Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Estudio de arquitecturas convolucionales con propiedades de invarianza respecto a la rotación aplicado a la clasificación morfológica de galaxias

Tesis presentada para optar al título de
Licenciado en Ciencias de la Computación

Bernabé Panarello

Director: Julio C. Jacobo Berlles

Buenos Aires, Noviembre 2017

*Agradecimientos

- A mi familia por apoyarme todo este tiempo y alentarme a que termine la licenciatura.
- A mi director Julio Jacobo por haberme orientado en este trabajo.
- A Kai Lars Polsterer (Astroinformatics group, HITS) por su valiosa contribución.
- A los integrantes del grupo de imágenes del DC por su input y feedback.
- A mis compañeros del DCAP (CGTIC-NEA) por brindarme su apoyo y recursos para elaborar esta Tesis.
- Al cuerpo de docentes del DC y a la Universidad de Buenos Aires por haberme brindado una valiosa educación.
- A mis amigos y compañeros de estudio.

Abstract

La tarea de clasificación morfológica de galaxias es de vital importancia en el campo de la cosmología. La reciente explosión en la disponibilidad de imágenes proporcionadas por distintos programas como la Sloan Digital Survey ha incrementado el interés en métodos de clasificación automática. Las redes neuronales convolucionales (CNN) han probado ser una excelente base para la tarea de clasificación de imágenes. Si bien estas generan representaciones jerárquicas que son invariantes o equivariantes respecto a la traslación, lo mismo no ocurre para otras transformaciones como la rotación. Las clasificación de galaxias debe ser invariante respecto a la rotación, debido a que las imágenes a clasificar no se presentan en una orientación específica. En este trabajo se implementan y evalúan arquitecturas CNN modificadas que por su diseño prometen ciertas propiedades de invarianza y/o equivarianza respecto a la rotación. Luego se evalúa su desempeño en dos tareas de clasificación y se reportan los resultados obtenidos.

Palabras clave: Redes convolucionales, invarianza rotacional, equivarianza rotacional, clasificación morfológica de galaxias.

Galaxy classification plays a vital role in the cosmology field. Over the last years, due to an explosion in the availability of image data produced by programs like the Sloan Digital Survey, the interest for automatic classification methods has increased. Deep convolutional neural networks (DCNN) have proven to be an excellent baseline to perform image classification tasks. Although CNN's generate hierarchical representations that are invariant or equivariant to translation, the same does not hold true for other spatial transformations like rotations. Galaxy classification must be rotation-invariant because the images to classify are presented to the networks in arbitrary orientations. In this work, we implement and evaluate modified CNN architectures that, due to their design, they promise rotation invariance and/or equivariance. We evaluate their performance in two galaxy classification tasks and report the results.

Key words: Convolutional neural networks, rotational invariance, rotational equivariance, galaxy morphological classification.

Índice

Introducción **6**

1 Métodos y herramientas de hardware y software **8**
1.1 Software base y lenguajes de programación 8
1.2 Visualización de Filtros 8
1.2.1 Visualización directa 8
1.2.2 Visualización por maximización respecto a la imagen de entrada 9
1.3 Aumentación de datos de entrenamiento 10
1.4 Hardware utilizado 10

2 Redes Convolucionales **11**
2.1 Introducción 11
2.2 Convolución 11
2.3 Arquitectura AlexNet 14
2.3.1 Data Augmentation 14
2.3.2 Dropout 15

3 Arquitecturas CNN con propiedades de equivarianza respecto a la rotación **16**
3.1 Introducción 16
3.2 Equivarianza e Invarianza 17
3.3 Arquitectura base 17
3.4 Utilización de aumentación de datos 18
3.5 Invarianza rotacional explotando simetría cíclica (Cyclic Simmetry) 20
3.6 Spatial Transformer Networks 24
3.7 Rotational Transformer Layers 26

4 Clasificación de Galaxias **29**
4.1 Esquemas de clasificación de galaxias 29
4.2 Clasificación automática de galaxias 32
4.3 Trabajo previo 32

5 Experimentos con modelos de clasificadores de galaxias **34**
5.1 Descripción del Dataset 34
5.2 Prevención de overfitting 36
5.3 Métricas y detalles de entrenamiento 37
5.4 Descripción de experimentos 37
5.4.1 Clasificación binaria Elípticas/Lenticulares y Espirales 37
5.4.2 Detección de barras nucleares 38
5.5 Modelo base 38
5.5.1 Descripción 38
5.5.2 Performance 39
5.5.3 Análisis de invarianza respecto a la rotación 41
5.6 Modelo Cyclic Simmetry 42
5.6.1 Descripción 42

5.6.2 Performance e Invarianza rotacional 42
5.7 Modelo RTL 43
5.7.1 Descripción 43
5.7.2 Performance e Invarianza rotacional 43
5.8 Experimento de detección de barras 45
5.9 Análisis 45

6 Conclusiones **46**
Bibliography48

Introducción

Las galaxias son entidades celestiales unidas gravitacionalmente, compuestas de polvo, gas y billones de estrellas (además de la reciente evidencia de materia oscura). La morfología de las mismas, esencialmente su figura y apariencia general, proporcionan a los astrónomos información sobre su formación, composición y evolución. La clasificación morfológica de galaxias es de vital importancia, debido a que frecuentemente los astrónomos confeccionan o trabajan con catálogos que les permiten contrastar teorías existentes o formar nuevas conjeturas sobre los procesos físicos que gobiernan las galaxias o la formación de estrellas.

