizado en cada capa de normalización:

$$(\text{BN1} + \text{BN2} + \text{BN3} + \text{BN4}) \cdot 2 + \text{Flatten} = (16 + 32 + 64 + 128) \cdot 2 + 512 = 992$$

Para lograr el objetivo de reutilizar la información, se ha aplicado la función Lambda a la salida de la capa Flatten. En esta función Lambda, se ha obtenido la información de

todas las capas anteriores de normalización y se han concatenado con la salida de la capa Flatten, dando como resultado un vector unidimensional que tiene la información: [Datos Flatten, Medias Normalización, Varianzas Normalización].

Posteriormente, se une con la capa densa, tanto los datos que se han ido obteniendo a medida que las imágenes han pasado por los diferentes bloques hasta la capa Flatten, como los datos que ha recibido de las capas de Batch Normalization para que en la última capa, clasifique la imagen entre las 100 posibles clases que hay.

VGG16 con Instance Normalization

A diferencia del BN, mediante Instance Normalization, no se normalizan las diferentes características por cada Batch, si no individualmente. Es decir, mediante IN, para cada muestra, se normaliza cada vector de características por separado, mientras que utilizando BN, se normalizan las características por cada Batch de manera conjunta.

La diferencia con el modelo VGG16 con BN es que se sustituyen las capas de Batch Normalization por capas de Instance Normalization, lo demás se mantiene igual.

VGG16 con Instance Normalization y reutilización de información

Como con el Batch Normalization, esta vez también se reutiliza la información de las capas de normalización, la media y la varianza.

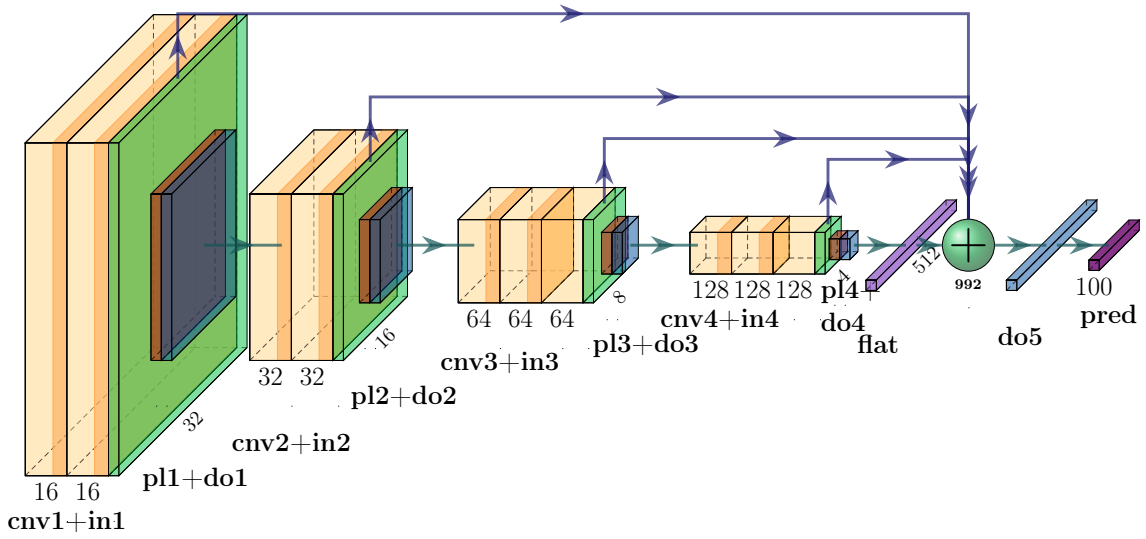

Figura 3.4: Estructura de la red neuronal VGG16 con Instance Normalization y reutilización.

En este caso, tras añadir la información de normalización de las capas de Instance Normalization a la salida de la capa flatten, también da como resultado 992 datos $((16 + 32 + 64 + 128) \cdot 2 + 512)$.

En resumen, la diferencia entre utilizar Batch Normalization e Instance Normalization radica en cómo se calculan y aplican las medias y varianzas durante el proceso de normalización.

En Batch Normalization, las medias y varianzas se calculan a partir de todas las imágenes/datos en un lote y se aplican a todas las imágenes/datos en ese lote. Esto significa que cada imagen en un lote se normalizará con las mismas medias y varianzas.

En Instance Normalization, las medias y varianzas se calculan individualmente para cada imagen/dato sin tener en cuenta las demás imágenes/datos del lote. Por ello, se normalizan según sus propias características estadísticas.

Por ello, utilizando BN, en cada batch/lote, las imágenes recibirán las mismas medias y varianzas obtenidas de las capas de normalización anteriores, unidas a la capa flatten. Mientras que en IN, aplicará medias y varianzas que varían entre imágenes, adaptándose a las características específicas de cada una y variando las que se juntarán a la capa flatten.

VGG16 con Generalized Divisive Normalization

Se analiza otro tipo de normalización llamada Generalized Divisive Normalization [3]. Se sigue manteniendo la misma arquitectura que con BN o IN, solo cambia la capa de normalización a la capa GDN.

VGG16 con Generalized Divisive Normalization y reutilización de información

En este caso, la normalización se realiza dividiendo la información por un valor que se calcula (explicación en 2.1.3). Por ello, ahora la información a reutilizar no es la media y la varianza si no el denominador de la división, la información que se reutilizará será $(\beta_i + \sum_j \gamma_{ij}|z_j|^{\alpha_{ij}})^{\epsilon_i}$.

