TRANSFERENCIA DE APRENDIZAJE EN REDES NEURONALES CONVOLUCIONALES PARA ESPACIOS EMBEBIDOS DE IMÁGENES

ROMEL CASADIEGOS BARRIOS

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
BUCARAMANGA
2016

TRANSFERENCIA DE APRENDIZAJE EN REDES NEURONALES CONVOLUCIONALES PARA ESPACIOS EMBEBIDOS DE IMÁGENES

ROMEL CASADIEGOS BARRIOS

Trabajo de Grado para optar al título de
Ingeniero de Sistemas

Director
RAÚL RAMOS-POLLÁN, PhD

Codirector
DARIO GARCIA-GASULLA, PhD

UNIVERSIDAD INDUSTRIAL DE SANTANDER
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
BUCARAMANGA
2016

DEDICATORIA

Dedico esta trabajo a mi familia, sin su apoyo esto no hubiera sido posible

A mis tutores y a los maestros que han hecho parte de mi formación,

A mis compañeros quienes han estado ahí para aprender unos de otros

A Daniela por todo.

Romel Casadiegos Barrios

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

Clustering (Agrupamiento): tarea de agrupar un conjunto de objetos de tal manera que los objetos en el mismo grupo (llamado un clúster) son más similares (en un sentido u otro) entre sí que con los de otros grupos (clusters) . Es una tarea principal de exploración de minería de datos, y una técnica común para los estadísticos de análisis de datos, que se utiliza en muchos campos, incluyendo el aprendizaje automático, reconocimiento de patrones, análisis de imágenes, recuperación de información, la bioinformática, la compresión de datos y gráficos por ordenador**.**

CNN (Red Neuronal Convolucional): Una red neuronal convolucional es un tipo de red neuronal artificial donde las neuronas corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria (V1) de un cerebro biológico. Este tipo de red es una variación de un perceptrón multicapa, pero su funcionamiento las hace mucho más efectivas para tareas de visión artificial, especialmente en la clasificación de imágenes.

Digits: El sistema de entrenamiento NVIDIA GPU para el aprendizaje profundo (dígits). DÍGITS le permite diseñar rápidamente la mejor red neuronal profunda (DNN) para tareas de clasificación de imágenes y reconocimiento de objetos utilizando la visualización comportamiento de la red en tiempo real.El uso de dígitos que puede realizar tareas de aprendizaje profundas comunes, tales como la gestión de datos, la definición de las redes, la formación de varios modelos en paralelo, la supervisión del rendimiento de entrenamiento en tiempo real, y elegir el mejor modelo desde el navegador de resultados. Por debajo está implementado usando Caffe y permite el entrenamiento muti-gpu.

Fine tuning (Ajuste fino): Proceso por el cual se toman redes con pesos ya entrenados, aprovechando las características ya aprendidas por esta red para realizar un entrenamiento con otro conjunto de datos y obtener un mejor resultado

Ivf: Por sus siglas en inglés image vector file, es un archivo donde se guarda un vector disperso que contiene las activaciones correspondiente a las capas seleccionadas de la red con anterioridad.

Machine Learning (Aprendizaje de Maquina): Rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que le permiten a las

máquinas aprender sin ser específicamente programadas, mediante un proceso de inducción. En un programa de aprendizaje automático se provee de un conjunto de datos de ejemplo (experiencia) a partir de los cuales se construye un modelo que calcula un resultado aproximado de los datos reales.

Python: Lenguaje de programación de alto nivel, interpretado y de código abierto ampliamente usado. Cuenta con una gran cantidad de librería que hacen de Python una herramienta adaptable a las necesidades de los desarrolladores . puede ser empleado para, desarrollo de aplicaciones web y de escritorio, computación científica, data science, machine learning y scripting.

RESUMEN

TÍTULO: TRANSFERENCIA DE APRENDIZAJE EN REDES NEURONALES CONVOLUCIONALES PARA ESPACIOS EMBEBIDOS DE IMÁGENES*

AUTOR: ROMEL CASADIEGOS BARRIOS**

PALABRAS CLAVE: Machine Learning, Deep Learning, Procesamiento de imágenes, CNNs, Clasificación, Clustering.

DESCRIPCIÓN: El proceso de entrenamiento de las redes neuronales convolucionales (CNNs) genera un gran conjunto de rasgos que caracterizan los datos de entrada. Estas características se utilizan normalmente para fines de discriminación, sin embargo, también se pueden entender como descriptores de datos. En este trabajo se estudia la repercusión de diversos parámetros meta-modelo (arquitectura, del conjunto de entrenamiento, de ajuste) tienen sobre los descriptores para los fines de representatividad. Utilizando el conjunto de datos CIFAR-100 construimos un vector embebido en el espacio alrededor de sus 100 clases de imágenes. A continuación, se evalúa la capacidad descriptiva de cada espacio de la inserción mediante la medición de su desempeño en la búsqueda no supervisada de las 20 súper clases definidas por CIFAR-100. Los resultados muestran que el aprendizaje de transferencia es generalmente más ventajoso para generar imágenes embebidas semánticamente útiles, aunque la formación de una CNN desde cero podría producir resultados similares. Por otra parte, nuestra incorporación semántica revela discrepancias entre CIFAR composición superclase y clases relacionadas visualmente. (como los chimpancés ser visualmente más similar a las personas que a otros omnívoros), esto muestra que sin importar como se compone las superclases, estos descriptores se agrupan por sus semejanzas visuales de la misma forma en que lo hace un ser humano, queda por explorar el impacto de utilizar arquitecturas de redes neuronales convolucionales con fines descriptivos mas que discriminativos que es el fin que cumplen actualmente.

*Trabajo de grado.

** Facultad de Ingenierías Fisico-Mecanicas, Escuela de Ingeniería de Sistemas e Informática. Director Raúl Ramos Pollan Ph.D. Codirector Dario Garcia Gasulla Ph. D.

ABSTRACT

TITLE: TRANSFER LEARNING IN CONVOLUTIONAL NEURAL NETWORKS FOR IMAGE EMBEDDING SPACES*

AUTHOR: ROMEL CASADIEGOS BARRIOS**

KEYWORDS: Machine Learning, Deep Learning, Image Processing, CNNs, Classification, Clustering

DESCRIPTION: The training process of convolutional neural networks (CNNs) generates a large set of features characterizing the input data. These features are typically used for discrimination purposes, however they can also be understood as data descriptors. In this paper we study the impact that various meta-model parameters (architecture, training set, fine-tuning) have on those descriptors for the purpose of representa-tiveness. Using the CIFAR-100 data set we build a vector embedding space around its 100 image classes. Then, we evaluate the descriptive power of each embedding space by measuring its performance at unsupervisedly finding the 20 superclasses dfined within CIFAR-100. Results show that transfer learning is generally ad-vantageous to generate semantically useful image embeddings, although training a CNN from scratch might produce similar results. Furthermore, our semantic embedding unveiled discrepancies between CIFAR superclass composition and visually related classes (such as chimpanzees beingvisually more similar to people than to other omnivores), this shows that no matter how the superclasses are composed these descriptors are grouped by their visual similarities in the same way a human being does, there remains to be explored the impact of using convolutional neural network architectures for descriptive rather than discriminative purposes which is the purpose they currently fulfill.

*Bachelor Thesis.

** Faculty of Physical-Mechanical Engineering. Shool of Engineering and Computer Science.. Director Raúl Ramos Pollan Ph.D. Codirector Dario Garcia Gasulla Ph. D.

INTRODUCCIÓN

Las redes neuronales convolucionales (CNNs) son entrenados para la clasificación de objetos, aprenden representaciones con el objetivo de encontrar un subconjunto óptimo para la discriminación. Este subconjunto corresponde con las características de la capa superior, que son alimentados en un conjunto totalmente conectada de capas para la estimación de la probabilidad de pertenecer a una clase. El resto de características aprendidas tienen un poder menos discriminativo pero sin embargo componen un lenguaje descriptivo rico y completo de los datos de entrada que se puede utilizar para diferentes propósitos. Estudiar el lenguaje de descripción definido por los modelos entrenados con CNN es un campo de interés ya que permitiría múltiples aplicaciones que exploten tales fuentes ricas de información (e.j., zero-shot learning). El desafío reside en el carácter puramente sub-simbólico de CNNs aprendió filtros que complica cualquier intento de interpretación sencillo. Para comprender el lenguaje descriptivo de los modelos de la CNN, uno puede mirar el lugar en el espacio embebido donde este lenguaje está definido. Un espacio de este tipo puede ser explorado mediante la asignación de vectores embebidos a los datos de entrada (imágenes), y la realización de operaciones con esos vectores. Por ejemplo, mediante el uso de medidas de distancia para saber en qué parte del espacio se encuentra cada entrada, se puede dar sentido al espacio embebido en su conjunto. Algoritmos de agrupación proporcionan una primera exploración de un espacio a través de medidas de distancia. Encontrar grupos de entidades cercanas entre sí da una idea de la estructura global del espacio, revelando cuáles características de alto nivel están codificados dentro del espacio. Para un modelo de CNN entrenado con fines discriminatorios esto es particularmente interesante, ya que el modelo no se le enseña a identificar las abstracciones, y sin embargo, éstos pueden aparecer de forma autónoma.

Las arquitecturas de las CNNs están continuamente mejorando con el objetivo de maximizar el poder de discriminación. Cambios en la arquitectura, los conjuntos de datos de entrenamiento y técnicas como el fine tuning permiten el desarrollo de cada vez mejores modelos de la CNN para tareas como la clasificación de objetos. Sin embargo, poco se sabe sobre el impacto que estos meta-parámetros de las CNNs tienen sobre otros aspectos de los modelos construidos, tales como el lenguaje descriptivo estos modelos definen. Claramente, la ambición de permitir representaciones CNNs para otros fines requiere la comprensión del impacto que la formación meta-parámetros tiene sobre las representaciones

aprendidas. En este trabajo se contrasta alguna información sobre este tema, mediante el uso de seis modelos diferentes de la CNN (dos arquitecturas diferentes, dos conjuntos de datos diferentes, y el rendimiento de fine tuning o no), y el análisis y la comparación de sus espacios embebidos definidos. Los resultados se mostrarán variaciones en el espacio de la CNN aprendido representaciones, así como en las variaciones de formación puede tener efectos en aquellos.

1. DEFINICIÓN DEL PROBLEMA

Deep Learning es conjunto de métodos para machine learning, uno de sus métodos al trabajar con imágenes consiste en, entrenar redes neuronales convolucionales para la discriminación de objetos determinados en clases definidas. Es usada con el fin de dar soporte a otras áreas de machine learning como reconocimiento automático del habla, y reconocimiento de señales de audio, sonido e imágenes, también en otras áreas como visión por computador. Cuando se trabaja con CNNs se busca tener alta precisión mediante el aprendizaje de características tanto de bajo y alto nivel con el fin de reconocer patrones para que las neuronas puedan activarse al momento que son ingresados y reconocidos, todo este entrenamiento se realiza de manera supervisada, es decir con etiquetas del resultado deseado durante el entrenamiento. Hoy en día se cuenta con diferentes métodos para intentar maximizar la precisión, que aunque han sido útiles, no han llegado a los niveles deseados (ej. hay conjuntos de datos con demasiadas clases, que dificultan el poder discriminativo de las CNNs).

Actualmente se ha demostrado que las características de alto nivel aprendidas por las redes neuronales convolucionales son ricas en semántica y pueden ser usada en para mejorar la discriminación mediante el entrenamiento zero-shot. Este tipo de entrenamientos buscan la extracción de mayor semántica de las redes neuronales no ha sido explorado a profundidad a la fecha y por tanto no se han podido explotar todas las posibles características y beneficios que puede traer, como la transferencia de aprendizaje obtenido durante entrenamiento con un conjunto de datos a un nuevo conjunto de datos de diferente naturaleza y poder clasificarlos apropiadamente.

2. OBJETIVO GENERAL Y ESPECÍFICOS

2.1. OBJETIVO GENERAL

Diseñar e implementar un esquema de extracción de representaciones embebidas de imágenes basado en redes neuronales convolucionales y evaluar su capacidad para aprender agregaciones semánticas de manera no supervisada.