El primer esquema de clasificación y el más conocido es el denominado *Hubble tuning Fork*, introducido por Edwin Hubble en 1926. Este esquema divide principalmente las galaxias en dos grandes grupos: Elípticas y Espirales. A su vez, el esquema define categorías dentro de estos dos grupos. Habiendo sido este sistema considerado como demasiado simple, se han propuesto otros esquemas de clasificación basados en principios similares, como por ejemplo el esquema revisado Hubble-de Vaucouleurs.

El proceso de clasificación puede ser llevado a cabo manualmente por expertos mediante la inspección visual de fotografías. Sin embargo, esta tarea es lenta y ciertamente no vale la pena emplear tiempo de un astrónomo para llevarla a cabo. Recientemente, se ha producido una explosión de datos astronómicos, generados por programas como la Sloan Digital Sky Survey (SDSS), el cual se espera que produzca 50 millones de imágenes de galaxias. Esto último ha incrementado el interés en el desarrollo de métodos de clasificación automática.

Desde el punto de vista del procesamiento de imágenes, clasificar galaxias constituye un caso de estudio interesante debido a que se trata de objetos invariantes respecto a la rotación. En la práctica, esto quiere decir que los objetos a clasificar pueden presentarse en las imágenes que los contienen en cualquier orientación no conocida de antemano, lo cual añade una complejidad extra a cualquier clasificador.

Las redes neuronales convolucionales profundas (DCNN) proveen una excelente base para construir un clasificador, e incorporan por diseño la invarianza a la traslación de los objetos a reconocer. Sin embargo, no ocurre lo mismo con la invarianza a la rotación. Aunque se conozca *a priori* que los datos a clasificar poseen esta propiedad, este conocimiento no es incorporado eficientemente en una CNN estándar. Las redes aprenden filtros que luego son utilizados para detectar ciertas características, como por ejemplo bordes, pero si el dataset presenta estos patrones en distintas orientaciones (lo cual es especialmente evidente en imágenes invariantes a la rotación), la red tenderá a aprender filtros repetidos que sólo difieren en su orientación. Esto se debe a que los pesos de los filtros son compartidos en el espacio X-Y (traslación) pero no se comparten en el espacio de la rotación. La existencia de filtros repetidos supone una utilización mayor de parámetros ajustables, lo cual eleva el costo computacional utilizado además de incrementar el riesgo de incurrir en overfitting.

En diversos trabajos publicados se han propuesto variantes a la arquitectura DCNN estandar que intentan incorporar el conocimiento *a priori* de la invarianza rotacional directamente en su diseño. Dieleman *et al.*

(2016) proponen una arquitectura que permite extender el *weight sharing* al grupo de rotaciones congruentes a 90°. Jaderberg *et al.* (2015) proponen la incorporación de las denominadas *Spatial Transformer Layers (STL)*, un mecanismo de *attention* que permite a la red identificar previamente a la clasificación los pixeles de interés dentro de la imagen de entrada. Esto se logra incorporando una subred dentro de la arquitectura, la cual aprende a generar una matriz de transformación en función de una imagen que luego es utilizada para re-muestrear la imagen de entrada antes de presentarla a la primer capa de la red principal. De esta manera, la clasificación se realiza sobre un recorte de la imagen original. Dado que la ventana o grilla de *resampling* no tiene restricciones respecto a su rotación, este método tiene el potencial de permitir la clasificación invariante a la rotación, dado que la sub-imagen de entrada puede ser presentada a la red alineada con los filtros aprendidos.

El objetivo de esta tesis es evaluar, bajo condiciones similirares, distintas arquitecturas convolucionales con propiedades de equivarianza respecto a la rotación aplicadas al caso de clasificación morfológia de galaxias . Para lograr este objetivo se propuso implementar en Keras dos arquitecturas: Una versión parcial de los propuesto en Dieleman *et al.* (2016) y una modificación de las STL a la que se denominó *Rotation Transformer Layers*. Se verificaron sus propiedades de equivarianza en contraposición con una arquitectura convolucional base y seguidamente se desarrollaron clasificadores de morfología de galaxias utilizando un dataset de referencia en el ámbito de la astronomía. Adicionalmente, se compararon los resultados obtenidos con un trabajo anterior que utiliza el mimo dataset.

El presente trabajo se encuentra organizado de la siguiente manera. El capitulo 1 repasa brevemente los conceptos básicos de redes convolucionales. El capítulo 2 describe las herramientas de software y hardware utilizadas durante el desarrollo de esta tesis, incluyendo la descripción de algunas metodologías de visualización. El capitulo 3 describe las propiedades de dos tipos de arquitecturas CNN extendidas que incorporan ciertas propiedades de equivarianza respecto a la rotación, en contraposición con las arquitecturas CNN estándar. El capitulo 4 provee cierto background sobre la problemática de clasificación de galaxias. El capitulo 5 describe la aplicación de las arquitecturas introducidas en el capitulo 3 a dos problemas distintos de clasificación de galaxias. En el capítulo 6 se presentan las conclusiones finales y trabajo futuro.

1 Métodos y herramientas de hardware y software

En la presente sección se describen los métodos, software y hardware utilizados para el desarrollo de esta tesis.

