UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

# TESINA DE LICENCIATURA


**TÍTULO:** Clasificación de gestos utilizando Deep Learning en datasets con pocos datos etiquetados.

**AUTORES:** Rios Gastón Gustavo

**DIRECTOR:** Hasperué Waldo y Ronchetti Franco.

**CODIRECTOR:**

**ASESOR PROFESIONAL:**

**CARRERA:** Licenciatura en Informática.



## Resumen

*El aumento en la cantidad de conjuntos de datos disponibles no ha alcanzado todas las problemáticas existentes, teniendo múltiples áreas donde los conjuntos de datos disponibles son pequeños para la aplicación efectiva de modelos de aprendizaje profundo. En esta tesis se exploraron diversos métodos para lograr alcanzar la mejor precisión posible utilizando la menor cantidad de datos. Llegando finalmente a lograr una precisión en la clasificación de señas estáticas del 99.26% en el conjunto de datos LSA16 y 94% con el conjunto de datos RWTH-PHOENIX-Wheater.*



## Palabras Clave

*Tesis de grado, Aprendizaje profundo, Reconocimiento de gestos, Pocos datos etiquetados, Few-shot learning, Data augmentation.*


## Conclusiones

*Los actuales modelos estado del arte como DenseNet han adquirido una capacidad superior a la de anteriores modelos para el tratamiento de pequeños conjuntos de datos permitiendo obtener mejores resultados sin necesidad de aplicar una multitud de técnicas especificas para el problema. Aún así, ha quedado demostrado que la aplicación de técnicas especificas, como redes prototípicas, puede aumentar aun más la exactitud de los modelos obtenidos, principalmente en situaciones más extremas donde los datos etiquetados disponibles son de unos pocos por clase.*

## Trabajos Realizados

*En los experimentos realizados analizamos y comparamos modelos estado del arte y específicos para problemas de conjuntos de datos pequeños, adicionalmente examinamos los efectos de la utilización de data augmentation. En particular realizamos experimentos con Wide-DenseNet, una arquitectura convolucional estado del arte, y Prototypical Networks, un modelo estado del arte utilizado en few-shot learning. En ambos casos cuantificamos el impacto de la aplicación de data augmentation sobre la exactitud de los modelos.*

## Trabajos Futuros

*En el futuro, el desarrollo de nuevos modelos generativos podría ayudar a resolver la problemática de los pocos datos etiquetados al permitir la generación de nuevos datos de entrenamiento a partir de los datos disponibles. Futuros avances en el área de meta-learning podría permitir el entrenamiento de modelos capaces de generalizar a partir del aprendizaje en múltiples tareas, lo que reduciría la cantidad de datos necesarios para cada tarea en particular.*


**Fecha de la presentación:** Febrero 2020

UNIVERSIDAD
NACIONAL
DE LA PLATA

Rios Gastón Gustavo

# Clasificación de gestos utilizando Deep Learning en datasets con pocos datos etiquetados.

Facultad de Informatica - Universidad Nacional de La Plata (UNLP)
Tesina de grado
Febrero 2020

# Resumen

Rios Gastón Gustavo: Clasificación de gestos utilizando Deep Learning en datasets con pocos datos etiquetados.
Tesina de grado
Directores: Ronchetti Franco y Hasperué Waldo
Universidad Nacional de La Plata
Licenciatura en Informatica
Febrero 2020

---

En los últimos años el aprendizaje profundo ha demostrado ser un método sumamente efectivo a la hora de realizar clasificación de imágenes. Esta efectividad es asociada en parte al aumento de poder de procesamiento, al desarrollo de nuevos algoritmos y al incremento en el tamaño y cantidad de conjuntos de datos disponibles. Pero este aumento en la cantidad de conjuntos de datos disponibles no ha alcanzado todas las problemáticas existentes, teniendo múltiples áreas donde los conjuntos de datos disponibles son pequeños para la aplicación efectiva de modelos de aprendizaje profundo o cuyos datos poseen información poco útil al no ser lo suficientemente representativa del problema o poseer ruido.

Esta limitación en la cantidad de datos etiquetados es una problemática actual existente en la clasificación de señas de la lengua de señas. En esta tesis se exploraron diversos métodos para lograr alcanzar la mejor precisión posible utilizando la menor cantidad de datos. Llegando finalmente a lograr una precisión en la clasificación de señas estáticas del 99.26 % en el conjunto de datos LSA16 y 94 % con el conjunto de datos RWTH-PHOENIX-Wheater.

**Palabras clave:** Tesis de grado, Aprendizaje profundo, Reconocimiento de gestos, Pocos datos etiquetados, Few-shot learning, Data augmentation.

# Índice general

# 1 Introducción

Con el paso de los años se ha visto un gran aumento en las capacidades del aprendizaje de máquina para la resolución de problemas complicados, uno de estos en los que se ha visto un gran avance es la visión por computadora [30]. Este aumento en las capacidades de los algoritmos de aprendizaje de máquina ha sido así gracias al gran avance en las técnicas utilizadas, el hardware disponible y el aumento en la disponibilidad de bases de datos de mayor tamaño. Actualmente la técnica más utilizada en la discriminación de clases en imágenes son las redes convolucionales, un tipo de red neuronal artificial que utiliza convoluciones para la extracción de información espacial, las cuales están compuestas por capas de neuronas artificiales que son entrenadas sobre una base de datos etiquetada dada.

La mejora en la performance del hardware disponible ha permitido aumentar el tamaño y la cantidad de las capas de las redes convolucionales. Este aumento en tamaño de las redes convolucionales permitió mejorar aún más la calidad de la clasificación en grandes conjuntos de datos ya que al agrandar las redes estas pueden obtener un entendimiento más profundo de los patrones que forman cada clase. No obstante, este aumento en tamaño también requiere un aumento en la cantidad de datos de entrenamiento debido a que de no ser así la red convolucional tenderá a sobreajustar, fallando en clasificar correctamente imágenes que no se encuentren entre los datos de entrenamiento [12].

Esta ausencia de datos de entrenamientos suficientes para entrenar modelos de redes neuronales profundas que obtengan una buena generalización de los datos se encuentra presente en la clasificación de señas estáticas de la lengua de señas. Con la cantidad de datos disponibles en cada conjunto de datos siendo de apenas miles de imágenes, lo cual no se compara a las millones de imágenes disponibles en otras áreas. Esto genera problemas al intentar utilizar modelos de aprendizaje profundo en aplicaciones reales debido a la baja generalización y fiabilidad que se obtendría en estos casos. Debido a esto es necesario la implementación de técnicas específicas para el entrenamiento de modelos en problemas en los que se dispone de una baja cantidad de datos etiquetados.

## 1.1. Motivación

Aunque la cantidad de grandes bases de datos disponibles ha ido en aumento, aún existen muchos problemas para los cuales no existen bases de datos lo suficientemente grandes como para entrenar una red neuronal de gran tamaño que permita mejorar aún más la precisión en la clasificación de los datos [91]. Uno de estos problemas

es la clasificación de señas de manos para lengua de señas, la cual cuenta con bases de datos como LSA16 [93] y RWTH-PHOENIX [26], entre otras [24][85], pero estas no poseen las suficientes imágenes etiquetadas como para poder entrenar una red neuronal profunda que obtenga una generalización correcta de los datos. Esto ocurre debido a que cada país posee una o más lenguas de señas, y cada lengua de señas está compuesta por una gran cantidad de señas, que son utilizadas en mayor o menor medida [20].

## 1.2. Objetivos

El tema de la clasificación de señas de la lengua de señas ha sido abordado previamente [56][68][95][17][84][5], pero nunca desde el punto de vista del análisis del tamaño de los conjuntos de datos disponibles y las posibles técnicas que permitan tratar con los problemas descubiertos de este análisis. En esta tesis se clasificaron las distintas señas de diversas lenguas de señas abordando el problema de los conjuntos de datos con pocos datos etiquetados. Esto facilitará la creación de sistemas traductores de lenguas de señas que permitan una comunicación más natural con personas que practiquen esta lengua. Por otro lado se obtendrá un análisis en profundidad de las dificultades a ser afrontadas al poseer conjuntos de datos con pocos datos etiquetados y los pasos a seguir para mitigar sus efectos en el resultado final.

## 1.3. Organización de la tesis

La organización de este documento se encuentra presentada de la siguiente forma:

En el Capítulo 2 se introduce el marco teórico sobre la visión por computadora con el que se trabajara en el resto de la tesis. En este capítulo se realiza una explicación de los conceptos que abarca el aprendizaje profundo y la visión por computadora. Además, se explicará el funcionamiento de las redes neuronales y en más detalle el de las redes neuronales utilizadas en la visión por computadora, las redes convolucionales. Se dará una explicación de las diferentes arquitecturas que han surgido junto a las ventajas de cada una.

En el Capítulo 3 se presenta una explicación de la lengua de señas, los conjuntos de datos disponibles para esta y el análisis de estos conjuntos de datos. Se indican los problemas de la lengua de señas para la creación de conjuntos de datos y la evolución de los conjuntos de datos disponibles para esta. Además se explica el problema de los conjuntos de datos con pocos datos etiquetados en aprendizaje profundo, un problema presente en la clasificación de señas estáticas.

En el Capítulo 4 se introducen las técnicas comúnmente utilizadas para tratar con el problema de poseer pocos datos etiquetados. Se realiza una explicación detallada de cada técnica y su funcionamiento junto a sus ventajas y desventajas. Este capítulo

servirá como una referencia comparativa entre las diferentes técnicas para tratar con pocos datos realizando comparaciones entre estas y sus casos de aplicación.

En el Capítulo 5 se desarrolla la experimentación aplicando algunas técnicas previamente definidas para alcanzar una mayor performance en la clasificación de señas estáticas de la lengua de señas. Se entrenaron varios modelos y se realizó un estudio comparativo de la aplicación cada modelo sobre distintos conjuntos de datos. Finalmente, se comparó el resultado obtenido al resultado sin la aplicación de las técnicas para trabajar con pocos datos etiquetados.

En el Capítulo 6 se detallan las conclusiones generales de la tesis realizando una compilación de lo aprendido a lo largo de esta. Esto incluye una reflexión del estado actual del entrenamiento de modelos con pocos datos etiquetados y del problema de la clasificación de señas estáticas. También se realiza una conclusión de los resultados obtenidos en el capítulo 6, cómo estos superan previos modelos estado del arte y en qué forma sus resultados podrían mejorarse aún más. Finalmente se detallan los posibles avances futuros en el entrenamiento de modelos con pocos datos etiquetados y en la clasificación de señas.

En el Capítulo 7 se realiza un listado de la documentación utilizada a lo largo de esta tesis.

# 2 Visión por computadora

El gran aumento en el número de cámaras, por ejemplo de celulares inteligentes, y el aumento en la utilización de internet y redes sociales como Instagram, Facebook o Youtube ha llevado a que la cantidad de imágenes disponibles en Internet incremente en gran medida [130]. Con los usuarios de Instagram subiendo más de 100 millones de imágenes por día en total y los usuarios de Youtube subiendo cientos de horas de vídeos por minuto es claro ver la cantidad de información disponible para analizar y explotar. Pero para obtener información valiosa de estas imágenes y vídeos primero es necesario que la computadora pueda extraerla y analizarla. La visión por computadora es un campo que integra métodos que permiten adquirir, procesar y analizar imágenes con el fin de obtener información de estas. Con la visión por computadora se busca lograr que las computadoras adquieran la visión y el entendimiento de imágenes humano. Este campo ha sido ampliamente estudiado, llegando al desarrollo de múltiples técnicas que permiten afrontar la visión por computadora alcanzando una muy alta precisión y performance [36][137][41], incluso superando la precisión humana en algunos casos [111] como se ve en la figura 2.1. Por ejemplo, en ImageNet, una base de datos de gran tamaño con más de 14 millones de imágenes, el error humano calculado en 2014 es de 5.1 % en exactitud top-5 mientras que actualmente el error alcanzado en 2019 en el articulo *Self-training with Noisy Student improves ImageNet classification* [128] en exactitud top-5 es de menos del 2 %.

## 2.1. Clasificación de imágenes

Uno de los subdominios que abarca la visión por computadora es la clasificación de imágenes. La clasificación de imágenes busca obtener la mejor clasificación posible de las imágenes que se introduzcan en el modelo. Un ejemplo de un problema de este tipo es la clasificación de imágenes de señas de manos en diferentes tipos de señas.

Para realizar esta clasificación es necesario un modelo que aprenda las características de cada clase y utilice estos conocimientos para realizar la clasificación de forma automática. Las redes neuronales convolucionales son el modelo más utilizado en este campo ya que poseen un relativamente bajo costo computacional además de la capacidad de obtener una buena extracción de características de las imágenes de entrada permitiendo así una salida más precisa [52].

Además del modelo clasificador también suele realizarse un procesamiento previo a las imágenes con el fin de adquirir más información de estas para aumentar la precisión, reducir el sobreajuste y acelerar el entrenamiento de los modelos. Un


***Figura 2.1*** *Reducción de error de clasificación top-5 a lo largo de los años en ImageNet hasta superar al error humano.*

ejemplo de este tipo de procesamiento es la normalización. Este es el proceso que busca acomodar los datos en una escala común, lo que facilita la convergencia del modelo. La normalización más comúnmente usada en redes neuronales es la llamada normalización Z 2.1.

$$x' = \frac{x - \mu}{\sigma} \tag{2.1}$$

Siendo $x$ es el elemento a normalizar, $\mu$ es la media y $\sigma$ la desviación estándar del conjunto de datos. Otros tipos de preprocesamiento son aquellos donde se introducen cambios a las imágenes para añadir datos adicionales al conjunto de datos, esto último es conocido como *data augmentation*.

## 2.2. Aprendizaje profundo

El aprendizaje automático es una disciplina científica que estudia algoritmos y modelos estadísticos cuyo objetivo es realizar tareas sin instrucciones explícitas mediante el aprendizaje de patrones y la inferencia buscando minimizar una función de error dada. Es decir, el aprendizaje automático busca enseñar a la computadora a realizar predicciones de datos no vistos basado en los patrones obtenidos de los datos estudiados sin intervención humana. Esto permite enfrentar problemas cuya solución no heurística es difícil de alcanzar o llevaría un tiempo demasiado alto.

Existe una gran variedad de modelos utilizados en tareas de aprendizaje automático, cada modelo posee sus ventajas y desventajas siendo utilizados para solucionar diversos problemas.

Las redes neuronales son un modelo muy utilizado para una gran cantidad de

tareas debido a su flexibilidad, ya que estas optimizan los pesos de sus neuronas aprendiendo patrones y características de los problemas y, de esta forma, minimizando el error a la hora de obtener la respuesta. Las redes neuronales más utilizadas actualmente en la clasificación de imágenes son las redes neuronales profundas (DNN - *Deep Neural Networks*). Este tipo de red neuronal artificial está compuesta por múltiples capas de neuronas obteniendo múltiples niveles de abstracción. Estos modelos permiten a la computadora aprender de la experiencia y entender el mundo en términos de una jerarquía de conceptos, donde cada concepto se define en cuanto a su relación con conceptos más simples. Aprendiendo por medio de la experiencia evita la necesidad de que los operadores humanos deban especificar los conocimientos que la computadora requiere. La jerarquía de conceptos le permite a la computadora aprender conceptos más complicados construyendolos sobre conceptos más simples [79].

Como fue mencionado anteriormente la fácil obtención de cámaras portátiles y su mejora en la calidad de imagen obtenible ha generado un gran aumento en la cantidad, variedad y calidad de imágenes disponibles.

Existen cada vez más conjuntos de datos públicos donde es posible encontrar una gran cantidad de imágenes (cientos de miles) etiquetadas. Ejemplos de estos conjuntos de datos son CIFAR-10 [58] (60.000 $32 \times 32$ imágenes a color divididas en 10 clases), ImageNet [21] (1000 imágenes en promedio por cada concepto significativo de WordNet), Googles Open Images [60] (aproximadamente 9 millones de imágenes etiquetadas), Labelled Faces in the Wild [43] (más de 13000 imágenes etiquetadas obtenidas de la web) entre muchas otras disponibles para diversos dominios de estudio.

Esto facilita el entrenamiento de DNN más profundas, que puedan extraer todas las características disponibles de las imágenes sin sobreajuste. También permite adentrarse en diversos dominios sin la necesidad inicial de desarrollar nuestros propios conjuntos de datos.

Es posible representar a las redes neuronales como una multiplicación de matrices o tensores. Esto facilita su entrenamiento al permitir su procesamiento en hardware específico que maximice el paralelismo en la ejecución de estas multiplicaciones aumentando en gran medida la velocidad de procesamiento [82]. Este aumento en la velocidad vuelve más feasible el entrenamiento de redes más anchas y profundas en más cantidad de datos por más epocas, ya que en caso de no poseer hardware específico el tiempo de entrenamiento se aumentaría en gran medida, volviendo imposible este entrenamiento. Esta posibilidad de agrandar las redes permite afrontar problemas más complicados y obtener mayor performance. A lo largo de los últimos años las redes neuronales han crecido en gran manera en profundidad, se ha pasado de unas pocas capas [117] a más de 1000 [42]. También se han desarrollado mode-

los ensamblados [120], que utilizan predicciones de múltiples modelos para obtener mejor precisión. La utilización de hardware específico fue una de las razones por las que este crecimiento fue posible reduciendo los tiempos de entrenamiento y el gasto energético en gran medida.

Actualmente, debido a la gran utilización de redes neuronales profundas, se ha seguido con el desarrollo a gran escala de nuevas tecnologías de hardware para mejorar la performance y reducir el consumo energético [76]. Entre estos nuevos desarrollos se encuentran las TPU de Google [51], las cuales han sido desarrolladas específicamente para la ejecución de algoritmos de aprendizaje profundo otorgando de esta forma una muy buena combinación de velocidad y poco consumo energético, el desarrollo de computadoras cuánticas [4], las que aún se encuentran en sus primeros pasos pero que podrían traer un gran aumento en la velocidad de ejecución de algunos algoritmos, los núcleos tensoriales de las GPU NVidia [74], las FPGA de Microsoft [13] , las cuales son de perfil más genérico que las TPU de Google pero que son más fáciles de adaptar ante posibles cambios de tecnología, entre otros nuevos desarrollos de hardware que en un futuro podrían ayudar al desarrollo de nuevas tecnologías de aprendizaje automático y a la resolución de problemas más grandes y complejos. También han sido desarrollados nuevos modelos y algoritmos a bajo nivel más eficientes que buscan minimizar este costo y adaptarse a los bajos recursos disponibles en el amplio mercado de las plataformas móviles [45].

### 2.2.1. Redes neuronales

Las redes neuronales artificiales son modelos inspirados en las neuronas biológicas y sus conexiones [32]. Las redes neuronales artificiales están compuestas por neuronas artificiales 2.2. Las neuronas artificiales de una red se comportan como funciones, las cuales reciben $m$ entradas con valores $x_1$ a $x_m$ y sus respectivos pesos $w_1$ a $w_m$. Además, se añadirá el bias para dar deslizamiento al resultado de la función con el objetivo de que esta se adapte mejor a los datos de entrenamiento.

Luego se le aplicará la función de activación al resultado del proceso, que permitirá a las siguientes capas conocer el nivel de activación que poseen las neuronas 2.2. Existen una gran variedad de funciones de activación utilizables, cada con su uso específico [81].

Una función de activación muy usada es la función sigmoide ploteada en la figura 2.3, la cual acota el resultado en el rango (0,1) y devuelve un valor en este rango de forma análoga. Esto permite a la capa de salida de una red neuronal tener un valor de confianza para la predicción de un objeto.

Otra función de activación comúnmente utilizada es la función ReLU ploteada en la figura 2.3. Esta función funciona como la función identidad para aquellos valores de entrada que sean positivos y devuelve 0 para todos los negativos. La función

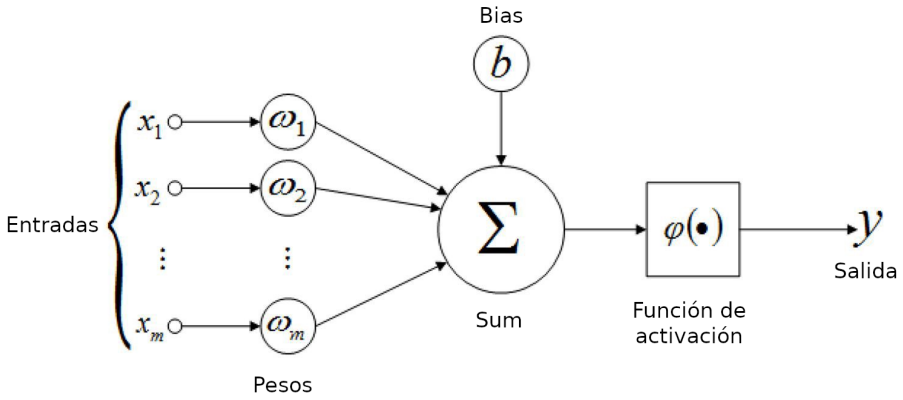



***Figura 2.2*** *Neurona artificial con m entradas.*

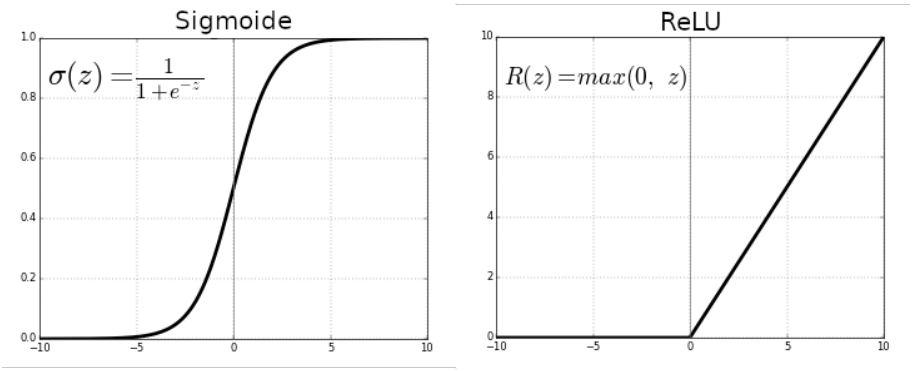



***Figura 2.3*** *Gráficos de las funciones de activación sigmoide y ReLU.*

ReLU posee numerosas ventajas sobre otras funciones, entre ellas esta función posee un bajo costo computacional lo que acelera el tiempo de entrenamiento e inferencia de las redes neuronales que la utilicen. Además permite una convergencia más rápida ya que al aumentar el tamaño de la entrada el valor de salida no se saturara ni se estancara. Esta característica también permite prevenir el problema del gradiente desvaneciente que surge con otras funciones de activación como la sigmoide. El gradiente desvaneciente es el problema que surge en aquellas redes neuronales de múltiples capas donde los pesos y bias de las primeras capas no se actualizarán efectivamente debido a la reducción de el gradiente de la función de costo en *backpropagation*. Otra característica es que esta función es *sparsely activated*, esto significa que la función solo se activará para algunos valores de entrada. De esta forma se simplifica el procesamiento de la red al haber menos neuronas activadas y se hace que las neuronas sólo procesen información útil.

$$y = \varphi(\sum_{j=1}^{m} w_j x_j + b) \tag{2.2}$$

Normalmente el bias se agregará a la entrada $x_0$ la cual típicamente recibe el valor 1 para facilitar la implementación de la neurona artificial y mejorar el tiempo de ejecución. De esta forma, si $X$ es el vector con las entradas $x_0$ a $x_m$ y $W$ el vector con los pesos $w_0$ a $w_m$ se obtiene una ecuación simplificada que permite procesar los pesos y bias sobre las entradas como una multiplicación entre un vector de pesos y un vector de entradas transpuesto 2.3.

$$y = \varphi(WX^T) \tag{2.3}$$

Las neuronas artificiales permiten realizar regresión lineal sobre un conjunto de datos permitiendo la separación de los datos linealmente en dos clases. Esto es útil en varios problemas, pero en problemas más complejos, como la clasificación de imágenes, es posible agrupar neuronas en múltiples capas. Cada capa estará compuesta de una o más neuronas las cuales normalmente se encuentran conectadas a neuronas de capas posteriores funcionando como entrada de estas [79].

De esta forma en una capa de $n$ neuronas se tendrán $n$ vectores de pesos diferentes y $n$ salidas. Por lo tanto, es posible acomodar los vectores de pesos de cada capa $i$ en matrices $W^i$ de $nxm$ siendo $m$ la cantidad de salidas de la capa anterior 2.4.

$$W^i = [[W_{1,1}, W_{1,2}, ..., W_{1,m}], ..., [W_{n,1}, W_{n,2}, ..., W_{n,m}]] \tag{2.4}$$

Esta matriz $W^i$ podrá ser multiplicada por su vector de entrada $X^i$ el cual será la entrada a la red o la salida de una capa anterior obteniendo los resultados de forma eficiente mediante una simple multiplicación de matrices en un vector $Z^i$ de tamaño $n$ 2.5.

$$Z^i = W^i {X^i}^T \tag{2.5}$$

Finalmente se aplicará la función de activación $\varphi^i$ a cada elemento del vector $Z^i$ para obtener la salida de la capa $s^i$. Siendo esta salida utilizada como entrada para la siguiente capa y, en la última capa, como la salida de la red neuronal 2.6.

$$s^i = \varphi(Z^i) \tag{2.6}$$

### 2.2.2. Entrenamiento

Para poder utilizar una red neuronal en inferencia y obtener predicciones primero es necesario entrenarla. Este entrenamiento consiste en utilizar un conjunto de datos de entrenamiento normalmente separados de los datos de prueba que permitirán a la red ajustarse a las características propias de los datos.

Para esto, la red neuronal necesitará de una función de costo. La función de

costo es una función que permitirá comparar los resultados predichos por la red neuronal contra los resultados esperados dictados por la entrada. Esta función tiene la condición de que su salida debe ser siempre positiva para indicar la diferencia que existe entre la predicción y el valor real, y su salida debe aumentar a medida que esta diferencia se incrementa. Existen múltiples funciones de costo, entre estas el error cuadrático medio y la muy utilizada función *cross-entropy*.

El error cuadrático medio 2.7 es una función simple cuyo resultado siempre tendrá un valor positivo debido a que el cuadrado de un número siempre es positivo y aumenta según la diferencia entre sus entradas. Siendo sus entradas en el caso de la función de costo la salida de la red neuronal y la salida esperada.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y})^2 \tag{2.7}$$

La función de costo será utilizada por el método de *backpropagation* para cambiar los valores de los pesos y bias de la red. Para esto se calcula el gradiente de la función de costo con respecto a estos parámetros. El gradiente indica cuánto deben cambiar los valores de los pesos y bias para que la salida de la red se acerque al valor esperado. Este gradiente se calcula mediante la derivada parcial de la función de costo en $W^i_{j,k}$ siendo $k$ el peso asociado a la entrada $k$ de la neurona $j$ en la capa $i$. Esta derivada parcial muestra en cuanto cambia el valor de la función de costo a medida que cambia el valor de $W^i_{j,k}$. El nuevo valor de cada peso y bias de la red podrá ser calculado restando el valor de la derivada parcial al valor anterior de parámetro ya que al restar se aleja al valor de los parámetros de aquellos que aumentan el resultado de la función de costo. De esta forma, el gradiente se reducirá a medida que el valor de la salida de la red se acerque más al valor esperado aumentando la velocidad con la que la red aprende cuando su error es muy alto y disminuyéndola a medida que su error se acerca al esperado.