2.2. OBJETIVOS ESPECÍFICOS

- Seleccionar arquitecturas de redes convolucionales y definir estrategias de entrenamiento de las redes para la transferencia de aprendizaje.
- Seleccionar diferentes datasets de imágenes, anotados con una ontología de clasificación
- Definir estrategias de uso de clustering para la agregación semántica
- Diseñar e implementar un esquema experimental y métricas de evaluación de las distintas combinaciones de arquitecturas de CNN y estrategias de entrenamiento y clustering.
- Evaluar el desempeño de las agregaciones semánticas obtenidas.

3. MARCO TEÓRICO

3.1 APRENDIZAJE AUTOMÁTICO

El aprendizaje automático es un campo de la ciencia de computación, que evolucionó del estudio de reconocimiento de patrones y la teoría computacional de aprendizaje de la inteligencia artificial. Este campo explora el estudio y la construcción de algoritmos que puedan aprender y hacer predicciones de los datos. Dichos algoritmos operan construyendo un modelo con ejemplos de entrada para hacer predicciones o tomar decisiones basado en estos, en lugar de seguir estrictamente instrucciones estáticas y es empleado en un rango de tareas donde la programación explícita no es factible, como el filtrado de spam, reconocimiento óptico de caracteres, motores de búsqueda y visión por computadora.

El objetivo principal del aprendizaje automático es generalizar desde la experiencia, desempeñarse acertadamente en ejemplos nuevos que no habían sido estudiados anteriormente, luego de obtener experiencia aprendida de los datos de entrenamiento, que por lo general vienen de una distribución de probabilidad desconocida y debe ser modelada por el sistema para poder producir predicciones para los nuevos casos.

3.1.1 Tareas en el aprendizaje automático: El aprendizaje automático nos permite hacer frente a las tareas que son demasiado difíciles para resolver con programas fijos escrito y diseñado por los seres humanos. Desde un punto de vista un científico y filosófico, el aprendizaje automático es interesante porque el desarrollo hacia la comprensión del aprendizaje de maquina implica desarrollar la compresión de los principios que subyacen a la inteligencia.

Muchos tipos de tareas se pueden resolver con el aprendizaje automático. Algunas de las tareas del aprendizaje automático más comunes incluyen los siguientes:

3.1.1.1 Clasificación: En este tipo de tarea, se espera que el programa informático especifique cuál de las categorías k alguna entrada pertenece. Para resolver esta tarea, se le preguntó por lo general el algoritmo de aprendizaje para producir una función

$f : \mathbb{R}^n \to \{1, \ldots, k\}$ Donde $y = f(\boldsymbol{x})$

el modelo asigna una entrada descrito por el vector x para una categoría identificado por código numérico y. Hay otras variantes de la codificación como k, por ejemplo, donde f da salida a una distribución de probabilidad sobre las clases.

3.1.1.2 Clasificación con entradas faltantes: Este tipo de tarea de clasificación se hace cuando al programa no se le garantiza que siempre se proporcionará todas las medidas en su vector de entrada. Con el fin de resolver la tarea de clasificación, el algoritmo de aprendizaje sólo se tiene para definir una única función de mapeo de un vector de entrada a una salida categórica. Cuando algunas de las entradas que faltan quizá, en lugar de proporcionar una única función de clasificación, el algoritmo de aprendizaje tiene que aprender un conjunto de funciones. Cada función corresponde a clasificar x con un subconjunto diferente de sus entradas faltantes. Este tipo de situaciones con frecuencia se dan en el diagnóstico médico, debido a que muchos tipos de tests médico son costoso o invasivo.

3.1.1.3 Regresión: Este tipo de tarea, se le pide al programa informático predecir un valor numérico dado alguna entrada. Para resolver esta tarea, los algoritmos de aprendizaje pidieron a la salida de una función $f : R^n \rightarrow R$. Este tipo de tarea es similar clasificación, excepto que el formato de salida es diferente.

3.1.1.4 Transcripción: En este tipo de tarea, se le pide al sistema de aprendizaje automático para observar una representación relativamente poco estructurado de algún tipo de datos y transcribir en forma discreta, textual. Por ejemplo, en el reconocimiento óptico de caracteres, el programa de ordenador se muestra una fotografía contiene una imagen de texto y se le pide que devuelva este texto en la forma de una secuencia de caracteres (por ejemplo, en formato ASCII o Unicode). Google Street View utiliza el aprendizaje profundo para procesar números de direcciones de esta manera.

3.1.1.5 Detección de anomalías: En este tipo de tarea, el programa filtra a través de un conjunto de eventos u objetos, y detecta aquellos que no son usuales. Un ejemplo de una tarea de detección de anomalías es detección de fraude con tarjeta de crédito. Al modelar sus hábitos de compra, una compañía de tarjetas de crédito puede detectar el mal uso de sus tarjetas. Si un ladrón le roba su tarjeta de crédito, las compras del ladrón suelen venir de una distribución de probabilidad diferente sobre los tipos de compra a la suya.

3.1.2 Categorización: Las tareas del aprendizaje automático se clasifican usualmente en 4 categorías, dependiendo de la naturaleza del aprendizaje.

Reinforcement Learming (Aprendizaje de Refuerzo): es un área de aprendizaje automático inspirado en la psicología conductista, refiere a cómo el software de agentes deben tomar acciones en un entorno con el fin de maximizar alguna noción de acumulativa recompensa. El problema, debido a su generalidad, se estudia en muchas otras disciplinas, como la teoría de juegos, teoría de control, investigación de operaciones, teoría de la información, la optimización basada en

la simulación, los sistemas multiagente , la inteligencia de enjambre , estadísticas y algoritmos genéticos . En la literatura de investigación y control de las operaciones, el campo donde se estudian los métodos de aprendizaje de refuerzo se llama programación dinámica aproximada. El problema se ha estudiado en la teoría de control óptimo, aunque la mayoría de los estudios se refieren a la existencia de soluciones óptimas y su caracterización, y no con los aspectos de aprendizaje o de aproximación. En la economía y la teoría de juegos, aprendizaje por refuerzo puede ser utilizado para explicar cómo puede surgir equilibrio bajo la racionalidad limitada.

- Aprendizaje Supervisado: Al equipo se le ingresan ejemplos y sus salidas deseadas y se busca generar una regla para crear una correspondencia entre entradas y salidas.

- Aprendizaje no Supervisado: Las etiquetas o salidas deseadas no le son dadas al sistema, haciendo que encuentre una estructura por sí mismo al reconocer patrones en los datos.

- Aprendizaje Semi-supervisado: Yace en medio de los 2 métodos anteriores, donde solo se le proporciona una señal de entrenamiento incompleta, con sólo algunas de las etiquetas de las salidas deseadas.

3.1.3 Algoritmos de Clasificación: Existen diversos tipos de algoritmos de clasificación. A continuación se listan brevemente

Árboles de decisión: El aprendizaje por este método, usa un árbol para la toma de decisiones como modelo predictivo, creando enlaces de lo observado en un elemento y su valor.
Redes Neuronales Artificiales: Algoritmos inspirados en la estructura y aspectos funcionales de las redes neuronales biológicas. La computación es estructurada en términos de un grupo artificial de neuronas interconectadas. Son utilizadas para modelar relaciones complejas entre las entradas y salidas de un sistema, hallar patrones en los datos o capturar la estructura estadística de una distribución de probabilidad conjunta desconocida entre las variables observadas.
Clustering: Asignación de un conjunto de observaciones en sub grupos, conteniendo cada uno de estos, datos con características similares de acuerdo a criterios pre establecidos. Es un método de aprendizaje no supervisado y una técnica común para el análisis estadístico de datos.
Aprendizaje por Similitud y métricas: Recibe pares de ejemplos considerados similares y pares de objetos poco similares. Necesita entonces aprender una

función de similitud o de distancia de métricas que pueda predecir si nuevos objetos son o no semejantes. Suele ser usando en sistemas de recomendación.

3.2 REDES NEURONALES ARTIFICIALES

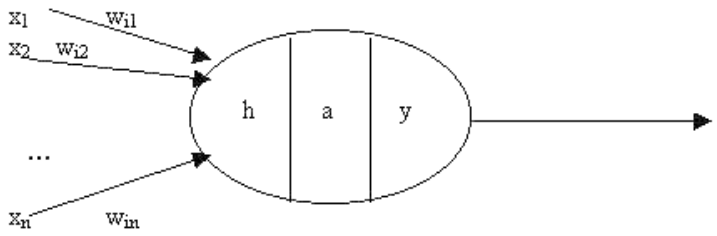
Las Redes Neuronales Artificiales son un paradigma del aprendizaje automático, inspirado en el funcionamiento del sistema nervioso animal. Es un sistema de interconexión de neuronas que colaboran entre sí para producir estímulos de salida. Puede definirse como un grafo dirigido con las siguientes restricciones:

- Las neuronas, también llamados nodos se llaman *elementos de proceso* (EP).
- Los enlaces se llaman *conexiones* y funcionan como caminos unidireccionales instantáneos.
- Cada EP puede tener cualquier número de conexiones.
- Todas las conexiones que salgan de un EP deben tener la misma señal.
- Los EP pueden tener *memoria local*.
- Cada EP posee una *función de transferencia* que, en función de las entradas y la memoria local produce una señal de salida y / o altera la memoria local.
- Las entradas a la RNA llegan del mundo exterior, mientras que sus salidas son conexiones que abandonan la RNA.

3.2.1 Modelo de Neurona Artificial: El modelo de neurona de Rumelhart y McClelland desarrollado en 1986 es una unidad de cálculo que intenta modelar el comportamiento de una neurona animal real. Es la unidad básica y esencial para la construcción de redes neuronales artificiales.

El modelo define un elemento de proceso (EP) o neurona artificial, como un dispositivo que genera una única salida a partir de un conjunto o vector de entradas.

Figura 1. Modelo de Neurona Artificial.



Fuente: http://avellano.usal.es/~lalonso/RNA/imagenes/modeloNA.jpg

Este modelo consta de un conjunto o vector de entradas **x**, un conjunto de pesos sinápticos w_{ij}, una regla de propagación que suele ser una suma ponderada del

producto escalar del vector de entrada y el vector de pesos, donde si las influencias excitadoras positivas dominan, se produce una señal positiva y transmite dicha señal a otras neuronas por su sinapsis de salida y puede ser expresada como,

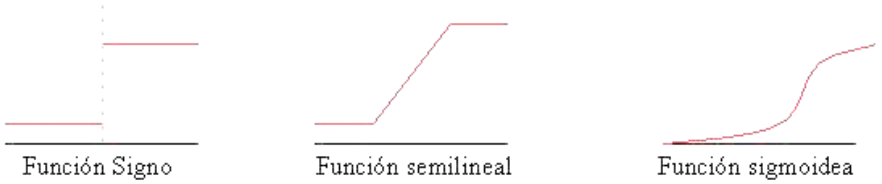
$$\mathbf{h}_i(\boldsymbol{t}) = \sum \boldsymbol{w_{ij}x_j}$$

También puede ser usada la distancia de Voronoi o de Mahalanobis, pero es más común el uso de la distancia euclidiana entre ambos vectores:

$$\mathbf{h}_i(\boldsymbol{t}) = \sum (\boldsymbol{x_j w_i})^2$$

También cuenta con una función de activación $a_i(t)=f(a_i(t-1), h_i(t))$ que proporciona el estado de activación de la neurona en función del estado anterior y el valor post sináptico que no suele tener en cuenta el estado anterior de la neurona, sino sólo el potencial *$h_i(t)$.* Suele ser una función determinista y, casi siempre, continua y monótona creciente y con una función de salida $F_i(t)$ que proporciona la salida $y_i(t)$ en función del estado de activación.