1.1 Software base y lenguajes de programación

La totalidad del código correspondiente a este trabajo fue desarrollada en Python 2.7. Particularmente, se utilizaron Theano (Theano Development Team (2016)) y Keras (Chollet *et al.* (2015)) para el desarrollo y entrenamiento de los distintos modelos propuestos. El código utilizado para desarrollar la denominada capa RTL está basado en https://github.com/taoyizhi68/keras-Spatial-Transformer-Layer.

1.2 Visualización de Filtros

Se utilizaron dos técnicas complementarias de visualización de filtros de capas convolucionales, descritas a continuación:

1.2.1 Visualización directa

Dado un filtro (kernel) de una capa convolucional, esta técnica simplemente asigna un valor positivo (entre 0 y 255) a cada valor de la matriz correspondiente, obteniéndose de esta manera una imagen del tamaño del filtro (típicamente cuadrada). La figura 1.1 es un ejemplo del resultado de esta técnica. El color se debe a que se combinaron los tres filtros correspondientes a los tres canales de la imagen.

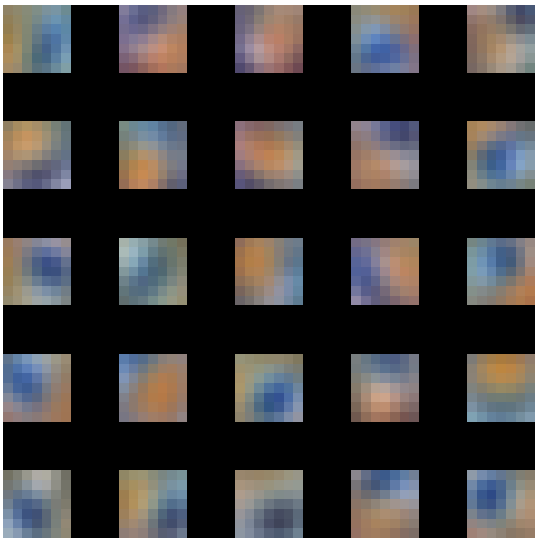

Figura 1.1: Visualización directa de filtros de una capa convolucional (la imagen está escalada)

1.2.2 Visualización por maximización respecto a la imagen de entrada

Definición 1: Se define $F_{k,j,l}$ como la matriz de pesos del k-ésimo filtro de la capa convolucional l aplicado al *feature map* generado por el filtro j de la capa l-1.

Definición 2: Se define $A_{k,j,l}(x)$ como el feature map generado por el filtro $F_{k,j,l}$ al ingresarse la imagen x a la red. $A_{k,j,l}(x) =$

Definición 3: Una imagen de activación máxima $I_{k,j,l}$ del filtro $F_{k,j,l}$ es aquella que cumple:

$I_{k,j,l} = argmax_x(l(A_{k,j,l}(x)))$

donde l se define como una función de pérdida, como por ejemplo el promedio los valores de la matriz del feature map.

En la práctica $I_{k,j,l}$ se obtiene en forma iterativa, utilizando un algoritmo de optimización como por ejemplo *ascenso de gradiente*, comenzando desde una imagen inicial I_0 (generalmente una imagen aleatoria). La figura 1.2 muestra un ejemplo de este tipo de visualización. No siempre es posible o sencillo obtener filtros representativos, dado que en muchos casos el algoritmo de optimización puede estancarse en máximos locales.

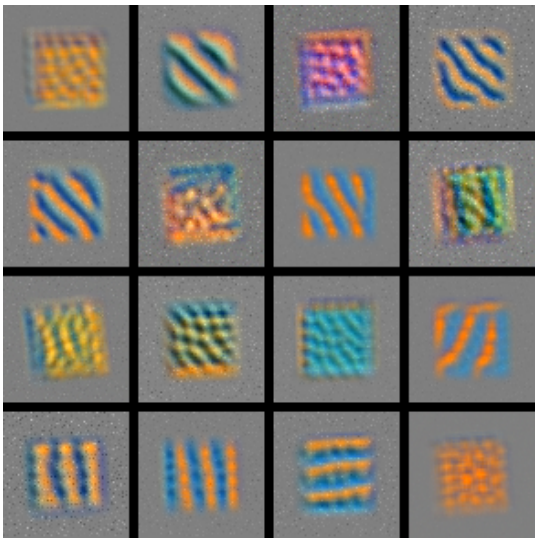

Figura 1.2: Visualización de filtros por maximización de la imagen de entrada respecto a una función de pérdida aplicada al feature map producido por el filtro.

1.3 Aumentación de datos de entrenamiento

Todos los modelos desarrollados en este trabajo fueron entrenados con imágenes generadas a partir transformaciones sobre un conjunto de imágenes mas pequeño. Para cada entrenamiento se utilizó un mínimo del orden de los 2.000.000 de imágenes generadas únicas, que equivalen aproximadamente a 26GB de datos. En lugar de efectuar las transformaciones en forma offline, lo cual requeriría que el algoritmo de entrenamiento realice constantes accesos a disco, se utilizó una librearía de aumentación en tiempo real Augmentor (DBLP:journals/corr/abs-1708-04680). Utilizando este sistema, las imágenes originales permanecen en memoria RAM y se generan batchs de imágenes aumentadas a pedido. Si el procesamiento se realiza en una GPU, Keras permite realizar la aumentación en forma asincrónica utilizando el tiempo ocioso de CPU. De esta manera es posible acelerar el tiempo de entrenamiento, evitando accesos innecesarios a disco.

1.4 Hardware utilizado

Los modelos desarrollados en este trabajo fueron entrenados utilizando un coprocesador nVidia Tesla K40c, perteneciente al cluster N.E.U.R.U.S. del Departamento de Computación de Alta Prestación (DCAP) de la Comisión Nacional de Energía Atómica (CNEA).