Adicionalmente se utilizará un parámetro llamado ratio de aprendizaje, el cual limitará o aumentará la velocidad de aprendizaje. De ser muy bajo el ratio de aprendizaje, el tiempo de convergencia aumentará demasiado por lo que el entrenamiento de la red llevará más tiempo en completar además de que se corre el riesgo de caer en un mínimo local de la función de error, lo que ocasiona que no se alcance el mejor entrenamiento posible. En caso de tener un ratio de aprendizaje demasiado alto, el entrenamiento podría llegar a divergir, debido a que posiblemente se salten zonas estrechas que posean mínimos con valor más bajos como se puede ver en 2.4.

Por otro lado, el ratio de aprendizaje obtiene ventajas al no ser fijo, ya que idealmente el error de la red impactará en sus pesos en mayor medida inicialmente y este impacto se disminuye a medida que la red se va entrenando. Es por esto que existen múltiples algoritmos de optimización que modifican el ratio de aprendizaje

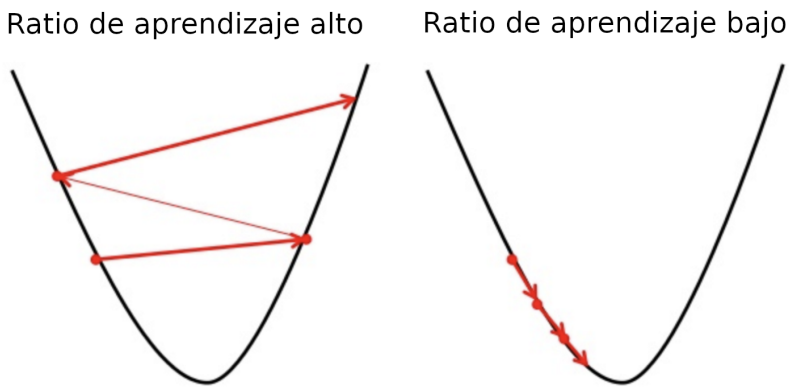



***Figura 2.4*** *Comparación de pasos realizados en el entrenamiento de una red con distintos ratios de aprendizaje. Izquierda: ratio de aprendizaje alto, luego de una cantidad de pasos el entrenamiento de la red comienza a divergir. Derecha: ratio de aprendizaje bajo, se realizan muchos pasos sin llegar al mínimo de la función de perdida L.*

como el algoritmo del momento, RMSProp y Adam.

Finalmente la operación para la actualización de un peso $w$ a $w'$ dada la función de costo $C$ quedaría escrita de la siguiente forma 2.8.

$$w' = w - \alpha \frac{\partial C}{\partial w} \tag{2.8}$$

Para entrenar la red neuronal hacen falta datos que permitan obtener los valores esperados de la función de costo. De esta forma es posible discernir diversos tipos de entrenamiento para los modelos según cómo son sus datos de entrenamiento en entrenamiento supervisado, no supervisado o semi-supervisado.

Un modelo supervisado es aquel cuyo entrenamiento está dado mediante datos de entrenamiento que poseen etiquetas de clase. El fin es obtener un modelo que trabaje como una función que dadas las etiquetas reales $Y$ de las datos de entrada $x$ los mapee de forma tal que $Y = f(x)$. El modelo podrá calcular de forma sencilla el error utilizado en su entrenamiento comparando el resultado obtenido por la clasificación de la imagen de entrada con sus verdaderas etiquetas.

Por otro lado, un modelo no supervisado es entrenado con datos de entrenamiento que no cuentan con etiquetas. Con el entrenamiento de este modelo se busca que aprenda la estructura y distribución de las imágenes de forma tal de obtener una clasificación en grupos de imágenes con estructura similar. Al no poseer las etiquetas de los datos de entrenamiento de antemano, el modelo no cuenta con las clases reales a las que debe asignar las imágenes, por lo que deberá aprender la estructura interna de estas y de esta forma encontrar la clasificación que mejor ajuste a los datos.

En muchos problemas es raro encontrar datos de entrenamiento los cuales posean la cantidad de datos etiquetados requerida para entrenar una red neuronal profunda

por medio de aprendizaje supervisado. En cambio la cantidad de datos no etiquetados disponibles suele ser mucho mayor, por lo que aprovecharlos podría facilitar el entrenamiento en este tipo de problemas. El aprendizaje semi-supervisado combina las características del aprendizaje supervisado y no supervisado. Este modelo intentará clasificar aquellas clases no etiquetadas utilizando un entrenamiento previo sobre los datos etiquetados, luego el modelo utilizara estos nuevos datos etiquetados para entrenarse y repetirá estos pasos sobre nuevos datos. De esta forma el modelo podrá obtener un conocimiento más amplio de la estructura de los datos.

### 2.2.3. Capas

La separación en capas le permitirá a la red neuronal extraer más información de la entrada para así poder realizar una predicción más precisa. Existen múltiples configuraciones de estas capas con sus propias cualidades y su uso específico. Estas capas son combinadas en diversas formas para formar múltiples tipos de arquitecturas de modelos. Muchos modelos comprimen una variedad de capas en bloques con características particulares que permiten afrontar nuevos problemas y aumentar la precisión o el tiempo de entrenamiento de las redes convolucionales que los utilicen.

#### Capas Fully Connected

Las DNN combinan múltiples tipos de capas de neuronas según el problema que busquen resolver. Las capas más básicas son las capas *fully connected* 2.5. Estas capas están compuestas por neuronas donde cada una recibirá entrada de cada una de las neuronas de la capa anterior multiplicado por un peso. En forma más específica, una capa *fully connected* es una función $\mathbb{R}^m$ a $\mathbb{R}^n$ donde siendo $x \in \mathbb{R}^m$ la entrada de una capa *fully connected* y $y \in \mathbb{R}^n$ la salida de esta, $y_i$ es el i-esimo elemento de la salida y $\sigma$ es la función de activación de las neuronas de la capa. $y_i$ se calcula de la siguiente forma 2.9

$$y_i = \sigma(x_1 w_1 + x_2 w_2 + ... + x_m w_m) \tag{2.9}$$

Por lo tanto la salida $y$ se calcula como 2.10

$$y = \sigma(x_1 w_{1,1} + ... + x_m w_{1,m}), ..., \sigma(x_1 w_{n,1} + ... + x_m w_{n,m}) \tag{2.10}$$

Se puede pensar en la salida como el resultado de la multiplicación de los filtros (pesos) y datos de entrada.

En visión por computadoras, la utilización de únicamente capas *fully connected* ha mostrado ser ineficiente, ya que la entrada x de cada neurona será cada píxel de la imagen llevada a un vector de una dimensión, haciendo crecer en gran medida el

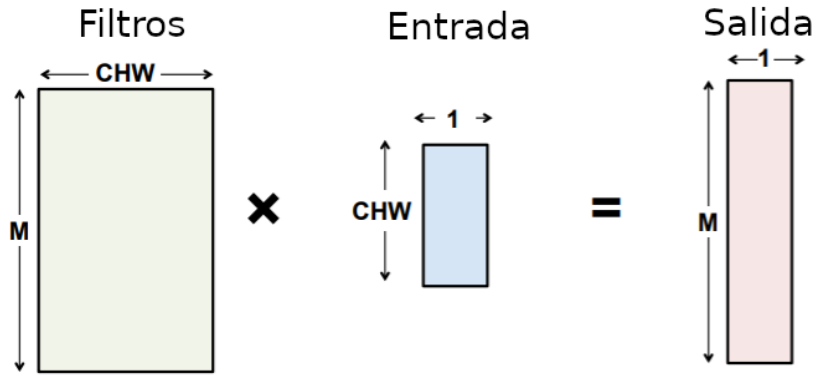



***Figura 2.5*** *Capa Fully Connected.*

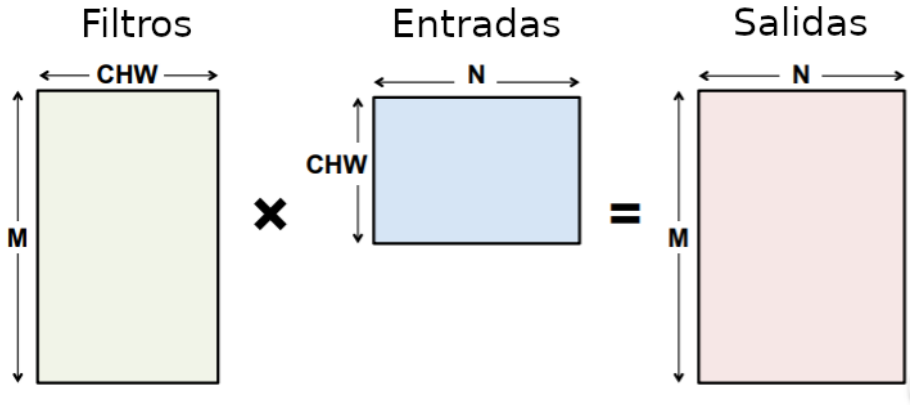



***Figura 2.6*** *Capa* fully connected *trabajando en* batches.

costo computacional a medida que crece la red o el tamaño de la imagen de entrada.

A pesar de esto, las capas *fully connected* han encontrado lugar en las redes convolucionales funcionando como capas de salida, colocándose al final de la red y aplicando sus pesos sobre los datos procesados por las capas convolucionales para obtener la etiqueta resultante. Utilizando capas *fully connected* como capa de salida se logra que la red sea completamente entrenable de comienzo a fin.

Es posible realizar esta operación sobre todo o parte del conjunto de datos al mismo tiempo, esto es llamado operar en *batches* o *mini batches* respectivamente. Utilizar esta forma de ejecución permite transformar la capa *fully connected* en una multiplicación de matrices como se puede ver en 2.6.

Esto otorga la posibilidad de obtener un gran aumento en la velocidad de procesamiento mediante la utilización de operaciones de multiplicación de matrices específicas de CPU y GPU.

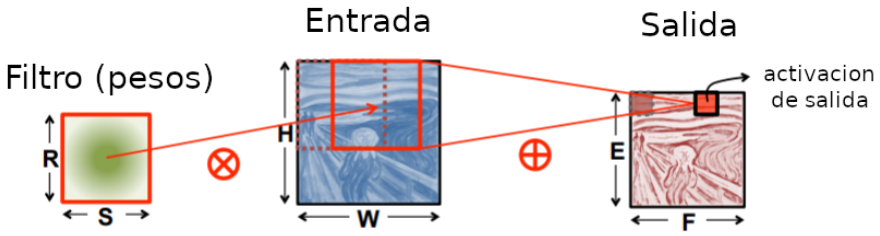



***Figura 2.7*** *Procesamiento de ventana deslizante.*

### Capas de convolución

Otro tipo de capa comúnmente usada para el clasificación de imágenes es la capa convolucional [10] 2.7. Esta capa funciona mediante una ventana de filtros deslizante de tamaño $RxS$, la cual irá construyendo la salida a medida que se desplaza por la entrada. La cantidad de espacio que la ventana se desplaza está dado por los pasos $s$. Además es posible utilizar *padding* $p$ en la entrada para aumentar o disminuir el tamaño de la salida. También será posible aumentar la cantidad de filtros utilizados en cada capa para aumentar la cantidad de características extraíbles por la capa convolucional.

Siendo $W_k \in \mathbb{R}^{R\times S\times c}$ los $k$ filtros de una capa convolucional, $X \in \mathbb{R}^{H\times W\times c}$ el mapa de características de entrada $y$ la salida $Y_k \in \mathbb{R}^{E\times F}$. La cantidad de canales $c$ de cada filtro convolucional será igual a la cantidad de canales de la entrada. El tamaño de la salida, de dimensiones $E \times F$, será igual a la cantidad de pasos que de la ventana de los filtros convolucionales sobre la entrada de forma tal que $E = (H - R + 2p)/s$ y simétricamente $F = (S - R + 2p)/s$. Dado un elemento en el mapa de características de salida $y_{i,j,k} \in Y_k$ donde $i, j \in \mathbb{N}, 0 \leq i < E$ y $0 \leq j < F$ y una matriz de elementos $x_{i,j} \in \mathbb{R}^{R\times S}$ tomados de la entrada $x_{i,j} \subseteq X$ el área que ocupa $x_{i,j}$ se desliza por $X$ a medida que cambian $i$ y $j$ 2.11.

$$y_{i,j,k} = \varphi(\sum_{i=0}^{E}\sum_{j=0}^{F}(W_k \otimes x_{i,j}) + b_k) \tag{2.11}$$

Con $b_k$ siendo el bias utilizado por el filtro $k$ y $\varphi$ la función de activación definida. El símbolo $\otimes$ indica multiplicación de elementos, por lo que $c_{i,j} = a_{i,j} \times b_{i,j}$ siendo $c$ el resultado de la multiplicación, y $a$ y $b$ las matrices a multiplicar.

Estas capas son utilizadas para formar redes neuronales convolucionales. Este tipo de redes permite una clasificación precisa de imágenes y objetos que se encuentren en estas aprendiendo en sus capas convolucionales las características de las imágenes a diferentes niveles de abstracción, similar a como un humano procesa las

imágenes [27].

Al igual que con las capas *fully connected*, es posible realizar el procesamiento en *batches* y a través de múltiples canales, de esta forma se aumenta la paralelización del procesamiento.

Existen múltiples formas de optimizar la ejecución de las capas convolucionales. Algunas de las más utilizadas son convolución directa, *unrolling-based convolution* [49] y *Fast Fourier Transform* (FFT) [106].

### Capas pooling

Las capas convolucionales extraen la información espacial de la entrada en sus filtros, los cuales generarán la salida deslizándose por la entrada. Debido a esto, si la entrada posee un tamaño demasiado grande, la información espacial que será observable por el los filtros convolucionales estará apuntada a pequeños detalles de la entrada. Si se entrena una red convolucional directamente con la entrada sin realizar cambios a su tamaño la red podría perder la capacidad de extraer detalles de más alto nivel en cuanto al posicionamiento de las características de la entrada. Por ejemplo, si se entrena sin reducir el tamaño de la entrada, un mínimo cambio en el posicionamiento del objeto a predecir podría generar una predicción errónea, debido a que la red no generaliza correctamente.

Por otro lado, reducir el tamaño de la entrada permite disminuir el tiempo de entrenamiento e inferencia de la red. Al ser más pequeña la entrada, la cantidad de movimientos que deberá realizar la ventana deslizante de una capa convolucional para obtener la salida será menor reduciendo así la cantidad de operaciones necesarias para cada capa y debido a esto el tiempo de procesamiento.

Para realizar esta reducción en el tamaño de las entradas es posible utilizar capas *pooling* 2.8. Estas capas funcionan como una operación aplicada sobre la entrada que devuelve una salida de menor tamaño pero igual cantidad de dimensiones. La operación de *pooling* utiliza dos parámetros, el tamaño indicará la ventana sobre la que se hará la operación de *pooling* y el paso indicará cuánto se moverá la ventana de *pooling* sobre la entrada. La capa *pooling* no realiza ningún aprendizaje, sus parámetros son constantes elegidas previamente al aprendizaje. La operación de *pooling* puede ser promedio o máximo, esto indica que la salida de la ventana será el promedio de todos sus elementos o el máximo elemento en su alcance respectivamente.

### Batchnormalization

Se sabe que la normalización de la entrada de la red neuronal es beneficiosa para su entrenamiento, obteniendo una convergencia más temprana. Con el surgimiento de las redes neuronales profundas es natural pensar en extender esta normalización

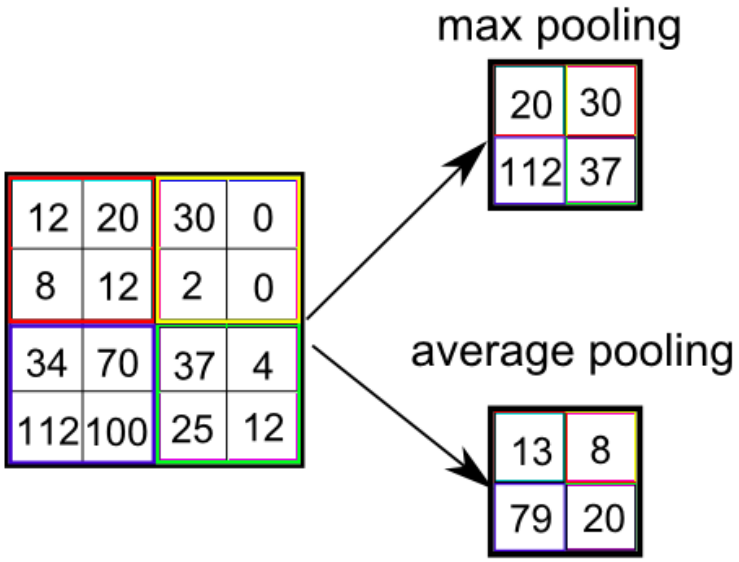



***Figura 2.8*** *Resultados de aplicar* max pooling *y* average pooling *a una matriz.*

a las capas intermedias. De esta idea surgieron las capas de *batch normalization* que realizan normalización de los mapas de características intermedios en *batches*. En cada *batch* se calculará la media y la desviación estándar usando los mapas de características de dicho *batch*, esto se apoya en el trabajo en *mini batches* usuales en el entrenamiento de redes neuronales ya que permiten aumentar la velocidad de entrenamiento [9].

La utilización de *batch normalization* permite la utilización de un ratio de aprendizaje mayor en entrenamiento, lo que permitirá alcanzar una convergencia más rápido y evitar mínimos locales. Esto ocurre ya que la capa *batch normalization* corrige las activaciones en cada paso de entrenamiento a media cero y unidad de desviación estándar. De esta forma las activación no crecerán demasiado generando divergencia.

Las capas de *batch normalization* son normalmente usadas en redes convolucionales, por lo que los mapas de características tendrán 4 dimensiones: cada elemento del *batch* $b$, cada canal $c$ y las dimensiones espaciales $x$ e $y$. La capa toma una entrada $X_{b,c,x,y}$ con la que calcula su media $\mu_c = \frac{1}{|B|}\sum_{b,x,y} X_{b,c,x,y}$ ,donde $B$ son todas las activaciones del *batch* de un canal $c$, y desviación estándar $\sigma_c$ (a la que sumará para obtener estabilidad numérica) de cada canal $c$. Además utilizará otros dos parámetros $\gamma_c$ y $B_c$ los cuales serán aprendidos en entrenamiento y realizarán una transformación afín para cada canal.

$$Y_{b,c,x,y} = \gamma_c \frac{X_{b,c,x,y} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}} + B_c \tag{2.12}$$

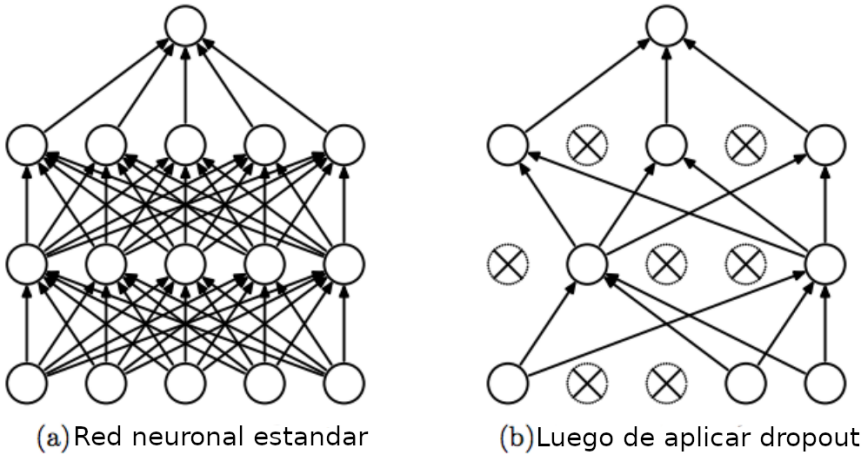



***Figura 2.9*** *Resultado de aplicar* dropout *a las capas* fully connected *de una red neuronal.*

### Dropout

En problemas donde los datos no son lo suficientemente variados puede surgir sobreajuste sobre los datos de entrenamiento. Esto generará que la red neuronal realice predicciones con un alto nivel de acierto sobre los datos de entrenamiento y un bajo nivel de acierto sobre datos de prueba ya que la red aprenderá características propias de los datos de entrenamiento en vez de aprender aquellas características generales que distinguen las clases.

Una técnica muy utilizada para prevenir el sobreajuste es el *dropout*. Esta técnica consiste en apagar neuronas de una capa de la red neuronal de forma tal que estas no reciben ninguna entrada ni generan ninguna salida a capas posteriores. Es posible aplicar *dropout* a cualquier capa de la red neuronal salvo a la capa de salida. El *dropout* generará un mejor balance en los pesos de las capas del modelo ya que las neuronas de cada capa no podrán depender exclusivamente del valor de una sola entrada que posiblemente se encuentre apagada.

Esta técnica permite utilizar redes neuronales más grandes que normalmente aumentan el sobreajuste sobre los datos de entrenamiento. De hecho, normalmente se aumenta el tamaño de la red neuronal al utilizar *dropout* para contrarrestar el hecho de que algunas neuronas se apaguen en entrenamiento 2.9.

*Dropout* es muy útil principalmente en capas *fully connected*, pero en capas convolucionales no se refleja completamente su utilidad. Esto es así debido a que la información en las entradas de las capas convolucionales se encuentra correlacionada espacialmente. Esto genera que a pesar del *dropout* la información de la entrada pueda ser pasada hacia la siguiente capa. Debido a esto existen otras técnicas que cumplen funciones similares a las del *dropout* regularizando la red convolucional. Algunas de estas técnicas son StochasticDepth que apaga capas completas de la red convolucional en entrenamiento, ShakeDrop regularization y DropBlock la cual,

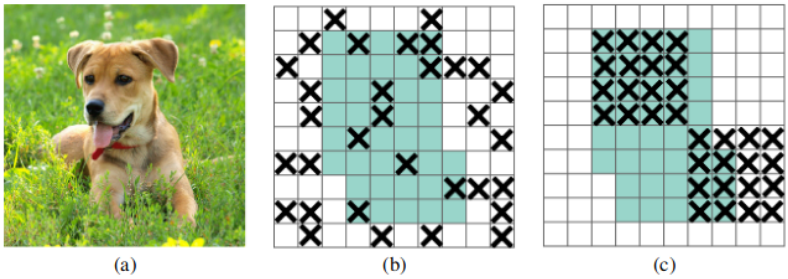



***Figura 2.10*** *(a) imagen de entrada de una red neuronal convolucional. (b) efecto de aplicar dropout regular. (c) efecto de aplicar dropblock. Las regiones verdes en (b) y (c) incluyen las unidades de activación que contienen información semántica de la imagen de entrada.*

inspirada en la técnica de preprocesamiento de cutout que remueve zonas completas de la entrada de la red, remueve zonas de los mapas de características de entrada de las capas intermedias de la red convolucional 2.10.

### Bottleneck

Las capas *bottleneck* son capas convolucionales que poseen un tamaño de filtro convolucional de $1 \times 1$ y un paso de 1. Su objetivo es el de reducir la cantidad de canales de la entrada a una capa convolucional con filtros de mayor tamaño y de esta forma reduciendo en gran medida el tiempo de ejecución. Además estas capas también pueden ser utilizadas para aumentar la cantidad de canales, este crecimiento o decrecimiento en la cantidad de canales será igual a la cantidad de filtros convolucionales utilizados 2.11.

### Bloque Squeeze and Excitation

Las redes convolucionales construyen características informativas fusionando información espacial y de los canales dentro de los campos receptivos locales de cada capa. Los bloques Squeeze and excitation (bloques SE) [40] se centran en la información de los canales que es utilizada en las capas convolucionales. Estos bloques mejoran la calidad de las representaciones producidas por la red neuronal modelando la interdependencia entre canales y realizando calibración de las características del modelo 2.12.

Los bloques SE pueden ser incluidos en cualquier modelo que utilice capas convolucionales para mejorar su performance a un bajo coste computacional.

Un bloque SE es una operación de 2 pasos en un mapa de características $V$. Primero se realiza la operación *squeeze* (apretar) en la cual se ejecuta una operación

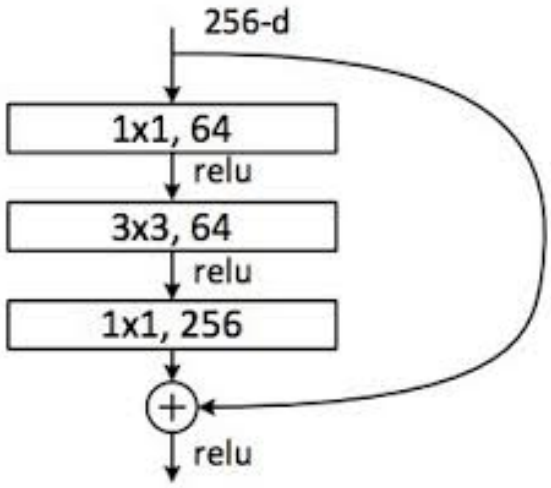



***Figura 2.11*** *Utilización de capas bottleneck con el fin de reducir la cantidad de canales de entrada de la capa convolucional* $3 \times 3$.

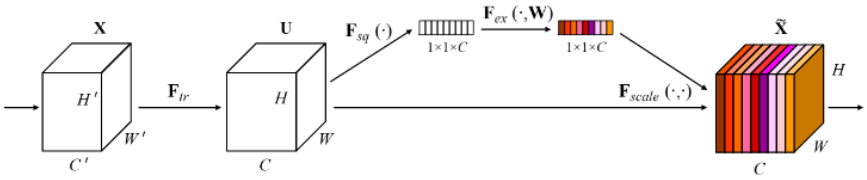



***Figura 2.12*** *Se introduce el bloque SE entre 2 capas convolucionales para modelar la interdependencia entre canales recalibrando las características que serán utilizadas como entrada de la siguiente capa.*

de *global average pooling* reduciendo la dimensionalidad de $V$ a la de un vector 2d con un tamaño igual a la cantidad de canales de $V$.

Luego se realiza la operación *excitation* (excitación). Esta operación captura las dependencias entre canales de forma flexible (aprendiendo la interacción no lineal entre canales) y no mutuamente excluyente (para que múltiples canales puedan ser enfatizados). Para esto utiliza una capa *fully connected* con activación ReLU y otra capa *fully connected* con activación sigmoide. La activación sigmoide de la última capa asegura que los canales no sean mutuamente excluyentes. Un ratio de reducción r es añadido a la primera capa *fully connected* del bloque SE con el propósito de reducir el coste computacional de los bloques SE de la red disminuyendo el tamaño de la primer capa *fully connected*. De esta forma, un ratio de reducción mayor otorgará un costo computacional menor pero peor ganancia en performance.