Figura 2. Funciones de Activación

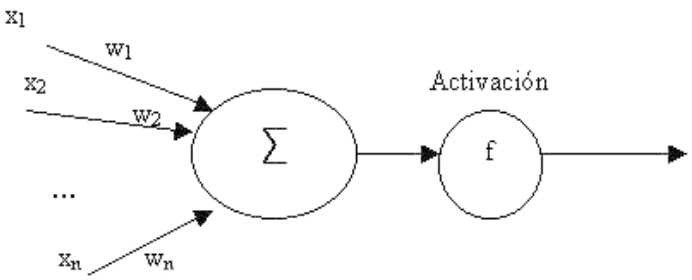


Fuente: http://avellano.usal.es/~lalonso/RNA/imagenes/funcionesActivacion.jpg

La función de salida suele ser la identidad. En algunos casos es un valor umbral (la neurona no se activa hasta que su estado supera un determinado valor).

Con todo esto, el modelo de neurona queda bastante simplificado:

Figura 3. Modelo de Neurona Simplificado

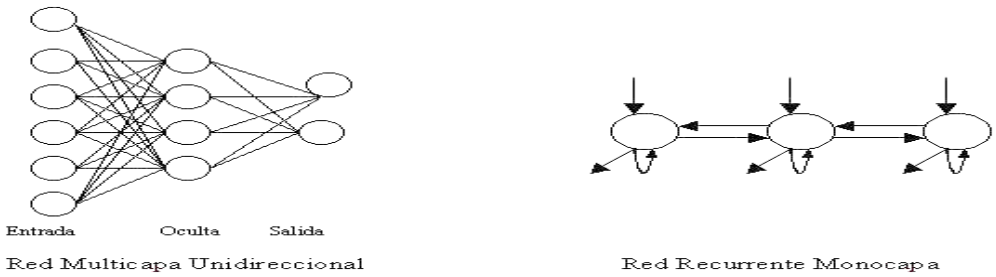


3.2.2 Arquitectura de las RNA: La arquitectura de una RNA es la estructura o patrón de conexiones de la red. Es conveniente recordar que las conexiones sinápticas son direccionales, es decir, la información sólo se transmite en un sentido.
En general, las neuronas suelen agruparse en unidades estructurales llamadas *capas*. Dentro de una capa, las neuronas suelen ser del mismo tipo. Se pueden distinguir tres tipos de capas:

- De *entrada*: reciben datos o señales procedentes del entorno.
- De *salida*: proporcionan la respuesta de la red a los estímulos de la entrada.
- *Ocultas*: no reciben ni suministran información al entorno (procesamiento interno de la red).

Generalmente las conexiones se realizan entre neuronas de distintas capas, pero puede haber conexiones intracapa o *laterales* y conexiones de *realimentación* que siguen un sentido contrario al de entrada-salida.

Figura 4. Arquitectura de RNA



Fuente: http://avellano.usal.es/~lalonso/RNA/imagenes/arquitecturaANS.jpg

3.2.3 Aprendizaje de las RNA: Es el proceso por el que una RNA actualiza los pesos (y, en algunos casos, la arquitectura) con el propósito de que la red pueda llevar a cabo de forma efectiva una tarea determinada.

Hay tres conceptos fundamentales en el aprendizaje:

- Paradigma de aprendizaje: información de la que dispone la red.
- Regla de aprendizaje: principios que gobiernan el aprendizaje.
- Algoritmo de aprendizaje: procedimiento numérico de ajuste de los pesos.

Para el aprendizaje de las redes neuronales se busca un algoritmo que permita encontrar los pesos y sesgos, para que el resultado obtenido se aproxime al valor real de la etiqueta *y(x)* para todas las entradas de entrenamiento *x*. Se define una función de costo, para determinar la precisión del método está definida así:

$$J\,(\theta)\;=\;\frac{1}{2}\sum_{t=1}^{m}(h_\theta(x^{(t)})-(y^{(t)}))^2$$

Donde θ representa los pesos de la red, $h_\theta(x^{(t)}$ la hipótesis realizada y $y^{(t)}$ las etiquetas reales. Para hallar el valor óptimo de los pesos, es necesario que estos varíen con el tiempo, moviéndose en dirección de la pendiente decreciente de J más pronunciada. Estos valores pueden ser extraídos y almacenados para no tener que realizar el proceso de entrenamiento para hallarlos nuevamente.

La tasa de aprendizaje está representada por α. Este algoritmo va dando pasos en la dirección de la pendiente decreciente más pronunciada de J. Éste es de tipo de minimización del error, pero existen cuatro tipos de algoritmos de aprendizaje:

- Minimización del error: gradiente descendente, retropropagación, etc. La modificación de pesos está orientada a que el error cometido sea mínimo.
- Boltzmann: para redes estocásticas, donde se contemplan parámetros aleatorios.
- Hebb: cuando el disparo de una célula activa otra, el peso de la conexión entre ambas tiende a reforzarse (Ley de Hebb).
- Competitivo: sólo aprenden las neuronas que se acercan más a la salida deseada.

Los algoritmos, y en general los procesos de aprendizaje, son complejos y suelen llevar bastante tiempo computacionalmente hablando. Su ventaja es que una vez ha aprendido, la red puede congelar sus pesos y funcionar en modo recuerdo o ejecución.

3.3 REDES NEURONALES CONVOLUCIONALES

En aprendizaje de máquina, una red neuronal convolucional, CNN por sus siglas en inglés, es un tipo de red neuronal artificial anticipativa, donde las neuronas individuales son apiladas de forma que respondan similarmente a regiones superpuestas en la corteza visual (la cual es responsable de estímulos visuales, especializada en procesar información de objetos estáticos y en movimiento, excelente en el manejo de reconocimiento de patrones). Las redes convolucionales fueron inspiradas en procesos biológicos y son variaciones de perceptores multicapa diseñados para utilizar la mínima cantidad de pre procesamiento. Tiene gran campo de aplicación en reconocimiento de imagen, video y procesamiento de lenguaje.

Para el reconocimiento de imágenes, las redes neuronales convolucionales consisten de múltiples capas de colecciones de pequeñas neuronas que procesan porciones de la imagen de entrada, llamadas campos receptores. Las salidas de

éstas son apiladas para superponerse y obtener una mejor representación de la imagen original, proceso que se repite para cada capa. El apilamiento permite a éstas redes tolerar dicha traducción de las imágenes de entrada.

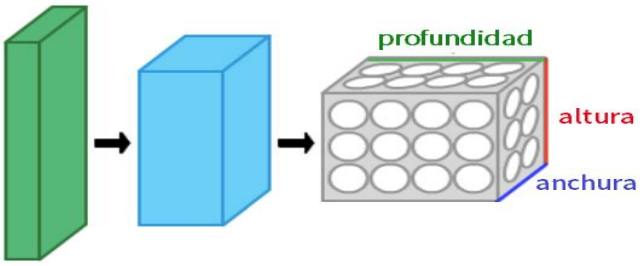
Las redes convolucionales pueden incluir capas locales o globales de Clustering que combinan las salidas del grupo de neuronas. También consisten en varias combinaciones de capas convolucionales y capas completamente conectadas. Para reducir el número de parámetros libres y mejorar la generalización, se introduce una operación de convolución en pequeñas regiones de la entrada. Una gran ventaja de las redes convolucionales es el uso de pesos compartidos en las capas convolucionales, es decir que el mismo filtro o banco de pesos es usado para cada pixel de la capa, reduciendo la huella de memoria y mejora el rendimiento. Comparado con otros algoritmos de clasificación de imágenes, las redes neuronales convolucionales usan relativamente poco pre procesamiento. Esto quiere decir que la red es responsable de aprender los filtros que deben ser programados manualmente en los algoritmos tradicionales. La falta de dependencia de conocimiento a priori y de esfuerzo humano en el diseño de características es una gran ventaja de las CNNs.

3.3.1 Características Distintivas: El tradicional modelo de los perceptrones multicapas fueron exitosamente usados para reconocimiento de imágenes, dada la completa conectividad entre los nodos, sufren del efecto Hughes (también conocido como maldición de la dimensión) y por tanto no se adaptan bien a las imágenes de mayor resolución, pues no toma en cuenta la estructura espacial, de los datos, tratando los pixeles de entrada cercanos en el mismo nivel que los muy lejanos. Una conectividad completa entre neuronas es un desperdicio en el framework de reconocimiento de imágenes (ya que el gran número de parámetros lleva rápidamente al overfitting, o sobre ajuste, además de que tener más parámetros indica mayor tamaño computacionalmente).

Estos modelos mitigan los retos de la arquitectura anterior al explotar la fuerte correlación espacial local presente en imágenes naturales. Opuesto al perceptrón multicapa, las redes convolucionales tienen las siguientes características distintivas:

- Volúmenes 3D de neuronas: Las capas de una CNN tienen neuronas en arreglos tridimensionales, a lo alto, ancho y profundo. Las neuronas dentro de la capa solo están conectadas a una pequeña región de la capa anterior, llamada campo receptivo. Distintas clases de capas son agrupadas para formar la arquitectura de la red convolucional.

Figura 5. Capas de una CNN en un arreglo tridimensional.



Fuente: http://cs231n.github.io/assets/cnn/cnn.jpeg

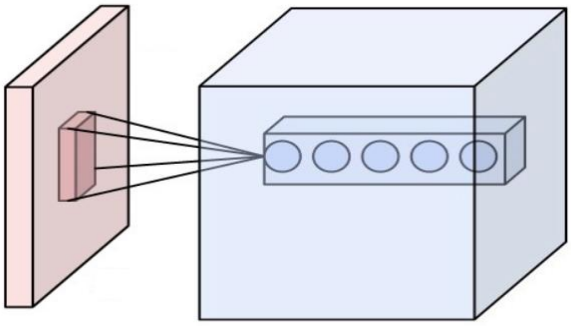
- Conectividad local: las CNNs explotan la correlación espacial al reforzar un patrón de conectividad local entre neuronas de capas adyacentes. La arquitectura asegura que los filtros aprendidos, produzcan la mejor respuesta a un patrón de entrada local espacialmente. Apilar tantas capas lleva a que los filtros no lineales se conviertan incrementalmente en filtros globales. Esto permite que la red inicialmente cree buenas representaciones de pequeñas partes de la entrada y luego ensamble representaciones de áreas más grandes de éstas
- Pesos Compartidos: Cada filtro es replicado a través de todo el campo visual. Dichas unidades replicadas comparten una misma parametrización y forman un mapa de características. Esto quiere decir que todas las neuronas en una capa convolucional, detecta exactamente la misma característica. Las unidades replicadas entonces permiten que las características sean detectadas a pesar de su posición en el campo visual, constituyendo así la propiedad de la traducción invariante.

3.3.2 Arquitectura: Una CNN consiste de un número de capas convolucionales y submuestreo, seguidas opcionalmente por capas completamente conectadas. La entrada a una capa convolucional es una imagen con alto m, ancho m y una cantidad de canales r. La capa convolucional tiene k filtros o kernels de tamaño $n * n * q$, donde $n \leq q$, puede variar para cada kernel. El tamaño de los filtros da lugar a la estructura localmente conectada, convolucionada con la imagen para producir k mapas de características de tamaño m-n+1. Cada mapa es posteriormente submuestreado con Clustering medio o máximo en pxp regiones contiguas donde p tiene un rango de 2 para imágenes pequeñas y no suele ser mayor que 5 para entradas más grandes. Las capas usadas son explicadas a continuación:

3.3.2.1 Capa convolucional: Es el bloque de construcción central de una CNN. Los parámetros de la capa consisten en un conjunto de filtros que pueden ser aprendidos, llamados kernels, que tienen un pequeño campo receptivo, pero se extienden a través de la totalidad de la profundidad del volumen de entrada. Durante el pase hacia adelante, cada filtro es convolucionado a lo alto y ancho del volumen de entrada, computando el producto punto entre las entradas del filtro y la entrada produciendo un mapa de activación de 2 dimensiones en dicho

filtro. Como resultado, la red aprende filtros que se activan al ver una característica específica en alguna posición espacial de la entrada. Apilar los mapas de activación para todos los filtros a lo largo de la dimensión de profundidad, forma el volumen de salida de la capa de convolución. Cada entrada en el volumen de salida también puede interpretarse como una salida de una neurona que mira una pequeña región en la entrada y comparte parámetros con otra en el mismo mapa de activación.

Figura 6. Neuronas de una capa convolucional conectada a su campo receptivo.