2 Redes Convolucionales

2.1 Introducción

Las redes convolucionales fueron introducidas por LeCun *et al.* (1989) como arquitecturas de redes neuronales para procesar más eficientemente los datos que presentan estructura de grilla. Como ejemplos de datasets con esta topología pueden citarse las señales discretas de audio en 1D o las imágenes representadas como una grilla 2D. La utilización de este conocimiento de la estructura *a priori* en una arquitectura de red especializada permite lograr las siguientes propiedades, muy deseables en sistemas de machine learning:

- Parameter sharing
- Interacciones esparsas
- Representaciones equivariantes respecto de la traslación

Formalmente, una red covolucional puede definirse como aquella que utiliza la operación de convolución en lugar de la multiplicación de matrices en al menos una de sus capas.

2.2 Convolución

La convolución en su forma discreta se define como una función de dos funciones con dominio discreto. En el caso de funciones de una sola variable, se corresponde con la siguiente definición:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$

donde x(t) y w(t) son funciones con dominio discreto.

De acuerdo a esta definición, en terminología de redes convolucionales, la función x se denomina entrada (input) mientras que la función w se denomina *kernel*.

La definición de convolución puede generalizarse también para dos dimensiones. Si además se consideran dominios finitos, puede llegarse a la siguiente expresión:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

o en forma equivalente:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(i,j).$$

En la figura 2.1 puede apreciarse graficamente un ejemplo de aplicación de un kernel o filtro a una imagen 2D. En este caso, el la función del kernel está definida por un tensor de dimensiones 2x2. El tensor resultante de la operación se denomina mapa de características (*feature map*). Cada punto del feature map resultante representa la aplicación del kernel a una región de la imagen de entrada de indénticas dimensiones al mismo. Si se restringe la operación solamente a los casos en que el kernel se aplica

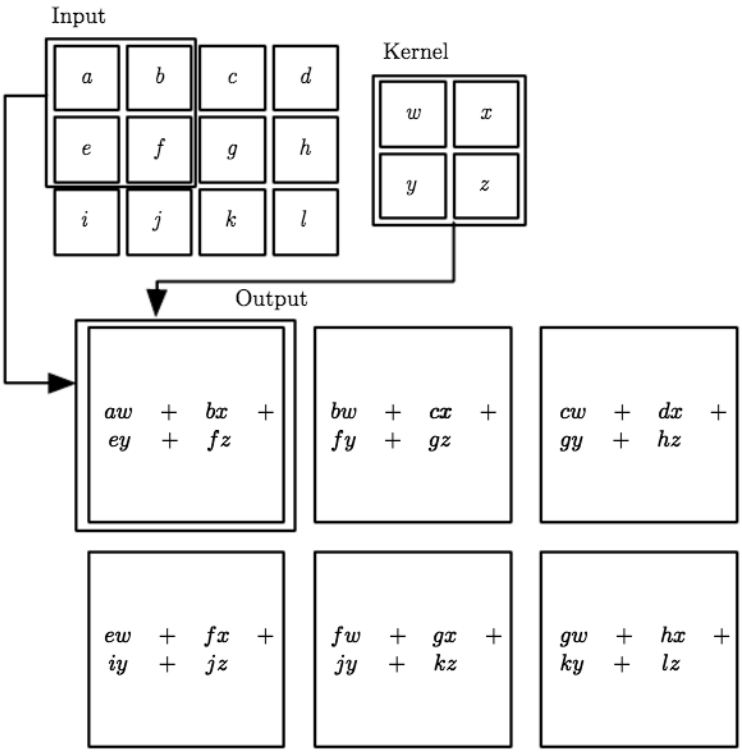

Figura 2.1: *
Ejemplo de convolución 2D. Un kernel de 2x2 es aplicado a un tensor de entrada de 3x4. Solamente se computa la salida para los casos en que el kernel cabe completamente dentro de la imagen, lo cual se denomina *convolución válida*. Fuente: Goodfellow *et al.* (2016).

únicamente dentro de los límites de la imagen de entrada, el feature map resultante tendrá dimensiones estríctamente menores a esta.

La utilización del mismo conjunto de parámetros de un kernel para aprender y detectar características locales sobre distintas posiciones (en el plano X-Y) de una imagen se denomina *parameter sharing*. En contraposición con las capas convencionales *fully connected* cada característica a detectar puede ser codificada en un solo conjunto de pesos, en lugar de ser codificada en forma redundante al aparecer en distintas posiciones en el set de entrenamiento. Esta reducción de redundancia permite además una utlización mucho menor de parámetros ajustables en la red, lo cual contribuye a reducir el riesgo de incurrir en *overfitting*. Las capas convolucionales permiten de esta manera generar mapas de características locales (2.2). Sin embargo, a diferencia de las capas *fully-connected*, no computan representaciones globales o de alto nivel de la imagen de entrada. La detección de características de más alto nivel se logra progresivamente