La salida del último paso será un valor escalar para cada canal. Para usar este escalar se incrementa la dimensionalidad de la salida de la capa *fully connected* de activación sigmoide para que coincida con la dimensionalidad de $V$. La salida

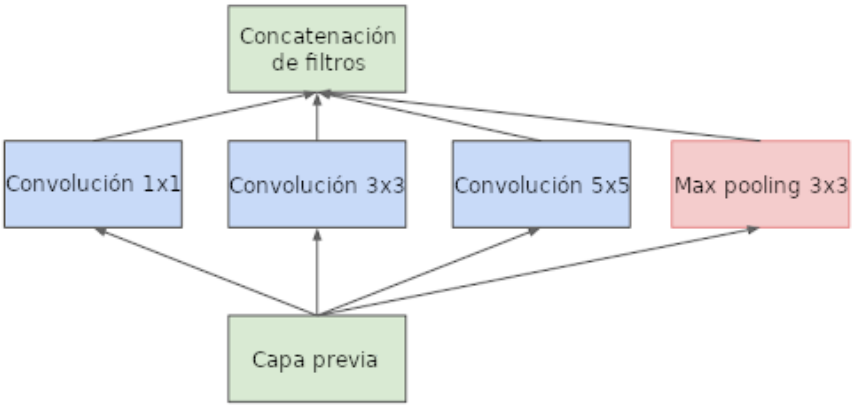



***Figura 2.13*** *Capa Inception.*

final del bloque SE será obtenida realizando una multiplicación de canales entre $V$ y el escalar. Es decir, dado el vector de entrada $V_{i,j,c}$ y el escalar $s_c$ siendo $c$ el canal, $\forall i, jV_{i,j,c} = V_{i,j,c} \times s_c$. De esta forma se obtendrá finalmente un mapa de características cuyos canales se encuentran pesados por el escalar.

### Modulo Inception

Una red dispersa óptima para un determinado problema, similar a las encontradas en sistemas biológicos, ayudaría a disminuir el tiempo de procesamiento y la posibilidad de sobreajuste. El problema está dado en que las estructuras de datos dispersas no uniformes requieren de mayor ingeniería y nueva infraestructura de computación dado que la existente no se adapta correctamente a las operaciones dispersas. La idea de la creación del módulo Inception [113] fue la de aproximar la estructura local dispersa mediante componentes densos ya disponibles y demostrados eficientes utilizando hardware contemporáneo como GPUs y CPUs (veloces en procesamiento de multiplicación de matrices).

Los módulos Inception están compuestos por bloques de múltiples capas convolucionales paralelas con distintos tamaños de filtro (por ejemplo $5 \times 5$, $3 \times 3$ y $1 \times 1$) y una capa de *max pooling* también paralela. Los resultados de estas capas paralelas son posteriormente concatenados para ser utilizados como la salida del bloque, de esta forma el siguiente bloque podrá abstraer características de diferentes escalas simultáneamente 2.13.

Si el módulo Inception fuera usado de esta forma se obtendría un gran crecimiento en el tamaño de las salidas y, por lo tanto, en el costo computacional en cada paso al concatenar las salidas de las capas paralelas internas de cada módulo. Para prevenir este crecimiento en las etapas más avanzadas de la red se incluyeron capas *bottleneck* (capas convolucionales de $1 \times 1$) que reducen la dimensionalidad de su entrada. De

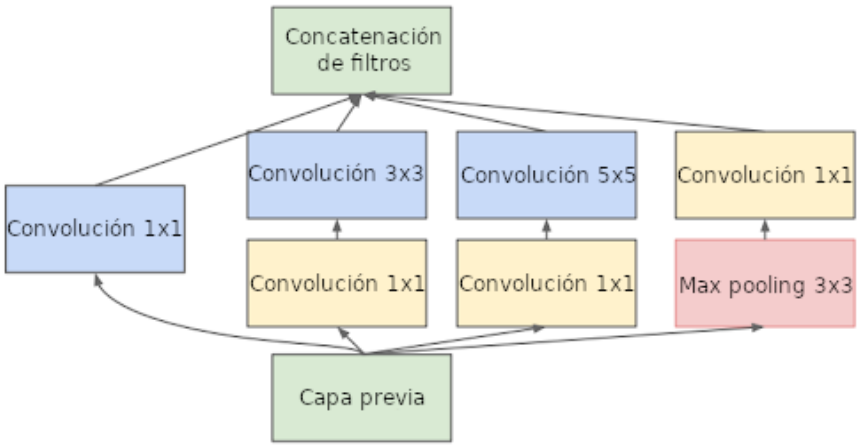



***Figura 2.14*** *Capa Inception optimizada.*

esta forma es posible aumentar el tamaño de la red en ancho y profundidad sin incrementar de forma excesiva su tiempo de cómputo 2.14.

En un articulo siguiente [114], los autores presentaron Inception v2 e Inception v3 con múltiples mejoras al funcionamiento de los módulos Inception permitiendo reducir el costo computacional de estos y aumentar su precisión.

### Bloque Residual

En búsqueda de mayor precisión las redes neuronales tienden a ser cada vez más anchas y profundas. Múltiples modelos de redes neuronales profundas han seguido este patrón para problemas cada vez más complejos y grandes.

El problema surge cuando al aplicar cada vez más capas a una red neuronal plana se introduce degradación, llegando al punto en el que agregar más capas a la red reduce su exactitud. Contrario a lo que se puede pensar, esta reducción en exactitud no es causada debido al sobreajuste ya que agregar más capas al modelo aumenta el error sobre los datos de entrenamiento, el cual en caso de darse sobreajuste no tendría por qué aumentar. La degradación es causada en el entrenamiento y optimizamiento de la red, apilar demasiadas transformaciones no lineales en arquitecturas de redes *feedforward* convencionales normalmente resulta en una mala propagación de los gradientes y las activaciones [110].

En las redes convolucionales convencionales *feedforward*, los conjuntos de capas convolucionales intentan ajustarse a un mapa subyacente óptimo $H(x)$. A medida que la red se entrena, los pesos de estos conjuntos de capas se intentan acercar a este mapa, pero en redes demasiado profundas este mapeo óptimo sería más difícil de alcanzar debido al gradiente desvaneciente o a la posibilidad de caer en mínimos locales en el entrenamiento [72]. Los módulos residuales [36] buscan ajustar el con-

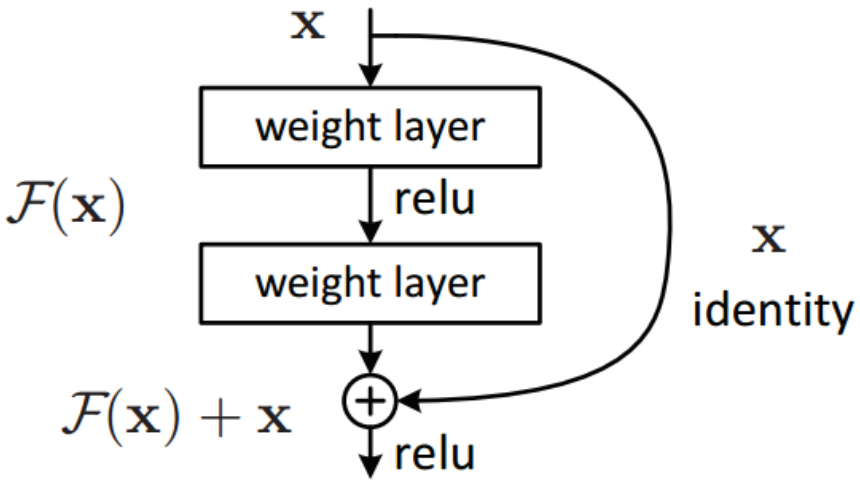



***Figura 2.15*** *Bloque residual.*

junto de capas convolucionales a un mapa residual $F(X) := H(x) - x$. Ajustar a este mapa residual es equivalente a ajustar a una función compleja $H(x)$ para obtener el mapeo óptimo. Por lo tanto se podría aproximar las capas a $F(X) := H(x) - x$ de la misma forma que a $H(x)$, volviendo a $F(x) + x$ la función original. Esta nueva función se implementa con conexiones shortcut de forma tal que a la salida de cada bloque se le sumará la entrada elemento a elemento 2.15. Esta suma elemento a elemento no añade complejidad computacional ni parámetros al no aumentar las dimensiones de la salida. En caso de que la salida y la entrada del bloque posean diferentes dimensiones, se utilizara una proyección lineal para equiparar los tamaños.

Existen múltiples estudios que investigan cuáles son las razones por las que, utilizando bloques residuales, una red neuronal residual (ResNet) es capaz de aumentar en gran medida su profundidad evitando la degradación. Veit A. [122] propuso en su articulo *Residual Networks Behave Like Ensembles of Relatively Shallow Networks* que las redes residuales funcionan como un ensamblamiento de redes neuronales menos profundas. La prueba de esto es que al remover capas o modificar el orden de algunas capas no se ve una gran reducción en la performance comparada a la reducción que sí se vería en una red VGG. El hecho de que una red residual funcione como un ensamblamiento podría justificar la reducción del efecto del gradiente desvaneciente, ya que se acortan los caminos efectivos. Otros estudios indican que las redes que utilizan bloques residuales tienden a evitar el mínimo local y converger más frecuentemente al mínimo global [72][67].

**Bloque Denso**

A medida que las redes convolucionales crecen en profundidad la información y el gradiente de las redes comienza a desvanecerse. Las capas residuales mostraron éxito tratando el gradiente desvaneciente conectando la salida de bloques anteriores a la entrada de bloques posteriores. Los bloques densos [41] profundizan esta conectividad conectando a cada entrada de cada bloque convolucional las salidas de todos los bloques convolucionales anteriores. A diferencia de los bloques densos que realizan suma elemento a elemento de la salida anterior a la entrada, los bloques densos realizan concatenación entre la entrada y las salidas anteriores. De esta forma se evita la pérdida de información preservando los mapas de características de salida de capas anteriores. Con esto es posible diferenciar cuál información es añadida a la red y cual es preservada. Esto permite mejorar el reutilizamiento de parámetros y, por lo tanto, disminuir su cantidad ya que a diferencia de las redes que utilizan bloques residuales, donde existen múltiples capas que no aportan demasiada información debido a su funcionamiento como red ensamblada, en las redes con bloques densos cada capa posee el conocimiento colectivo del bloque permitiendo así aprender nuevos patrones.

Adicionalmente, las conexiones densas proveen un efecto regularizador. Esto significa que los valores de los parámetros en la red se mantienen bajo evitando así el sobreajuste y facilitando el entrenamiento de la red en bases de datos con pocos datos.

Dada una red de L capas que busca ajustar una transformación no lineal en cada capa $H_l(x)$, donde $x$ representa la entrada a de la capa y $l$ el índice de la capa.

Siendo $x_i$ con $i \in 0...l$ la salida de la capa $i$ es posible formular la salida de una capa de un bloque denso como

$$x_l = H_l([x_0, ..., x_{l-1}])$$

Siendo $[x_0, ..., x_{l-1}]$ la concatenación de los mapas de características de salida de capas anteriores en el bloque denso 2.16.

### 2.2.4. Arquitecturas convencionales para la clasificación de imágenes

Una vez comprendido el funcionamiento de las capas y bloques, queda entender como organizar estas capas y bloques de forma tal que la red obtenga la mejor precisión utilizando la menor cantidad de recursos, lo que permitirá agrandar la red para intentar mejorar la precisión y ahorrar tiempo y recursos[11].

Modelos de redes neuronales poco profundos han existido desde hace varias décadas. En las décadas 1960 y 1970 comenzaron a surgir modelos de múltiples capas

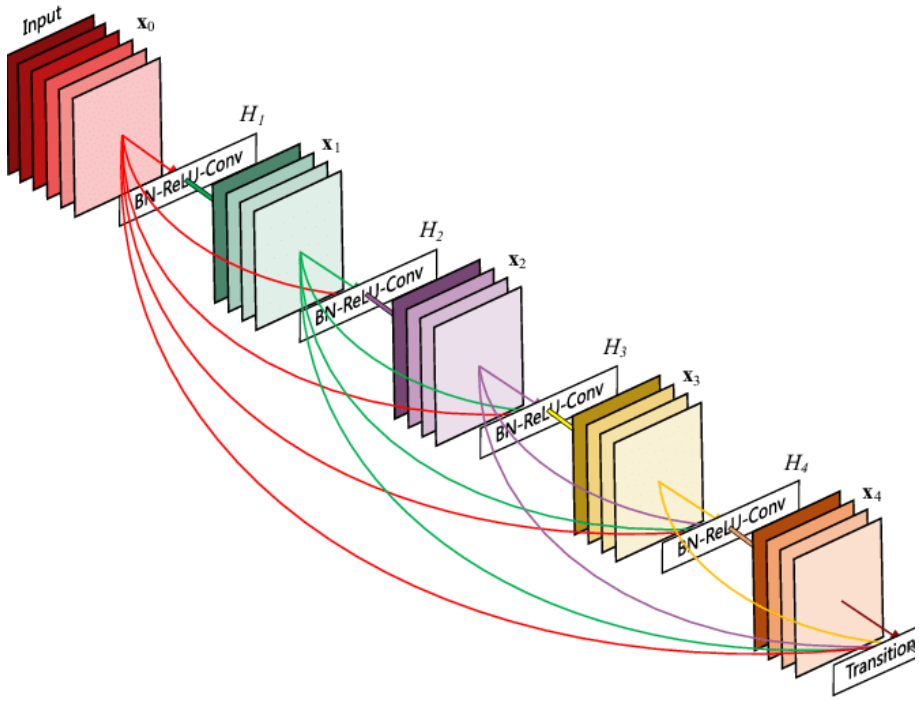



***Figura 2.16*** *Bloque denso.*

de neuronas. Desde esos tiempos se ha experimentado con distintas topologías y profundidades de redes. En 1965 comenzaron a entrenarse redes con *Group Method of Data Handling* (GMDH) donde la cantidad de capas y unidades por capa se adapta al conjunto de datos de entrenamiento. En 1979 fue desarrollado Neocognitron, el primer modelo en implementar capas convolucionales [98].

Con el paso de los años se han ido desarrollando arquitecturas de redes neuronales cada vez más precisas en la visión por computadora. Estas arquitecturas aprovechan de diversas maneras las características de las redes neuronales para lograr amplias mejoras sobre arquitecturas anteriores obteniendo mayor precisión [36][41][15][40], menos requerimientos en datos [25][107] o tiempo de entrenamiento e inferencia [119][39][137][70][118].

Los modelos estado del arte son aquellos que, en la actualidad, han logrado la mejores resultados en sus respectivas áreas. Cada una de estas arquitecturas ha sido diseñada para afrontar distintos problemas, por lo que no existe una que pueda ser aplicada universalmente en todas las situaciones. Desde modelos creados para intentar lograr la mayor precisión en problemas de clasificación de miles de clases [33], donde el tiempo de inferencia pasa a segundo plano, hasta modelos diseñados para su implementación en plataformas móviles [39], que intentan alcanzar un bajo tiempo de inferencia utilizando los limitados recursos de hardware disponibles en los dispositivos móviles, la elección de la arquitectura a utilizar depende específicamente de los requerimientos dados.

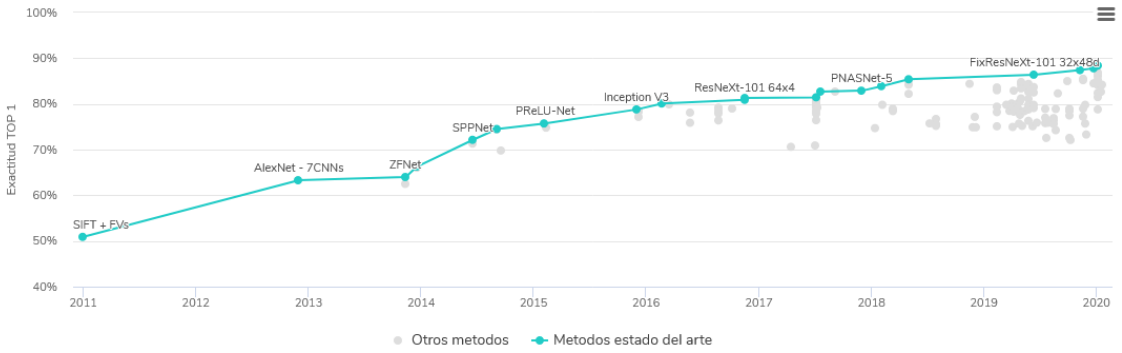



***Figura 2.17*** *Gráfico con las precisiones de los modelos estado del arte a lo largo de los años en el conjunto de datos ImageNet.*

En la visión por computadora, existen múltiples conjuntos de datos de referencia sobre los que se realizan las comparaciones entre los diferentes modelos estado del arte. Entre estos conjuntos ImageNet [21] destaca como uno de los conjuntos de datos imágenes más utilizados para medir la performance de nuevos modelos 2.17. ImageNet presenta 3 retos: localización de objetos, detección de objetos y detección de objetos en vídeos.

### AlexNet

En 2012 Alex Krizhevsky publicó el articulo *ImageNet Classification with Deep Convolutional Neural Networks* [59] en el cual presento una red convolucional profunda, que posteriormente pasaría a ser llamada AlexNet, la cual alcanzó un ratio de error top-1 y top-5 de 39.7 % y 18.9 % respectivamente un resultado muy llamativo para su época, en la cual se creía a *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [94] como una tarea sumamente complicada para su tiempo donde se debían clasificar imágenes en 1000 categorías diferentes.

AlexNet cuenta con 5 capas convolucionales con activaciones ReLU, capas de *max pooling* y 3 capas *fully connected* en la parte de clasificación del modelo con activación sigmoide en su salida. También se utilizó *dropout* en las capas *fully connected* para evitar sobreajuste. Adicionalmente se realizó *data augmentation* para incrementar artificialmente el tamaño del conjunto de datos de entrenamiento y poder extraer más información de este. Todas estas eran técnicas conocidas y funcionales pero no ampliamente utilizadas en su tiempo. Actualmente se siguen utilizando estas técnicas en modelos estado del arte.

### VGG

Posterior a la creación de AlexNet, VGG [104] ayudó a estandarizar la creación de modelos más profundos y performantes de redes convolucionales. VGG redujo el tamaño de los filtros convolucionales pasando de los filtros de $11 \times 11$ de AlexNet a filtros de $3 \times 3$. Esto permitió a VGG incrementar la cantidad de capas convolucionales al requerirá menos cantidad de parámetros y por lo tanto procesamiento por cada capa. Estas capas convolucionales se encontraban apiladas en pequeñas cantidades entre las capas de *max pooling*. Fueron desarrollados VGG-16 y VGG-19, haciendo alusión a la cantidad de capas convolucionales que poseen siendo estas 16 y 19 respectivamente.

### GoogLeNet

En el reto ILSVRC de 2014 [94] se presento el modelo GoogLeNet, el cual obtuvo muy altos resultados. En el mismo papel que GoogLeNet se introdujeron los modelos Inception con los módulos Inception. Los módulos Inception permitieron reducir en gran medida la cantidad de parámetros necesarios en la red. Previo a los modelos Inception las topologías de redes neuronales estaban conformadas por capas convolucionales de gran cantidad de filtros apiladas con capas de *max pooling* entre estas pilas. Como comparación, en el mismo año, VGG-19 presentaban 144 millones de parámetros con un error top 5 de 92.0 % mientras que GoogLeNet poseía apenas 5 millones con un error top 5 de 89.9 %.

Utilizando los eficientes módulos Inception, GoogLeNet, pudo aumentar la profundidad del modelo a 22 capas. Debido a el aumento en la profundidad se incrementó el problema del gradiente desvaneciente. Para contrarrestar esto, los autores de GoogLeNet agregaron salidas en diferentes puntos del modelo creando pequeñas redes de salida entrenadas para realizar predicciones. Utilizando estas predicciones se calcula una pérdida auxiliar la cual es sumada con un peso de descuento (por ejemplo un peso de 0.3 fue utilizado en GoogLeNet) a la pérdida general de la red 2.18. Posteriormente, estas salidas auxiliares serán removidas para realizar la inferencia.

### Resnet

A pesar de los avances anteriormente mencionados, aún existían problemas para entrenar redes extremadamente profundas, no solo debido al sobreajuste, sino que comienza a surgir el problema del gradiente desvaneciente a medida que se agregan más capas. Las redes residuales (ResNet) [36] afrontaron este problema utilizando bloques residuales. Estos bloques utilizan conexiones *shortcut* sumando la entrada del bloque a la salida de este. Esto permitió facilitar el entrenamiento de redes aún más profundas. Como comparación, en el reto ILSVRC de 2016 en el que compitieron

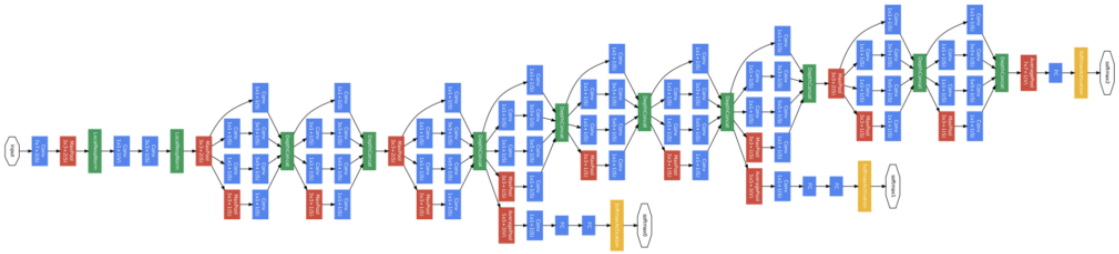


***Figura 2.18*** *Arquitectura GoogleNet.*

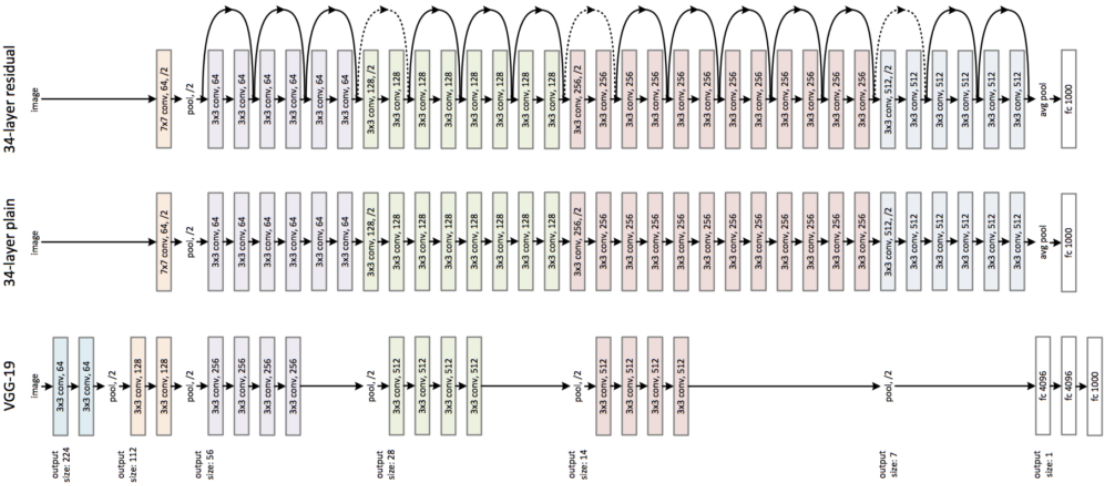



***Figura 2.19*** *Arquitectura del modelo ResNet utilizado en el articulo original.*

tanto Inception v3 [114] (el cual aplicó múltiples mejoras sobre las redes Inception) como ResNet, Inception v3 presentó un modelo con 42 capas mientras que el modelo ResNet poseía unas impresionantes 152 capas.

El modelo ResNet original se construyó basando su arquitectura en un modelo VGG simplificado el cual posee menos filtros y complejidad que modelos VGG convencionales. A este modelo luego se le agregaron las conexiones *shortcut* para transformarlo en una red residual. Los bloques residuales del modelo están compuestos de 2 capas convolucionales y realizan conexiones *shortcut* de identidad o conexiones *shortcut* proyectadas. La conexión *shortcut* de identidad realiza mapeo de identidad de la entrada del bloque a la adición con la salida de este y es utilizada cuando las dimensiones de la salida del bloque coinciden con las de la entrada. Cuando estas dimensiones no coincidan serán utilizadas conexiones *shortcut* proyectadas (indicado en la imagen del modelo 2.19 por flechas punteadas). Las conexiones *shortcut* proyectadas utilizan convoluciones $1 \times 1$ para igualar las dimensiones de la entrada a las de la salida 2.19.

**Densenet**

En la *Conference on Computer Vision and Pattern Recognition 2017* (CVPR) fue presentado el modelo DenseNet en el paper [41] titulado *Densely Connected Convolutional Networks* con el cual ganó el premio *Best Paper Awards*.

DenseNet introdujo los bloques densos. Estos bloques funcionan concatenando los mapas de características de entrada de un bloque convolucional a los mapas de características salida de todos los bloques convolucionales previos dentro del bloque denso para luego utilizar esta concatenación como entrada al bloque convolucional. De esta forma, cada bloque convolucional recibe el conocimiento colectivo de todos los bloques anteriores manteniendo el estado global de la red y permitiendo su acceso desde cualquier sitio de esta.

Utilizando la arquitectura de DenseNet la red puede ser menos profunda que otras como ResNet pero aun así dando mejores resultados. Esto es debido al reutilizamiento de características a través de la red que resulta de la concatenación en la entrada.

Cada bloque convolucional de la arquitectura DenseNet presentada en el articulo original está compuesto de una capa *bottleneck*, la cual contiene una convolución $1 \times 1$ y ayuda a reducir la complejidad de la red, y una función compuesta. La función compuesta está formada por 3 operaciones: *batch normalization*, ReLU y una capa convolucional de $3 \times 3$. Este tipo de arquitectura DenseNet que posee una capa *bottleneck* es referida como DenseNet-B.

DenseNet utiliza un ratio de crecimiento en sus capas con el cual controla el incremento en la cantidad de canales de cada capa. Utilizando este ratio se aumenta el tamaño de las capas convolucionales de cada bloque denso de forma incremental. Es decir, dada una capa convolucional $l$ perteneciente a un bloque denso, la cantidad de canales de salida de la capa $l$ será de $k_0 + k \times (l - 1)$ donde $k_0$ es la cantidad de canales de la entrada y $k$ es el ratio de crecimiento. A medida que se avanza por cada capa convolucional en el bloque denso la cantidad de canales y, por lo tanto, de información que aporta cada capa incrementará en $k$.

Entre los bloques densos de los modelos DenseNet son ubicados bloques de transición. Estos bloques están a cargo de realizar *downsampling*, reduciendo la dimensionalidad de los mapas de características. Esto lo hacen utilizando una capa convolucional *bottleneck* seguida de una capa *average pooling* de $2 \times 2$. De esta forma es posible aumentar la profundidad de la red contrarrestando el valor del ratio de crecimiento para mantener una cantidad de parámetros relativamente bajo que permitan un buen tiempo de cómputo.