Para ello, se almacena el denominador a medida que la información pasa las capas de normalización GDN y se une a la información que se tiene en la capa flatten. En este caso, el número de parámetros que tendrá el modelo antes de llegar a la capa de clasificación es diferente al número de parámetros de utilizar Batch Normalization e Instance Normalization.

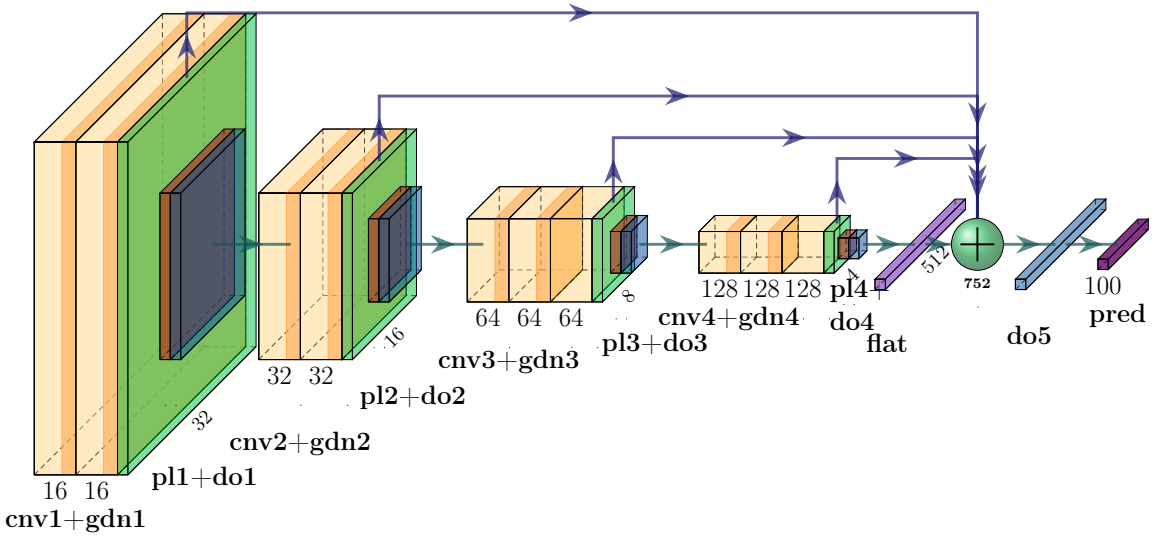

Figura 3.5: Estructura de la red neuronal VGG16 con Generalized Divisive Normalization y reutilización.

En este caso, el número de datos que se reutiliza en las capas posteriores es inferior porque, a diferencia de BN e IN, no se extraen 2 datos de las capas de normalización (media y varianza) por imagen, si no un único dato por característica. Por ello, antes de la capa Dense, se tienen 752 datos, 512 datos de la capa flatten, 16 datos de la primera normalización GDN, 32 de la segunda GDN, 64 de la tercera y 128 de la última normalización:

$$\text{GDN1} + \text{GDN2} + \text{GDN3} + \text{GDN4} + \text{Flatten} = 16 + 32 + 64 + 128 + 512 = 752$$

3.1.2. Nuevas arquitecturas a partir de PerceptNet

Para continuar con la experimentación, se ha partido de la estructura principal de la red neuronal PerceptNet 2.2.2 y se han probado diferentes tipos de normalizaciones, como en el caso anterior, Batch Normalization, Instance Normalization y Generative Divisive Normalization.

A la arquitectura base, se le ha añadido unas capas al final: una capa "Flatten" y una capa "Dense" para así poder clasificar las imágenes. Además, se le han añadido varias capas Dropout.

PerceptNet sin normalización

Primero, se ha empleado la arquitectura PerceptNet sin ninguna clase de normalización. Así, al final del estudio, se puede comprobar los beneficios que pueden aportar las diferentes normalizaciones e incluso la reutilización de la información de las diferentes normalizaciones.