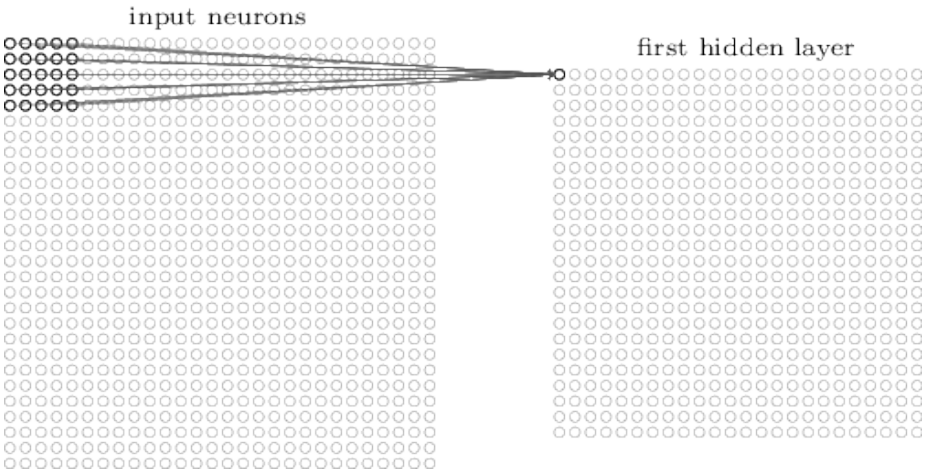

Figura 2.2: *
Ejemplo de conexión de una capa de entrada a una capa intermedia. Cada región de 5x5 se encuentra conectada a una unidad oculta mediante los pesos representados en un kernel. Fuente: Nielsen (2015).

> apilando capas convolucionales, de manera que las capas más altas van detectando progresivamente características más complejas (y más globales respecto a la imagen original). De esta manera se genera una jerarquía de detectores características.

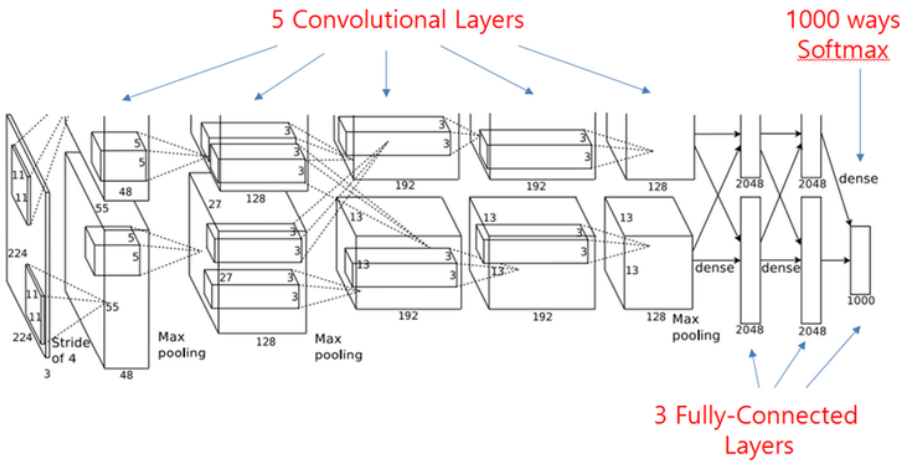

Figura 2.3: *
Arquitectura AlexNet, compuesta de 5 capas convolucionales y 3 capas fully-connected. Fuente: Krizhevsky *et al.* (2012).

2.3 Arquitectura AlexNet

La arquitectura denominada AlexNet (figura 2.3) fue introducida por Krizhevsky *et al.* (2012) con el fin de construir un clasificador para el reconocimiento de objetos en imágenes del dataset denominado ImageNet, correspondiente a la competencia ILSVRC 2012. El dataset presenta 1.2 millones de imágenes clasificadas en 1000 categorías por clasificadores humanos. El clasificador basado en esta arquitectura logró un error de clasificación sobre datos de validación de 37.5% para el top-1 (es decir, la categoría que el clasificador arroja con probabilidad más alta dada una imagen de entrada) y de 17.5% para el top-5 (es decir, que la categoría correcta se encuentre dentro de las 5 categorías más probables según el clasificador).

La arquitectura AlexNet es considerada un caso de estudio interesante dado que incorpora en un mismo modelo varias técnicas introducidas anteriormente:

- Capas convolucionales combinadas con capas de pooling
- Aumentación de datos o *data augmentation*
- Dropout

2.3.1 Data Augmentation

La técnica de aumentación de datos consiste en tomar un dataset y aumentar la cantidad de training samples mediante la alteración de los samples originales, pero preservando los labels. En el caso puntual de las imágenes, esto se logra mediante transformaciones afines (es decir, rotaciones, cambios de escala o cualquier otra operación que mantenga la

relación de paralelismo de cualquier par de lineas arbitrario de la imagen) o bien mediante la introducción de algun tipo de ruido o alteración cromática. La aumentación de datos tiene dos funciones. Por un lado, al aumentar la cantidad de samples de entrenamiento, el modelo resultante es menos propenso a presentar overfitting. En el caso de AlexNet, las variaciones espaciales utilizadas consisten en traslaciones y reflejos horizontales. Las variaciones de color son producidas mediante un algorítmo basado en PCA. Estos procedimientos se encuentran descritos en más detalle en Krizhevsky *et al.* (2012).