Para incrementar aún más la compacidad del modelo es posible añadir un factor de compresión a los bloques de transición. En cada bloque de transición, si su entrada es un mapa de características de tamaño $m$ y se utiliza un factor de compresión

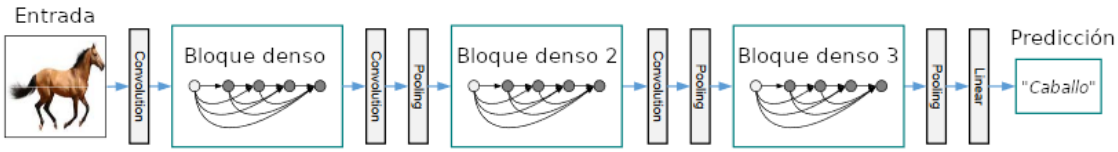



***Figura 2.20*** *Arquitectura DenseNet.*

$0 < \theta < 1$, la salida del bloque sea un mapa de características de tamaño $\theta m$. Con un factor de compresión de 1 el tamaño de los mapas de características se mantendrá intacto. Un modelo DenseNet con un valor de $\theta$ menor a 1 es referido como DenseNet-C, y si este modelo también utiliza capas *bottleneck* será referido como DenseNet-BC 2.20.

Una ventaja de utilizar DenseNet es que las señales de error de la clasificación puede ser fácilmente propagadas a capas anteriores. Cada capa posee acceso directo a los gradientes de la función de costo y a su vez a la entrada original de la red. El clasificador al final de la red provee supervisión directa a todas las capas con solo unos pocos bloques de transición intermedios. Esta supervisión implícita facilita el entrenamiento otorgando transparencia a las capas profundas de la clasificación, además previene el problema del gradiente desvaneciente o explotado y aumenta la robustez y discriminación de las primeras capas [62].

DenseNet ha mostrado conseguir muy buenos resultados tanto en bases de datos chicas como grandes. En el articulo original DenseNet-201 alcanza una exactitud top 5 de 94.46 % con 20 millones de parámetros mientras que ResNet-101 un año antes requirió de 40 millones de parámetros para alcanzar una exactitud top 5 de 93.95 %.

### NASNet

Cómo es posible ver con las arquitecturas presentadas anteriormente, la complejidad de las arquitecturas de redes neuronales se han ido acercando al punto en el que el diseño humano comienza a limitar el potencial de encontrar la arquitectura óptima para un determinado problema. La selección de la arquitectura óptima normalmente requeriría una gran cantidad de tiempo y de ingeniería de arquitecturas por una persona experta.

Para solucionar esto fue propuesto el *framework Neural Architecture Search* [136] el cual utiliza una red neuronal recurrente (RNN) como controlador para buscar obtener la mejor arquitectura. Las redes neuronales normalmente pueden ser explicitadas como un string de configuración, el cual puede ser buscado por una RNN. Para esto el controlador buscará configuraciones para formar nuevas redes neuronales hijas. Posteriormente estas redes neuronales hijas pueden ser entrenadas sobre los datos

de entrenamiento para conseguir la exactitud sobre el conjunto de validación. Esta exactitud será utilizada como la recompensa para realizar aprendizaje por refuerzo actualizando los pesos del controlador de forma tal que en cada iteración aumentarán las probabilidades de obtener mejores arquitecturas.

En la CVPR 2018 fue presentado el articulo *Learning Transferable Architectures for Scalable Image Recognition* [137] que introdujo NASNet. Esta red se construyó utilizando el espacio de búsqueda NASNet. Implementar NAS sobre una base de datos de gran tamaño llevaría un tiempo demasiado largo, ya que es requerido el entrenamiento de las redes hijas para obtener el resultado del controlador. El espacio de búsqueda NASNet permite buscar arquitecturas óptimas para bases de datos de mayor complejidad (por ejemplo ImageNet) partiendo de bases de datos de menor complejidad (por ejemplo CIFAR-10). Para esta transferibilidad se diseña un espacio de búsqueda tal que la complejidad de la estructura sea independiente de la profundidad de la red. De esta forma la red estará compuesta de bloques convolucionales con estructuras idénticas, los cuales se repetirán múltiples veces según la profundidad de la red requerida, que varían en los valores de sus pesos 2.21.

NASNet utiliza 2 tipos de bloques convolucionales: celdas normales, las cuales retornan mapas de características de la misma dimensión que la entrada y celdas de reducción, las cuales reducen el alto y ancho de los mapas de características por un factor de 2. Por lo tanto, NASNet está construida por conjuntos de N celdas normales separadas por celdas de reducción. El algoritmo buscará configuraciones 2.22 para las celdas utilizando un proceso de 5 pasos el cual será repetido $B$ veces. En el espacio de búsqueda las celdas reciben 2 estados ocultos anteriores y a partir de estos se generará un nuevo bloque interno. Este paso se iterara $B$ veces agregando los nuevos bloques como estados ocultos para los siguientes bloques. Los 5 pasos son:

- Seleccionar un estado oculto de los 2 estados ocultos anteriores o los estados ocultos agregados por los nuevos bloques creados.
- Seleccionar un segundo estado oculto de las mismas opciones que en el primer paso.
- Seleccionar una operación para aplicar al estado elegido en el paso 1. Estas operaciones son elegidas de un conjunto de operaciones predefinidas comúnmente utilizadas en redes convolucionales.
- Seleccionar una operación para aplicar al estado elegido en el paso 2. Selecciona del mismo conjunto de operaciones que el paso anterior.
- Seleccionar un método para combinar las salidas de las operaciones de los pasos 3 y 4. Este método puede ser la adición o la concatenación de los resultados.

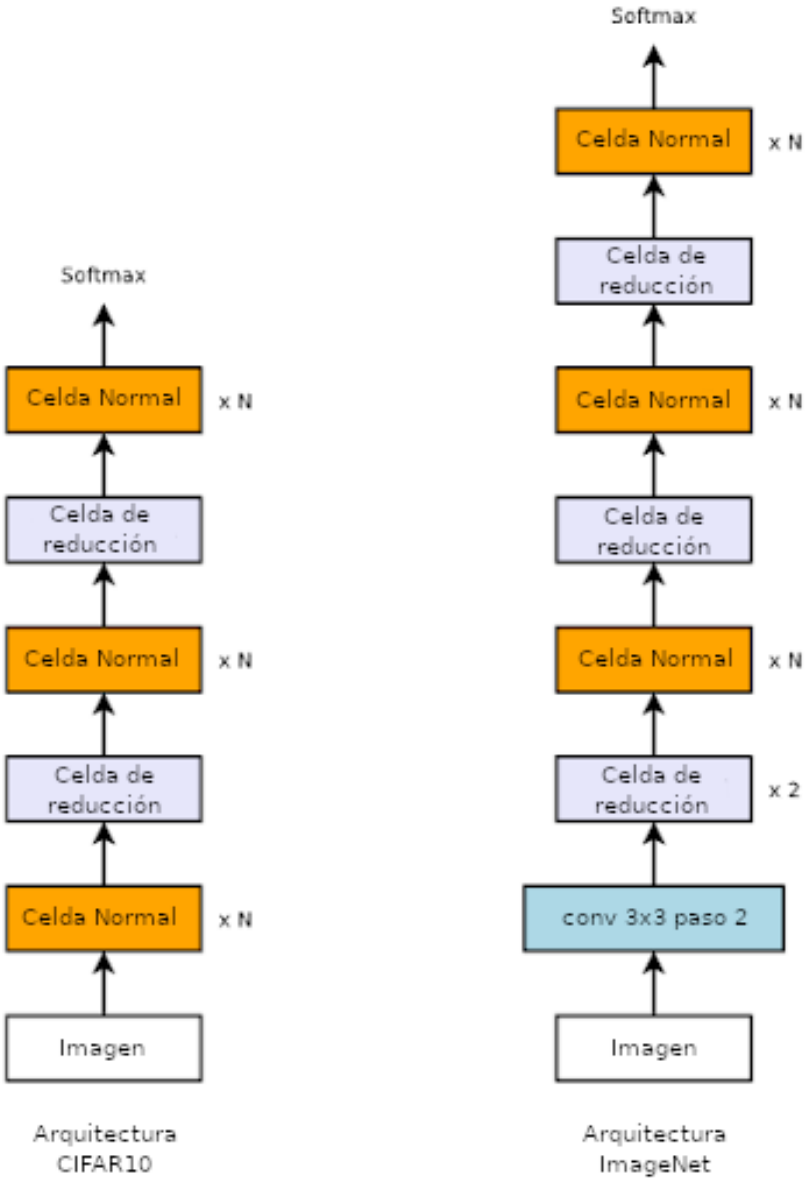



***Figura 2.21*** *Arquitectura NasNet para CIFAR10 e ImageNet.*

Utilizando este método se construyó NASNet, con la cual se alcanzaron muy buenos resultados tanto en CIFAR-10 como en ImageNet consiguiendo en este último conjunto de datos una exactitud top 5 de %96.2, similar a los resultados obtenidos por SENet el mismo año pero con aproximadamente un %60 de los parámetros utilizados en este último.

En artículos posteriores se realizaron cambios basándose en el funcionamiento de la búsqueda de arquitecturas de NASNet que permitieron obtener mejores tiempos de búsqueda de estructuras mediante búsqueda heurística, buscando nuevas estructuras de forma progresiva, de más simples a mas complejas [70]. También fueron desarrolladas búsquedas de arquitecturas para plataformas móviles (MnasNet) [118]

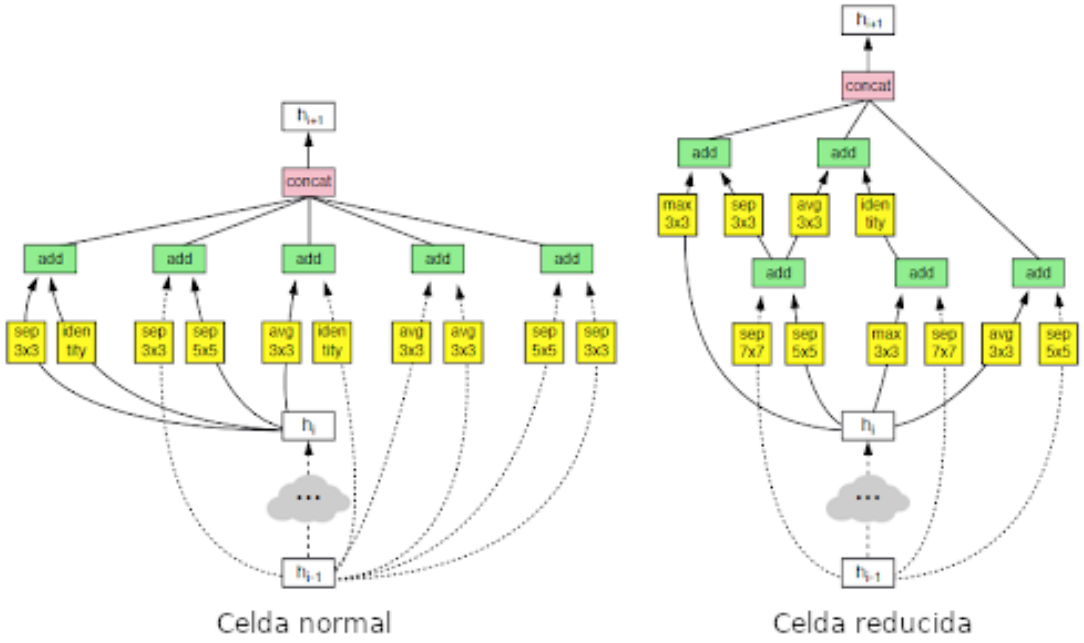



***Figura 2.22*** *Configuraciones alcanzadas en NASNet.*

que tomaron en cuenta no sólo la precisión de la arquitectura obtenible sino también el tiempo inferencia real ejecutando las redes neuronales hijas en dispositivos móviles.

Adicionalmente vale la pena mencionar EfficientNet [119]. Este modelo fue construido utilizando la técnica propuesta en el mismo articulo para el escalado de arquitecturas. EfficientNet fue construido a partir de una arquitectura creada de forma similar a MnasNet teniendo en cuenta tanto la exactitud de la arquitectura generada como sus FLOPS ya que esta arquitectura no fue diseñada para un hardware específico. Esta arquitectura luego fue escalada a la cantidad de FLOPS objetivo. Se utiliza un método de escalado compuesto según el cual el ancho, la profundidad y el tamaño de la entrada cambiarán en conjunto según unos coeficientes de escalado fijos definidos previamente. Este escalado le permite a EfficientNet obtener precisión estado del arte pero reduciendo en gran medida la cantidad de parámetros necesarios para alcanzar esta precisión. Actualmente los modelos que alcanzan mayor exactitud top 1 y top 5 en ImageNet están basados en EfficientNet [128][127].

# 3 Lengua de señas

La lengua de señas es un método de comunicación comúnmente utilizado por personas sordas para facilitar su comunicación. La lengua de señas es expresada mediante diversas articulaciones manuales y elementos no manuales. Existen una gran variedad de lenguas de señas diferentes y no mutuamente inteligibles (esto significa que personas que utilicen distintas lenguas de señas pueden no sé entenderse entre sí si no poseen un aprendizaje previo de la otra lengua), normalmente cada país posee su propia lengua de señas. De acuerdo a la Federación Mundial de Sordos existen más de 300 lenguas de señas utilizadas alrededor del mundo con más de 70 millones de sordos utilizándolas [19].

A lo largo de los años esta lengua ha ido evolucionando adaptándose a cada comunidad desarrollando sus propias señas y sintaxis. Por lo tanto la lengua de señas de cada comunidad ha crecido y obtenido sus propias características. Algunas ejemplos de lenguas de señas son la lengua de señas argentina (LSA), la lengua de señas americana (ASL) y la lengua de señas alemana (GSL).

La lengua de señas no es simplemente mímicas sino que poseen su propia gramática compleja de forma tal que sus señas no están siempre visualmente relacionadas a aquello que representan. Esta gramática permite a la lengua de señas expresar cualquier tipo de idea desde las más simples a las más abstractas.

Esta lengua es considerada una lengua natural al igual que otras lenguas naturales orales como el español. Esto es así ya que posee ciertas características que permiten que así sea clasificado. La lengua de señas posee una fonología abstracta compuesta de unidades representadas por diferentes posiciones de manos, orientaciones, localizaciones, movimiento y expresiones no manuales. Esta lengua posee su propia sintaxis con mecanismos propios de formación de palabras productivos.

Además de la forma de la mano y su posición, la lengua de señas utiliza el movimiento de cejas, boca, cabeza, hombros y vista para añadir diferentes significados a las señas. Un ejemplo de esto en ASL es el levantamiento de cejas para realizar preguntas abiertas y el las cejas fruncidas para indicar preguntas cuya respuesta es sí o no.

## 3.1. Descripción del problema

Los usuarios de la lengua de señas forman una minoría cultural, unida por el uso de un lenguaje y experiencias en común. Siendo la lengua de señas una de las partes más significativas de esta cultura es necesario que cualquier software utilizado para el reconocimiento de señas logre una precisión que esté a la altura de las necesidades

de estos grupos.

Han sido estudiados métodos intrusivos para la clasificación de señas que requieren que el usuario utilice guantes, ropa o sensores con el fin de reconocer sus señas. Con sus primeros trabajos datando a 1983, donde se desarrolló una patente describiendo un guante electrónico capaz de reconocer deletreo de dedos de ASL.

Actualmente la visión por computadora no intrusiva es el método más utilizado para afrontar este problema facilitando la usabilidad del usuario pero introduciendo las complejidades de la visión por computadora. El desarrollo del aprendizaje profundo de los últimos años ha permitido aumentar la precisión de las soluciones estado del arte. El problema con estos tipos de modelos es que necesitan grandes conjuntos de datos para generalizar correctamente los cuales no abundan en este área, lo que lleva a limitar la precisión de modelos de aprendizaje profundo sobre situaciones de la vida real donde la se introduce una gran variabilidad a los datos con, por ejemplo, cambios en la iluminación, posición, ángulo y ejecución de diferentes señas.

Existen dos posibles métodos para la clasificación de señas, estos son la clasificación estática o dinámica [71]. La clasificación estática consta de clasificar imágenes de señas de manos. Este tipo de clasificación de señas puede ser utilizado, por ejemplo, en la creación de diccionarios que permitan identificar señas individuales. La desventaja de este tipo de clasificación es que en conversaciones del mundo real utilizando lenguas de señas, estas señas forman diversas oraciones complejas alterando su significado según su contexto y solapando señas entre sí. La clasificación dinámica clasificara señas en videos aprovechando la dimensión temporal para lograr una clasificación en un ambiente más realista.

El problema afrontado en esta tesis es el de la clasificación de señas estáticas de diferentes lenguas de señas utilizando una baja cantidad de datos etiquetados. Se optó por la clasificación estática sobre la dinámica debido a la existencia de una gran variedad de conjuntos de datos de diferentes dimensiones de este tipo, lo que permite obtener una comprensión más amplia al estudiar el efecto de los diferentes tipos de bases de datos y sus tamaños. Posteriormente, los modelos desarrollados podrán ser utilizados en aplicaciones que busquen realizar una clasificación estática de señas con precisión alta.

Cada seña posee sus propias características distintivas, estas son la forma de las manos, posición, orientación y elementos no manuales como la forma del cuerpo o el gesto facial. La red neuronal debe ser capaz de utilizar todas estas características para una correcta clasificación de las señas. Otro efecto de estas características distintivas es la limitación de los cambios introducidos a las imágenes para realizar *data augmentation* ya que introducir demasiada rotación o cambio en el ratio podría tener el efecto de cambiar la clase de un gesto. Por otro lado, las imágenes de las señas naturalmente poseen muchas ambigüedades como la inclusión de movimiento

o de elementos no propios de la misma seña como lo sería un cambio momentáneo en la expresión facial.

Para lograr esta clasificación serán entrenados diferentes modelos de redes neuronales, de esta forma se podrá comparar el efecto del tamaño del conjunto de datos en la precisión del modelo.

## 3.2. Conjuntos de datos existentes

Los conjuntos de datos existentes en el área de la lengua de señas están compuestos por vídeos o imágenes de personas realizando señas. Estos datos pueden encontrarse en diferentes formatos, según el objetivo particular del conjunto de datos. Los datos en formato RGB tienen la ventaja de aprovechar la amplia disponibilidad de cámaras, requiriendo el mínimo esfuerzo de los usuarios a la hora de utilizar los programas que aprovechen estos datos [93][26][75]. Otro formato utilizado es el formato de *motion capture* [38], que captura el movimiento de las personas realizando las señas y lo traslada a un modelo digital. De esta forma se obtienen datos más detallados de los movimientos de las personas pudiendo alcanzar así un reconocimiento y generación de lengua de señas más preciso [29][48][73]. Adicionalmente también es posible utilizar cámaras de profundidad para capturar el posicionamiento 3d de la persona realizando las señas. Esto permite obtener datos más precisos de la realización de cada seña y realizar un reconocimiento más preciso ignorando variabilidades propias de las imágenes tomadas por cámaras estándar. El problema es que las cámaras de profundidad son menos comunes que las estándar, generando una dificultad extra a aquellas personas que quieran utilizar sistemas de reconocimiento de señas entrenados mediante datos capturados por cámaras de profundidad [105].

Estos datos pueden o no estar etiquetados diferenciando las señas, o indicando el comienzo y fin de cada una en video. Los datos etiquetados son necesarios para la realización de entrenamiento supervisado sobre modelos de aprendizaje de máquina. Existen múltiples dificultades a la hora de etiquetar las distintas señas, ya que este proceso requiere mucho tiempo y la participación de personas que posean conocimientos sobre el área para alcanzar conjuntos de datos correctamente etiquetados lo suficientemente grandes como para entrenar un modelo de aprendizaje profundo.

Algunos conjuntos de datos existentes utilizan a personas novatas en la lengua de señas para su interpretación, esto puede generar múltiples errores en la interpretación por simple inexperiencia, generando datos erróneos que limitan su aplicabilidad en el entrenamiento de modelos que luego serán utilizados por verdaderos usuarios de la lengua de seña. Otros conjuntos de datos utilizan datos obtenidos de internet, por ejemplo de YouTube, en los que la habilidad de los intérpretes para realizar las señas es desconocida. Este método permite obtener una gran cantidad de datos en ambiente real fácilmente con intérpretes variados realizando señas de forma natural

(sin los cambios intrínsecos incluidos en la interpretación individual de señas para obtener más claridad) pero estos datos obtenidos deben ser etiquetados, lo que conlleva mucho tiempo y por ende dinero.

Finalmente, otros conjuntos de datos utilizan intérpretes profesionales los cuales poseen mayores habilidades a la hora de realizar las señas correctamente pero que limitan la cantidad de datos y la variabilidad de estos, ya que obtener una gran cantidad de intérpretes profesionales para recolectar los datos de múltiples señas incluye un gran costo económico y de tiempo.

Adicionalmente existe el problema de la diversidad de las lenguas de señas y de las etiquetas utilizadas, lo que dificulta la posibilidad de combinar múltiples conjuntos de datos para aumentar la cantidad de datos disponibles. Esto genera la necesidad de crear conjuntos de datos de calidad lo suficientemente grandes para cada lengua de señas. Lo que aumenta en gran medida la complejidad el problema, principalmente para comunidades pequeñas o con bajos recursos.

Todo esto lleva a que la cantidad de datos disponibles en conjuntos de datos etiquetados de lengua de señas sea mucho menor al tamaño ideal para entrenar modelos de aprendizaje profundo. El éxito en el área del reconocimiento de voz, el cual se asemeja en complejidad al reconocimiento de señas, ha sido posible en parte gracias a la gran cantidad de datos disponibles, con conjuntos de datos de millones de palabras. En comparación, los conjuntos de datos de lengua de señas suelen poseer menos de 100.000 datos, siendo esto un gran obstáculo en el éxito del aprendizaje profundo sobre esta área 3.1.

| | *Lengua de señas* | *Habla* |
|---|---|---|
| Modalidad | visual-gestual | auditivo-oral |
| Serialidad | baja | alta |
| Iconicidad | alta | baja |
| Tamaño típico de los conjuntos articulados | <100.000 señas | 5 millones de palabras |
| Tamaño típico de los conjuntos anotados | <100.000 señas | 1 billón de palabras |
| Tamaño típico del vocabulario | 1.500 señas | 300.000 palabras |
| Qué es modelado | 1.500 señas completas | 1.500 tri-fonemas |
| Cantidad de interpretes típica | 10 | 1.000 |

***Cuadro 3.1*** *Comparación de los conjuntos de datos de la lengua de seña contra los de habla. Los conjuntos articulados refiere a conjuntos individuales generados, mientras que los conjuntos anotados refiere a conjuntos generados a partir de la combinación de múltiples conjuntos de datos que utilizan el mismo sistema de anotación.*

La variabilidad de los datos es limitada al ser poca la cantidad de datos y/o in-

***Figura 3.1*** *Imágenes de señas de manos extraídas del conjunto de datos LSA16.*

térpretes en los conjuntos de datos. Esto genera que sea difícil de aplicar los modelos entrenados en situaciones reales, ya que no se dispondrá de la suficiente generalización para los distintos intérpretes que fueran a utilizarlo, los cuales podrían poseer distintas edades, etnias, forma de cuerpo o manos entre otras cosas.

### 3.2.1. LSA64 / LSA16

Como fue mencionado anteriormente, existen distintas lenguas de señas para cada grupo cultural. Esto hace necesaria la creación de conjuntos de datos específicos para cada una de estas lenguas, ya que las lenguas de señas no son mutuamente inteligibles. En el caso de la Lengua de Señas Argentinas (LSA) fueron desarrollados los conjuntos de datos LSA16 y LSA64 [93][92] con el fin de poseer conjuntos de datos para el entrenamiento de modelos de aprendizaje de máquina en dicha lengua de señas 3.1.

LSA16 contiene imágenes estáticas de 16 formas de manos de la LSA, cada una de estas formas de manos fue interpretada 5 veces por 10 sujetos diferentes formando un total de 800 imágenes de tamaño $32 \times 32$. Los sujetos llevan guantes de colores y ropas oscuras sobre un fondo blanco, esto con el objetivo de facilitar la segmentación de las manos. El conjunto de datos se encuentra balanceado con 50 imágenes por clase, lo que facilita el entrenamiento al evitar que existan clases minoritarias difíciles de aprender por el modelo. El conjunto de datos también posee las manos segmentadas donde cada imagen solo posee una mano centrada y separada del fondo, lo que simplifica el problema reduciendo el tamaño y cantidad de información que el modelo debe aprender.

Por otro lado LSA64 contiene videos de señas con el fin de producir un diccionario para la LSA y de ser utilizado en el reconocimiento de señales dinámicas. Este conjunto de datos cuenta con 3200 videos donde 10 intérpretes ejecutaron 5 repeticiones de 64 diferentes tipos de señas. Los sujetos llevan guantes de colores diferentes para cada mano y ropas oscuras con fondo blanco. Las clases se encuentran balanceadas con 50 videos por clase. Este conjunto de datos también provee una versión preprocesada de los videos, donde se cuenta con la posición de la cabeza y las manos de los intérpretes extraídas de cada cuadro. Adicionalmente se cuenta con las imágenes de las manos extraídas de cada cuadro y puestas sobre un fondo

***Figura 3.2*** *Imágenes de señas de manos extraídas del conjunto de datos RWTH-PHOENIX-Weather 2014 MS Handshapes.*

negro. De esta forma es posible ahorrar *overhead* en entrenamiento en caso de querer utilizar las manos segmentadas.

Ambos conjuntos de datos fueron interpretados por personas no expertas en la lengua de señas en ambiente de laboratorio. Esto podría limitar su aplicabilidad a situaciones de la vida real donde los intérpretes nativos pueden tener una forma ligeramente diferentes de ejecutar las señas.

Actualmente estos conjuntos de datos son los únicos existentes sobre la LSA con tamaño suficiente para el entrenamiento de modelos de aprendizaje de máquina.

### 3.2.2. RWTH-PHOENIX-Weather

Este conjunto de datos [26] está compuesto por una selección de 3359 imágenes etiquetadas de manos de tamaño $132 \times 92$ recortadas de videos de intérpretes de lengua de seña de la estación de tv pública alemana PHOENIX. Los intérpretes visten ropa oscura enfrente de un fondo gris. Muchas imágenes del conjunto de datos poseen un desenfoque de movimiento significantes, otras poseen ambas manos del intérprete y las manos no siempre se encuentran perfectamente centradas 3.2.

El conjunto de datos posee un total de 45 señas de manos diferentes. Debido a la fuente de la que fueron tomadas las imágenes, existe un gran desbalance entre clases, con clases de tan solo 1 elemento mientras que otras poseen hasta 529 elementos. Este desbalance deberá ser tratado para poder realizar un entrenamiento efectivo de un modelo de aprendizaje de máquina.

Este conjunto de datos también cuenta con 17 Gb de datos de entrenamiento no etiquetados. Estos datos pueden ser utilizados con el fin de realizar entrenamiento semi-supervisado o débilmente supervisado.