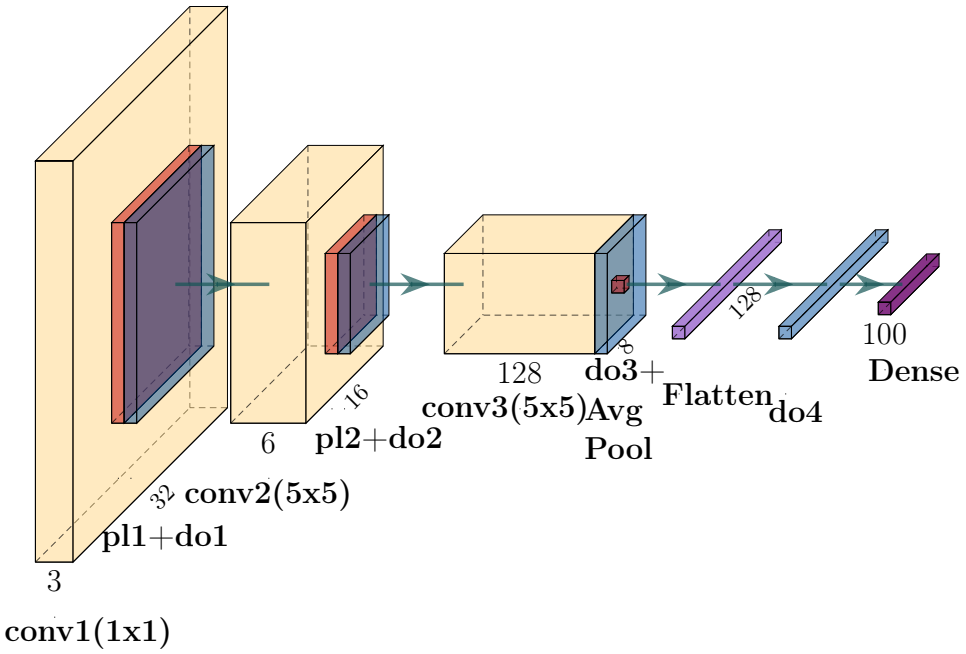

Figura 3.6: Estructura de la red neuronal PerceptNet sin normalizaciones.

1. **Entrada:**
   - Input: tamaño de entrada (32, 32, 3).
2. **Bloque 1:**
   - Conv2D: 3 filtros, tamaño de kernel (1x1), padding 'same'.
   - MaxPooling2D: tamaño de pool (2x2).
   - Dropout: la proporción es 0.2 (rate).
3. **Bloque 2:**
   - Conv2D: 6 filtros, tamaño de kernel (5x5), padding 'same'.

- MaxPooling2D: tamaño de pool (2x2).
- Dropout: la proporción es 0.2 (rate).

4. **Bloque 3:**
   - Conv2D: 128 filtros, tamaño de kernel (5x5), padding 'same'.
   - Dropout: la proporción es 0.2 (rate).
5. **Capa de reducción y clasificación:**
   - GlobalAveragePooling2D: reduce la dimensionalidad de las características extraídas.
   - Flatten: convierte la matriz en un vector.
   - Dropout: la proporción es 0.2 (rate).
   - Dense: número de unidades igual al número de clases (en este caso, 100), activación softmax.

PerceptNet con Generalized Divisive Normalization

La red neuronal PerceptNet [7] se ha diseñado utilizando la capa Generalized Divisive Normalization. Como antes se ha comentado, a la arquitectura base, se le ha añadido unas capas al final: una capa "Flatten" y una capa "Dense" para así poder clasificar las imágenes en las 100 clases diferentes que tenemos en el dataset CIFAR-100 [5].

PerceptNet con Generalized Divisive Normalization y reutilización de la información

Para poder reutilizar la información de estas capas Generalized Divisive Normalization, se ha extraido el denominador de la fórmula que se les aplica a las imágenes 2.1.3. Este valor del denominador, se extraerá de cada capa GDN y se añadirá al final para que la última capa la procese.

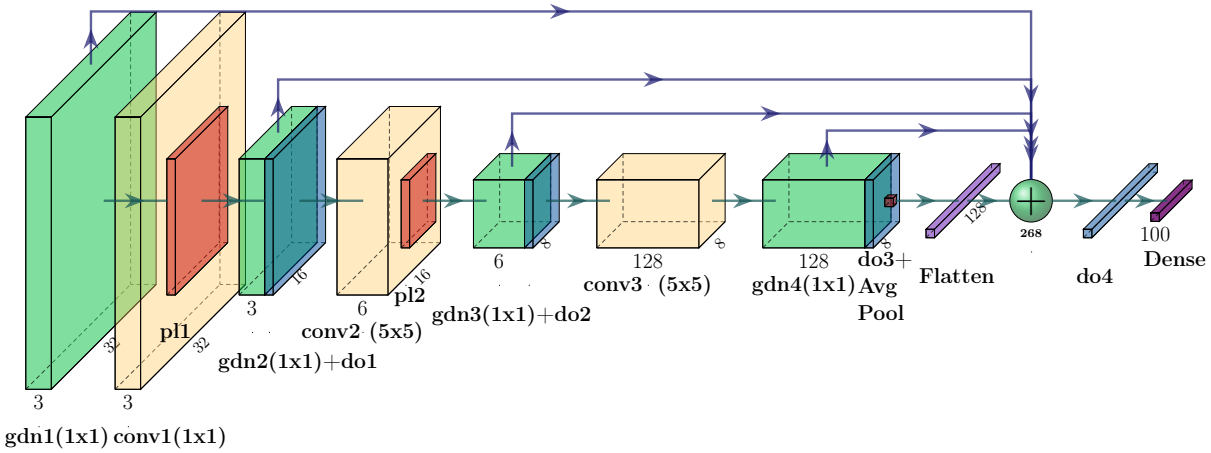

Figura 3.7: Estructura de la red neuronal PerceptNet con Generalized Divisive Normalization y reutilización.

En la figura se observa que antes de la última capa densa, hay un total de 256 datos. Estos datos serán los 128 datos de la capa "Flatten" anterior más los datos que se reutilizarán de las capas GDN:

$$\text{GDN1} + \text{GDN2} + \text{GDN3} + \text{GDN4} + \text{Flatten} = 3 + 3 + 6 + 128 + 128 = 752$$

PerceptNet con Batch Normalization

Se han sustituido las capas de normalización GDN de la red neuronal PerceptNet por capas Batch Normalization para así poder compararlas posteriormente con su versión con reutilización.