2.3.2 Dropout

Una forma de reducir el error de predicción de los datos de testeo es combinar las predicciones de varios modelos a la vez. Sin embargo, los recursos para hacerlo son prohibitivos para redes grandes que requieren varios días de entrenamiento. La técnica de *dropout* (Hinton *et al.* (2012)) permite de cierta manera combinar los resultados de diferentes "versiones" un modelo. Para cada unidad, se setea a 0 su salida con una determinada probabilidad (por ejemplo 0.5). Las neuronas que son "descartadas" de este modo no contribuyen a la pasada de feedforward ni participan en el back-propagation. De esta manera, cada vez que se introduce una entrada a la red, el algorítmo de entrenamiento virtualmente utiliza una arquitectura distinta, aunque los pesos son compartidos. Esta técnica evita la formación de co-adaptaciones complejas entre neuronas, dado que las mismas no pueden "confiar" en la presencia de otras unidades específicas. De esta manera, son forzadas a aprender features más robustas que son útiles para ser utilizadas en conjunto con muchos subconjuntos aleatorios de las demás neuronas. En el caso de AlexNet, la técnica de dropout es utilizada en sus primeras dos capas fully-connected. Esto reduce substancialmente el overfitting, pero también duplica el tiempo de entrenamiento.

3 Arquitecturas CNN con propiedades de equivarianza respecto a la rotación

3.1 Introducción

En el contexto de clasificación de objetos en imágenes, algunos datasets presentan la propiedad de contener objetos *invariantes* respecto a la rotación. Informalmente, esto significa que los objetos a reconocer pueden presentarse en la imagen en cualquier ángulo de rotación y su clasificación (y representaciones) deben ser invariante respecto a este. Dentro de los datasets que cumplen la propiedad de invarianza respecto a la rotación, se pueden nombrar las imágenes de galaxias o de plancton. Estos datasets generan una dificultad adicional a los modelos basados aprendizaje supervisado, dado que al momento de utilizarse los mismos para inferir una categoría o valor de regresión, la imagen de entrada podría presentar al objeto a reconocer en un ángulo distinto al aprendido por el modelo en base a los datos de entrenamiento. Para clarificar esto, puede suponerse por ejemplo que el modelo basado en CNN aprende a reconocer un determinado borde o curva distintiva del algún tipo de objeto (o subconjunto de tipos de objetos). Si al realizarse la inferencia esta característica se encuentra presente, pero en un ángulo muy distinto a la orientación con que aparece en el kernel que la detecta, este no enviará la activación correspondiente (o al menos no con un nivel de señal óptimo), perdiéndose así la detección de dicho feature. Las redes convolucionales como por ejemplo AlexNet, proveen una base muy conveniente para construir una clasificador o un modelo de regresión sobre imágenes. Más aún, permiten clasificar objetos en forma independiente a su traslación dentro de la imagen de entrada, debido a que la convolución es equivariante respecto a dicha operación. Sin embargo, sus arquitecturas no incorporan en forma nativa el conocimiento *a priori* de que los objetos pueden encontrarse en distintas orientaciones.

Lo que resta de este capítulo se organiza de la siguiente manera: Primero se define formalmente invarianza y equivarianza. Seguidamente se evalúan las propiedades de equivarianza e invarianza rotacional de tres arquitecturas distintas. Para esto se entrenan los modelos con el dataset MNIST con rotaciones (figura 3.1) y se evalúa la visualización de las representaciones de la capa convolucional 2 de cada modelo.

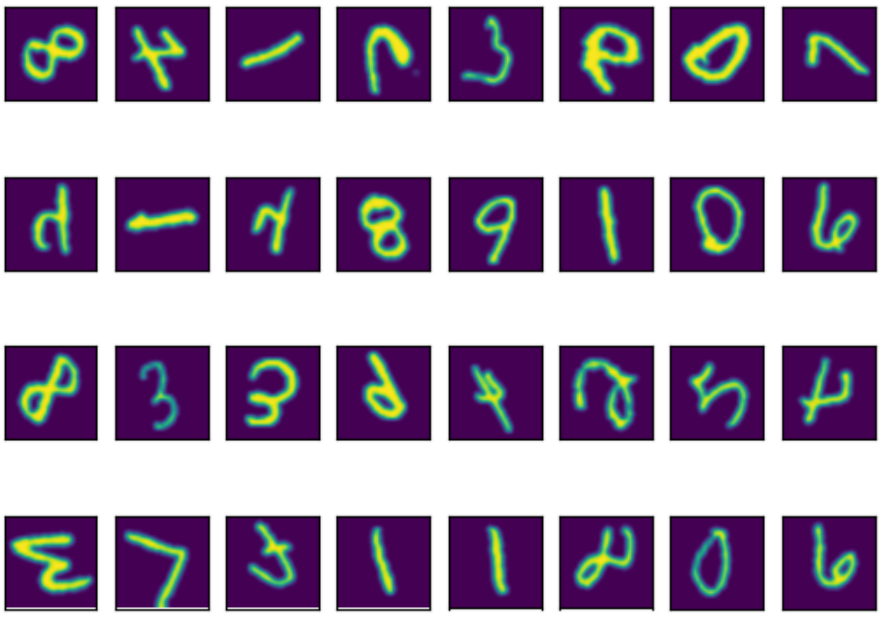

Figura 3.1: Ejemplos del dataset MNIST rotado, generado aplicando rotaciones aleatorias a las imágenes del dataset MNIST original.

3.2 Equivarianza e Invarianza

Dada una función f y una clase de transformaciones $\mathcal{T}$, f es *equivariante* con respecto a $\mathcal{T}$ si para toda transformación $T \in \mathcal{T}$ de la entrada x, se puede encontrar una transformación T' correspondiente a la salida $f(\mathrm{x})$ de manera que $f(T\mathrm{x}) = T'f(\mathrm{x})$ para todo x. (Schmidt (2012)).