### 3.2.3. CIARP

Este conjunto de datos [24] contiene 6000 imágenes de tamaño $38 \times 38$ tomadas mediante una cámara monocromática. Las imágenes fueron etiquetadas manualmente y corresponden a 10 clases de gestos de manos. Las manos se encuentran centradas y fueron segmentadas del fondo, el cual fue reemplazado por un fondo negro. El pequeño tamaño de las imágenes y la baja cantidad de clases le otorga a

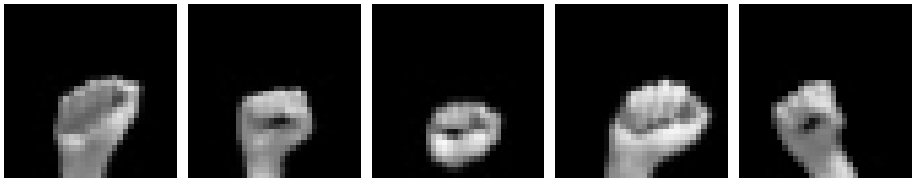


***Figura 3.3*** *Imágenes de señas de manos extraídas del conjunto de datos CIARP.*

este conjunto de datos una complejidad menor a la vista en los conjuntos LSA16 y RWTH-PHOENIX-weather 2014 3.3.

Las clases del conjunto de datos corresponden a formas de manos no basadas en una lengua de señas, pero aun así son lo suficientemente similares manteniendo la complejidad del problema como para que la comparación aún sea válida.

### 3.2.4. Purdue RVL-SLLL American Sign Language Database

Para el desarrollo de sistemas de reconocimiento automático de la lengua de señas americana (ASL) fue desarrollado este conjunto de datos [75], el cual cuenta con 2576 videos de 14 intérpretes nativos de ASL. Fueron utilizadas cámaras a color para capturar los videos en resolución de $640 \times 480$.

Los videos fueron capturados con iluminación controlada para garantizar su uniformidad en todos los sujetos. Se utilizaron 2 tipos diferentes de iluminación, iluminación difusa para reducir las sombras e iluminación potente directa para aumentar el contraste. De esta forma se puede utilizar la ventaja de cada tipo de iluminación, con la iluminación difusa reduciendo el ruido que agrega la existencia de sombras en los cuadros mientras que con la iluminación potente directa se potencia el contraste facilitando el trabajo de segmentar formas a los algoritmos de visión de máquina.

El conjunto de datos esta posee 2 partes, una parte compuesta por primitivas de movimientos y formas de manos y otra parte compuesta por grabaciones de sujetos realizando las señas de diversas frases. La primera parte contiene 39 primitivas de movimiento comúnmente encontradas en ASL junto a 62 formas de manos 3.4. Para otorgarle contexto de articulación a estas formas de manos cada sujeto interpretó 2 palabras que utilicen las formas de manos. Esto permite facilitar la aplicación de un modelo entrenado con este conjunto de datos sobre el reconocimiento de formas de manos en palabras o frases ya que el modelo tendrá una mejor generalización de cada forma de mano superior a la que obtendría de una imagen estática.

Este conjunto de datos posee una buena cantidad de formas de manos junto a buena calidad de imagen y cantidad de videos volviéndolo muy útil para el entrenamiento de un modelo reconocedor de ASL. El método de selección de participantes


***Figura 3.4*** *Imágenes de señas de manos extraídas del conjunto de datos Purdue RVL-SLLL American Sign Language Database.*

también es correcto al seleccionar participantes que hablan ASL de forma nativa, lo cual es un gran punto positivo porque de esta forma se reducen los errores de ejecución y se obtiene una ejecución natural de las señas. Al poseer 14 intérpretes también se obtiene una buena generalización al ser diferentes personas con su propia formas de ejecutar cada seña. Aun así la cantidad de participantes y de datos recolectados es opacada frente a la cantidad de participantes y datos obtenibles en otros problemas de dificultad similar como el reconocimiento de voz donde existen conjuntos de datos como Google Audioset el cual posee 2.084.320 de clips de audio de una cantidad de sujetos mucho mayor.

## 3.3. Bases de datos con pocos datos etiquetados

El aprendizaje profundo ha demostrado alcanzar una gran precisión en tareas de reconocimiento de gestos [46] y señas [8][121] pero esta precisión es demostrada sobre bases de datos pequeñas de 1500 o menos imágenes por imágenes por clase las cuales suelen estar tomadas de una pequeña cantidad de personas, usualmente menos de 10. Por otro lado, las imágenes suelen estar tomadas en condiciones uniformes, lo cual facilita su reconocimiento. Esto es necesario para lograr un buen reconocimiento utilizando la limitada cantidad de datos disponibles. Dada la complejidad del problema debida la gran cantidad de señas, las diferencias en la forma de realizarlas y en las personas que las realizan es necesario una gran cantidad de datos o un tratamiento especial del problema para lograr una buena precisión [23].

Este problema de poseer menos datos etiquetados de los que requeriría el problema para alcanzar una alta precisión existe en una gran variedad de áreas. Por ejemplo, en el área de la medicina [101][115] donde existe una gran complejidad en sus problemas pero las bases de datos disponibles se encuentran sin etiquetar o mal etiquetadas. Otro área en el que se observa el problema de trabajar con datos etiquetados limitados es en el área de análisis de sentimiento o perspectiva [129][63] donde la cantidad de datos etiquetados es eclipsada por la cantidad de datos no etiquetados existentes, los cuales aumentan constantemente.

Las redes neuronales artificiales buscan modelar la distribución inherente de los elementos del conjunto de datos en el que son entrenadas. El problema existe cuando

esta distribución modelada sobre los datos de entrenamiento no refleja correctamente la totalidad de los datos por no poseer suficiente información para encontrar los patrones distintivos de los elementos o poseer información que se aleja de la distribución de los datos totales.

Para acercar la distribución del modelo a los datos totales es necesario poseer un conjunto de datos de entrenamiento que refleje correctamente la distribución de los datos totales. Mediante una selección de un conjunto de datos de entrenamiento que caracterice correctamente al conjunto de datos totales [86] y aumentando la cantidad de datos de entrenamiento hasta que estos abarquen una cantidad de elementos suficientemente representativa del total es posible acercarse a esta distribución. Entre más datos se posean será más sencillo distinguir aquellos casos difusos en los que el modelo sufra problemas para distinguir las diferentes clases de elementos.

Como fue mencionado anteriormente, los modelos de aprendizaje profundo son entrenados minimizando el error hasta converger. Si la cantidad de datos disponibles no es suficiente el modelo podría converger en un mínimo local, y de esta forma, generalizando incorrectamente. A medida que se aumenta el tamaño de la red, y con ello la cantidad de parámetros a entrenar, este problema se acentúa ya que aumenta la dimensionalidad de la función de costo y se facilita aún más caer en mínimos locales.

Conseguir aumentar la cantidad de datos etiquetados suele ser una tarea difícil, ya que esto requiere de grandes cantidades de dinero y tiempo. Debido a esto se han desarrollado múltiples técnicas para permitir obtener mejor precisión utilizando una cantidad baja de datos etiquetados. Algunas de estas técnicas creadas para afrontar el problema del entrenamiento de modelos con datos etiquetados limitados serán explicadas más detalladamente en el siguiente capítulo.

# 4 Trabajando con pocos datos etiquetados

La ausencia de los datos etiquetados necesarios para entrenar de forma convencional una red neuronal existente en muchos problemas llevó al desarrollo de nuevas tecnologías para trabajar con pocos datos etiquetados. A pesar del aumento en la cantidad y la facilidad de acceso a conjuntos de datos en los últimos años [80][1] este problema aún persiste. Adicionalmente, esta ausencia de datos genera un dilema ético al afectar principalmente a minorías y países que no cuentan con los recursos necesarios para recolectar y etiquetar los datos.

Las tecnologías desarrolladas para entrenar modelos con una cantidad limitada de datos etiquetados suelen inclinarse a 2 categorías:

1. concentrándose en los datos utilizados en el entrenamiento
    - Data augmentation
    - Weak supervisión
    - Active learning
2. concentrándose en el método de entrenamiento y el modelo
    - Transfer learning
    - Few-shot learning
    - Aprendizaje semi-supervisado
    - Robust learning

Muchas de estas tecnologías pueden utilizarse en conjunto para intentar alcanzar mejores resultados [69][65] y algunas de estas técnicas no son muy difíciles de aplicar a diversas tareas que requieran entrenar redes utilizando pocos datos etiquetados lo que incentiva aún más su utilización.

## 4.1. Data augmentation

*Data augmentation* de imágenes es un conjunto de técnicas que buscan aumentar artificialmente la cantidad de datos que pueden ser obtenidos de las imágenes disponibles en el conjunto de datos. Estas técnicas modifican las imágenes del conjunto de datos creando nueva información que podrá ser utilizada como entrada para el modelo. Esta es una de las técnicas más utilizadas debido a la facilidad de su implementación y la mejora en los resultados que otorga incluso para conjuntos de datos grandes.

Usando *data augmentation* es posible trabajar con menos imágenes teniendo una varianza menor, lo que baja las posibilidades de sobreajuste de la red, y permite obtener un entendimiento mayor de las características más importantes que conforman cada clase del conjunto de datos. Varianza es definida por Abhijit Ghatak [28] como la tendencia al aprendizaje de características al azar desconsiderado la señal real. Tener una varianza alta implica que el modelo está aprendiendo del ruido en los datos y posiblemente sobreajustando. Mediante *data augmentation* es posible lograr un resultado similar a redes entrenadas utilizando una cantidad mayor de datos pero necesitando una cantidad menor de datos para alcanzar estos resultados [125].

Existe una gran variedad de funciones aplicables a imágenes empleadas para realizar *data augmentation*. Estas funciones modifican las imágenes de forma tal que se reduzcan las invarianzas propias del conjunto de datos. Entre las funciones existentes comúnmente utilizadas se puede encontrar rotación, traslación, inclinación, escalado, distorsionado, recortado y varias modificaciones en los colores. Por ejemplo la invarianza de rotación existente en conjuntos de datos de señas de manos, que podrían generar problemas a la hora de la generalización para usuarios de lengua de señas zurdos. Normalmente estas funciones son seleccionadas utilizando conocimiento experto y requiriendo tiempo para encontrar las funciones óptimas para un determinado conjunto de datos.

En la CVPR 2019 fue presentado AutoAugment [16] el cual, de forma similar a como las redes NAS buscan las mejores arquitecturas de redes neuronales, busca las funciones óptimas para aplicar a los datos de entrenamiento y alcanzar la mayor precisión. Durante el entrenamiento el controlador (una RNN al igual que en NAS) tomará políticas de *data augmentation* S, las cuales estarán compuestas de sub-políticas conformadas por 2 operaciones en imágenes junto a sus respectivas probabilidades de ser aplicadas y sus magnitudes. Utilizando esta política S se entrena una red hija aplicando las sub-politicas de forma aleatoria en cada elemento de los *mini batchs*. Las operaciones de las sub-políticas se aplicarán en el orden especificado. Como resultado del entrenamiento se obtendrá la exactitud sobre el conjunto de datos de validación, y con esta exactitud se actualiza el controlador mediante aprendizaje por refuerzo.  4.1.

También es posible la utilización de *generative adversarial networks* (GAN) [31] para realizar *data augmentation*. Las GAN realizan entrenamiento no supervisado sobre el conjunto de datos para extraer las distribuciones de estos de forma tal de poder generar datos sintéticos a partir de ruido. Este tipo de red está compuesta por 2 partes: un generador y un discriminador. Comúnmente el generador tomará como entrada un mapa de características generado a partir de ruido y generará datos sintéticos que serán utilizada como entrada para el discriminador, el cual intentara discernir si estos datos son reales o generados. De esta forma, a medida que se

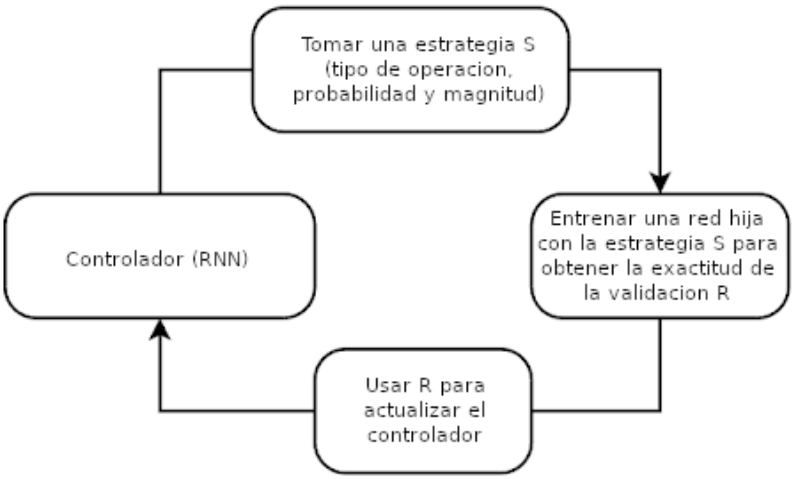



***Figura 4.1*** *Proceso utilizado por AutoAugment para encontrar estrategias óptimas de* data augmentation.

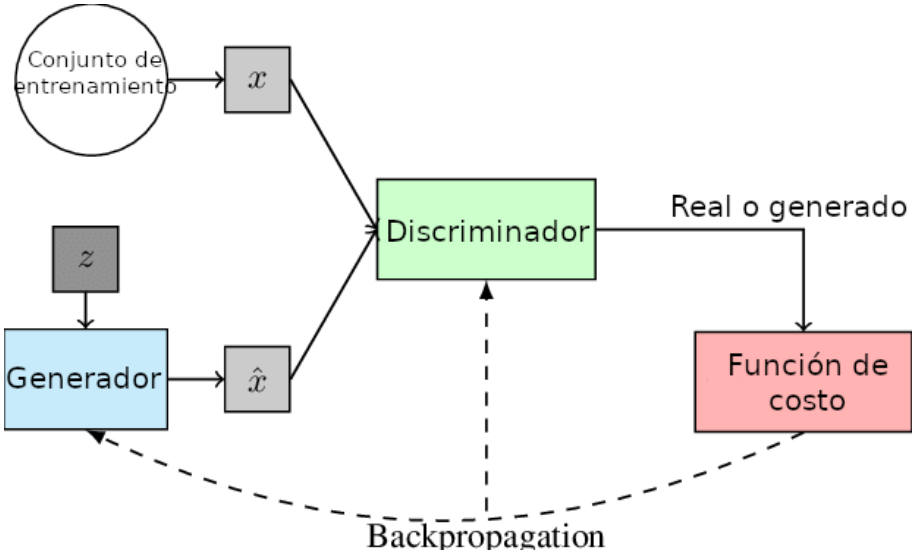



***Figura 4.2*** *Proceso realizado por GAN.*

entrena la red el generador se volverá mejor generando datos sintéticos que parecen reales mientras que el discriminador aprenderá a discriminarlos de forma más exacta 4.2.

La capacidad de las GAN de generar nuevos datos sintéticos ha demostrado ser útil a la hora de reducir la invarianza y el desbalance de los datos de un conjunto de datos. Un ejemplo de una red GAN utilizada en *data augmentation* es *Data Augmentation GAN* (DAGAN) [2]. Este modelo generativo es entrenado en un dominio fuente para luego poder aplicar este entrenamiento en datos de un dominio objetivo con una baja cantidad de datos. La forma en la que aplica este entrenamiento es aprendiendo las transformaciones válidas para la generación de datos propias del

dominio fuente, el cual tendrá los datos suficientes para realizar un entrenamiento correcto de la GAN, y luego aplicando estas transformaciones a datos del dominio objetivo, de forma tal que los datos del dominio objetivo puedan ser transformados manteniéndose en la misma clase y así añadiendo varianza al conjunto de datos. DA-GAN toma como entrada un elemento del conjunto de datos del dominio objetivo, le agrega ruido y los utiliza para generar un nuevo dato sintético de la misma clase que la entrada. Posteriormente estos datos podrán ser utilizados para entrenar una red neuronal para clasificación.

Otro tipo de utilización de GAN para *data augmentation* es la implementación utilizada en el articulo titulado *Data Augmentation Using GANs* [97] en el cual se utiliza el modelo generativo para crear nuevos datos a partir de ruido y luego utiliza estos nuevos datos para entrenar otros modelos. Esta generación de datos tiene la ventaja de permitir trabajar con modelos desbalanceados o crear conjuntos de datos enteramente sintéticos, lo ultimo podría ser útil en casos en los que el conjunto de datos original no se desea hacer público, como pasa en algunos conjuntos de datos con información sensible. En los experimentos realizados entrenando un modelo utilizando los datos generados se llegó a obtener resultado similares o incluso mayores a modelos entrenados utilizando los datos originales. Esto demuestra el potencial de la aplicación de GAN para el data augmentation.

## 4.2. Weak supervisión

Obtener grandes conjuntos de datos para entrenar redes neuronales es una tarea difícil, que lleva tiempo y que requiere de grandes sumas de dinero. Principalmente en áreas donde el etiquetamiento de los datos requiere del conocimiento de un experto el tiempo y dinero necesario para realizar este etiquetamiento sobre una amplia cantidad de datos puede escalar rápidamente. Una alternativa a la utilización de datos etiquetados con conocimiento experto es *weak supervision*. Esta es una rama del aprendizaje de máquina donde son utilizados datos de una o más distribuciones condicionales con ruido para entrenar modelos.

*Weak supervision* permite adquirir datos etiquetados baratos y de forma eficiente otorgando supervisión de alto nivel y menos precisa (por ejemplo mediante heurísticas o con distribución de etiquetas conocida), supervisión barata y de menor calidad (por ejemplo *crowdsourcing*) o aprovechando fuentes ya existentes (por ejemplo modelos pre-entrenados o bases de conocimiento) [88].

De esta forma los métodos de *data augmentation* entran dentro de lo que se podría definir como *weak supervision* al extender la cobertura de la distribución de datos etiquetados con los datos transformados, los datos transformados etiquetados no son discriminados de los datos originales.

Otra forma de *weak supervision* muy utilizada es *weak labeling* en el cual se les

asignan etiquetas con ruido débiles a un conjunto de datos etiquetados. Existen múltiples formas en las que pueden ser generadas dichas etiquetas, entre estas formas destacan *crowdsourcing*, reglas heurísticas y *distant supervision*. Todos estos métodos asignan etiquetas a los datos pero introduciendo ruido al poseer estas etiquetas menor precisión y calidad que las realizadas por expertos. También es posible tomar los datos etiquetados de otros conjuntos de datos, para esto se puede comparar la información extraíble de estos para elegir la mejor fuente de la cual tomar los datos con ruido que se utilizaran para el entrenamiento del modelo [69].

*Crowdsourcing* es un método común para obtener etiquetas de conjuntos de datos en el cual estas son obtenidas de grupos de personas (*crowd*) cuya habilidad y dedicación a la hora de etiquetar varía ampliamente. Esto genera que las etiquetas obtenidas posean distintas distribuciones dependiendo de la persona que realizó el etiquetado. Una forma de solventar este ruido introducido a las etiquetas de los elementos del conjunto es utilizando un puntaje de confianza, el cual indicará cuán confiable es que el etiquetado del elemento provisto por el individuo sea el mismo que el etiquetado real de dicho elemento. Entre las formas de obtener la confianza esta puede ser obtenida de manera automática utilizando diversos algoritmos [57] o utilizando un nivel de confianza provisto por el mismo individuo que realiza el etiquetado [83]. Otra forma de aumentar la calidad de las etiquetas obtenidas es mediante la colaboración con expertos en el dominio [64] en la cual estos corregirán los casos de altos complejidad en datos etiquetados utilizando *crowdsourcing* de menor calidad.

Otro método para realizar *weak supervision* comúnmente usado es el poseer acceso directo a la distribución de probabilidad de los datos. Esto puede ser realizado mediante la utilización de clasificadores débiles o con bias [77] entrenados en otros conjuntos de datos. Estos clasificadores se ejecutarán sobre el conjunto de datos a etiquetar generando nuevas etiquetas débiles cuya precisión estará ligada a la precisión del modelo clasificador.

De cualquier forma en las que se obtengan los datos mediante *weak supervision*, el modelo deberá tener en cuenta la calidad de estos para realizar un entrenamiento correcto. Existen múltiples formas de supervisar la calidad de los datos obtenidos, dependiendo o no del método utilizado para su obtención, por ejemplo se pueden eliminar los elementos etiquetados de manera automática cuyo nivel de confianza se encuentre por debajo de una constante predefinida. También existen formas de volver a los modelos más resistentes al ruido modificando el peso que tienen los elementos según cuán confiable son estos comparados a un conjunto de datos confiable de referencia [57].

## 4.3. Active learning

Tradicionalmente en el aprendizaje de máquina el modelo aprende de los datos pasivamente sin interactuar directamente en cómo estos están formados. Esto puede llevar a que ocurran redundancias en los datos de entrenamiento, los datos no se diferencian entre sí lo suficiente como para extraer información que ayude a discriminar entre diferentes clases. A medida que el modelo estudiante $s$ se entrena, este se volverá mejor en lograr ajustar el mapa subyacente $X \mapsto Y : s(x) = y$ siendo $x$ una muestra del espacio de entrada $X$ e $y$ la salida esperada para dicha entrada en el espacio de salida $Y$. Si se agregan datos en las áreas en las que el estudiante devuelve las salidas esperadas estos datos no tendrán impacto en el entrenamiento del modelo, en cambio si estos datos se agregan en valores del espacio de entrada en los que el estudiante y el maestro difieren entonces el estudiante obtendrá información útil y logrará realizar un mejor ajuste del mapeo óptimo. Si se entrena el modelo a partir de datos etiquetados tomados al azar serán necesarios muchos más datos y tiempo de entrenamiento para alcanzar una generalización correcta ya que los datos que realmente aportan información útil se encontrarán más dispersos en el conjunto de datos. Adicionalmente, si se posee una baja cantidad de datos este problema se intensifica ya que no se podrá alcanzar una buena generalización al no poseer suficiente información de los límites entre las diferentes salidas esperadas.

*Active learning* busca solucionar estos problemas introduciendo al modelo como estudiante el cual tomará un rol activo en la selección de los elementos con los cuales se entrenará mediante una comunicación con el maestro quien le proveerá de datos [34]. El estudiante elegirá inteligentemente cuales son aquellas áreas en las que necesita de datos para lograr un mejor ajuste del mapa óptimo $s(x)$ 4.3. Existen variaciones de cómo esto es logrado, 2 de estas variaciones son: *active sampling* y *active data selection*.

En *active data selection* son seleccionados subconjuntos del conjunto de datos de entrenamiento con el fin de aumentar la eficiencia y fiabilidad de la red al poseer un pequeño subconjunto que posea la información necesaria para lograr generalizar sin sobreajuste a los datos de entrenamiento. *Active data selection* es útil cuando se posee un subconjunto con una gran cantidad de elementos y se busca obtener un entrenamiento rápido y efectivo sobre estos datos.

En cambio, en *active sampling* el estudiante seleccionará elementos no etiquetados del conjunto de datos que considera útiles debido a su información extraíble o reducción en el error de generalización [109], o por encontrarse al borde de los límites de clasificación [44][55]. Esta variación de *active learning* es especialmente útil en problemas donde no se posean los suficientes datos etiquetados ya que permitirá elegir iterativamente muestras de datos para etiquetar, lo que es conveniente ya que

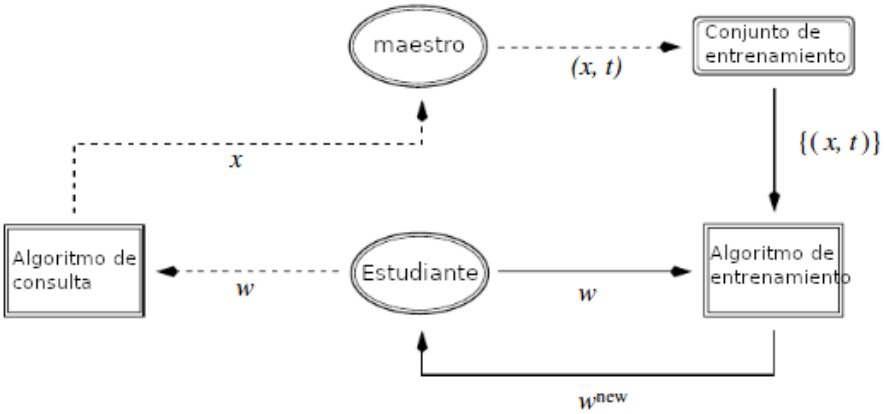



***Figura 4.3*** *Proceso de Active Learning.*

permite ajustar el entrenamiento al tiempo o dinero disponible etiquetando aquellos datos que realmente aporten información útil en el entrenamiento del modelo.

## 4.4. Transfer learning

Los modelos de aprendizaje de máquina han sido utilizados en una gran cantidad de tareas de clasificación de imágenes sobre grandes conjuntos de datos de diferentes tipos obteniendo muy buenos resultados [119][137]. El problema surge cuando los conjuntos de datos no son lo suficientemente grandes como para realizar buen uso de estos modelos, ya que su entrenamiento sin una correcta adaptación a la problemática llevaría a una pobre generalización y sobreajuste. *Transfer learning* otorga la posibilidad de utilizar estos modelos entrenados en grandes conjuntos de datos en problemas relacionados reentrenando las redes completa [116] o parcialmente en las ultimas capas *fully connected* que realizan la clasificación [90]. En el ICLR 2019 se presentó *SOurce SELEction for Target Optimization* (SOSELETO), este método permite optimizar la selección de conjuntos de datos de origen teniendo en cuenta la cantidad de información útil que estos presenten al ser aplicados para realizar *transfer learning* a otros conjuntos de datos.

Utilizando *transfer learning* es posible reducir en gran medida el tiempo de entrenamiento ya que el modelo transferido poseerá desde un principio sus pesos configurados de forma tal que el modelo realice una extracción genética de características al menos parcialmente relevantes. El modelo posteriormente será entrenado para ajustarse al conjunto de datos objetivo mediante *fine-tuning* o el reentrenamiento de la red completa. Las redes convolucionales pueden ser separadas en 2 partes, un núcleo convolucional donde se realiza la extracción de características de la entrada mediante un conjunto de capas convolucionales y la capa de salida compuesta de una o más capas *fully connected*. Normalmente en *transfer learning* se entrena solo

la capa de salida dejando intacto el núcleo convolucional ya que esto otorga un entrenamiento más rápido y menos costoso además de evitar sobreajuste en caso de que el conjunto de datos objetivo tenga poca cantidad de datos de entrenamiento. También se opta por entrenar solo la capa de salida cuando los conjuntos de datos objetivo y origen son lo suficientemente similares, como es en el caso de que el conjunto de dato objetivo sea un subconjunto del conjunto de datos de origen. A este reentrenamiento de solo la capa de salida se lo conoce como *fine-tuning*. Por otro lado, si se desea obtener un modelo que ajuste mejor a los datos del conjunto de datos objetivo y este es lo suficientemente grande y diferente es posible entrenar el modelo transferido por completo. Este reentrenamiento parte con los pesos ya con valores cercanos a los óptimos, por lo que se obtendrá una convergencia más rápidamente y se evitará caer en óptimos locales. Existen estudios [66] que indican que mantener un bias hacia el modelo original podría traer beneficios a la hora del reentrenamiento del modelo evitando perder parte del conocimiento inicial de la red.