Fuente: http://cs231n.github.io/assets/cnn/depthcol.jpeg

- Conectividad local: Al tratar con entradas de grandes dimensiones, como imágenes, es impráctico conectar neuronas a todas las neuronas del volumen anterior. Estas arquitecturas de redes no tienen en cuenta la estructura espacial de los datos. Las redes convolucionales explotan la correlación espacial local al reforzar un patrón de conectividad local entre neuronas de capas adyacentes, cada neurona está conectada solo a una pequeña región del volumen de entrada. El alcance de ésta es un hiper parámetro llamado campo receptivo de la neurona. Las conexiones son locales en el espacio, siempre se extienden a lo largo y ancho de la profundidad total del volumen de entrada. Dicha arquitectura asegura que los filtros aprendidos produzcan la respuesta más fuerte a un patrón de entrada espacialmente local.

- Arreglo espacial: El tamaño del volumen de salida de una red convolucional es controlado por 3 hiperparámetros, profundidad, paso y rellenado con ceros.
    - La profundidad del volumen de salida controla el número de neuronas en la capa que conecta a la misma región del volumen de entrada. Todas éstas aprenden a activarse para diferentes características en la entrada.

- El paso controla cómo se distribuyen las columnas de profundidad alrededor de las dimensiones espaciales. Cuando el paso es 1, una nueva columna de neuronas se asigna a una posición espacial a una unidad espacial de distancia. Esto lleva a la rápida superposición de campos receptivos entre columnas y a una gran cantidad de volúmenes de salida. Inversamente, al usar mayores pasos los campos receptivos se solapan menos y el volumen de salida tendrá menos dimensionalidad espacial.
- En ocasiones es conveniente rellenar la entrada con ceros en el borde del volumen. El tamaño de este rellenado es el tercer hiperparámetro. Permite controlar el tamaño de los volúmenes de salida, aunque en ocasiones es deseable conservar el tamaño espacial exacto.
  El tamaño espacial del volumen de salida puede ser computado como una función del tamaño del volumen de entrada *W*, el tamaño del campo receptor de las neuronas de la capa convolucional *F*, el paso aplicado S y la cantidad de rellenado con ceros en el borde *P*. La fórmula para calcular cuántas neuronas cabrán en el volumen dado es (W-F+2P)/S+1 Si el resultado obtenido no es un entero, los pasos han sido asignados erróneamente y las neuronas no pueden ser apiladas en el volumen de entrada simétricamente. En general asignar el rellenado en $p = \frac{F-1}{2}$ cuando el paso es 1, asegura que el volumen de entrada y de salida tenga el mismo tamaño espacial.

- Intercambio de parámetros: Es usado para controlar el número de parámetros libre. Asume que con una porción bidimensional a profundidad, las neuronas en cada porción de profundidad usarán los mismos pesos y sesgo. Como todas las neuronas a una misma profundidad comparten parametrización, el paso de la información hacia la siguiente capa en cada porción a profundidad de la capa, puede ser computada como una convolución de los pesos de las neuronas con el volumen de entrada. Es común llamar filtro o kernel al conjunto de pesos que es convolucionado con la entrada. El resultado de dicha operación es un mapa de activación y los conjuntos de éstos para cada filtro diferente son apilados juntos en la dimensión de profundidad para producir el volumen de salida. Compartir parámetros contribuye a la invariancia en la traducción de la arquitectura de las redes neuronales convolucionales.

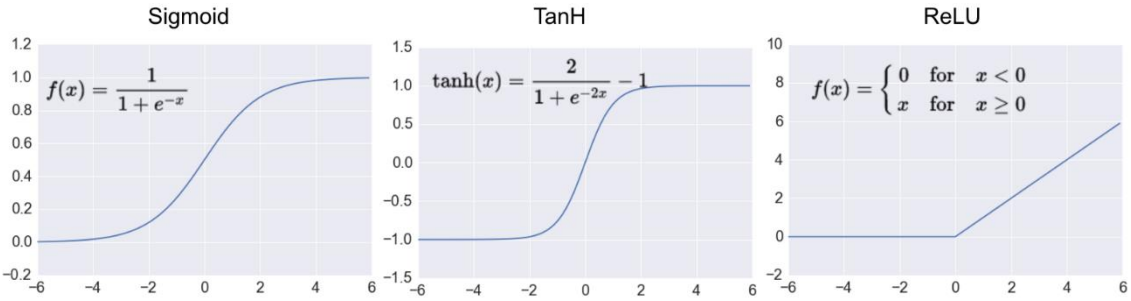
3.3.2.2 Capa de Clustering o Pooling: Es una capa altamente importante, pues se encarga de disminuir el tamaño después de extraer las características por medio de la convolución. El Pooling particiona la imagen de entrada en un conjunto de rectángulos sin superposición y para cada una de estas subregiones, da como resultado, el valor máximo. Intuitivamente, es posible decir que una vez encontrada una característica, su posición exacta no es tan importante como la

relativa a otras características. La función de la capa de Pooling es reducir la cantidad de parámetros y de computación en la red, al disminuir progresivamente el tamaño espacial de la representación consiguiendo controlar también el sobreajuste a los datos- Es común insertar capas de Pooling entre capas de convolución. Ésta provee una invariancia en la traducción de los datos, lo que significa que una misma característica estará activa aún si la imagen es sometida a traducciones y reducciones. Son deseadas características invariantes.

Actualmente es usado al Max-Pooling, cuya salida es el valor máximo de cada región tomada, pero es posible realizar otras funciones de Pooling, como la media y la norma L2. La media era la usada anteriormente, pero en la práctica el máximo ha demostrado obtener mejores resultados. Dada la agresiva reducción en el tamaño de la representación, se tiende a usar filtros pequeños.

3.3.2.3 Capa ReLu: Es una capa de neuronas que aplica la función de activación no saturante *f(x) = max (0, x)* Aumenta las propiedades no lineales de la función decisoria y de la red en general sin afectar los campos receptivos de la capa convolucional. Entre otras funciones usadas para aumentar la no linealidad se encuentran la tangente hiperbólica *f(x) = tanh(x)* y la función sigmoide $f(x) = 1/ (1+e^{-x})$ Comparada a otras funciones, es preferible el uso de ReLu dando un entrenamiento más ràpido de la red sin causar cambios significativos en la precisión de la generalización.

Figura 7. Funciones de activación comunes en redes neuronales artificiales



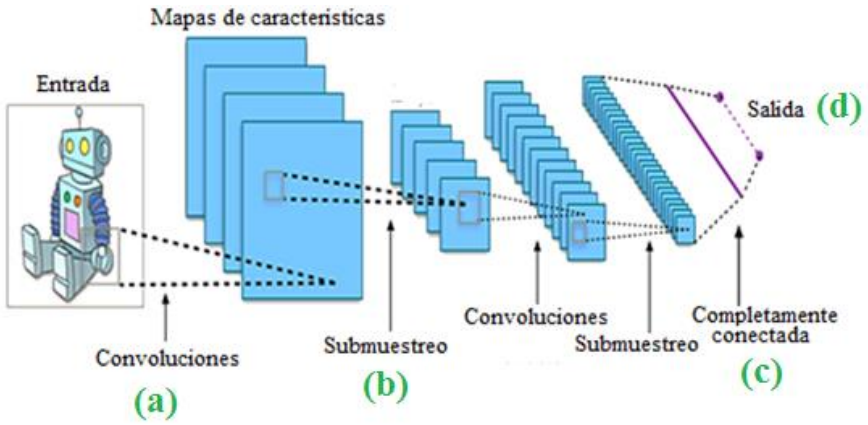
Fuente: http://adilmoujahid.com/images/activation.png

3.3.2.4 Capa completamente conectada: Después de varias capas convolucionales y de Clustering máximo, el razonamiento de alto nivel en la red neuronal es realizado por medio de estas capas. Las neuronas de ésta, tienen conexiones completas a todas las activaciones en la capa inmediatamente anterior, por tanto sus activaciones pueden ser computadas con una multiplicación de matrices seguida por desplazamiento del sesgo.

3.3.2.5 Capa de pérdida: Especifica la penalización hecha por la red, en la diferencia entre el valor predicho y valor real de la etiqueta. Diversas funciones de pérdida son aplicables para las diferentes tareas. Softmax es utilizado para predecir una sola clase, que es mutuamente excluyente respecto a las demás. La entropía sigmoide cruzada es usada en predicciones donde se espera un valor continuo entre 0 y 1 y la función de pérdida Euclideana, para la regresión de etiquetas con valores reales.

Una arquitectura típica de redes neuronales convolucionales está conformada por algunas capas ReLu, seguidas por capas de Clustering, repitiendo este patrón, hasta que el valor de entrada haya sido unido espacialmente hasta alcanzar un tamaño mucho menor, que pueda considerarse pequeño. Llegado un punto, es natural, hacer la transición a una capa completamente conectada, que mostrará el valor o valores de salida correspondientes.

Figura 8. Arquitectura típica de una CNN



Fuente: https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png

- (a) En esta parte de la figura podemos observar como una característica aprendida por la CNN pasa por la imagen, esto con el fin de activarse al encontrar el patrón para el cual fue entrenada aprender (color, borde, rasgo ..)
- (b) En esta parte de la figura las activaciones proveniente de (a) son reducidas por medio de capas de Pooling, con el fin de reducir el gran tamaño de los datos que surgen tras cada capa de convolución
- (c) En esta parte de la figura podemos observar las capas totalmente conectadas están son las que tienen el poder discriminativo y son quienes determinan la probabilidad de pertenecer a una clase
- (d) En esta parte de la figura podemos observar la salida que es un vector de probabilidad de pertenecer a cada clase, del cual tomamos el de mayor valor como la clase a la cual pertenece la imagen pasada por la CNN.

3.3.3 Métodos de Regularización: Regularización es una *técnica* utilizada en un intento de resolver el sobre entrenamiento, en la mayoría de los modelos estadísticos sucede que al ser sobre entrenados aprenden el conjunto de datos del entrenamiento perdiendo generalidad sobre nuevos datos y ajustándose a características muy específicas sobre el conjunto de entrenamiento, en muchos casos dándole más importancia a ciertas características del dataset, casi que olvidando o teniendo muy poco en cuenta otras, lo que buscan los métodos de regularización es darle igual importancia a mas características a la hora de entrenar el modelo y obtener mejores resultados a la hora de predecir con datos nunca antes visto durante el entrenamiento

3.3.3.1 Métodos empíricos

- Dropout (perdida) : Las capas completamente conectadas, tienden a estar sobre ajustadas, pues tienen la mayoría de los parámetros. Este método es introducido para evitar dicho problema al no entrenar todos los nodos con todos datos de entrada, además aumenta la velocidad de entrenamiento y las permite que las características aprendidas sean más robustas lo que da como resultado una mejor generalización. En cada etapa del entrenamiento, nodos individuales son eliminados de la nueva red con una probabilidad de 1-p o se mantienen con probabilidad p, produciendo una red reducida, eliminando también todos los nodos entrantes o salientes de un nodo eliminado y se entrena con la red obtenida. Posteriormente se reinsertan los nodos con sus pesos originales.

- DropConnect: Es la generalización del Dropout, para regularizar grandes capas completamente conectadas, donde cada conexión puede ser eliminada, en lugar de eliminar unidades de salida, con una probabilidad 1-p. Cada unidad recibe su entrada de un subconjunto aleatorio de unidades de la capa anterior. En vez de poner en cero un subconjunto aleatorio de activaciones, en este método se cambian los valores de un subconjunto de pesos. Una capa completamente conectada con DropConnect se convierte en una capa escasamente conectada donde las conexiones se realizan aleatoriamente durante la etapa de entrenamiento.

- Datos Artificiales: El grado de sobre ajuste en la curva de aprendizaje de un modelo está determinado por su poder y la cantidad de datos de entrenamiento que recibe, proveer más datos de entrenamiento a una red convolucional, puede ayudar a reducirlo. Teniendo en cuenta que se suelen usar todos los datos disponibles para entrenar las redes, si es posible se debe buscar crear u obtener nuevos datos reales, de lo contrario es posible perturbar los existentes para obtener nuevos ejemplos.