PerceptNet con Batch Normalization y reutilización de la información

Para reutilizar la información de la información de las capas de normalización Batch Normalization, se han replicado la media y la varianza para cada imagen dentro del batch. La media y la varianza es la misma para todo el batch, es decir, se calcula conjuntamente con todas las imágenes de un batch. Asimismo, ésta sería la estructura que tendría la red neuronal con reutilización de la media y la varianza:

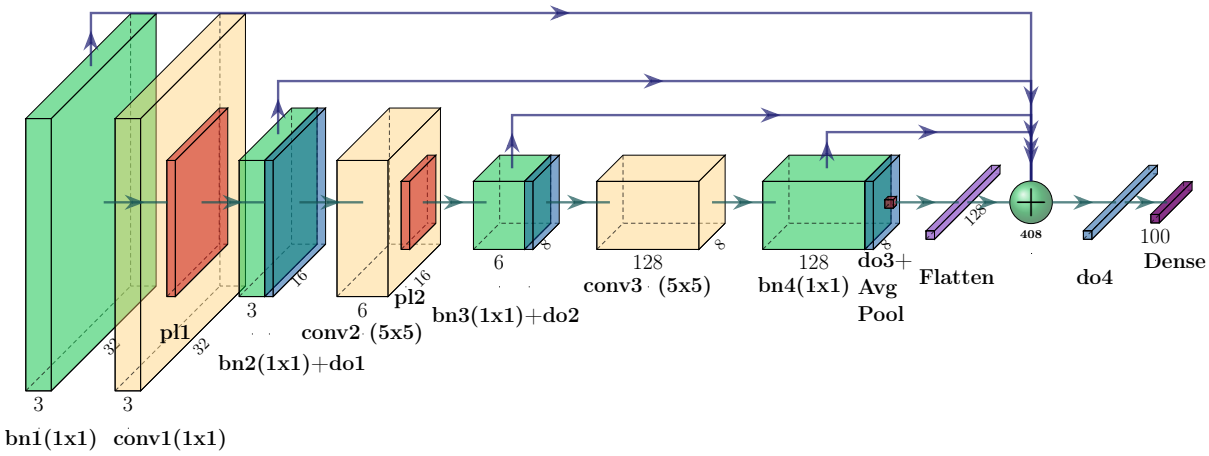

Figura 3.8: Estructura de la red neuronal PerceptNet con Batch Normalization y reutilización.

Después de la capa flatten, se le añaden las medias y varianzas de las anteriores capas, dando como resultado 408 datos que se procesarán en la última capa densa.

El resultado de 408 viene de la suma de los valores de la capa densa con las medias y varianzas de cada capa de normalización:

$$(\text{BN1} + \text{BN2} + \text{BN3} + \text{BN4}) \cdot 2 + \text{Flatten} = (3 + 3 + 6 + 128) \cdot 2 + 128 = 408$$

Se multiplica por 2 el número de características que se han normalizado en las capas de Batch Normalization porque se extraen 2 elementos: la media y la varianza.

Para la reutilización de la información, después de la capa “Flatten”, se ha aplicado una Lambda a cada imagen. En esta Lambda, se ha empleado una función para obtener la información de todas las capas anteriores de normalización y se han repetido las medias y las varianzas de las diferentes capas tantas veces como el tamaño del batch sea. Posteriormente, se han concatenado las varianzas y las medias con la salida de la capa “Flatten”, dando como resultado un vector unidimensional que contiene los datos procesados hasta la capa flatten, las medias y las varianzas. Gracias a ello, se han podido procesar correctamente en la última capa densa de clasificación con 100 clases diferentes.

PerceptNet con Instance Normalization

A la arquitectura PerceptNet, se le ha cambiado la capa de normalización por la capa Instance Normalization. En esta normalización, se normalizan las imágenes de una en una y de característica en característica (o canal).

En esta red neuronal, se tendrán 4 capas de Instance Normalization.

PerceptNet con Instance Normalization y reutilización de la información

La estructura con la reutilización será idéntica a la del caso de utilizar Batch Normalization. La diferencia reside en el tipo de normalización. En este caso, en vez de repetir la misma media entre todas las imágenes del batch, cada imagen tendrá sus propias medias y varianzas calculadas respecto a sus valores en las diferentes capas Instance Normalization para cada canal.

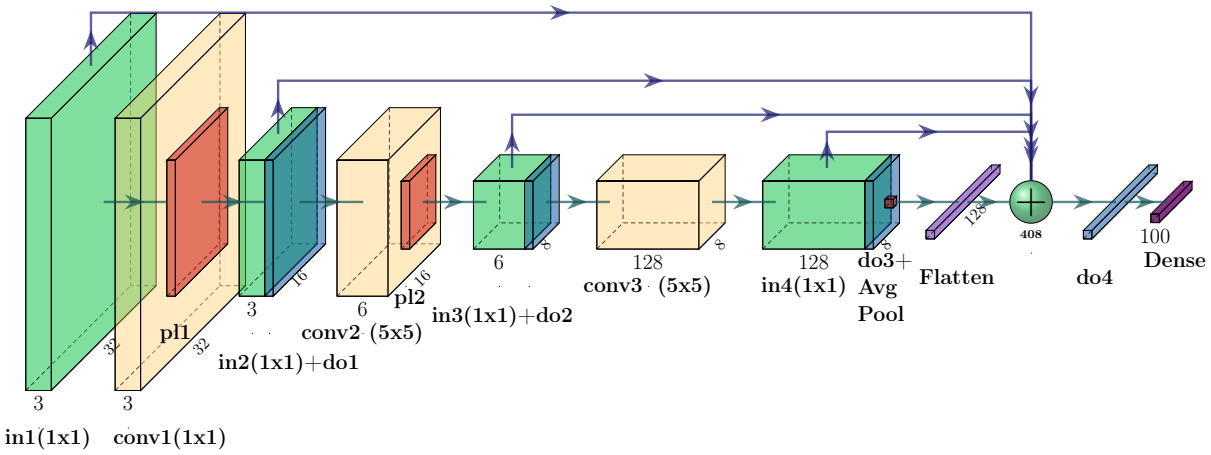

Figura 3.9: Estructura de la red neuronal PerceptNet con Instance Normalization y reutilización.