Transferir modelos entrenados en grandes conjuntos de datos para ser utilizados en conjuntos de datos con menor cantidad de datos permite una mejor generalización del problema ya que el modelo transferido habrá aprendido a extraer características más generales del modelo original [18]. De esta forma solo una pequeña cantidad de datos serán necesarios para reentrenar el modelo ya que solo se requiere adaptar el modelo al problema en específico y los conocimientos para la extracción de características ya existen en el núcleo convolucional.

Se han demostrado en numerosas ocasiones los beneficios de utilizar *transfer learning* [22][90][102] siempre y cuando se cuente con un modelo pre-entrenado en un gran conjunto de datos es posible reutilizarlo para disminuir en gran medida los tiempos y costos de entrenamiento. De esta forma se vuelve más accesible la posibilidad de utilizar grandes modelos con capacidades de extraer características de mucha complejidad aprovechando estos modelos ya entrenados y con gran capacidad de generalización.

## 4.5. Few-shot learning

Normalmente los seres humanos son capaces de aprender conceptos con solo una mirada incluso cuando las personas en cuestión son niños [61]. Por ejemplo un niño que ve una jirafa sabría distinguirla de un perro inmediatamente con solo una mirada. De poder realizar este aprendizaje con un modelo de aprendizaje de maquina podríamos reducir en gran medida la necesidad de grandes bases de datos y grandes tiempos de entrenamiento, ya que actualmente las arquitecturas más utilizadas para aprendizaje profundo requieren de grandes cantidades de datos para un entrenamiento óptimo.

*Few-shot learning* afronta este problema de lograr generalizar conceptos a partir

de unas pocas muestras. Los seres humanos al aprender un nuevo concepto contamos con todos los conocimientos previos a éste. Similarmente *few-shot learning* toma un conjunto de datos base $D_base = (x_1, y_1), ..., (x_N, y_N)$ que contiene $N$ elementos de de $A$ clases base donde $x_i$ es un elemento del conjunto de datos e $y_i \in 1, ..., A$ la etiqueta asociada a dicho elemento. Posteriormente será presentado un nuevo conjunto de datos de soporte $D_support = (x_1, y_1), ..., (x_K, y_K)$ con $K$ elementos pertenecientes a $C$ nuevas clases. El objetivo será poder entrenar un modelo capaz de reconocer correctamente estas $C$ nuevas clases. Este tipo de aprendizaje es conocido como K-learn-C-shot [124]. Por ejemplo, *one-shot learning* se refiere a los problemas donde se tiene un conjunto de datos de soporte donde solo se posee un elemento por clase mientras que *zero-shot learning* intenta entrenar modelos capaces de reconocer objetos cuyas instancias pueden no haber sido vistas durante el entrenamiento [126].

Han sido desarrolladas una amplia variedad de técnicas para afrontar problemas de *few-shot learning* alcanzando buenos resultados [124][53][96][25][89][107][108]. Estas técnicas son cuentan con una multitud de conjuntos de datos sobre las que son aplicadas y que permite una comparación directa en la performance de los modelos entrenados con cada técnica. Entre estos conjuntos de datos algunos de los más utilizados son Omniglot [61], CIFAR-10 [58] y mini-imagenet [123]. Omniglot es un conjunto de datos compuesto de 1623 caracteres escritos a mano pertenecientes a 50 sistemas de escritura, similar al conjunto de datos MNIST utilizado en aprendizaje profundo convencional este conjunto de datos permite una comparación sencilla de distintos modelos. CIFAR-10 consiste de 60000 imágenes a color de $32 \times 32$ pertenecientes a 10 clases. Por otro lado, mini-imagenet posee un total de 60000 imágenes a color de $84 \times 84$ distribuidas entre 100 clases con 600 elementos por clase extraídas del conjunto de datos ImageNet.

Normalmente estas técnicas de *few-shot learning* pueden ser clasificadas en 4 categorías no mutuamente excluyentes [99]: metric learning[107][124], meta learning [96][25][89][112], data augmentation [133] y semantic learning [100].

*Metric learning* busca calcular mapeos del espacio empotrado a las imágenes del conjunto de entrenamiento, de forma tal que aquellas imágenes de la misma clase se encuentren más cerca entre sí y aquellas de otras clases más alejadas siguiendo la función utilizada de distancia 4.4. Algunos ejemplos de este tipo de modelos son las Matching Networks [123], las Prototypical Networks [107] y, más recientemente, SimpleShot [124]. Este tipo de modelos son sencillos y eficientes comparados a otros modelos de *few-shot learning*. Adicionalmente estos modelos alcanzan performance estado del arte llegando a una exactitud de %81.50 con SimpleShot en Mini-ImageNet, uno de los conjuntos de datos de referencia más complejos a la hora de la prueba de algoritmos de *few-shot learning*.

Por otro lado, *meta learning* busca entrenar sus modelos en una variedad de

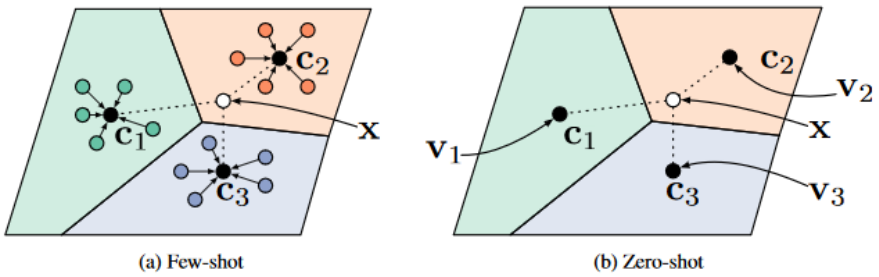



***Figura 4.4*** *Redes prototípicas en escenarios one-shot y zero-shot. Los puntos embebidos de consulta son clasificados según su distancia a los prototipos de clase.*

tareas diferentes de la tarea objetivo para obtener una mayor generalización requiriendo solo una pequeña cantidad de elementos en esta última. Similar a como un humano utiliza la experiencia previa de su vida para facilitar el aprendizaje de nuevas tareas se intenta lograr este objetivo aplicando meta aprendizaje aprendiendo a aprender. Los modelos de meta aprendizaje entrenan en episodios, donde cada episodio es una tarea de *few-shot learning* y utilizaran este entrenamiento para lograr alcanzar una rápida adaptación del modelo para afrontar nuevas tareas de clasificación *few-shot*. Entre las variaciones que se han desarrollado de meta aprendizaje a lo largo de los años se encuentran: Modelos que utilizan *metric learning*, como Prototypical Networks [107] o Matching Networks [123] que aprenden del conjunto de soporte como minimizar la distancia entre elementos de la misma clase y de esta forma aprendiendo cómo aprender nuevos elementos.

*Memory networks* que aprenden a almacenar experiencias que luego podrán ser utilizadas para aprender nuevas tareas. Esto generalmente se realiza utilizando una memoria externa como en el modelo Neural Turing Machines o mediante LSTM [96].

Métodos basados en descenso de gradiente que intentan ajustar el algoritmo de optimización y los pesos iniciales de forma tal que el modelo converja con una mínima cantidad de pasos en el conjunto de soporte [25][112].

*Data augmentation* es otro método viable de afrontar few-shot learning. Este método funciona generando nuevos elementos para aumentar la cantidad de datos del conjunto de entrenamiento [133]. Los nuevos elementos generados permiten refinar los límites de clasificación al brindar nuevos datos sobre los cuales entrenar 4.5.

Finalmente, *semantic learning* busca utilizar información semántica adicionalmente a los datos utilizados comúnmente de forma tal que el modelo pueda obtener más información contextual a los datos y lograr una mejor generalización de forma similar a como un infante logra asociar imágenes a conceptos semánticos. Para hacer esto se alimenta al modelo no solo con las imágenes del conjunto de entrenamiento, sino con la información semántica asociada que puede estar en forma de etiqueta de categoría, información semántica descriptiva y atributos [100].

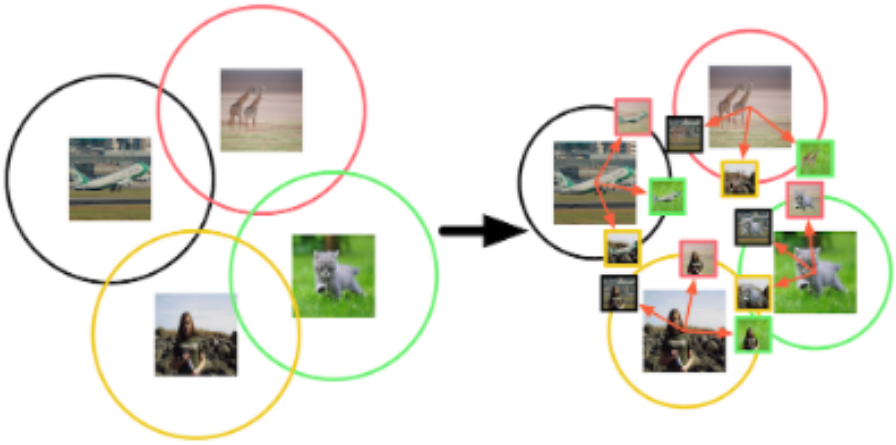


***Figura 4.5*** *Ilustración de la generación de datos para one-shot learning.*

## 4.6. Aprendizaje semi-supervisado

Como fue mencionado anteriormente 2.2.2 es posible clasificar el entrenamiento de modelos de aprendizaje profundo en 3 categorías según la disponibilidad de datos etiquetados: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje semi-supervisado.

El aprendizaje supervisado cuenta con un conjunto de datos de entrenamiento $X = (x_1, ..., x_N)$ acompañado de las respectivas etiquetas para cada uno de sus elementos $Y = (y_1, ..., y_N)$, de esta forma es sencillo calcular la función de costo simplemente comparando la salida del modelo a la salida esperada y utilizando este valor para calcular los gradientes.

Por otro lado, el aprendizaje no supervisado no cuenta con las etiquetas de sus elementos, sólo contará con $X$. El aprendizaje no supervisado permite encontrar fácilmente patrones en los datos sobre una gran cantidad de estos debido a la amplia disponibilidad de datos etiquetados. Por esta última razón es posible utilizar estos modelos para encontrar características en la estructura interna de los datos que de otra forma hubieran sido imposibles de ubicar. Con este tipo de entrenamiento se pueden diseñar algoritmos de agrupamiento que permitan agrupar aquellos elementos cuya estructura y características sean similares entre sí. La desventaja sería que este agrupamiento será menos preciso que la clasificación utilizando modelos supervisados y podrían no coincidir los agrupamientos encontrados con las etiquetas al no estar directamente relacionados.

El aprendizaje semi-supervisado cae en el medio entre el aprendizaje supervisado y el no supervisado. Debido a la dificultad y el coste en tiempo y dinero existente en el etiquetado de datos no etiquetados, la cantidad del total de los datos que es

efectivamente etiquetada es en general pequeña en comparación a la cantidad de todos los datos disponibles. Este fenómeno genera que una gran cantidad de datos que podrían ser aprovechados para mejorar los modelos sean desperdiciados en el aprendizaje supervisado o, desde otro punto de vista, los datos etiquetados que podrían ayudar a guiar el aprendizaje no supervisado son desaprovechados [6]. En el aprendizaje semi-supervisado se cuenta con un conjunto de datos $X = (x_i)_{i\in[n]}$ el cual puede ser separado en los subconjuntos $X_l = (x_1, ..., x_l)$ acompañado de las etiquetas $Y = (y_1, ..., y_l)$ para cada uno de sus elementos, y $X_u = (x_l + 1, ..., x_u)$ el cual no cuenta con las etiquetas de sus elementos. Para que $X_u$ sea relevante en el problema es necesario que la distribución de $X$ que estos datos adicionales ayudarán a dilucidar colabore en la tarea de clasificación, en caso contrario el aprendizaje semi-supervisado podría no incrementar la performance del modelo o incluso disminuirla [14]. Un estudio reciente ha logrado aliviar este problema de cambio de dominio, donde la distribución del conjunto de datos no etiquetado difiere de la del conjunto de datos etiquetados, mediante la utilización de *batch normalization* y el cálculo separado de las estadísticas de normalización de los conjuntos de datos etiquetado y no etiquetado [132].

Similar al aprendizaje semi-supervisado existe el aprendizaje transductivo, el cual utiliza los datos de prueba para entrenar el modelo además de los datos de entrenamiento. De esta forma se busca especializar en mayor medida el modelo para clasificar estos datos de prueba, cambiando el objetivo de la generalización de los datos al objetivo de alcanzar mayor performance sobre esos datos de prueba aprendiendo directamente sobre ellos.

Una forma comúnmente utilizada de realizar entrenamiento semi-supervisado es mediante pseudo-etiquetado. Esto se logra entrenando uno o más modelos sobre la limitada cantidad de datos etiquetados con los que se cuente para después utilizar estos modelos entrenado de forma supervisada en el etiquetado de los elementos del conjunto no etiquetado con alta confianza en la clasificación. Una vez obtenidos estos nuevos datos con pseudo-etiquetas, estos se utilizan para alimentar el modelo como nuevas entradas. No obstante, el uso incorrecto de esta técnica puede llevar a bias de confirmación, también llamado acumulación de ruido, donde el pseudo-etiquetado incorrecto de elementos del conjunto de datos no etiquetado a lo largo de los épocas lleva a aumentar la confianza en estos etiquetados incorrectos y así generando resistencia a los cambios en el modelo y sobreajuste. Se ha demostrado que utilizar *data augmentation* y configurando una cantidad mínima de datos etiquetados por época ayuda a evitar el problema del bias de confirmación mejorando la performance de los modelos que utilicen pseudo-etiquetado [3]. Otro método utilizado para mejorar la performance de los modelos que utilizan pseudo-etiquetado es la utilización de múltiples modelos mediante co-entrenamiento, los modelos cooperan enseñándose

entre sí mediante la selección de pseudo-etiquetas en las que posean confianza alta. Para que este tipo de entrenamiento otorgue los mayores beneficios es necesario que los modelos colaboradores posean la mayor diversidad posible, de forma tal que estos aprendan de las diferencias entre sus etiquetados [135].

## 4.7. Robust learning

Al adquirir datos para el entrenamiento de modelos de aprendizaje profundo siempre existe la posibilidad de que los datos adquiridos posean ruido, es decir, las etiquetas asociadas a los elementos podrían no ser las correctas, lo que dificulta el entrenamiento de los modelos [78]. Esto podría darse debido al etiquetado por personas no expertas (problema común en *crowdsourcing*), al etiquetado automático o a la corrupción de etiquetas por parte de terceros. Normalmente las redes neuronales son entrenadas para minimizar el error sobre un conjunto de entrenamiento, lo cual facilita que los modelos realicen memorización de los datos dañando la generalización y generando sensibilidad a ejemplos adversariales. La memorización ocurre debido a que esta es la forma más sencilla de ajustar al conjunto de entrenamiento generando sobreajuste. Esto genera que se dañe la generalización de la red y que el modelo se vuelva más vulnerable a memorizar etiquetas corruptas.

*Robust learning* es una rama del aprendizaje profundo que investiga métodos para el entrenamiento de redes neuronales resistentes a datos con presencia de ruido. Esto es, dado una distribución real de los datos $p(x, y)$ siendo $x$ un elemento e $y$ su etiqueta asociada, se cuenta con un conjunto de datos que se asume corrupto $\tilde{D}$ con $n$ elementos $(x, \tilde{y})$ cuya corrupción es especificada por la distribución de ruido de etiquetas $p(\tilde{y}|x, y)$. Las etiquetas del conjunto de datos $D$ podrían no estar correctamente etiquetadas, esto llevaría a cambiar la distribución de $D$ con respecto a la distribución real de los datos con lo que entrenar ciegamente sobre $D$ haría que el modelo aprenda una distribución errónea de los datos.

Esta demostrado [103] que *data augmentation* puede servir como regularización para evitar el sobreajuste y, de esta forma, minimizar el efecto de ejemplos adversariales en los modelos. Esto lo logran entrenando al modelo sobre una distribución vecinal, la cual se alcanza introduciendo ejemplos entre los elementos del conjunto de datos original. Entrenando con *data augmentation* el modelo es alentado a actuar linealmente entre ejemplos reduciendo oscilaciones no deseadas en predicciones sobre elementos fuera del conjunto de entrenamiento. Un ejemplo de *data augmentation* que puede ser utilizada en *robust learning* es *mixup* [134], un método sencillo, eficiente y agnóstico a los datos que facilita la regularización de modelos generando ejemplos de entrenamiento virtuales combinando pares de ejemplos y sus etiquetas.

El aprendizaje por currículo [7] es un método en donde el modelo será entrenado

de forma ordenada empezando su entrenamiento desde ejemplos más sencillos y aumentando la dificultad del entrenamiento gradualmente. Actualmente este tipo de entrenamiento ha demostrado ser útil en *robust learning* para el entrenamiento de redes mejorando su generalización con datos corruptos [50]. El método moderno de realizar aprendizaje por currículo es mediante la utilización de un modelo maestro y un modelo estudiante, ambos modelos interactúan entre sí para formar un currículo que permita al modelo alumno obtener un mejor entrenamiento. El maestro aprenderá iterativamente la dificultad de los ejemplos del conjunto de entrenamiento mediante su interacción con el alumno y utilizará esta dificultad para enseñarle al alumno de forma incremental subconjuntos de los datos del conjunto de entrenamiento.

Existen casos en los que además del conjunto de datos corrupto es posible contar con un conjunto de datos de menor tamaño con datos fiables. En este caso es posible aumentar aún más la performance de los modelos de *robust learning* apoyándose sobre estos datos fiables. Esta perspectiva de entrenamiento utilizando datos fiables se acerca a la definición de aprendizaje semi-supervisado [35], donde se cuenta con un conjunto de datos etiquetado $X_l$ y un conjunto de datos no etiquetados $X_u$. En *robust learning* el mismo conjunto de entrenamiento utilizado para el aprendizaje semi-supervisado podría ser reclasificado como datos fiables $X_l$ y datos corruptos $X_u$ utilizando pseudo-etiquetado sobre los datos cuyas etiquetas se desconoce. Por otro lado, de poseerse un conjunto de datos para *robust learning*, ignorando las etiquetas del conjunto de datos corrupto se podría aplicar fácilmente aprendizaje semi-supervisado. Estos conjuntos de datos fiables pueden ser aprovechados mediante la utilización de matrices de corrupción [37] o mediante el calculo de distancia entre elementos del conjunto de datos fiable y los corruptos [57]. Ambos métodos tienen funcionalidad similar, calculando un peso para la corrupción de las etiquetas o fuente respectivamente y utilizando este peso para guiar el entrenamiento de los modelos.

# 5 Experimentación

Con el fin de demostrar las mejoras que otorga el uso de las técnicas mencionadas anteriormente hemos desarrollado y comparado métodos utilizados en el entrenamiento de modelos con conjuntos de datos con poca cantidad de datos etiquetados [23]. Estos experimentos y sus resultados fueron publicados en el Congreso Argentino de Ciencias de la Computación CACIC 2019 [23].

La clasificación de señas de la lengua de señas sufre de falta de suficientes datos etiquetados para poder entrenar efectivamente modelos de aprendizaje profundo de manera convencional. Debido a esto aplicar técnicas para el entrenamiento utilizando conjuntos de datos pequeños permite obtener resultados superiores a los obtenidos en trabajos previos.

En los experimentos realizados analizamos y comparamos modelos estado del arte y específicos para problemas de conjuntos de datos pequeños, adicionalmente examinamos los efectos de la utilización de *data augmentation*. En particular realizamos experimentos con Wide-DenseNet, una arquitectura convolucional estado del arte, y Prototypical Networks, un modelo estado del arte utilizado en *few-shot learning*. En ambos casos cuantificamos el impacto de la aplicación de data augmentation sobre la exactitud de los modelos.

## 5.1. Preparación de los conjunto de datos

Seleccionamos 3 conjuntos de datos sobre los que entrenamos los modelos: LSA16 [93], RWTH-PHOENIX-Weather [26] y CIARP [24]. Estos conjuntos de datos poseen imágenes cuyas composiciones varían en gran medida, ya han sido evaluados y poseen diferentes cantidades de muestras o de distribuciones de ejemplos por clase.

### 5.1.1. CIARP

CIARP [24] contiene 6000 imágenes de tamaño $38 \times 38$ adquiridas de una cámara a color. Las imágenes fueron etiquetadas manualmente y corresponden 10 clases de gestos de manos. Las manos se encuentran centradas y segmentadas del fondo, el cual fue reemplazado por píxeles negros. El pequeño tamaño de las imágenes y la baja cantidad de clases le otorgan a este conjunto de datos una baja complejidad. Las clases en los datos corresponden a formas de manos no basadas en la lengua de señas pero lo suficientemente similares como para que la comparación se mantenga como valida.

### 5.1.2. LSA16

LSA16 [93] contiene imágenes de 16 formas de manos de la lengua de señas argentina (LSA) donde cada forma de mano fue realizada 5 veces por 10 personas diferentes generando un total de 800 imágenes de tamaño $32 \times 32$. Las señas fueron tomadas utilizando guantes con colores y ropas negras sobre un fondo blanco, lo que aumenta el contraste de las señas. El conjunto de datos se encuentra balanceado, con 50 imágenes para cada clase, por lo que no es necesario realizar ningún trabajo adicional para balancear el conjunto de datos. Solo hay una mano por imagen, la cual se encuentra centrada y aislada del fondo.

### 5.1.3. RWTH

RWTH [26] está compuesto por una selección de imágenes de manos de tamaño $132 \times 92$ recortadas de videos de intérpretes de señas de manos de la estación de tv pública alemana PHOENIX. Hay un total de 45 señas de manos diferentes. Los intérpretes utilizan ropas negras en frente de un fondo gris, lo que ayuda a aumentar el contraste de las manos. Muchas de las imágenes poseen desenfoque de movimiento y otras poseen ambas manos del intérprete en la imagen. Las manos no se encuentran siempre perfectamente centradas. Esto aumenta la complejidad del conjunto al ser menos uniforme en cuanto al formato de las manos en las imágenes y poseer ruido representado en manos desenfocadas. El conjunto de datos se encuentra desbalanceado en gran medida, con algunas clases teniendo solo 1 ejemplo mientras que otras tienen tantos como 529 ejemplos. Removimos aquellas clases que poseían menos de 20 ejemplos siguiendo, de esta forma se garantiza una mínima cantidad de imágenes por clase para que la red aprenda.

## 5.2. Preparación de los modelos

Para este experimento elegimos utilizar 2 arquitecturas de modelos: Wide-DenseNet [47] y Prototypical Network [107]. Las redes prototípicas fueron desarrolladas para afrontar problemas de *few-shot learning*, por lo que su diseño esta orientado a obtener buena performance en problemas con baja cantidad de datos etiquetados. En cambio, DenseNet es una arquitectura de red neuronal estado del arte no desarrollada específicamente para conjuntos de datos pequeños pero, aun así, ha demostrado buenos resultados en la clasificación entrenando sobre estos.

### 5.2.1. Prototypical Network

Las redes prototípicas [107] son modelos de meta aprendizaje utilizados para problemas de clasificación *few-shot*, donde el clasificador debe generalizar a nuevas

clases de clases no vistas en el conjunto de entrenamiento dado solo un pequeño número de elementos de cada nueva clase. La habilidad del algoritmo para realizar aprendizaje *few-shot* es medida típicamente por su performance en tareas de clasificación K-learn-C-shot. Primero el modelo recibe conjunto de consulta perteneciente a una nueva clase no vista previamente. Posteriormente el modelo es provisto con un conjunto de soporte $S$, compuesto de $K$ nuevos ejemplos de diferentes clases no vistas. Finalmente el algoritmo debe determinar a cuál de las clases del conjunto de soporte pertenecen los elementos del conjunto de consulta. Sistemas para tareas de clasificación *few-shot*, como las redes prototípicas, también pueden ser aplicados a pequeños conjuntos de datos donde todas las clases son conocidas. Las redes prototípicas aplican un convincente bias inductivo en forma de prototipos de clases alcanzando una muy buena performance en tareas de *few-shot*. La suposición clave es que existe una asignación en la cual los ejemplos de cada clase se agrupan alrededor de una representación prototípica la cual es la media de los elementos individuales. De esta forma se realiza clasificación C-shot en el caso de que $C > 1$ seleccionando la etiqueta para el elemento dado según cual sea el prototipo de clase más cercano.

La Prototypical Network con la que realizamos los experimentos utiliza 4 bloques convolucionales. Cada bloque está compuesto de una capa convolucional $3 \times 3$ de tamaño de filtro 64, una capa de *batch normalization*, una función ReLU y una capa *max-pooling* de $2 \times 2$. Todos los modelos fueron entrenados utilizando optimizador ADAM [54]. Se utilizó un ratio de aprendizaje inicial de $10^{-3}$ y un recorte del ratio de aprendizaje a la mitad cada 2000 episodios. Entrenamos la Prototypical Network utilizando distancia euclídea en escenarios 1-shot y 5-shot con episodios de entrenamiento conteniendo 16, 20 y 10 clases (para LSA16, RWTH y CIARP respectivamente) y 5 puntos de consulta por clase. Encontramos ventajoso equiparar la misma cantidad de valores de $C$ para los escenarios de entrenamiento y prueba, y la utilización de un valor alto de $K$ (más clases) por episodio de entrenamiento. Computamos la exactitud de los modelos promediando la exactitud sobre 1000 episodios generados aleatoriamente del conjunto de entrenamiento.

En los experimentos realizados con RWTH utilizamos la misma arquitectura de bloques del codificador pero utilizando 8 capas en lugar de 4, con la idea de que dada la mayor complejidad del conjunto de datos será necesaria una red de mayor tamaño para obtener resultados óptimos. Se obtuvo una gran diferencia en los resultados obtenidos en los escenarios 1-shot y 5-shot para este conjunto de datos. Descubrimos que los escenarios 5-shot otorgaban mejores resultados. Usando esta información decidimos realizar aprendizaje 5-shot en los experimentos realizados sobre los demás conjuntos de datos. Las mejores configuraciones para los conjuntos de datos que encontramos fue un escenario 5-shot con igual C tanto para las etapas de entrenamiento como de prueba y utilizando $K \geq 5$, con 5 o más clases por episodio

de entrenamiento. Los mejores resultados fueron obtenidos cuando el valor de $K$ se acerca a la cantidad total de clases en el conjunto de datos excepto en CIARP donde los mejores resultados fueron obtenidos utilizando un número de clases por episodio de entrenamiento de 5. Adicionalmente, la mejor configuración de la red fue de 64 filtros para todos los conjuntos de datos.