3.3.3.2 Métodos explícitos

- Tamaño de la red: En cuanto a la regulaizacion con respecto al tamaño de la red se puede limitar el número de unidades ocultas y parámetros libres en la red, es la forma más sencilla de prevenir su sobreajuste, Esto restringe directamente el poder predictivo de la red, reduciendo la complejidad de la función que puede realizar en los datos.

- Restringir la norma máxima: Es posible aplicar un límite superior absoluto a la magnitud del vector peso para cada neurona y usar gradientes descendentes proyectados para reforzar la restricción. En la práctica, corresponde a la actualización de parámetros y luego realizar la restricción fijando el vector de pesos *w* de cada neurona para satisfacer que

$$|| \vec{w} ||_2 < c$$

3.4 CONJUNTO DE DATOS

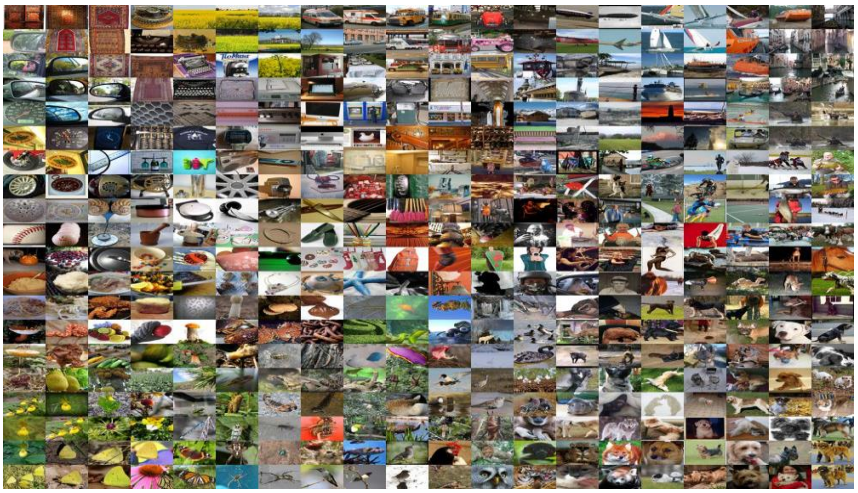
Actualmente en la literatura podemos encontrar 2 dataset que pueden ser usados con el fin de cumplir uno de nuestros objetivos (dataset anotados con una ontología de clasificación), ImageNet y CIFAR-100, estos dos conjuntos de datos tienen una jerarquía entre sus clases que ayuda a entender las relaciones entre ella, en el caso de Imagenet wornet y para cifar unas superclases previamente definidas

ImageNet cuenta con 1000 clases en el caso de CIFAR-100 contamos con 100

El principal uso de la semántica en tareas de aprendizaje automático se da para conjuntos de datos clasificados en gran cantidad de clases y anotados con una ontología de clasificación. En la literatura es posible encontrar 2 dataset que pueden ser usados para este fin, ImageNet y CIFAR-100. Son conjuntos de datos con una jerarquía entre clases, que permite entender más fácilmente sus relaciones, WordNet para el caso de ImageNet y para CIFAR-100 unas superclases definas previamente.

3.4.1 ImageNet.

Figura 9. Algunas imágenes del dataset ImageNet

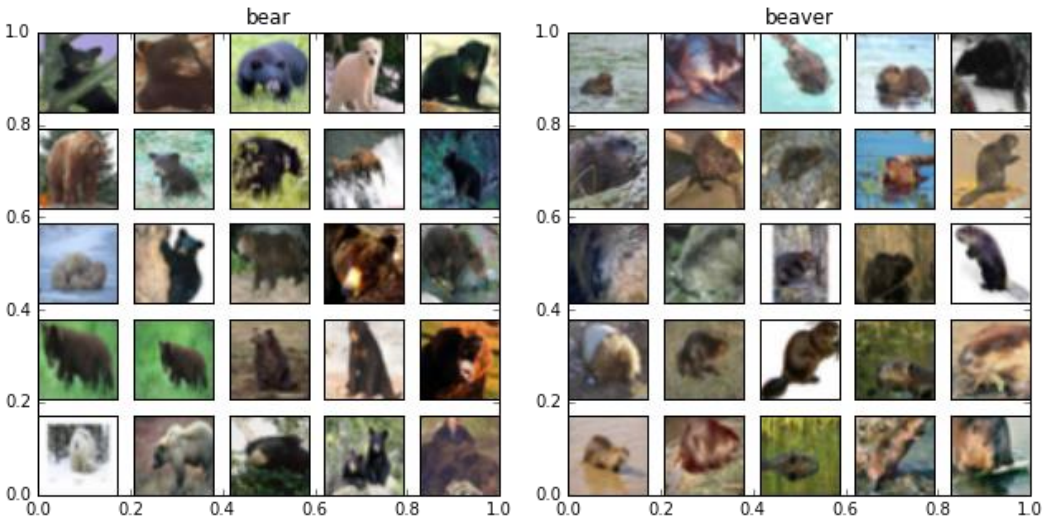


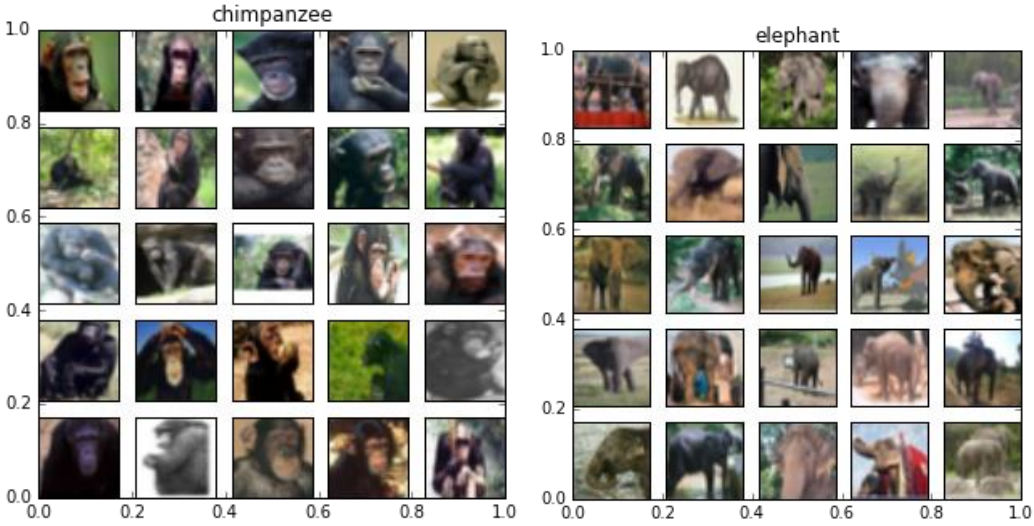
Fuente: https://assets.wired.com/photos/w_730/wp-content/uploads/2015/01/cnn-visualization-crop.jpg

ImageNet es una base de datos de la imagen organizadas de acuerdo con el WordNet jerarquía (en la actualidad sólo los sustantivos), en el que cada nodo de la jerarquía está representado por cientos y miles de imágenes. Donde actualmente se cuenta con un promedio de más de quinientas imágenes por nodo, las cuales están organizadas en 1000 clases, contando con un total de 14’197.122 imágenes, teniendo para el entrenamiento entre 5.000 a 20.000 imágenes por clase.

3.4.2 CIFAR-100

Figura 10. Algunas imágenes del dataset CIFAR-100





CIFAR-100 cuenta con 500 imágenes de entrenamiento y 100 imágenes de prueba por clase. Las 100 clases en el CIFAR-100 se agrupan en 20 superclases. Cada imagen viene con una etiqueta de "fina" (la clase a la que pertenece) y una etiqueta "gruesa" (la superclase a la que pertenece). En total contando con 50.000 imágenes para el entrenamiento y 10.000 para la validación.

Figura 11. Clases y superclases del dataset CIFAR-100

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Fuente: https://www.cs.toronto.edu/~kriz/cifar.html

3.5 VALIDACION DE MODELOS

Después de construir un modelo podemos evaluar su precisión mediante métricas como son el TPR, TNR, accuracy, precision, F1-Score, AUC, entre otras. . Como los datos que permiten construir el modelo contienen variaciones y ruido, existen técnicas para disminuir el error midiendo la precisión sobre observaciones que no habían sido vistas durante el entrenamiento.

3.5.1 Métricas: Las métricas son estadísticos que revelan características en una predicción, algunas son útiles para medir la precisión sobre una clase dada, otras calculan el desempeño general de la predicción. Se usan como el objetivo de un modelo o para medir el desempeño del mismo.
Hay métricas aceptadas de forma general y con objetivos claros, definidas sobre la falsedad o veracidad de la predicción. A continuación se presentan métricas que fueron usada para la validación de nuestros modelos.

3.5.1.1 Recall: Recall es una métrica para evaluar la predicción de las instancias positivas, generalmente la clase uno (1) para un problema de clasificación binario. Hay que notar que las instancias negativas verdaderas, clase cero (0), no se toman en cuenta en esta métrica.

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

3.5.1.2 Precision o Positive Predictive Value (PPV): El valor predictivo positivo se define como el objetivo de aumentar los sujetos clasificados correctamente como positivos, pero busca disminuir los sujetos negativos clasificados como positivos (falsos positivos).

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

3.5.1.3 Accuracy, (Precisión o Exactitud): Accuracy se traduce a español como precisión o exactitud, por lo que se debe distinguir bien entre la precisión que tiene un modelo en la tarea de clasificación y la medida de la métrica precisión (accuracy) para un modelo. El objetivo de la accuracy es mejorar el desempeño general, clasificar sujetos positivos y negativos de forma correcta. Pero por su definición matemática, generalmente no es la métrica correcta para conjuntos de entrada desbalanceados.

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

3.5.1.4 F1 Score: También se conoce como la puntuación F o la Medida F. Dicho de otro modo, la puntuación F1 transmite el equilibrio entre la precisión y recall.

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\text{recall}} + \frac{1}{\text{precision}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

3.6 ESPACIOS EMBEBIDOS

Un embebido es una representación de un objeto topológico, grafico campo, etc., espacio determinado de manera que su conectividad o propiedades algebraicas se conservan. Por ejemplo un embebido grafico conserva la conectividad un embebido de cambo conserva la estructura algebraica.

Un espacio X esta embebido en otro espacio Y cuando las propiedades de Y restringidas a X son los mismos que las propiedades de X. Por ejemplo, los números racionales están embebidos en los reales, y los números enteros están embebidos en los racionales. En la geometría, la esfera está embebida en R ^ 3 como la esfera unidad.

En el caso del conjunto de imágenes que vamos a trabajar mas adelante, al generar vectores embebidos por cada una de ellas (generando una nueva representación de cada una de las imágenes a partir de las capas de convolución), teniendo nuevas representaciones que conservaran las propiedades entre ellas para así poder evaluar el desempeño de las agregaciones semánticas generadas a partir de las capas de convolución.

4. ESTADO DEL ARTE

Las CNN nacieron de la necesidad de tener una herramienta más potente para clasificación de imágenes y hasta el momento se han mantenido como tal. Sin embargo, recientemente se ha estudiado el posible aporte que pueden ofrecer las distintas características aprendidas en las capas de convolución, se ha planteado que éstas aprenden una semántica y que es posible extraer de ellas dicho conocimiento aprendido. Hasta hace poco, la clasificación de imágenes se había realizado con redes neuronales convolucionales profundas, entrenadas con una capa de salida con tantas unidades como número de clases, lo que hace que la distinción sea más difusa y difícil de obtener a medida que éstas clases aumentan.