En el caso particular en que se cumple $T' = T$, se dice que f es misma-equivariante (*same-equivariant*) respecto a $\mathcal{T}$.

En el contexto en que f representa un clasificador, se requiere que la salida de $f(T\mathrm{x})$ sea independiente de la transformación f, es decir, que T' sea la identidad para todo T. En este caso se dice que f es *invariante* respecto a $\mathcal{T}$.

3.3 Arquitectura base

La figura 3.2 ilustra la arquitectura convolucional de referencia que luego fue extendida para evaluar los métodos descritos en las siguientes secciones. A modo de referencia, pueden visualizarse las propiedades de equivarianza del modelo base cuando es entrenado con el dataset MNIST original (sin inducir rotaciones en el dataset de entrenamiento). La figura 3.3 muestra las activaciones generadas por la capa 2 del modelo cuando se le presenta una imagen de un mismo dígito en distintas orientaciones. Si se observan los niveles de activación, claramente puede determinarse

que en la mayoría de los casos, las representaciones no son invariantes ni misma-equivariantes. Se puede demostrar formalmente que la convolución es equivariante respecto a traslaciones pero no respecto a rotaciones (Cohen & Welling (2016)).

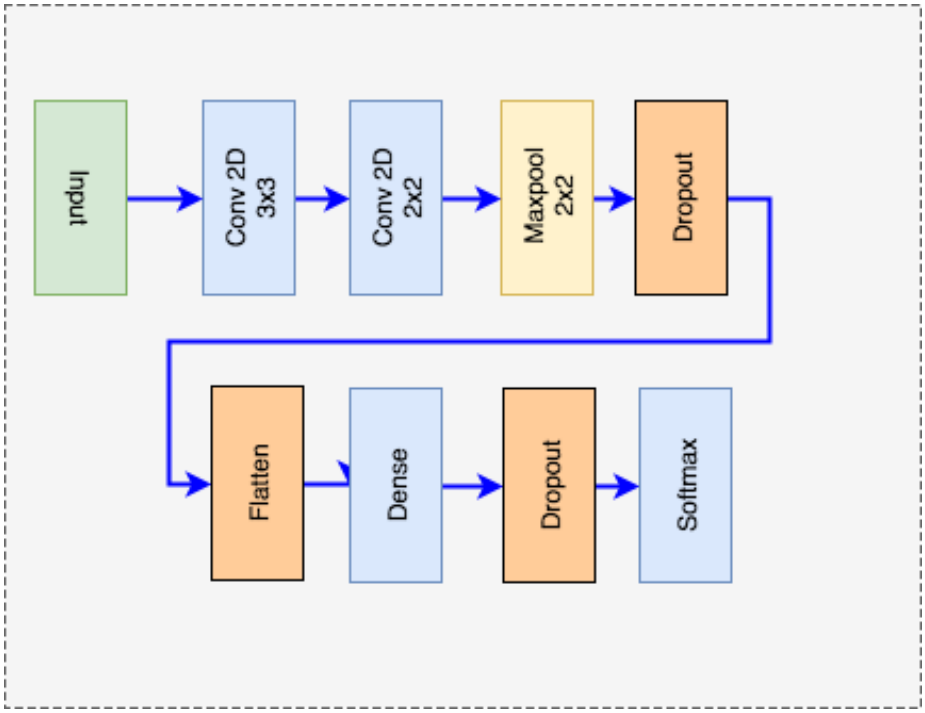

Figura 3.2: Arquitectura convolucional base para detectar dígitos MNIST

3.4 Utilización de aumentación de datos

Una de las formas más simples de introducir en un clasificador invarianza con respecto a una transformación para un determinado dataset es mediante la aumentación de datos (Simard et al 2003). Como se mencionó anteriormente, esto consiste en generar nuevos samples de entrenamiento mediante la aplicación de perturbaciones sobre los originales (en este caso, las perturbaciones se originan aplicando la transformación que se quiere codificar). Si la red tiene capacidad suficiente (es decir, un determinado número de parámetros ajustables), esta debería aprender en muchos casos las variantes de los objetos a partir de los datos introducidos. La desventaja de confiar únicamente en la aumentación de datos sin introducir ningún otro tipo de información previa en la arquitectura de la red es que requiere obviamente incrementar la cantidad de parámetros en el modelo, los cuales son utilizados para codificar las variantes rotacionales de un mismo feature. El requerimiento de más parámetros implica más costo computacional y el riesgo de incurrir en overfitting. La aumentación