### 5.2.2. DenseNet

Seleccionamos DenseNet ya que es un modelo estado del arte en múltiples dominios y puede utilizarse en la clasificación de pequeños conjuntos de datos alcanzando un ratio de error bajo. Utilizamos una variación de DenseNet llamada Wide-Densenet la cual sigue la estrategia utilizada por Wide-ResNet [131]. Esta variación utiliza modelos de menor profundidad pero mayor ratio de crecimiento, de esta forma se obtienen modelos más compactos reduciendo el tiempo de procesamiento requerido y obteniendo una exactitud igual o mayor a la obtenida en las arquitecturas más profundas. A esta arquitectura le fueron añadidos bloques SE ya que estos pueden ser incluidos en cualquier tipo de modelo que utilice capas convolucionales mejorando su performance a un bajo costo computacional. Utilizando estos bloques fue posible disminuir el ratio de error del modelo aún más. Utilizamos un ratio de reducción en los bloques SE de 16 como es recomendado por los autores originales. Introducimos los bloques SE luego de cada bloque denso y luego de cada bloque de transición 5.1, de esta forma podemos aumentar la información obtenida por canal con un bajo costo computacional.

Utilizamos búsqueda de cuadrícula de los hiperparametros para encontrar el modelo con la mayor exactitud en cada conjunto de datos. Se probaron ratios de crecimiento con valores de 32, 64 y 128 y profundidad de las capas densas de hasta [6,12,24,16], siendo el valor en la posición $i$ del conjunto la cantidad de bloques convolucionales pertenecientes al bloque denso $i$. Entrenamos los modelos utilizando un tamaño del *batch* de 16, un ratio de aprendizaje inicial de $10^{-3}$ con optimizador *categorical cross entropy*. Se configuró una cantidad máxima de 400 épocas con una paciencia máxima con valor 25, por lo tanto el entrenamiento del modelo se detendrá una vez alcanzado la época 400 o una vez ocurran 25 épocas sin mejorar la performance sobre el conjunto de prueba, de esta forma se evita el sobreajuste pero se mantiene una ventana lo suficientemente grande para evitar caer en un mínimo local. El mejor modelo encontrado para todos los conjuntos de datos posee un ratio de crecimiento de 64 y 2 bloques densos de 6 y 12 capas respectivamente.

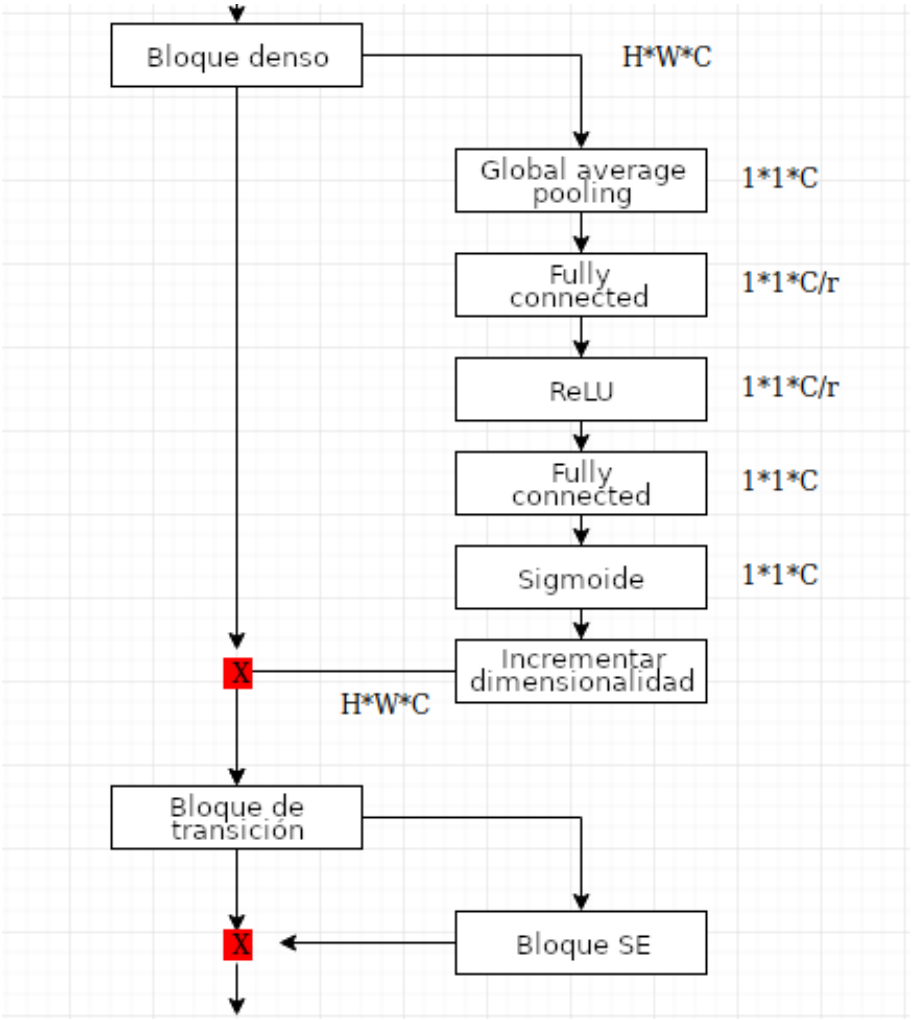



***Figura 5.1*** *Bloques SE añadidos a Wide-DenseNet.*

## 5.3. Entrenamiento

Realizamos los experimentos de clasificación en los conjuntos de datos de formas de manos LSA16, RWTH y CIARP. Para cada experimento dividimos el conjunto de datos en conjuntos de entrenamiento y prueba, con el conjunto de prueba abarcando el 25 % de los elementos. La división fue estratificada manteniendo las proporciones de elementos para cada clase en ambos conjuntos. Aplicamos normalización en cuanto a las características restando la media y dividiendo por la desviación estándar cada característica. Para *data augmentation* utilizamos vuelta horizontal, rotación de hasta 10 o 30 grados, y modificación en el tamaño de las imágenes reduciéndolo hasta 10 % o 20 % en altura y ancho. Descubrimos que una rotación de 10 grados otorgaba mejores resultados en el conjunto de prueba, lo que se debe a que una rotación de 30 grados era demasiado alta para la naturaleza de los conjuntos de datos pudiendo llegar a cambiar el significado de la seña.

Con el fin de optimizar las arquitecturas utilizadas se realizaron múltiples experimentos con redes prototípicas y Wide-DenseNets con distintas variaciones de configuraciones de hiperparametros para cada conjunto de datos con y sin *data augmentation*. También se realizaron experimentos con varias configuraciones de *data augmentation*. Para esto se creó un conjunto de configuraciones $C$ posibles seleccionadas de forma experta basándose en el problema dado. Utilizando estas configuraciones procedimos a entrenar los modelos para cada conjunto de datos obteniendo los resultados en los conjuntos de prueba los cuales utilizamos como comparación para obtener la mejor configuración de hiperparametros y de *data augmentation*.

## 5.4. Resultados

En la tabla 5.1, podemos observar que todos los modelos poseen una menor exactitud en el conjunto de datos RWTH, lo cual es esperado debido a que este posee mayor cantidad de clases, imágenes de manos no segmentadas y desbalance de clases. Las redes prototípicas poseen una exactitud similar tanto para LSA16 como para CIARP superando a los demás modelos, lo cual también es esperado debido a que estos conjuntos de datos poseen muy pocos elementos. En LSA16 Prototypical Network obtuvo mejor exactitud que VGG16 y Wide-DenseNet, y en CIARP obtuvo resultados similar a los obtenidos por LeNet CNN y Wide-DenseNet. La exactitud obtenida por Wide-DenseNet en RWTH es ligeramente superior que para los demás modelo. Nuestra hipótesis es que las redes prototípicas obtuvieron menor exactitud debido a que las imágenes de las manos no se encontraban segmentadas en este conjunto de datos. Debe notarse que la utilización de *data augmentation* no otorgo mejoras significativas en la exactitud obtenida en los conjuntos de datos LSA16 y CIARP.

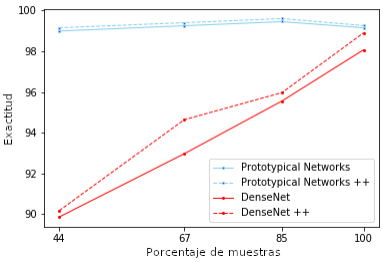
Otro factor a considerar es que se obtuvieron mejores resultados utilizando parámetros que generen arquitecturas de tamaño reducido.

En la imagen 5.2 podemos observar la exactitud de las redes prototípicas y los modelos DenseNet entrenados con distintos tamaños de muestra. Realizamos experimentos utilizando las mismas arquitecturas y configuraciones descriptas en esta sección variando los tamaños de las muestras de entrenamiento en porcentajes de 44 %, 67 % y 85 % con un tamaño de conjunto de entrenamiento fijo del 25 %. De los resultados obtenidos es posible ver que la performance de los modelos de DenseNet incrementa al añadir mas ejemplos al conjunto de entrenamiento. De la imagen 5.2(b) podemos ver que el modelo DenseNet entrenado utilizando *data augmentation* obtiene mejores resultados que el modelo entrenado sin utilizarla. Por otro lado, los modelos de redes prototípicas no muestran un incremento significativo en la performance al añadir más porcentaje de elementos a la muestra de entrenamiento. En las imágenes 5.2(b) y 5.2(c) podemos observar como la utilización de *data augmentation*,

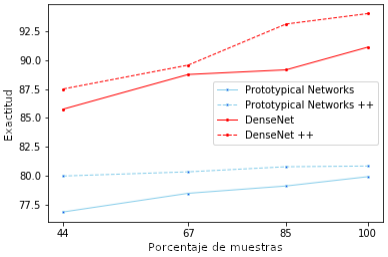
| *Method* | *LSA16* | *RWTH* | *CIARP* |
|---|---|---|---|
| LeNet [24] | - | - | 99.20 |
| Inception (fine-tuning) [56] | - | 85.50 | |
| VGG16 [87] | 95.92 | 82.88 | |
| Inception+SVM (pre-trained) [87] | 93.67 | 78.12 | - |
| DenseNet | 98.07 | 91.10 | 99.93 |
| DenseNet ++ | 98.90 | **94.00** | 99.99 |
| Prototypical Networks | 99.15 | 79.93 | 99.98 |
| Prototypical Networks ++ | **99.26** | 80.85 | **100.00** |

***Cuadro 5.1*** *Exactitud de varios modelos basados en redes neuronales convolucionales en los 3 conjuntos de datos: LSA16, RWTH y CIARP. Los modelos indicados con ++ utilizaron* data augmentation.

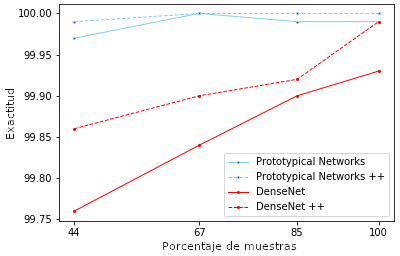
en los conjuntos de datos RWTH Y CIARP respectivamente, resulta en la obtención de modelos con mayor mejora en la exactitud comparado a los resultados obtenidos en LSA16 5.2(a) donde el incremento en performance obtenido de la utilización de *data augmentation* es mínimo.




(a) LSA16




(b) RWTH




(c) CIARP

***Figura 5.2*** *Exactitud de redes prototípicas y modelo DenseNet entrenados variando el tamaño de muestra en los 3 conjuntos de datos: LSA16, RWTH y CIARP. Cada gráfico representa un conjunto de datos diferentes donde el axis x es el porcentaje de elementos de la muestra utilizado y el axis y es la exactitud obtenida. Los modelos con ++ utilizaron* data augmentation.

## 5.5. Conclusiones

Para todos los conjuntos de datos encontramos modelos que mostraron una performance a la par o mejor que anteriores modelos estado del arte.

Todos los modelos logran exactitud casi perfecta en CIARP. Esto muestra que el conjunto de datos es demasiado simple para ser utilizado como punto de referencia para el reconocimiento de formas de manos. Mientras que posee mas elementos que otros conjuntos de datos (6000), estas imágenes son demasiado homogéneas y no poseen suficientes variación como para poder generalizar los resultados a aplicación del mundo real.

Las redes prototípicas proveen un nuevo estado del arte en el conjunto de datos LSA16 superando todos los demás métodos conocidos. Wide-DenseNet también mejoran sobre los modelos estado del arte y se acercan a la performance obtenida con redes prototípicas en esta conjunto de datos. Sin embargo, es posible observar que la diferencia en performance entre los 2 modelos se reduce en gran medida cuando el tamaño de la muestra de entrenamiento incrementa.

También alcanzamos un nuevo estado del arte en el conjunto de datos RWTH con Wide-DenseNet, mientras que las redes prototípicas también lograron mejorar sobre resultados previos.

Esto demuestra que las nuevas arquitecturas convolucionales pueden trabajar mejor con pocos datos, pero aun existe espacio para mejoras usando modelos especializados

# 6 Conclusiones y trabajos futuros

## 6.1. Conclusiones

Con los experimentos realizados se ha demostrado la utilidad de la implementación de técnicas especificas para el tratamiento de conjuntos de datos con pocos datos etiquetados superando previos modelos del estado del arte en el área dada. Llegando finalmente a lograr una precisión en la clasificación de señas estáticas del 99.26 % en el conjunto de datos LSA16 utilizando un modelo de Prototypical Network con *data augmentation* y 94 % en el conjunto de datos RWTH-PHOENIX-Wheater utilizando un modelo DenseNet con *data augmentation*.

Los actuales modelos del estado del arte, como DenseNet, han adquirido una capacidad superior a la de anteriores modelos para el tratamiento de pequeños conjuntos de datos permitiendo obtener mejores resultados sin necesidad de aplicar una multitud de técnicas especificas para el problema.

Aún así, ha quedado demostrado que la aplicación de técnicas especificas, como redes prototípicas, puede aumentar aun más la exactitud de los modelos obtenidos, principalmente en situaciones más extremas donde los datos etiquetados disponibles son de unos pocos por clase.

Por lo tanto, según los recursos disponibles y el problema afrontado debe seleccionarse cuidadosamente el método a utilizar para el tratamiento de conjuntos de datos con pocos datos etiquetados, de forma tal que se puedan obtener los modelos con mayor exactitud y generalización posible.

Adicionalmente, la aplicación de *data augmentation* provee un incremento en la performance de los modelos, principalmente de aquellos modelos que requieren una mayor cantidad de datos para obtener una buena generalización. Las políticas de *data augmentation* utilizadas en los experimentos podrán ser transferidas a otros problemas de reconocimiento de formas de manos estáticas, lo que facilitara trabajos futuros.

*Data augmentation* ha demostrado ser una de las técnicas más fácilmente aplicables a cualquier problema de este tipo, pudiendo ser aplicada simplemente utilizando librerías existentes. Pero *data augmentation* no se limita simplemente a la modificación de las características de las imágenes con el fin de generar nuevas, sino que se han desarrollado nuevas metodologías de *data augmentation* como la utilización de redes generativas para obtener elementos completamente nuevos a partir de ruido. También se han desarrollado formas de optimizar las estrategias sobre qué funciones se aplicaran sobre las imágenes para realizar *data augmentation* sobre un conjunto de datos especifico mediante aprendizaje por refuerzo.

*Weak supervision* y *active learning* otorgan métodos efectivos para la obtención de información adicional con la cual entrenar los modelos sin necesidad de la inversión de grandes cantidades de recursos y tiempo. Las grandes cantidades de datos obtenidos mediante *weak supervision* facilitan a los modelos la obtención de mayor generalización al aumentar en gran medida la cantidad de ejemplos de entrenamiento disponibles. *Active learning*, por otro lado, otorga una forma eficiente de etiquetar datos con ruido de forma fiable para ayudar a los modelos a diferenciar clases mas correctamente. Ambas técnicas se apoyan en la necesidad de obtener mas datos, lo que permite alcanzar una mayor generalización al acercarse más a la distribución real del conjunto total de los datos, de forma económica y rápida ya que el etiquetado de la totalidad de los datos es usualmente una tarea sumamente costosa y que requiere una gran cantidad de tiempo.

Debido a la dificultad encontrada al etiquetar datos, es normal encontrar la situación en la cual solo una parte del total de los datos dispone de etiquetas. En estos casos es posible aplicar aprendizaje semi-supervisado para mejorar la calidad de los modelos producidos al permitir la utilización de la totalidad de los datos aprovechando los datos no etiquetados disponibles. Otra problemática común es la adquisición de datos con baja calidad. Los conjuntos de datos con ruido, los cuales poseen datos incorrectamente etiquetados, de no ser tratados con cuidado, podrían afectar en gran medida el funcionamiento de los modelos entrenados utilizándolos, los cuales podrían aprender erróneamente de estos memorizando asociaciones erróneas. *Robust learning* otorga métodos confiables de enfrentar el problema de poseer datos con ruido dentro de los datos de entrenamiento. Existe una gran cantidad de técnicas de *robust learning* que permitirán efectuar entrenamiento aprovechando estos conjuntos de datos de entrenamiento con ruido para aumentar la generalización de los modelos producidos reduciendo al mínimo el efecto de los datos con ruido en el modelo.

Adicionalmente existen problemas en los cuales es posible encontrar modelos previamente entrenados en grandes conjuntos de datos de tareas similares. Estos modelos pueden ser reutilizados mediante *transfer learning* aprovechando su conocimiento previo y su capacidad de generalización, al haber sido entrenados en conjuntos de datos de gran tamaño. Esto provee una forma rápida y económica de entrenar modelos para tareas especificas con gran capacidad de generalización.

Aun así, existen casos en los que se cuenta con una cantidad muy limitada de datos y es imposible la obtención de datos adicionales para el entrenamiento de los modelos. *Few-shot learning* permite en estos casos realizar clasificación entrenando modelos que son capaces de generalizar los datos aprendiendo de una muy pequeña cantidad de datos etiquetados.

La existencia de estos métodos anteriormente mencionados permite la creación

de modelos con gran generalización entrenados a partir de una pequeña cantidad de datos etiquetados de forma rápida y económica. De esta forma se facilita la accesibilidad de modelos de aprendizaje profundo para proyectos de menor escala con pocos recursos o problemáticas en las que los datos intrínsecos son limitados. Esto es importante al expandir en gran medida la cantidad de tareas sobre las que es posible aplicar modelos de aprendizaje profundo.

## 6.2. Trabajos futuros

En el futuro, el desarrollo de nuevos modelos generativos podría ayudar a resolver la problemática de los pocos datos etiquetados al permitir la generación de nuevos datos de entrenamiento a partir de los datos disponibles. Futuros avances en el área de *meta-learning* podría permitir el entrenamiento de modelos capaces de generalizar a partir del aprendizaje en múltiples tareas, lo que reduciría la cantidad de datos necesarios para cada tarea en particular. Combinando ambas técnicas mencionadas, la creación de modelos generativos que utilicen *meta-learning* podría permitir la creación de nuevos datos que vayan mas allá de la distribución del conjunto de datos de entrenamiento y se acerquen mas a la distribución total de los datos aplicando los conocimientos generales aprendidos mediante el meta aprendizaje a la tarea en particular sobre la que se quieren generar nuevos datos.

También sería interesante la experimentación utilizando otras técnicas para el tratamiento de conjuntos de datos con pocos datos etiquetados. Constantemente se esta generando nueva información en imágenes y vídeos de lengua de señas, la utilización de *weak supervision* para extraer y utilizar estos datos podría otorgar una mejora significativa en la generalización de los modelos entrenados. Estos datos podrían ser posteriormente utilizados en un modelo que realice *robust learning* o aprendizaje semi-supervisado.

# 7 Bibliografía

[1] Amazon, ed. *Open Data on AWS*. 2020. URL: https://aws.amazon.com/es/opendata/?wwps-cards.sort-by=item.additionalFields.sortDate&wwps-cards.sort-order=desc (visitado 01-03-2020).

[2] Antreas Antoniou, Amos J. Storkey y Harrison Edwards. «Data Augmentation Generative Adversarial Networks». En: *ArXiv* abs/1711.04340 (2017).

[3] Eric Arazo, Diego Ortego, Paul Albert, Noel E. Oonnor y Kevin McGuinness. «Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning». En: *ArXiv* abs/1908.02983 (2019).

[4] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven y John M. Martinis. «Quantum supremacy using a programmable superconducting processor». En: *Nature* 574.7779 (2019), págs. 505-510. ISSN: 1476-4687. DOI: 10.1038/s41586-019-1666-5. URL: https://doi.org/10.1038/s41586-019-1666-5.

[5] Haitham Badi. «Recent methods in vision-based hand gesture recognition». En: *International Journal of Data Science and Analytics* 1.2 (2016), págs. 77-87. ISSN: 2364-4168. DOI: 10.1007/s41060-016-0008-z. URL: https://doi.org/10.1007/s41060-016-0008-z.

[6] Sugato Basu, Mikhail Bilenko, Arindam Banerjee, Raymond J. Mooney y Misha Bilenko. *Probabilistic Semi-Supervised Clustering with Constraints*. Semi-Supervised Learning. MIT Press, sep. de 2006, págs. 71-98. URL: https://www.microsoft.com/en-us/research/publication/probabilistic-semi-supervised-clustering-with-constraints/.

[7] Yoshua Bengio, Jérôme Louradour, Ronan Collobert y Jason Weston. «Curriculum Learning». En: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML 09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, págs. 41-48. ISBN: 9781605585161. DOI: 10.1145/1553374.1553380. URL: https://doi.org/10.1145/1553374.1553380.

[8] Vivek Bheda y Dianna Radpour. «Using Deep Convolutional Networks for Gesture Recognition in American Sign Language». En: *CoRR* abs/1710.06836 (2017). arXiv: 1710.06836. URL: http://arxiv.org/abs/1710.06836.

[9] Johan Bjorck, Carla P. Gomes y Bart Selman. «Understanding Batch Normalization». En: *CoRR* abs/1806.02375 (2018). arXiv: 1806.02375. URL: http://arxiv.org/abs/1806.02375.

[10] Jake Bouvrie. «Notes on Convolutional Neural Networks». Nov. de 2006. URL: http://cogprints.org/5869/.

[11] Jason Brownlee. *Convolutional Neural Network Model Innovations for Image Classification*. Jul. de 2019. URL: https://machinelearningmastery.com/review-of-architectural-innovations-for-convolutional-neural-networks-for-image-classification/.

[12] Rich Caruana, Steve Lawrence y C. Lee Giles. «Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping». En: *NIPS*. 2000.

[13] A. M. Caulfield, E. S. Chung, A. Putnam, H. Angepat, J. Fowers, M. Haselman, S. Heil, M. Humphrey, P. Kaur, J. Kim, D. Lo, T. Massengill, K. Ovtcharov, M. Papamichael, L. Woods, S. Lanka, D. Chiou y D. Burger. «A cloud-scale acceleration architecture». En: *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. Oct. de 2016, págs. 1-13. DOI: 10.1109/MICRO.2016.7783710.

[14] Olivier Chapelle, Schölkopf Bernhard y Alexander Zien. *Semi-supervised learning*. MIT, 2006.

[15] François Chollet. «Xception: Deep Learning with Depthwise Separable Convolutions». En: *CoRR* abs/1610.02357 (2016). arXiv: 1610.02357. URL: http://arxiv.org/abs/1610.02357.

[16] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan y Quoc V. Le. «AutoAugment: Learning Augmentation Policies from Data». En: *CoRR* abs/1805.09501 (2018). arXiv: 1805.09501. URL: http://arxiv.org/abs/1805.09501.

[17] Amirhossein Dadashzadeh, Alireza Tavakoli Targhi, Maryam Tahmasbi y Majid Mirmehdi. «HGR-Net: a fusion network for hand gesture segmentation and recognition». En: *IET Computer Vision* 13 (2018), págs. 700-707.

[18] Wenyuan Dai, Qiang Yang, Gui-Rong Xue y Yong Yu. «Boosting for Transfer Learning». En: *Proceedings of the 24th International Conference on Machine Learning*. ICML 07. Corvalis, Oregon, USA: Association for Computing Machinery, 2007, págs. 193-200. ISBN: 9781595937933. DOI: 10.1145/1273496.1273521. URL: https://doi.org/10.1145/1273496.1273521.

[19] World Federation of the Deaf, ed. *Our Work*. 2016. URL: http://wfdeaf.org/our-work (visitado 03-02-2020).

[20] National Institute of Deafness y Other Communication Disorders. *American Sign Language*. 2019. URL: https://www.nidcd.nih.gov/health/american-sign-language (visitado 03-02-2020).

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li y L. Fei-Fei. «ImageNet: A Large-Scale Hierarchical Image Database». En: *CVPR09*. 2009.

[22] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng y Trevor Darrell. «DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition». En: *CoRR* abs/1310.1531 (2013). arXiv: 1310.1531. URL: http://arxiv.org/abs/1310.1531.

[23] Ulises Jeremias Cornejo Fandos, Gaston Gustavo Rios, Franco Ronchetti, Facundo Quiroga, Waldo Hasperu y Laura Lanzarini. «Recognizing Handshapes using Small Datasets». En: *Congreso Argentino de Ciencias de la Computación (CACIC) 19* (2019).

[24] Dennis Núñez Fernández y Bogdan Kwolek. «Hand Posture Recognition Using Convolutional Neural Network». En: *SpringerLink* (nov. de 2017). URL: https://link.springer.com/chapter/10.1007/978-3-319-75193-1_53.

[25] Chelsea Finn, Pieter Abbeel y Sergey Levine. «Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks». En: *CoRR* abs/1703.03400 (2017). arXiv: 1703.03400. URL: http://arxiv.org/abs/1703.03400.

[26] Jens Forster, Christoph Schmidt, Thomas Hoyoux, Oscar Koller, Uwe Zelle, Justus H. Piater y Hermann Ney. «RWTH-PHOENIX-Weather: A Large Vocabulary Sign Language Recognition and Translation Corpus». En: *LREC*. 2012.

[27] Nilay Ganatra y Atul Patel. «A Comprehensive Study of Deep Learning Architectures, Applications and Tools». En: *International Journal of Computer Sciences and Engineering* 6 (dic. de 2018), págs. 701-705. DOI: 10.26438/ijcse/v6i12.701705.

[28] Abhijit Ghatak. *Machine Learning with R*. Springer Verlag, Singapor, 2018.

[29] Sylvie Gibet. «Building French Sign Language Motion Capture Corpora for Signing Avatars.» En: *LREC 2018. Workshop on the Representation and Processing of Sign Languages: Involving the Language Community*. 2018.

[30] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[31] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville y Yoshua Bengio. «Generative Adversarial Nets». En: *Advances in Neural Information Processing Systems 27*. Ed. por Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence y K. Q. Weinberger. Curran Associates, Inc., 2014, págs. 2672-2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[32] Martin T. Hagan. *Neural Network Design - 2nd Edtion*. 2014. URL: http://hagan.okstate.edu/NNDesign.pdf.

[33] L. K. Hansen y P. Salamon. «Neural network ensembles». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12.10 (oct. de 1990), págs. 993-1001. ISSN: 1939-3539. DOI: 10.1109/34.58871.