Se han propuesto métodos para mapear imágenes en espacios embebidos semánticos continuos. En algunos casos, el espacio embebido es entrenado junto a la transformación de la imagen, en otros, la semántica del espacio embebido es establecida por una tarea de procesamiento de lenguaje natural independiente y luego esa transformación de la imagen en dicho espacio es aprendida en una segunda etapa. Los proponentes de estos sistemas embebidos de imágenes han hecho hincapié en sus ventajas sobre la clasificación tradicional n-way (n clases fijas sin extraer información semántica) formulando el entendimiento de las imágenes, particularmente en términos de la promesa del aprendizaje zero-shot, aprendizaje donde es posible reconocer clases no usadas en el entrenamiento y con las que el sistema nunca ha tenido contacto ni conocimiento previo. En los años 2013 y 2014 se desarrollaron 2 metodologías De ViSE y CON-Se que son utilizadas en el aprendizaje zero-shot para aprovechar las ventajas que éste ofrece. El método mapea las imágenes en el espacio semántico embebido por medio de la combinación convexa del vector de la etiqueta de la clase y no requiere entrenamiento adicional. Confiere muchas ventajas asociadas con esquemas de imágenes embebidas más complejos superando los métodos aprendizaje anteriores, el conocimiento semántico mejora estas predicciones alcanzando hasta un 18% de acierto a través de miles de nuevas etiquetas nunca vistas por el modelo visual.

Gracias a la extracción de la semántica del lenguaje natural, se ha mejorado la clasificación de objetos. Sin embargo, estudios recientes han mostrado que las capas de convolución de una CNN entrenada son espacios ricos en semántica. Ésta semántica extraída puede ser utilizada en las metodologías ConSE y De ViSE, pero aún no se tiene conocimiento de qué tan buena sea esta semántica, qué factores podrían ayudar a mejorarla, para su uso en entrenamientos zero-shot u otras tareas en machine learning.

5. TRABAJO REALIZADO

Características aprendidas por CNNs en las capas de convolución se han explorado a través de técnicas de visualización [14]. El patrón que esta codificado en una característica en particular puede ser mostrado a través de tales métodos, por ejemplo, mostrar los parámetros de entrada que maximizan la activación de esa característica en particular [13]. Sin embargo, para explorar el espacio embebido como un todo requiere de métodos más generales. Algunos autores exploran los embebidos definidos por capas de neuronas que se encuentran a diferentes profundidades, y encontraron que al tomar la última capa de características, estos datos están basados en la naturaleza exterior o interior de la imagen [1]. Siguiendo este mismo método otros autores usando la primera capa totalmente conectada de un modelo para definir el espacio embebido [8]. Usando este tipo de representaciones para definir el espacio embebido los autores buscan explorar la detección de atributos entre otros desafíos, donde encontraron que las máquinas de soporte vectorial pueden utilizarse para identificar clases abstractas como hombre o ella usa gafas.

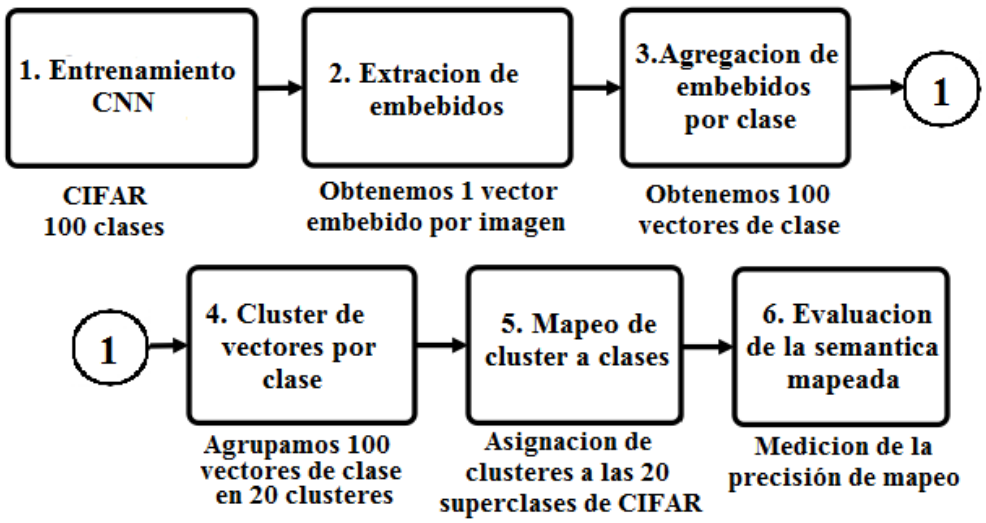
El objetivo del proyecto es transferir todo el aprendizaje aprendido por las capas de convolución de la CNNs creando nuevas representaciones de las imágenes mediante vectores embebidos que contiene las activaciones de las imágenes al ser pasadas por las capas de convolución.

CNNs profundas requieren de grandes conjuntos de datos para lograr un rendimiento competitivo y evitar el sobreajuste, debido al gran número de parámetros que se pueden aprender. Un ejemplo de esto es, el ganador del desafío 2015 IMAGEnet ILSVRC utiliza un modelo con más de 150 capas [5]. Lo que busca la transferencia de aprendizaje es utilizar tales modelos para ser reutilizados para nuevos conjuntos de datos y contextos a través de un proceso de fine tuning que es mucho menos costoso. Sin embargo, dicha transferencia no es inmediata cuando los conjuntos de datos son de naturaleza muy diferente [7][9][12] y arquitecturas y estrategias de entrenamiento tienen que ser reconsideradas [10].

6. METODOLOGÍA

El objetivo de este proyecto es entender el impacto de las diferentes arquitecturas de red y la transferencia de técnicas de aprendizaje en la representatividad embebida de las imágenes extraídas de las CNNs. Exploramos el comportamiento de los espacios embebidos definidos por los modelos de la CNN mediante la medición de la correlación de cada espacio embebido con las 20 superclases del conjunto de datos CIFAR-100. Nuestro enfoque consiste en agrupar los vectores de manera no supervisada 100 clases correspondientes a las 100 clases CIFAR-100 en 20 grupos, por lo que podemos obtener una medida de la capacidad de abstracción para cada embebido. Los vectores embebidos que contienen grupos más similares a las superclases de CIFAR-100 corresponderan con un espacio más rico en términos de semántica visual, como la abstracción es la fuente de muchos procesos de aprendizaje (es decir, de inferencia). Se extrajeron y analizaron vectores embebidos a partir de dos arquitecturas diferentes, CNN AlexNet [6] y GoogleNet [11], utilizando diferentes parametrizaciones en un flujo de trabajo como se muestra en la siguiente figura.

Figura 12 : Flujo de trabajo



7. DESARROLLO DEL PROYECTO

En este capítulo se presenta una descripción de la selección implementación de las redes neuronales empleadas para la clasificación así como el dataset escogido para el entrenamiento las herramientas usadas para el entrenamiento, la extracción de los vectores embebidos su clustering y los resultados obtenidos.

7.1 ETAPA 1: SELECCIÓN DEL DATASET

Después de hacer una revisión en los diferentes datasets disponibles, queda claro que la evaluación de este tipo de embebidos es un reto debido a la falta de semántica visual formales para comparar. Aun cuando estos datasets estan mapeados a palabras, lo que permite utilizar bases de datos léxicas como WordNet, no hay tal baseline para las imágenes que pueden permitir una evaluación formal. Para evitar esta limitación se utiliza el conjunto de datos CIFAR-100, que contiene imágenes asociadas a 100 clases diferentes, pero que también contiene 20 superclases compuestas por 5 clases cada uno, donde las "clases dentro de la misma superclase son similares y por lo tanto más difíciles de distinguir de las clases que pertenecen a diferentes superclases."[5]. En consecuencia, cada superclase representa ciertas semántica visual que pueden ser codificadas dentro del espacio de embebido de esas clases.

El uso de las 20 superclases del conjunto de datos CIFAR-100 como Ground Truth tiene ciertos inconvenientes. Como se muestra en la figura 10 algunas superclases parecen parecer arbitrariamente definidas (p. ej vehículos 1 vehículos 2) y la clasificación definida es solo uno de estos inconvenientes (existen otras características visuales p. ej color, forma, contexto). De todos modos se seguirá estrictamente esta clasificación, debido a que así fue definido el dataset.

7.2 ETAPA 2 : SELECCIÓN DE LAS ARQUITECTURAS Y ENTRENAMIENTO DE LAS CNNS

7.2.1 Selección de las arquitecturas de CNNs: Para la selección de arquitecturas de CNNs se tuvo en cuenta los resultados de accuracy obtenidos con el dataset ImageNet y la complejidad de sus capas de convolución, entre los candidatos encontramos:

- **AlexNet** (5 capas de convolución 3 totalmente conectadas)

- **GoogleNet** (9 modulos de incepcion 3 totalmente conectadas)

Descartamos otras arquitecturas tales cómo Vgg16, Vgg19 debido a la cantidad de memoria requerida para su entrenamiento, haciendo imposible su uso sobre la infraestructura con la que contamos.

Tabla 1. Arquitectura CNN Googlenet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Tabla 2. Arquitectura CNN Alexnet

Layer	Weights	FLOP
conv1	35K	211M
conv2	307K	448M
conv3	885K	299M
conv4	663K	224M
conv5	442K	150M
fc1	38M	75M
fc2	17M	34M
fc3	4M	8M
Total	61M	1.5B

Podemos observar información importante respecto a las arquitecturas que usamos donde Weights = Params, Ops = FLOP, donde se indican los pesos por capa y las operaciones que se hace en cada capa respectivamente, esto a simple vista nos muestra la complejidad de GoogleNet que busca tener capas de

convolución mas compleja y a la vez trata de tener la menor cantidad de pesos posible, a diferencia de AlexNet, que su poder de clasificación se centra en las capas totalmente conectadas llegando a tener hasta 38 millones de pesos en solo una capa, esto demuestra la superioridad de GoogleNet que logra un número similar de operaciones a AlexNet, pero manteniendo un capas de convolución más complejas (debido que mas capas de convolución logran reconocer patrones de más alto nivel) y logrando reducir el uso de memoria al tener menos pesos.

7.2.2 Entrenamiento de las CNNs: Tres espacios embebidos visuales fueron construidas utilizando cada una de las dos arquitecturas CNN, con el uso de las siguientes estrategias:

- No Train: el uso de un modelo de pre-entrenado con el conjunto de datos Imagenet, la CNN (AlexNet o GoogleNet) se utilizó para obtener un vector embebido para cada imagen en el conjunto de prueba CIFAR-100.

- Scratch: el entrenamiento del modelo CNN desde cero utilizando el conjunto de datos CIFAR-100, y la inicialización de la red con los pesos aleatorios. Una vez más, cada vector embebido se obtiene de cada imagen de el conjunto de prueba CIFAR-100.

- Fine tune: partiendo de los pesos en el modelo IMAGEnet pre entrenada la CNN se entrenó con el conjunto de datos de prueba CIFAR-100. Los mismos vectores embebidos se construyen para el set de prueba de CIFAR-100.

El entrenamiento fue llevado a cabo el supercomputador Guane con el uso del framework Digits, el cual permite el uso de múltiples GPUs para el entrenamiento de las CNNs, debido a que la cantidad de memoria de una sola GPU no era suficiente para cargar toda la red, se conto con un Clúster en Guane que contaba con 8 GPUs Tesla M2075

7.3 ETAPA 3 : ESPACIOS EMBEBIDOS

7.3.1 Extracción de vectores embebidos: Dado un modelo de CNN, construimos su espacio vectorial embebido considerando la salida de varias capas a una profundidad variable. Capas superiores son óptimas para fines de discriminación, pero buscamos poder descriptivo en lugar de discriminativo, con el propósito de identificar grupos abstractos. Dado que las características de

todos los niveles han demostrado que incluyen semántica descriptiva [15], enriqueciendo el espacio embebido, para ambos modelos consideramos todas las capas de convolución, exceptuando el primero, que normalmente produce patrones excesivamente simples que difícilmente pueden contribuir semánticamente al espacio embebido.

Tabla 3. Capas seleccionadas para la construcción del vector embebió por arquitectura.

Arquitectura	Capas seleccionadas para la construcción del vector embebido
AlexNet	conv2 conv3 conv4 conv5
GoogleNet	inception_3a/output inception_3b/output inception_4a/output inception_4b/output inception_4c/output inception_4d/output inception_4e/output inception_5a/output inception_5b/output

Los vectores embebidos para cada arquitectura se construyen mediante la concatenación de las activaciones en cada una de las capas previamente definidas. Como resultado, para AlexNet tenemos vectores de 359,680 valores, y para GoogLeNet tenemos vectores de 1,235,584 valores. Esto corresponde al paso 2 de la figura 11 Flujo de trabajo.