Como en el caso de Batch Normalization, al realizar la concatenación se obtienen 408 datos, la suma de los valores de la capa densa con las medias y varianzas de las capas Instance Normalization:

$$(\text{IN1} + \text{IN2} + \text{IN3} + \text{IN4}) \cdot 2 + \text{Flatten} = (3 + 3 + 6 + 128) \cdot 2 + 128 = 408$$

Se guardan tanto la media como la varianza en cada capa para cada imagen que pasa, por ello, se multiplica por 2.

Capítulo 4

Resultados

4.1. Configuración de los experimentos

4.1.1. Explicación previa

Para la experimentación, se han probado para los modelos VGG16 y PerceptNet las diferentes técnicas de normalización y la reutilización comentadas anteriormente.

Para todos los modelos, se ha utilizado el mismo dropout y learning rate (tanto para VGG16 como para PerceptNet). Así, se realiza una comparación en condiciones similares para ambos casos y para todos los diferentes modelos.

Se ha trabajado con el conjunto de datos CIFAR 100 y para analizar los resultados, para todos los modelos y combinaciones. Cada modelo se ha dejado entrenar 500 épocas, lo suficiente para que converjan. Asimismo, se ha escogido para cada red neuronal, la época que mejor resultado ha obtenido. La métrica que se ha comparado para decidir la época ha sido el accuracy para el conjunto de validación. Para ello, se ha utilizado un callback. Posteriormente, se ha utilizado un conjunto de test, separado antes de cualquier entrenamiento, para calcular las diferentes métricas.

4.1.2. Métricas

Para cada modelo, se han evaluado diferentes métricas para ver su rendimiento: accuracy, loss, precision, recall y AUC.

Accuracy (Precisión)

Proporción de predicciones correctas sobre el total de predicciones realizadas.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

- TP: Verdaderos positivos
- TN: Verdaderos negativos
- FP: Falsos positivos

- FN: Falsos negativos

Loss (Pérdida)

Mide la discrepancia entre las predicciones del modelo y los valores reales. Cuanto menor es el valor, mejor es el rendimiento del modelo y también sirve para que el modelo aprenda. Se utiliza la fórmula de la entropía cruzada categórica:

$$\text{Categorical crossentropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

- N: Número de muestras
- C: Número de clases
- y_{ij}: Valor de la etiqueta real de la i-ésima muestra para la j-ésima clase (1 si pertenece, 0 si no)
- $\hat{y}_{ij}$: Probabilidad predicha por el modelo para la i-ésima muestra para la j-ésima clase

Precision (Precisión)

La proporción de verdaderos positivos sobre el total de positivos predichos. No se debe confundir con Accuracy, que la traducción al Español es la misma.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall (Sensibilidad)

La proporción de verdaderos positivos sobre el total de positivos reales.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

AUC (Área bajo la curva ROC)

Es el área bajo la curva ROC. Mide la capacidad del modelo para distinguir entre clases. Un valor cercano a 1 indica un buen modelo, mientras que un valor de 0.5 sugiere un modelo sin capacidad discriminatoria.

Es una métrica muy robusta ante el desbalanceo de clases.

4.2. Resultados obtenidos

4.2.1. VGG16

Métricas

Los resultados obtenidos para las diferentes configuraciones del modelo VGG16 son los siguientes ("v2" significa que se reutiliza la información en el modelo):

Modelo	Loss	Precission	Recall	AUC	Accuracy	Params
VGG16	2.974	0.544	0.331	0.878	0.394	529,332
VGG16_BN	3.275	0.532	0.464	0.859	0.483	530,292
VGG16_BN_v2	53879.512	0.475	0.475	0.735	0.475	578,292
VGG16_GDN	2.923	0.613	0.204	0.886	0.335	725,892
VGG16_GDN_v2	2.324	0.737	0.218	0.935	0.398	749,892
VGG16_IN	3.948	0.483	0.410	0.825	0.429	529,812
VGG16_IN_v2	2.383	0.633	0.306	0.916	0.413	577,812