[34] M. Hasenjäger y H. Ritter. «Active Learning in Neural Networks». En: *New Learning Paradigms in Soft Computing*. Ed. por Lakhmi C. Jain y Janusz Kacprzyk. Heidelberg: Physica-Verlag HD, 2002, págs. 137-169. ISBN: 978-3-7908-1803-1. DOI: 10.1007/978-3-7908-1803-1_5. URL: https://doi.org/10.1007/978-3-7908-1803-1_5.

[35] Ryuichiro Hataya e Hideki Nakayama. «Unifying semi-supervised and robust learning by mixup». En: *International Conference on Learning Representations*. 2019.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun. «Deep Residual Learning for Image Recognition». En: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[37] Dan Hendrycks, Mantas Mazeika, Duncan Wilson y Kevin Gimpel. «Using Trusted Data to Train Deep Networks on Labels Corrupted by Severe Noise». En: *CoRR* abs/1802.05300 (2018). arXiv: 1802.05300. URL: http://arxiv.org/abs/1802.05300.

[38] Vincent Hernandez y Gentiane Venture. *American Sign Language - LeapMotion - 25 subjects - 60 signs*. 2018. URL: http://dx.doi.org/10.17632/c7zmhcfnyd.1.

[39] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto y Hartwig Adam. «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications». En: *CoRR* abs/1704.04861 (2017). arXiv: 1704.04861. URL: http://arxiv.org/abs/1704.04861.

[40] Jie Hu, Li Shen y Gang Sun. «Squeeze-and-Excitation Networks». En: *CoRR* abs/1709.01507 (2017). arXiv: 1709.01507. URL: http://arxiv.org/abs/1709.01507.

[41] Gao Huang, Zhuang Liu y Kilian Q. Weinberger. «Densely Connected Convolutional Networks». En: *CoRR* abs/1608.06993 (2016). arXiv: 1608.06993. URL: http://arxiv.org/abs/1608.06993.

[42] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra y Kilian Q. Weinberger. «Deep Networks with Stochastic Depth». En: *CoRR* abs/1603.09382 (2016). arXiv: 1603.09382. URL: http://arxiv.org/abs/1603.09382.

[43] Gary B. Huang, Manu Ramesh, Tamara Berg y Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Inf. téc. 07-49. University of Massachusetts, Amherst, oct. de 2007.

[44] J.-N Hwang, Jai Choi, Shinil Oh y Robert II. «Query-based learning applied to partially trained multilayer perceptrons». En: *Neural Networks, IEEE Transactions on* 2 (feb. de 1991), págs. 131-136. DOI: 10.1109/72.80299.

[45] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley y Luc Van Gool. «AI Benchmark: Running Deep Neural Networks on Android Smartphones». En: *CoRR* abs/1810.01109 (2018). arXiv: 1810.01109. URL: http://arxiv.org/abs/1810.01109.

[46] M. Z. Islam, M. S. Hossain, R. ul Islam y K. Andersson. «Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation». En: *2019 Joint 8th International Conference on Informatics, Electronics Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision Pattern Recognition (icIVPR)*. Mayo de 2019, págs. 324-329. DOI: 10.1109/ICIEV.2019.8858563.

[47] C. V. Jawahar, Hongdong Li, Greg Mori y Konrad Schindler. *Computer vision - ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2-6, 2018, revised selected papers*. Springer, 2019.

[48] Pavel Jedlicka. «Sign Language Motion Capture database recorded by one device». En: 2018.

[49] Zhuoran Ji. «ILP-M Conv: Optimize Convolution Algorithm for Single-Image Convolution Neural Network Inference on Mobile GPUs». En: *arXiv e-prints*, arXiv:1909.02765 (sep. de 2019), arXiv:1909.02765. arXiv: 1909.02765 [cs.DC].

[50] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li y Li Fei-Fei. «MentorNet: Regularizing Very Deep Neural Networks on Corrupted Labels». En: *CoRR* abs/1712.05055 (2017). arXiv: 1712.05055. URL: http://arxiv.org/abs/1712.05055.

[51] Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, Richard C. Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox y Doe Hyun Yoon. «In-Datacenter Performance Analysis of a Tensor Processing Unit». En: *CoRR* abs/1704.04760 (2017). arXiv: 1704.04760. URL: http://arxiv.org/abs/1704.04760.

[52] Asifullah Khan, Anabia Sohail, Umme Zahoora y Aqsa Saeed Qureshi. «A Survey of the Recent Architectures of Deep Convolutional Neural Networks». En: *CoRR* abs/1901.06032 (2019). arXiv: 1901.06032. URL: http://arxiv.org/abs/1901.06032.

[53] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox y Bernt Schiele. «Lucid Data Dreaming for Object Tracking». En: *CoRR* abs/1703.09554 (2017). arXiv: 1703.09554. URL: http://arxiv.org/abs/1703.09554.

[54] Diederik P Kingma y Jimmy Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014).

[55] W. Kinzel y Pal Rujan. «Improving a Network Generalization Ability by Selecting Examples». En: *EPL (Europhysics Letters)* 13 (jul. de 2007), pág. 473. DOI: 10.1209/0295-5075/13/5/016.

[56] O. Koller, H. Ney y R. Bowden. «Deep Hand: How to Train a CNN on 1 Million Hand Images When Your Data is Continuous and Weakly Labelled». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2016, págs. 3793-3802. DOI: 10.1109/CVPR.2016.412.

[57] Nikola Konstantinov y Christoph Lampert. «Robust Learning from Untrusted Sources». En: *CoRR* abs/1901.10310 (2019). arXiv: 1901.10310. URL: http://arxiv.org/abs/1901.10310.

[58] Alex Krizhevsky. «Learning Multiple Layers of Features from Tiny Images». En: *University of Toronto* (mayo de 2012).

[59] Alex Krizhevsky, Ilya Sutskever y Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Advances in Neural Information Processing Systems 25*. Ed. por F. Pereira, C. J. C. Burges, L. Bottou y K. Q. Weinberger. Curran Associates, Inc., 2012, págs. 1097-1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[60] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig y Vittorio Ferrari. «The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale». En: *CoRR* abs/1811.00982 (2018). arXiv: 1811.00982. URL: http://arxiv.org/abs/1811.00982.

[61] Brenden Lake, Ruslan Salakhutdinov y Joshua Tenenbaum. «Human-level concept learning through probabilistic program induction». En: *Science* 350 (dic. de 2015), págs. 1332-1338. DOI: 10.1126/science.aab3050.

[62] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang y Zhuowen Tu. «Deeply-Supervised Nets». En: *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*. Ed. por Guy Lebanon y S. V. N. Vishwanathan. Vol. 38. Proceedings of Machine Learning Research. San Diego, California, USA: PMLR, mayo de 2015, págs. 562-570. URL: http://proceedings.mlr.press/v38/lee15a.html.

[63] Vivian Lee, Keng Hoon Gan, Tien Tan y Rosni Abdullah. «Semi-supervised Learning for Sentiment Classification using Small Number of Labeled Data». En: *Procedia Computer Science* 161 (ene. de 2019), págs. 577-584. DOI: 10.1016/j.procs.2019.11.159.

[64] Wei Lee, Chien-Wei Chang, Po-An Yang, Chi-Hsuan Huang, Ming-Kuang Wu, Chu-Cheng Hsieh y Kun-Ta Chuang. «Effective Quality Assurance for Data Labels through Crowdsourcing and Domain Expert Collaboration». En: *EDBT*. 2018.

[65] Owen T Lewis y Katherine L. Hermann. «Data for free: Fewer-shot algorithm learning with parametricity data augmentation». En: *International Conference on Learning Representations 2019*. 2019.

[66] Xuhong Li, Yves Grandvalet y Franck Davoine. «A Baseline Regularization Scheme for Transfer Learning with Convolutional Neural Networks». En: *Pattern Recognition* 98 (sep. de 2019), pág. 107049. DOI: 10.1016/j.patcog.2019.107049.

[67] Yuanzhi Li y Yang Yuan. «Convergence Analysis of Two-layer Neural Networks with ReLU Activation». En: *CoRR* abs/1705.09886 (2017). arXiv: 1705.09886. URL: http://arxiv.org/abs/1705.09886.

[68] Jeroen Lichtenauer, Emile Hendriks y Marcel Reinders. «Sign Language Recognition by Combining Statistical DTW and Independent Classification». En: *IEEE transactions on pattern analysis and machine intelligence* 30 (dic. de 2008), págs. 2040-6. DOI: 10.1109/TPAMI.2008.123.

[69] Or Litany y Daniel Freedman. «SOSELETO: A Unified Approach to Transfer Learning and Training with Noisy Labels». En: *CoRR* abs/1805.09622 (2018). arXiv: 1805.09622. URL: http://arxiv.org/abs/1805.09622.

[70] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang y Kevin Murphy. «Progressive Neural Architecture Search». En: *CoRR* abs/1712.00559 (2017). arXiv: 1712.00559. URL: http://arxiv.org/abs/1712.00559.

[71] Shuping Liu, Yu Liu, Jun Yu y Zengfu Wang. *A Static Hand Gesture Recognition Algorithm Based on Krawtchouk Moments*. Nov. de 2014. URL: https://link.springer.com/chapter/10.1007/978-3-662-45643-9_34.

[72] Tianyi Liu, Minshuo Chen, Mo Zhou, Simon S. Du, Enlu Zhou y Tuo Zhao. «Towards Understanding the Importance of Shortcut Connections in Residual Networks». En: *arXiv e-prints*, arXiv:1909.04653 (sep. de 2019), arXiv:1909.04653. arXiv: 1909.04653 [cs.LG].

[73] Pengfei Lu y Matt Huenerfauth. «Collecting a Motion-Capture Corpus of American Sign Language for Data-Driven Generation Research». En: *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing*

*for Assistive Technologies*. Los Angeles, California: Association for Computational Linguistics, jun. de 2010, págs. 89-97. URL: https://www.aclweb.org/anthology/W10-1312.

[74] Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng y Jeffrey S. Vetter. «NVIDIA Tensor Core Programmability, Performance & Precision». En: *CoRR* abs/1803.04014 (2018). arXiv: 1803.04014. URL: http://arxiv.org/abs/1803.04014.

[75] A. M. Martinez, R. B. Wilbur, R. Shay y A. C. Kak. «Purdue RVL-SLLL ASL database for automatic recognition of American Sign Language». En: *Proceedings. Fourth IEEE International Conference on Multimodal Interfaces*. Oct. de 2002, págs. 167-172. DOI: 10.1109/ICMI.2002.1166987.

[76] Luis Piñuel Moreno. *Aceleración de algoritmos de machine learning desde un enfoque arquitectónico.* Jun. de 2018.

[77] Ivars Namatevs, Kaspars Sudars e Inese Polaka. «Automatic data labeling by neural networks for the counting of objects in videos». En: *Procedia Computer Science* 149 (ene. de 2019), págs. 151-158. DOI: 10.1016/j.procs.2019.01.118.

[78] David F. Nettleton, Albert Orriols-Puig y Albert Fornells. «A study of the effect of different types of noise on the precision of supervised learning techniques». En: *Artificial Intelligence Review* 33.4 (2010), págs. 275-306. ISSN: 1573-7462. DOI: 10.1007/s10462-010-9156-z. URL: https://doi.org/10.1007/s10462-010-9156-z.

[79] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[80] Natasha Noy. *blog google*. Ene. de 2020. URL: https://blog.google/products/search/discovering-millions-datasets-web.

[81] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan y Stephen Marshall. «Activation Functions: Comparison of trends in Practice and Research for Deep Learning». En: *CoRR* abs/1811.03378 (2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378.

[82] Kyoung-Su Oh y Keechul Jung. «GPU implementation of neural networks». En: *Pattern Recognition* 37.6 (2004), págs. 1311-1314. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2004.01.013. URL: http://www.sciencedirect.com/science/article/pii/S0031320304000524.

[83] Satoshi Oyama, Yukino Baba, Yuko Sakurai e Hisashi Kashima. «Accurate Integration of Crowdsourced Labels Using Workerself-reported Confidence Scores». En: *IJCAI*. 2013.

[84] Oyebade K. Oyedotun y Adnan Khashman. «Deep learning in vision-based static hand gesture recognition». En: *Neural Computing and Applications* 28.12 (2017), págs. 3941-3951. ISSN: 1433-3058. DOI: 10.1007/s00521-016-2294-8. URL: https://doi.org/10.1007/s00521-016-2294-8.

[85] Raimundo Farrapo Pinto Junior; Ialis Cavalvante de Paula Junior. *Static Hand Gesture ASL Dataset*. 2019. DOI: 10.21227/gzpc-k936. URL: http://dx.doi.org/10.21227/gzpc-k936.

[86] Junfei Qiu, Qihui Wu, Guoru Ding, Yuhua Xu y Shuo Feng. «A survey of machine learning for big data processing». En: *EURASIP Journal on Advances in Signal Processing* 2016.1 (2016), pág. 67. ISSN: 1687-6180. DOI: 10.1186/s13634-016-0355-x. URL: https://doi.org/10.1186/s13634-016-0355-x.

[87] Facundo Quiroga, Ramiro Antonio, Franco Ronchetti, Laura Cristina Lanzarini y Alejandro Rosete. «A study of convolutional architectures for handshape recognition applied to sign language». En: *XXIII Congreso Argentino de Ciencias de la Computación (La Plata, 2017)*. 2017.

[88] Alex Ratner, Stephen Bach, Paroma Varma y Chris Ré. *Weak Supervision: The New Programming Paradigm for Machine Learning*. 2020. URL: https://hazyresearch.github.io/snorkel/blog/ws_blog_post.html.

[89] Sachin Ravi y Hugo Larochelle. *OPTIMIZATION AS A MODEL FOR FEW-SHOT LEARNING*. 2017. URL: https://openreview.net/pdf?id=rJY0-Kcll.

[90] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan y Stefan Carlsson. «CNN Features off-the-shelf: an Astounding Baseline for Recognition». En: *CoRR* abs/1403.6382 (2014). arXiv: 1403.6382. URL: http://arxiv.org/abs/1403.6382.

[91] Yuji Roh, Geon Heo y Steven Euijong Whang. «A Survey on Data Collection for Machine Learning: a Big Data – AI Integration Perspective». En: *arXiv e-prints*, arXiv:1811.03402 (nov. de 2018), arXiv:1811.03402. arXiv: 1811.03402 [cs.LG].

[92] Franco Ronchetti, Facundo Quiroga, Cesar Estrebou, Laura Lanzarini y Alejandro Rosete. «LSA64: A Dataset of Argentinian Sign Language». En: *XX II Congreso Argentino de Ciencias de la Computación (CACIC)* (2016).

[93] Franco Ronchetti, Facundo Quiroga, Laura Lanzarini y Cesar Estrebou. «Handshape Recognition for Argentinian Sign Language using ProbSom». En: *Journal of Computer Science and Technology* 16.1 (2016), págs. 1-5. ISSN: 1666-6038.

[94] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg y Li Fei-Fei. «ImageNet Large Scale Visual Recognition Challenge». En: *International Journal of Computer Vision* 115.3 (2015), págs. 211-252. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[95] Griselda SaldaÃ GonzÃ¡lez, Jorge Cerezo SÃ¡nchez, Mario Mauricio Bustillo DÃaz y Apolonio Ata PÃ. «Recognition and Classification of Sign Language for Spanish». en. En: *ComputaciÃy Sistemas* 22 (mar. de 2018), págs. 271-277. ISSN: 1405-5546. URL: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462018000100271&nrm=iso.

[96] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra y Timothy P. Lillicrap. «One-shot Learning with Memory-Augmented Neural Networks». En: *CoRR* abs/1605.06065 (2016). arXiv: 1605.06065. URL: http://arxiv.org/abs/1605.06065.

[97] Fabio Henrique Kiyoiti dos Santos Tanaka y Claus Aranha. «Data Augmentation Using GANs». En: *CoRR* abs/1904.09135 (2019). arXiv: 1904.09135. URL: http://arxiv.org/abs/1904.09135.

[98] Jürgen Schmidhuber. «Deep Learning in Neural Networks: An Overview». En: *CoRR* abs/1404.7828 (2014). arXiv: 1404.7828. URL: http://arxiv.org/abs/1404.7828.

[99] Eli Schwartz. *Few-Shot Learning in CVPR 2019*. 2019. URL: https://towardsdatascience.com/few-shot-learning-in-cvpr19-6c6892fc8c5 (visitado 01-02-2020).

[100] Eli Schwartz, Leonid Karlinsky, Rogério Schmidt Feris, Raja Giryes y Alexander M. Bronstein. «Baby steps towards few-shot learning with multiple semantics». En: *CoRR* abs/1906.01905 (2019). arXiv: 1906.01905. URL: http://arxiv.org/abs/1906.01905.

[101] Dhruv Sharma, Zahil Shanis, Chandan K. Reddy, Samuel Gerber y Andinet Enquobahrie. «Active Learning Technique for Multimodal Brain Tumor Segmentation Using Limited Labeled Images». En: *Domain Adaptation and Representation Transfer and Medical Image Learning with Less Labels and Imperfect Data*. Ed. por Qian Wang, Fausto Milletari, Hien V. Nguyen, Shadi Albarqouni, M. Jorge Cardoso, Nicola Rieke, Ziyue Xu, Konstantinos Kamnitsas, Vishal Patel, Badri Roysam, Steve Jiang, Kevin Zhou, Khoa Luu y Ngan Le. Cham: Springer International Publishing, 2019, págs. 148-156. ISBN: 978-3-030-33391-1.

[102] Evan Shelhamer, Jonathon Long y Trevor Darrell. «Fully Convolutional Networks for Semantic Segmentation». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (mayo de 2016), págs. 1-1. DOI: 10.1109/TPAMI.2016.2572683.

[103] Patrice Y. Simard, Yann A. LeCun, John S. Denker y Bernard Victorri. «Transformation Invariance in Pattern Recognition – Tangent Distance and Tangent Propagation». En: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. por Grégoire Montavon, Geneviève B. Orr y Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, págs. 235-269. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_17. URL: https://doi.org/10.1007/978-3-642-35289-8_17.

[104] Karen Simonyan y Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». En: *arXiv e-prints*, arXiv:1409.1556 (sep. de 2014), arXiv:1409.1556. arXiv: 1409.1556 [cs.CV].

[105] Quentin De Smedt, Hazem Wannous, Jean-Philippe Vandeborre, J. Guerry, B. Le Saux y D. Filliat. «3D Hand Gesture Recognition Using a Depth and Skeletal Dataset». En: *Eurographics Workshop on 3D Object Retrieval*. Ed. por Ioannis Pratikakis, Florent Dupont y Maks Ovsjanikov. The Eurographics Association, 2017. ISBN: 978-3-03868-030-7. DOI: 10.2312/3dor.20171049.

[106] Steven W. Smith. *The scientist and engineers guide to digital signal processing*. California Technical Pub., 1999.

[107] Jake Snell, Kevin Swersky y Richard S. Zemel. «Prototypical Networks for Few-shot Learning». En: *CoRR* abs/1703.05175 (2017). arXiv: 1703.05175. URL: http://arxiv.org/abs/1703.05175.

[108] Richard Socher, Milind Ganjoo, Hamsa Bastani, Osbert Bastani, Christoper Manning y Andrew Ng. «Zero-Shot Learning Through Cross-Modal Transfer». En: *Advances in Neural Information Processing Systems* (ene. de 2013).

[109] Peter Sollich. «Query construction, entropy, and generalization in neural-network models». En: *Physical Review E* 49 (5 mayo de 1994), págs. 4637-4651. DOI: 10.1103/PhysRevE.49.4637. URL: https://link.aps.org/doi/10.1103/PhysRevE.49.4637.

[110] Rupesh Kumar Srivastava, Klaus Greff y Jürgen Schmidhuber. «Training Very Deep Networks». En: *CoRR* abs/1507.06228 (2015). arXiv: 1507.06228. URL: http://arxiv.org/abs/1507.06228.

[111] Yusuke Sugomori, Bostjan Kaluza, Fabio M. Soares y Alan M. F. Souza. *Deep Learning*. Packt Publishing, 2017.

[112] Qianru Sun, Yaoyao Liu, Zhaozheng Chen, Tat-Seng Chua y Bernt Schiele. «Meta-Transfer Learning through Hard Tasks». En: *ArXiv* abs/1910.03648 (2019).

[113] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke y Andrew Rabinovich. «Going Deeper with Convolutions». En: *CoRR* abs/1409.4842 (2014). arXiv: 1409.4842. URL: http://arxiv.org/abs/1409.4842.

[114] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens y Zbigniew Wojna. «Rethinking the Inception Architecture for Computer Vision». En: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[115] Nima Tajbakhsh, Yufei Hu, Junli Cao, Xingjian Yan, Yi Xiao, Yong Lu, Jianming Liang, Demetri Terzopoulos y Xiaowei Ding. «Surrogate Supervision for Medical Image Analysis: Effective Deep Learning From Limited Quantities of Labeled Data». En: *CoRR* abs/1901.08707 (2019). arXiv: 1901.08707. URL: http://arxiv.org/abs/1901.08707.

[116] Muhammed Talo, Ulas Baloglu, Özal yldrm y U Rajendra Acharya. «Application of Deep Transfer Learning for Automated Brain Abnormality Classification Using MR Images». En: *Cognitive Systems Research* (dic. de 2018). DOI: 10.1016/j.cogsys.2018.12.007.

[117] S. Tamura y M. Tateishi. «Capabilities of a four-layered feedforward neural network: four layers versus three». En: *IEEE Transactions on Neural Networks* 8.2 (mar. de 1997), págs. 251-255. ISSN: 1941-0093. DOI: 10.1109/72.557662.

[118] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan y Quoc V. Le. «MnasNet: Platform-Aware Neural Architecture Search for Mobile». En: *CoRR* abs/1807.11626 (2018). arXiv: 1807.11626. URL: http://arxiv.org/abs/1807.11626.

[119] Mingxing Tan y Quoc V. Le. «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks». En: *CoRR* abs/1905.11946 (2019). arXiv: 1905.11946. URL: http://arxiv.org/abs/1905.11946.

[120] Sean Tao. «Deep Neural Network Ensembles». En: *CoRR* abs/1904.05488 (2019). arXiv: 1904.05488. URL: http://arxiv.org/abs/1904.05488.

[121] Lean Karlo S. Tolentino, Ronnie O. Serfa Juan, August C. Thio-ac, Maria Abigail B. Pamahoy, Joni Rose R. Forteza y Xavier Jet O. Garcia. «Static Sign Language Recognition Using Deep Learning». En: *International Journal of Machine Learning and Computing* 9 (2019), págs. 821-827.

[122] Andreas Veit, Michael J. Wilber y Serge J. Belongie. «Residual Networks are Exponential Ensembles of Relatively Shallow Networks». En: *CoRR* abs/1605.06431 (2016). arXiv: 1605.06431. URL: http://arxiv.org/abs/1605.06431.

[123] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu y Daan Wierstra. «Matching Networks for One Shot Learning». En: *CoRR* abs/1606.04080 (2016). arXiv: 1606.04080. URL: http://arxiv.org/abs/1606.04080.

[124] Yan Wang, Wei-Lun Chao, Kilian Q. Weinberger y Laurens van der Maaten. «SimpleShot: Revisiting Nearest-Neighbor Classification for Few-Shot Learning». En: *arXiv e-prints*, arXiv:1911.04623 (nov. de 2019), arXiv:1911.04623. arXiv: 1911.04623 [cs.CV].

[125] Jason W. Wei y Kai Zou. «EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks». En: *CoRR* abs/1901.11196 (2019). arXiv: 1901.11196. URL: http://arxiv.org/abs/1901.11196.

[126] Yongqin Xian, Christoph H. Lampert, Bernt Schiele y Zeynep Akata. «Zero-Shot Learning - A Comprehensive Evaluation of the Good, the Bad and the Ugly». En: *CoRR* abs/1707.00600 (2017). arXiv: 1707.00600. URL: http://arxiv.org/abs/1707.00600.

[127] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille y Quoc V. Le. «Adversarial Examples Improve Image Recognition». En: *arXiv e-prints*, arXiv:1911.09665 (nov. de 2019), arXiv:1911.09665. arXiv: 1911.09665 [cs.CV].

[128] Qizhe Xie, Minh-Thang Luong, Eduard Hovy y Quoc V. Le. «Self-training with Noisy Student improves ImageNet classification». En: *arXiv e-prints*, arXiv:1911.04252 (nov. de 2019), arXiv:1911.04252. arXiv: 1911.04252 [cs.LG].

[129] Brian Xu, Mitra Mohtarami y James R. Glass. «Adversarial Domain Adaptation for Stance Detection». En: *CoRR* abs/1902.02401 (2019). arXiv: 1902.02401. URL: http://arxiv.org/abs/1902.02401.

[130] Ibrar Yaqoob, Ibrahim Hashem, Abdullah Gani, Salimah Mokhtar, Ejaz Ahmed, Nor Anuar y Athanasios Vasilakos. «Big Data: From Beginning to Future». En: *International Journal of Information Management* 36 (dic. de 2016). DOI: 10.1016/j.ijinfomgt.2016.07.009.

[131] Sergey Zagoruyko y Nikos Komodakis. «Wide Residual Networks». En: *CoRR* abs/1605.07146 (2016). arXiv: 1605.07146. URL: http://arxiv.org/abs/1605.07146.

[132] Michal Zajac, Konrad Zolna y Stanislaw Jastrzkebski. «Split Batch Normalization: Improving Semi-Supervised Learning under Domain Shift». En: *CoRR* abs/1904.03515 (2019). arXiv: 1904.03515. URL: http://arxiv.org/abs/1904.03515.

[133] Hongguang Zhang, Jing Zhang y Piotr Koniusz. «Few-Shot Learning via Saliency-guided Hallucination of Samples». En: *CoRR* abs/1904.03472 (2019). arXiv: 1904.03472. URL: http://arxiv.org/abs/1904.03472.

[134] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin y David Lopez-Paz. «mixup: Beyond Empirical Risk Minimization». En: *CoRR* abs/1710.09412 (2017). arXiv: 1710.09412. URL: http://arxiv.org/abs/1710.09412.

[135] Zhi-Hua Zhou. «When semi-supervised learning meets ensemble learning». En: *Frontiers of Electrical and Electronic Engineering in China* 6.1 (2011), págs. 6-16. ISSN: 1673-3584. DOI: 10.1007/s11460-011-0126-2. URL: https://doi.org/10.1007/s11460-011-0126-2.

[136] Barret Zoph y Quoc V. Le. «Neural Architecture Search with Reinforcement Learning». En: *CoRR* abs/1611.01578 (2016). arXiv: 1611.01578. URL: http://arxiv.org/abs/1611.01578.

[137] Barret Zoph, Vijay Vasudevan, Jonathon Shlens y Quoc V. Le. «Learning Transferable Architectures for Scalable Image Recognition». En: *CoRR* abs/1707.07012 (2017). arXiv: 1707.07012. URL: http://arxiv.org/abs/1707.07012.