Nuestro objetivo es analizar cómo los espacios embebidos capturan la semántica visual definida por las 20 superclases del conjunto de datos CIFAR-100. Una sola imagen proporciona una representación imperfecta (una perspectiva única, un único contexto, etc), lo que puede dar lugar a resultados no fiables. Para evitar esta limitación construimos vectores embebidos para cada una de las 100 clases de CIFAR-100, mediante la agregación de los vectores embebidos de todas sus imágenes contenidas través de una media aritmética. El resultante **vector embebido de clase** corresponderá a una descripción más coherente de las características visuales de la clase abstracta. Por último, se normaliza cada vector de clase por cada capa. Este último paso se lleva a cabo con el fin de equilibrar la evidencia proporcionada por cada capa, independientemente de su tamaño y la activación de la magnitud de respuesta. Al hacerlo, el vector tendrá en cuenta igualmente las características identificadas en cada nivel de abstracción de la CNN.

7.3.2 Vectores embebidos por clase: Para construir nuestros vectores de clase utilizamos las 10.000 imágenes del conjunto de datos de prueba CIFAR-100, la agregación de las 100 imágenes que pertenecen a cada una de las 100 clases. Como resultado se obtiene, para cada una de las seis combinaciones de

arquitectura de la CNN y la estrategia de extracción, 100 vectores de clase de igual tamaño a los vectores embebidos de las imágenes. Se espera que cada uno de esos vectores de clase pueda contener la semántica visual asociada con su clase según la percepción de la CNN. Estos vectores compondrán nuestro espacio embebido, lo que corresponde al paso 3 en la figura 11 Flujo de trabajo.

7.4 ETAPA 4 : CLUSTERING Y MAPEO A SUPERCLASES

7.4.1 Clustering de vectores de clase por superclase: Agrupamos el vector embebido de las representaciones de las 100 clases en los datos CIFAR-100 en 20 clusters. La mayoría de los algoritmos de Clustering no están diseñados para hacer frente a millones de las funciones y de nuestras inmersiones vectoriales se componen de hasta 1,2 millones de características (para los modelos GoogleNet). Para evitar esta limitación se calcula la matriz de distancia entre los vectores 100 de clase de cada espacio embebido utilizando la distancia coseno. Esto se traduce en una matriz de 100x100 para cada modelo CNN, matriz que captura su correspondiente espacio embebido. El proceso de Clustering se realiza en esta matriz mediante la Spectral Clustering. Estamos utilizando este algoritmo de Clustering porque se supone que el espacio embebido no es lineal y que los grupos se definen por las relaciones entre los ejemplos de localidad. Esto tiene ventajas sobre los algoritmos más clásicos como k-means que asumen el espacio euclidiano embebido lineal y agrupaciones esféricas.

Spectral Clustering puede utilizar diferentes métodos para dividir los datos y la asignación de etiquetas a los ejemplos. Desde nuestros experimentos buscan más bien pequeños grupos (el cluster promedio está compuesta por 5 elementos) usando k-means como su método de partición después de la transformación espectral produce más particiones variables; es decir, diferentes Clusterings utilizando la misma matriz de distancias producen diferentes agrupaciones. Para reducir la variabilidad del proceso de Clustering, se utiliza el método de discretización para la partición, que es menos sensible a la inicialización. Sin embargo, incluso con este método, diferentes ejecuciones ceden ligeramente diferentes agrupaciones. Para obtener una muestra estable para cada espacio embebido calculamos 100 ejecuciones de agrupación independientes en cada matriz de distancia.
A partir de toda la agrupación aplicamos un método de agrupación de consenso [3] a cada espacio vectorial embebido para obtener una partición final. El método de consenso de agrupación utilizada se basa en la matriz de co-asociación de los ejemplos [2]. Para cada clúster una matriz se calcula donde cuenta cuantas veces

aparecen don ejemplos dentro del mismo clúster. Esta matriz se transforma en una matriz de distancia y se aplica la agrupación jerárquica ascendente utilizando criterios de enlace completo para obtener un dendrograma que representa la partición de consenso. El dendrograma resultante se corta para obtener 20 clúster.

7.4.2 Mapeo de clusters a superclases: En este punto, en cada configuración experimental, cada grupo contiene un conjunto de las clases de CIFAR-100 que están cerca uno del otro en el espacio embebido correspondiente. En el paso 5, se utiliza el conjunto de clases que componen cada grupo para encontrar la superclase CIFAR-100 más similar entre los 20 que se muestra en la Figura 10. Si un espacio embebido capturó perfectamente la semántica que definen los 20 superclases, cada grupo contendría aproximadamente las mismas clases de la superclase a la cual corresponden. Utilizamos tres estrategias para dirigir este proceso de asignación: el uso de precisión, recall o la F1-score. Por último, dado un resultado de la asignación, se mide el promedio de similitud entre las agrupaciones y sus superclases mapeadas.

8. RESULTADOS OBTENIDOS

8.1 PROCESO DE CLUSTERING

Los 20 grupos obtenidos para cada uno de los seis modelos se muestran en la Tabla 4 y 5. Las agrupaciones de diferentes modelos se suman manualmente para maximizar la similitud entre los experimentos. Desde una perspectiva general se observa una gran coherencia en el contenido agrupaciones a través de los seis modelos, aunque sólo uno de clúster es universalmente define en todos los espacios de la embebidos (línea 3). Por ejemplo, todos los grupos en la línea 1 contienen los mismos cuatro elementos, con la excepción de serpiente que además aparece en tres de los modelos. La línea 2 muestra cómo la clase chimpancé se agrupa por GoogleNet junto con las clases de humanos (sobre todo en las superclase en CIFAR-100), mientras que en CIFAR-100 chimpancé pertenece a la superclase grandes omnívoros y herbívoros. De hecho, no es razonable argumentar que un chimpancé es visualmente más cerca de las personas que a algunos omnívoros como canguro.

8.2 MAPEO DE CLUSTERES A SUPERCLASES

A pesar de las limitaciones identificadas previamente, tratamos de asignar automáticamente las agrupaciones a superclases CIFAR-100 con el fin de cuantificar la correspondencia. Se utilizó en cada caso tres estrategias: precisión (asignar cada grupo a la superclase que contiene la mayor parte de sus componentes), Recall (asignar a cada superclase en el clúster que contiene la mayor parte de sus componentes) F1 score (asignar cada grupo a la superclase producir la mejor puntuación F1). En todos los casos hemos calculado las tres métricas y les informa en la Tabla 6, que muestra la métrica de promedio en todos los 20 grupos para cada modelo / mapeo / combinación de métricas. La desviación estándar fue siempre entre 0020 y 0025 que señala la estabilidad de los parámetros utilizados y los grupos que se encuentran. La Tabla 6 también muestra la precisión calculada sobre una unidad de prueba de CIFAR-100 cuando se utiliza la CNN como clasificador. Tenga en cuenta que en el caso de configuración NoTrain, la respuesta CNN a una imagen de entrada es una de las clases ImageNet, no CIFAR-100, por lo tanto la precisión no se puede medir.

En términos de arquitectura, GoogleNet constantemente supera AlexNet en todos los escenarios considerados. El poder descriptivo de un modelo de CNN está fuertemente limitada por el tamaño y la complejidad de las características que

captura. GoogleNet, siendo una CNN profunda con más características que define el vector embebio, en consecuencia, se obtienen mejores resultados.

Los resultados de la tabla 6 indican que la estrategia de mapeo no tiene ningún efecto práctico, ya que todas las métricas proporcionan resultados casi idénticos. En casi todos los casos, la estrategia NoTrain produce las métricas de mapeo peores (excepto en AlexNet-NoTrain con precisión), que muestra los beneficios de la realización de un proceso de formación, ya sea desde el principio o a través de finetuning, utilizando el mismo conjunto de datos que desea extraer el vector embebido. Sin embargo, los resultados de la estrategia NoTrain no son tan malo como podría haberse esperado. Esto indica que, en cierto grado, un modelo puede generar vectores embebidos que sirven como descriptor en otros dominios. (Como ejemplo esta el uso de vectores embebidos como descriptores de video para el reconocimiento de acciones)

Por último, GoogleNet-Finetune apenas mejora GoogleNet-Scratch, lo que sugiere que afinar el modelo pre entrenado plantea poca ventaja. Lo contrario sucede con AlexNet, donde finetune parece inducir mejores asignaciones. Por lo tanto, finetuning parece tener poco impacto en los espacios embebidos. Estos estas observaciones deben ser tomados con precaución, ya que, como se mencionó en la sección 7.1 las superclases CIFAR-100 parecen estar definido no sólo a través de criterios visuales.

Tabla 4. Consenso de los 20 cluster por cada uno de los espacios embebidos en AlexNet

	AlexNet Scratch	AlexNet NoTrain	AlexNet Finetune
1	bowl, clock, plate, snake, worm	bowl, clock, plate, worm	bowl, clock, plate, worm
2	baby, boy, girl, man, woman	baby, boy, girl, man, woman	baby, boy, girl, man, woman
3	maple_tree, oak_tree, palm_tree, pine_tree, willow_tree		
4	bicycle, keyboard, motorcycle, telephone	bicycle, motorcycle, tank	bicycle, motorcycle
5	apple, orchid, poppy, rose, tulip	orchid, poppy, rose, tulip	orchid, poppy, rose, tulip
6	orange, pear, sunflower, sweet_pepper	apple, orange, pear, sunflower, sweet_pepper	apple, orange, pear, sweet_pepper
7	bus, pickup_truck, streetcar,tank, tractor, train	bus, can, house, pickup_truck,streetcar, tractor, train	bus, house, pickup_truck, streetcar, tank, tractor, train
8	fox, hamster, leopard,lion, mushroom, tiger	bee, crab, keyboard, leopard, lion,lobster, raccoon, snake, tiger, wolf	crocodile, fox, leopard, lion, wolf,porcupine, raccoon, shrew, tiger
9	bed, couch, table, television, wardrobe	bed, couch, table, television, wardrobe	bed, couch, table, television, wardrobe
10	bottle, can, cup, lamp, lawn_mower, rocket	bottle, rocket, skyscraper, telephone	bottle, can, cup, lamp, rocket
11	camel, cattle, elephant, house	camel, cattle, cup, lamp, mushroom	bear, beaver, camel, cattle, chimpanzee, elephant, kangaroo, otter, seal, skunk
12	dolphin, ray, shark, turtle, whale	dolphin, ray, shark, turtle, whale	dolphin, ray, shark, turtle, whale
13	butterfly, caterpillar, forest, spider	beetle, caterpillar, forest, lizard, road, spider	butterfly, caterpillar, forest, snail, spider
14	chair, cockroach, dinosaur	chair, cockroach, dinosaur	chair, cockroach, telephone
15	bridge, cloud, mountain, plain, road, sea	bridge, castle, cloud, mountain, plain, sea	bridge, cloud, mountain, plain, road, sea
16	aquarium_fish, flatfish, possum, rabbit, raccoon, trout, wolf	flatfish, trout	aquarium_fish, dinosaur, flatfish, mushroom, trout
17	beaver, crocodile, kangaroo, otter,orcupine, seal, snail, squirrel	crocodile, mouse, possum, shrew	hamster, mouse, possum,rabbit, squirrel
18	bear, chimpanzee, skunk	aquarium_fish, fox, hamster, kangaroo, rabbit, snail, squirrel	bee, beetle, lawn_mower, sunflower
19	bee, beetle, crab, lizard, lobster, mouse, shrew	butterfly, lawn_mower, skunk	crab, keyboard, lizard, lobster, snake
20	castle, skyscraper	bear, beaver, chimpanzee, elephant, otter, porcupine, seal	castle, skyscraper

Tabla 5. Consenso de los 20 cluster por cada uno de los espacios embebidos en GoogleNet