Cuadro 4.1: Resultados obtenidos para los diferentes modelos VGG16

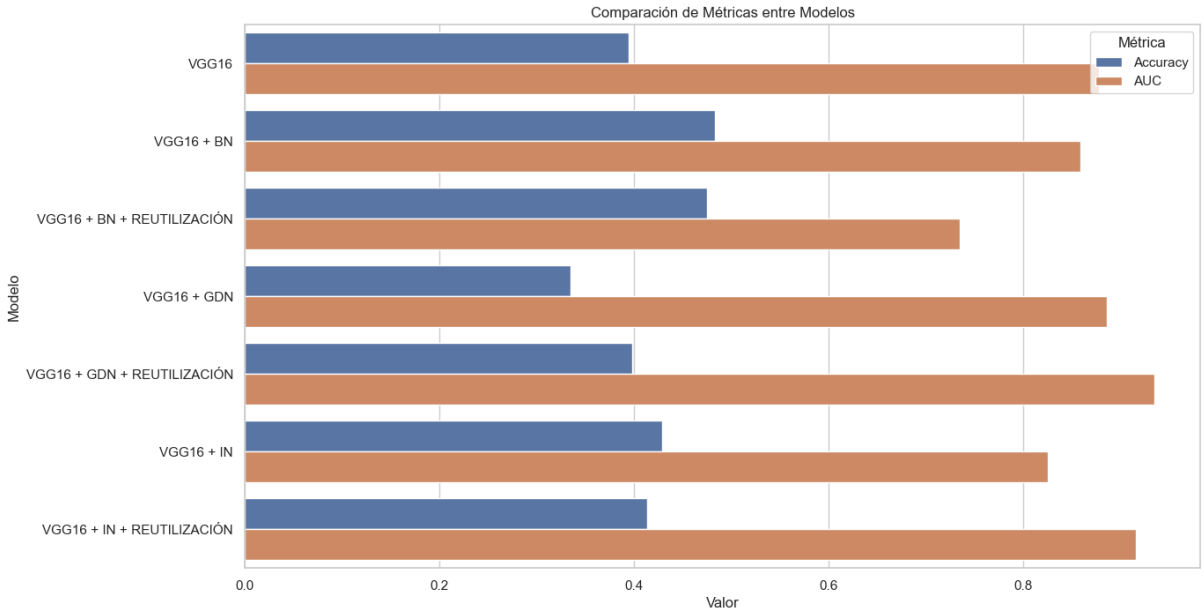

Figura 4.1: Gráfico de barras de las métricas de las arquitecturas VGG16 utilizadas.

Comparación de los resultados

La comparación de los resultados se realizará con la métrica AUC, ya que es más robusta y menos influenciada por el desbalanceo de clases que el Accuracy.

Tras comparar los resultados, se puede observar que para las arquitecturas VGG16 que utilizan las capas Generalized Divisive Normalization e Instance Normalization se benefician de la reutilización de la información de las capas de normalización.

Mirando en el AUC, para la arquitectura VGG16 con Generalized Divisive Normalization, tiene 0.886, mientras que si se introduce la reutilización de el denominador (explicación en 3.1.1) se logra aumentar hasta 0.935.

Observando el modelo VGG16 con Instance Normalization, se obtiene un AUC de 0.825, mientras que reutilizando las medias y varianzas de las imágenes (explicación en 3.1.1) se incrementa su valor hasta 0.916.

Por otro lado, en el caso de utilizar Batch Normalization, no se ha observado mejoría, incluso ha empeorado el modelo de 0.859 a 0.735. Esto se debe a que en el caso de la normalización Batch Normalization, se repite la misma media y varianza para cada imagen dentro de un batch (explicación en 3.1.1).

4.2.2. PerceptNet

Métricas

Los resultados obtenidos para las diferentes configuraciones del modelo PerceptNet son los siguientes (como antes, "v2" significa que se reutiliza la información en el modelo):

Modelo	Loss	Precission	Recall	AUC	Accuracy	Params
PerceptNet	3.430	0.598	0.032	0.855	0.193	32,696
PerceptNet_BN	3.519	0.566	0.026	0.842	0.181	33,256
PerceptNet_BN_v2	3.641	0.561	0.028	0.825	0.162	61,256
PerceptNet_GDN	3.133	0.644	0.073	0.878	0.242	49,554
PerceptNet_GDN_v2	2.904	0.679	0.090	0.900	0.285	195,058
PerceptNet_IN	4.606	0.0	0.0	0.500	0.010	32,976
PerceptNet_IN_v2	3.102	0.621	0.089	0.881	0.265	60,976

Cuadro 4.2: Resultados obtenidos para los diferentes modelos PerceptNet

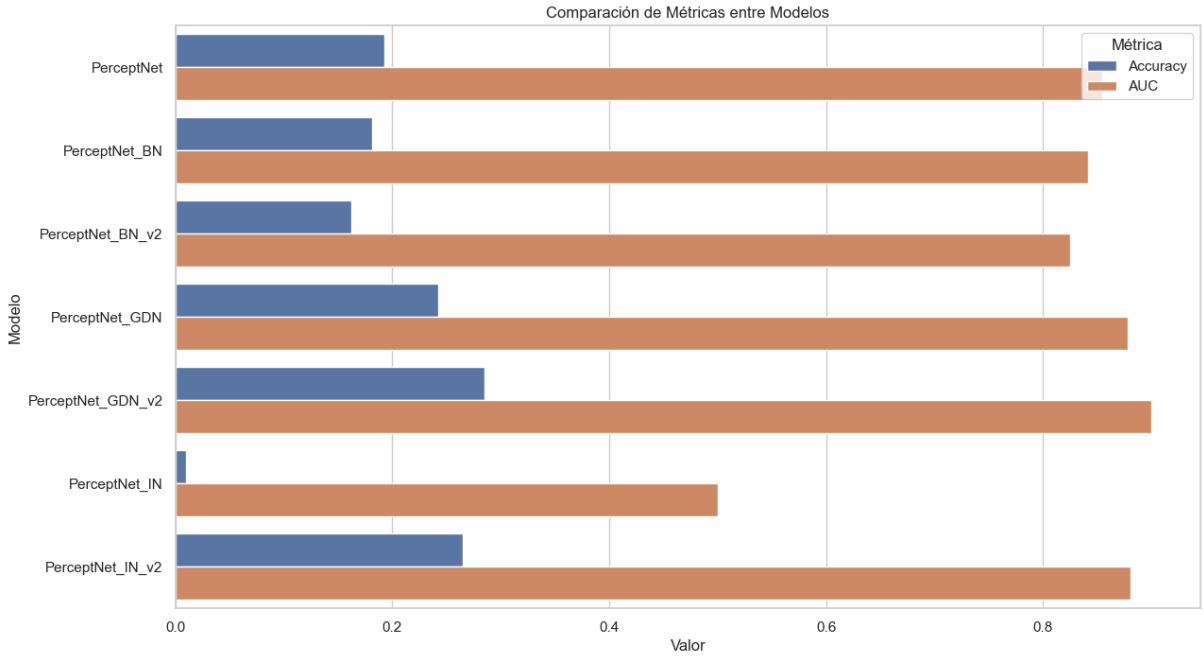

Figura 4.2: Gráfico de barras de las métricas de las arquitecturas VGG16 utilizadas.

Comparación de los resultados

En el caso de la arquitectura PerceptNet, también se han observado mejoras tras reutilizar la información en la capa Generalized Divisive Normlaization. El modelo con la capa GDN obtiene un AUC de 0.878 y reutilizando la información se consigue llegar a 0.900.

Por otro lado, si se emplea la capa Instance Normalization, se obtiene un AUC de 0.5, mientras que si reutilizamos la información se consigue llegar hasta 0.881, logrando que el modelo aprenda.

Como sucedía con la anterior arquitectura analizada (VGG16), la reutilización de información de las capas Batch Normalization empeora los resultados obtenidos de un AUC de 0.842 a 0.825.

En comparación con la arquitectura VGG16, se obtienen resultados peores con PerceptNet, pero esto se debe a que el modelo es mucho más pequeño y con menos parámetros. Por ello, al contrastar los resultados de un modelo con el otro se observa tanta diferencia, ya que se debe a la complejidad y extensión de los diferentes modelos.

Capítulo 5

Conclusiones

5.1. Efectividad de la reutilización de información en capas de normalización

Tras realizar la experimentación, se ha comprobado que la reutilización de información en las capas Generalized Divisive Normalization (GDN) e Instance Nomralization (IN) ha mostrado mejoras significativas en el rendimiento de ambas arquitecturas. Estas mejoras se pueden observar en los resultados obtenidos en el AUC.

Contrariamente, la reutilización de información en las capas Batch Normalization no resultó en mejoras, sino que empeoró el rendimiento en las 2 arquitecturas. Este deterioro puede deberse a que Batch Normalization utiliza la misma media y varianza para cada imagen dentro de un batch, lo que podría introducir redundancias y limitaciones para que generalice el modelo.

Estos hallazgos abren nuevas vías de investigación y desarrollo para optimizar las arquitecturas de redes neuronales con el fin de mejorar su rendimiento.

5.2. Trabajo futuro

Los resultados sugieren que la reutilización de información en capas de normalización específicas puede ser una técnica efectiva para mejorar el rendimiento de las redes neuronales. Sin embargo, depende de la naturaleza de la capa de normalización utilizada.

Por ello, se debería de realizar estudios adicionales con otras capas de normalización para comprobar su efectividad y para entender mejor en qué condiciones la reutilización de información es beneficiosa.

Además, probar con otras arquitecturas u otros conjuntos de datos podría proporcionar una visión más amplia.

Apéndice A

Apéndice

A.1. Generación de Arquitecturas de Redes Neuronales

A.1.1. Introducción

En el desarrollo del trabajo de fin de grado, hice varias representaciones gráficas de las arquitecturas de las redes utilizadas. Para ello, se empleó la herramienta PlotNeuralNet, la cuál facilita la creación de diagramas de diferentes redes neuronales.

A.1.2. Herramienta utilizada: PlotNeuralNet

PlotNeuralNet [8] es una herramienta de código abierto disponible en GitHub. Permite generar visualizaciones de arquitecturas de las redes neuronales utilizando LaTeX como entorno para la creación.

A.1.3. Generación de diagramas

Para generar diagramas con PlotNeuralNet, se realizan varios pasos:

1. **Definición de arquitectura**: Se define la arquitectura de la red que se desea visualizar en un formato compatible con PlotNeuralNet. Esto implica especificar cada capa de la red, sus filtros, el tamaño de los filtros, tamaño de las representaciones, etc.

2. **Creación del diagrama**: Tras definir la arquitectura, se genera un diagrama en formato LaTex, capaz de representar visualmente cada capa de la red y sus conexiones a través de la red.

3. **Renderización del diagrama**: Finalmente, el diagrama en formato LaTeX se renderiza para obtener la imagen final. Para ello, se ha logrado implementarlo en el mismo documento que se ha desarrollado el TFG.

A.1.4. Diagramas utilizado en el trabajo

En este TFG se generaron varios diagramas para ilustrar y explicar de una manera más detallada alguna estructura de las redes neuronales utilizadas.

Se ha representado principalmente las 2 arquitecturas utilizadas en el proyecto: VGG16 y PerceptNet.