	GoogLeNet Scratch	GoogLeNet NoTrain	GoogLeNet Finetune
1	bowl, clock, plate, snake, worm	bowl, clock, plate, worm	bowl, clock, plate, snake, worm
2	baby, boy, chimpanzee, girl, man, woman	baby, boy, chimpanzee, girl, man, woman	baby, boy, flatfish, girl, man, woman
3	maple_tree, oak_tree, palm_tree, pine_tree, willow_tree		
4	bicycle, crab, lobster, motorcycle	bicycle, lawn_mower, motorcycle	bicycle, motorcycle
5	aquarium_fish, mushroom, orchid, poppy, rose, tulip	aquarium_fish, orchid, poppy, rose, tulip	orchid, poppy, rose, tulip
6	apple, orange, pear, sweet_pepper	apple, flatfish, orange, pear, sweet_pepper	apple, orange, pear, sweet_pepper
7	bus, pickup_truck, streetcar, tractor, train	bus, pickup_truck, tank, streetcar, tractor, train	bus, pickup_truck, tank, streetcar, tractor, train
8	fox, leopard, lion, raccoon, tiger, wolf	fox, kangaroo, leopard, lion,raccoon, tiger, wolf	crab, fox, leopard,lion, lobster, tiger
9	bed, chair, couch, table, television, wardrobe	bed, couch, telephone, television, wardrobe	bed, chair, couch, dinosaur, table, trout
10	bottle, can, cup, lamp, lawn_mower, rocket	bottle, can, rocket, skyscraper	bottle, can, cup, lamp, lawn_mower, rocket
11	bear, camel, cattle, dinosaur, elephant, kangaroo	bear, bridge, camel, castle, cattle, elephant, house	bear, camel, cattle, elephant,kangaroo, otter, seal
12	dolphin, ray, shark, turtle, whale	dolphin, ray, seal, shark, turtle, whale	aquarium_fish, dolphin, ray, shark, turtle, whale
13	caterpillar, crocodile, forest, lizard, snail, spider	beetle, butterfly, caterpillar, forest, snail, spider	bee, beetle, butterfly, cockroach, spider, sunflower
14	bee, beetle, butterfly, cockroach, sunflower	chair, cockroach, dinosaur, trout	caterpillar, crocodile, forest, lizard, mushroom, snail
15	bridge, castle, house, mountain, skyscraper	hamster, mouse, porcupine, possum, rabbit, squirrel	bridge, castle, house, skyscraper
16	flatfish, tank, trout	cup, lamp, mushroom, table	chimpanzee, raccoon,skunk, wolf
17	beaver, otter, seal	bee, sunflower	television, wardrobe
18	keyboard, skunk, telephone	beaver, crocodile, lizard, otter, shrew, skunk, snake	keyboard, telephone
19	hamster, mouse, porcupine, possum, rabbit, shrew, squirrel	crab, keyboard, lobster	beaver, hamster, mouse, porcupine, possum, rabbit, shrew, squirrel
20	cloud, plain, road, sea	cloud, mountain, plain, road, sea	cloud, mountain, plain, road, sea

Figura 13 : Muestre de imágenes por clases con divergencia el el clustering de embebidos de clase y superclase CIFAR.

En la parte superior, los embebidos de clase tienden a asociar con girasoles naranjas, mientras que en CIFAR aparecen junto con las rosas en la superclase flores. En pocas palabras, lo mismo para los cohetes que se agrupan con botellas, pero no con los tranvías como en los vehículos de la superclase de CIFAR.

8.3 SIMILITUD SUPERCLASES

Para lograr una mejor comprensión que mide la similitud para cada clase CIFAR a través de los vectores embebidos dentro de cada arquitectura CNN. Esto es, por GoogleNet, se mide la similitud entre los embebidos Scratch y NoTrain, Scratch y Finetune y entre Finetune y NoTrain. Para cada superclase y par de embebidos, se calculó la similitud coseno entre el vector embebido en cada representación y la media de todas las imágenes de la superclase. Esto se muestra en la Tabla 7

Estos resultados muestran cómo los vectores embebidos obtenidos con configuraciones Finetune y NoTrain son los más similares (como se esperaba como modelos afinarse se obtienen a partir NoTrain), mientras que los embebidos a partir de la estrategia Scratch difieren sistemáticamente del resto de

todos los superclases. Configuraciones de Scratch son un poco más cercanos de los modelos Finetune, causado por que ambas configuraciones comparten el ser entrenadas con CIFAR-100.

Si tenemos en cuenta el rendimiento evaluado en la Tabla 6, estas correlaciones dar una idea interesante. En términos de rendimiento los modelos de GoogLeNet Scratch y Finetune se comportan de manera similar, y superan a NoTrain. Sin embargo en la Tabla 7: similitud modelos Finetune y NoTrain son claramente más cercanos en comparación con el modelo Scratch. Estos comportamientos indican que, mientras que los valores del espacio embebido se pueden modificar de manera significativa por el conjunto de datos de formación inicial (por ejemplo, Scratch vs FineTune), el espacio capturado es eventualmente consistente en términos de la semántica.

Tabla 6 : Métricas en el mapeo de clusters a las superclases de CIFAR100

	Mapping by F1			Mapping by Precision			Mapping by Recall			
Model	F1	Prec	Recall	F1	Prec	Recall	F1	Prec	Recall	Discrimination accuracy
AlexNet Scratch	.544	.580	.548	.545	.581	.550	.540	.582	.549	.566
AlexNet NoTrain	.541	.621	.543	.541	.623	.544	.540	.620	.543	N/A
AlexNet Finetune	.608	.661	.606	.609	.663	.607	.608	.662	.606	.721
GoogLeNet Scratch	.642	.682	.643	.643	.684	.644	.643	.683	.644	.653
GoogLeNet NoTrain	.574	.635	.575	.575	.635	.579	.575	.635	.578	N/A
GoogLeNet Finetune	.651	.698	.651	.653	.699	.653	.652	.699	.652	.770

Tabla 7 : Similitud coseno por pares entre las inmersiones. Un valor de 0 es la más cercana, 1 es ortogonal.

	GoogLeNet			AlexNet		
SuperClass	Scratch vs. NoTrain	Scratch vs. Finetune	Finetune vs. NoTrain	Scratch vs. NoTrain	Scratch vs. Finetune	Finetune vs. NoTrain
Aquatic Mammals	.69	.59	.42	.68	.60	.39
Fish	.69	.59	.39	.68	.61	.39
Flowers	.69	.64	.40	.68	.63	.42
Food Containers	.66	.60	.40	.69	.65	.35
Fruit and Vegetables	.69	.64	.38	.69	.64	.37
Household electrical devices	.65	.57	.42	.67	.61	.37
Household furniture	.65	.57	.42	.67	.61	.39
Insects	.67	.59	.45	.67	.60	.40
Large Carnivores	.67	.58	.49	.68	.58	.45
Large man-made outdoor things	.69	.62	.42	.68	.64	.43
Large natural outdoor scenes	.75	.70	.35	.72	.68	.37
Large omnivores	.66	.55	.47	.66	.57	.43
Medium sized mammals	.66	.56	.48	.67	.57	.44
Non insect invertebrates	.64	.55	.45	.65	.56	.39
People	.65	.54	.44	.67	.59	.39
Reptiles	.65	.54	.46	.65	.54	.40
Small mammals	.66	.55	.46	.66	.55	.43
Trees	.70	.65	.46	.71	.68	.47
Vehicles 1	.66	.56	.47	.68	.63	.45
Vehicles 2	.68	.59	.45	.68	.62	.42

8.4 PRODUCTO

Como resultado del proyecto de investigación se obtuvo un producto paper el cual fue presentado en la 19th International Conference of the Catalan Association for Artificial Intelligence [15].

9. CONCLUSIONES

- Con la construcción de embebidos por clase de imagen y la agrupación de ellos, se muestran como hay conceptos abstractos significativos de manera no supervisada, aunque parcialmente en desacuerdo con la Ground truth que consideramos (CIFAR-100 superclases). Esto revela discrepancias en los criterios conceptuales y visuales en la construcción de los clusters/abstracciones.

- Los resultados mostraron el impacto que tienen el tipo de arquitectura de CNN a usar, GoogleNet al tener más capas de convolución que aprenden características más complejas logra obtener semántica visual más rica.

- Nuestras pruebas sugieren que el aprendizaje de transferencia es en general ventajoso construir imagen embebidas semánticamente útiles, pero no tanto como se podría esperar.

- El entrenamiento de un CNN desde cero podría producir resultados similares, y la formación con un conjunto de datos diferente todavía logra un rendimiento coherente. Esto podría ser beneficioso en los casos en que no hay CNNs entrenadas previamente o que pueden no ser conveniente, como en conjuntos de datos especializados (médicos, bio, etc.) cuya naturaleza es muy diferente de los grandes conjuntos de datos genéricos (como ImageNet) utilizados para crear modelos pre-entrenados.

- La dependencia de los modelos pre-formados CNNs se puede reducir, lo que facilita el diseño de CNNs que busquen generar vectores embebidos descriptivos en lugar de discriminación en imágenes.

REFERENCIAS BIBLIOGRÁFICAS

[1] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531.

[2] Fred, A. L. N. and Jain, A. K. (2005). Combining Multiple Clusterings Using Evidence Accumulation IEEE Trans. Pattern Anal. Mach. Intell., 27(6):835–850.

[3] Ghaemi, R., Sulaiman, M. N., Ibrahim, H., and Mustapha, N. (2009). A survey: clustering ensembles techniques. World Academy of Science, Engineering and Technology, 50:636–645.

[4] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385.

[5] Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.

[6] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105.

[7] Näppi, J. J., Hironaka, T., Regge, D., and Yoshida, H. (2016). Deep transfer learning of virtual endoluminal views for the detection of polyps in ct colonography. In SPIE Medical Imaging, pages 97852B–97852B. International Society for Optics and Photonics.

[8] Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 806–813.

[9] Reyes, A. K., Caicedo, J. C., and Camargo, J. E. (2015). Fine-tuning deep convolutional networks for plant recognition. In Working notes of CLEF 2015 conference.

[10] Rueda-Plata, D., Ramos-Pollán, R., and González, F. A. (2015). Supervised greedy layer-wise training for deep convolutional networks with small datasets. In Computational Collective Intelligence, pages 275–284. Springer

[11] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9.

[12] Xu, M., Cheng, W., Zhao, Q., Ma, L., and Xu, F. (2015). Facial expression recognition based on transfer learning from deep convolutional networks. In Natural Computation (ICNC), 2015 11th International Conference on, pages 702–708. IEEE.

[13] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., and Lipson, H. (2015). Understanding neural networks through deep visualization. arXiv preprint arXiv:1506.06579.

[14] Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer vision–ECCV 2014, pages 818–833. Springer.

[15] Garcia-Gasulla, D., Moreno, J., Ramos-Pollàn, R., Casadiegos, R., Bejar, J., Cortés, U., Ayguadé, E., and Labarta, J. (2016). On the representativeness of convolutional neural networks layers. In Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence.

BIBLIOGRÁFIA

DEEP LEARNING DOCUMENTATION [en línea] <http://www.neuralnetworksanddeeplearning.com>

MACHINE LEARNING DOCUMENTATION [en línea]<http://cs229.stanford.edu/notes/cs229-notes1.pdf

DEEP LEARNING DOCUMENTATION [en línea]<http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/>

DEEP LEARNING DOCUMENTATION [en línea]<http://techtalks.tv/talks/stochastic-pooling-for-regularization-of-deep-convolutional-neural-networks/58106/>

DIGITS DOCUMENTATION [en línea]<https://developer.nvidia.com/digits>

CAFFE DOCUMENTATION [en línea]<http://caffe.berkeleyvision.org/>

CIFAR-100 DOCUMENTATION [en línea]<https://www.cs.toronto.edu/~kriz/cifar.html>

IMAGENET DOCUMENTATION [en línea]<http://image-net.org/>

DEEP LEARNING DOCUMENTATION [en línea]<http://www.deeplearningbook.org/contents/ml.html>

GOOGLENET DOCUMENTATION [en línea]<https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf>

MACHINE LEARNING DOCUMENTATION [en línea]<http://ebooks.iospress.nl/publication/45307>