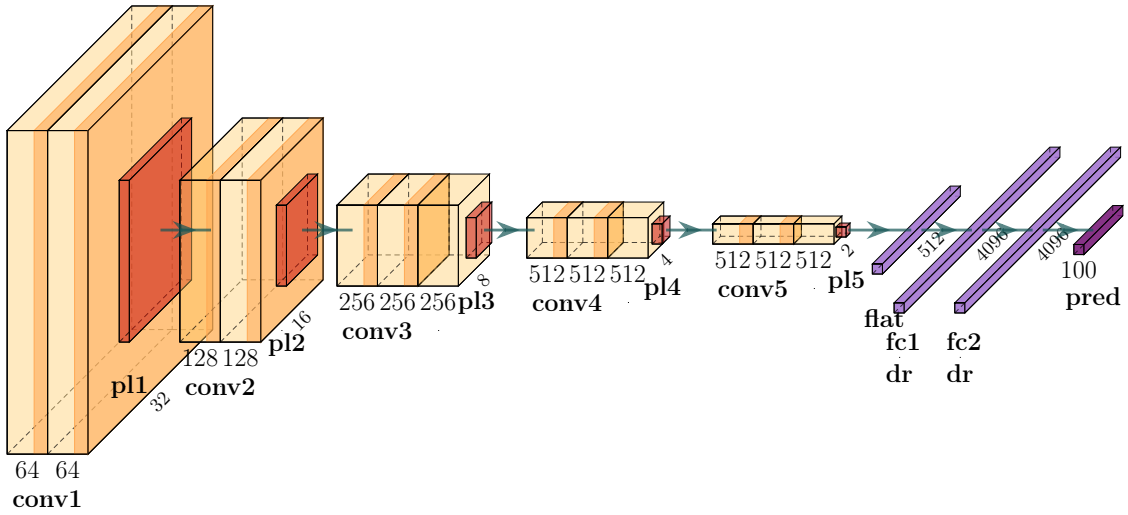

Figura A.1: Estructura de la red neuronal VGG16 [6].

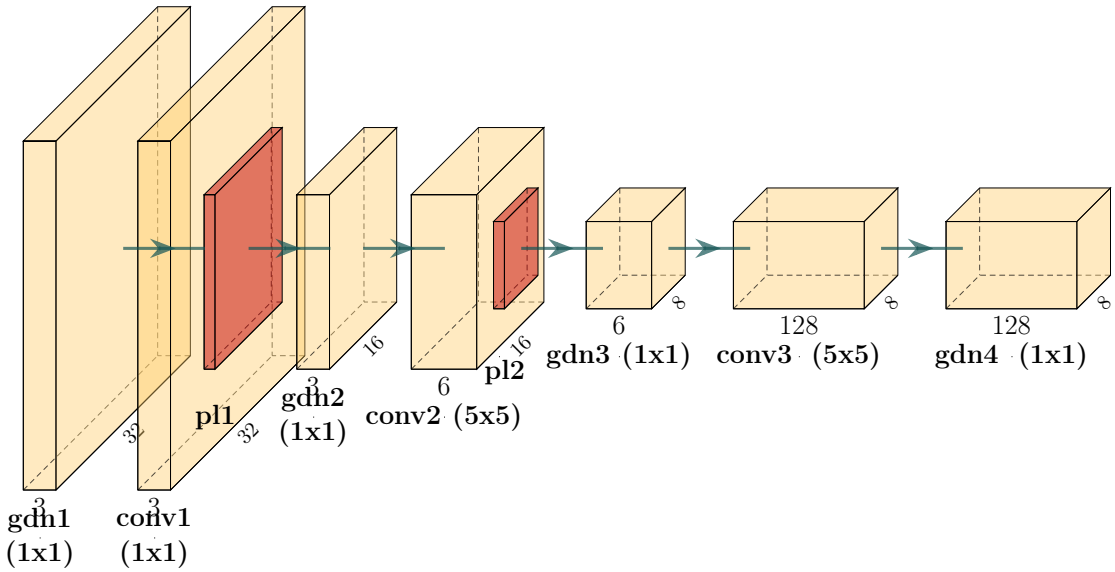

Figura A.2: Estructura de la red neuronal PerceptNet [7].

Se han generado diagramas de las diferentes normalizaciones utilizadas y, también, con la reutilización de la información para que se puedan entender mejor las estructuras de las redes neuronales empleadas.

Además. se ha logrado, de una manera muy visual, explicar en qué consiste la reutilización de la información mediante flechas salientes de las capas de normalización hasta el punto donde se concatena toda la información.

Bibliografía

[1] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015.

[2] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization, 2017.

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. Density Modeling of Images using a Generalized Normalization Transformation, 2016.

[4] Xu Pan, Luis Gonzalo Sánchez Giraldo, Elif Kartal, and Odelia Schwartz. Brain-inspired Weighted Normalization for CNN Image Classification. *bioRxiv*, 2021.

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[6] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, 2015.

[7] Alexander Hepburn, Valero Laparra, Jesus Malo, Ryan McConville, and Raul Santos-Rodriguez. Perceptnet: A Human Visual System Inspired Neural Network For Estimating Perceptual Distance. In *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, October 2020.

[8] Haris Iqbal. PlotNeuralNet. https://github.com/HarisIqbal88/PlotNeuralNet, 2018. GitHub repository.