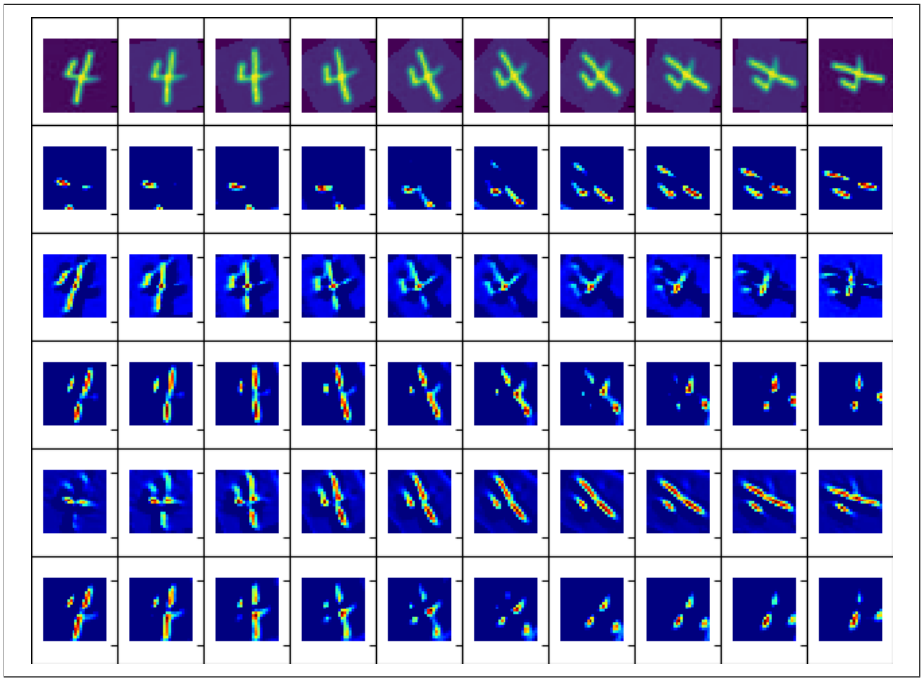

Figura 3.3: Salida (feature maps) de la capa convolucional 2 del modelo base entrenado SIN aumentación de datos cuando se le presenta el dígito 4 en ángulos de 0º a 90º. Los colores fríos representan valores de activación bajos y los cálidos valores altos.

de datos no garantiza la equivarianza de las representaciones en todos los casos, como se puede observar en la figura 3.4. Cohen & Welling (2016) demuestran también (para el caso de rotaciones cíclicas) que si la red aprende versiones rotadas de un mismo filtro, el conjunto de feature maps es equivariante respecto a las rotaciones, pero cada feature map individualmente no.

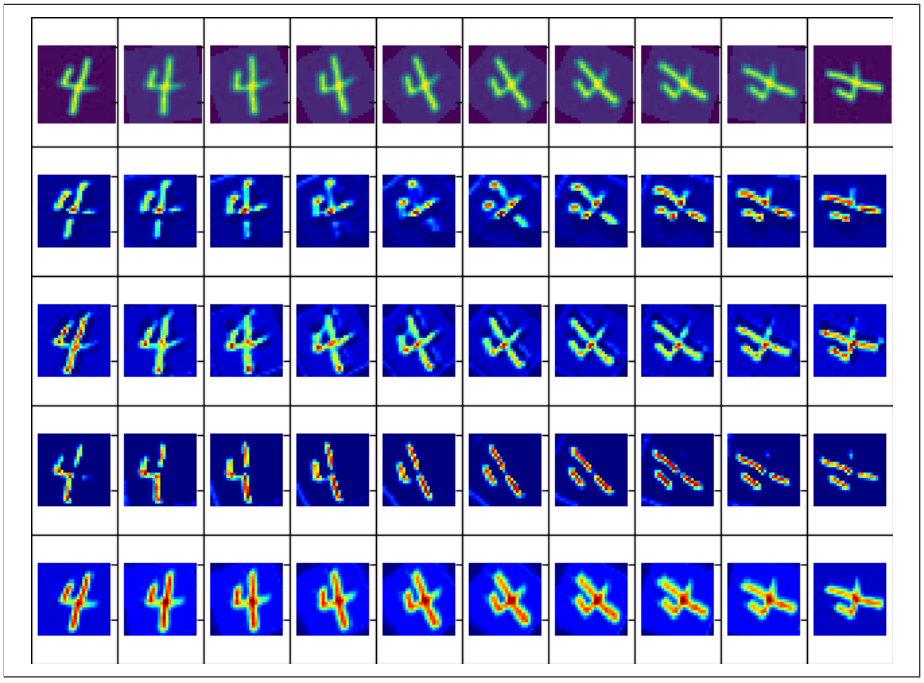

Figura 3.4: Salida (feature maps) de la capa convolucional 2 de una red entrenada con aumentación de datos cuando se le presenta el dígito 4 en ángulos del 0 al 90. Los colores fríos representan valores de activación bajos y los cálidos valores altos.

3.5 Invarianza rotacional explotando simetría cíclica (Cyclic Simmetry)

Los filtros de las CNN aprenden a detectar patrones en múltiples localizaciones de una imagen. Muchas veces esos patrones ocurren en distintas orientaciones, como por ejemplo un determinado borde. Un ejemplo de esto se puede apreciar visualizando los pesos de los filtros de la primer capa de una red CNN como AlexNet (figura 3.5). Como consecuencia de esto, las redes tienden a aprender múltiples copias de un mismo filtro en distintas orientaciones. Esto es particularmente evidente cuando los datos de entrada exhiben invarianza rotacional. Como se mencionó anteriormente, la aumentación de datos de entrenamiento tampoco garantiza equivarianza. Las CNN efectúan la detección de features (i.e. la convolución) solamente en el espacio X-Y de la imagen o feature map de entrada, pero no en el espacio de la rotación. Una forma de garantizar invarianza (dentro de un grupo de ángulos de rotación) es extender la detección de features a dicho espacio. Para ilustrar esta idea, se presenta una analogía muy simple en la figura 3.6, donde se requiere detectar una figura que puede tener dos orientaciones posibles. En el caso (a) se puede

determinar que para detectar la figura son necesarios al menos tres filtros, los cuales producen 3 feature maps. En el caso (b) se puede apreciar que para detectar la misma figura en una orientación distinta al caso (a), es necesario un filtro adicional. Por lo tanto, son necesarios cuatro filtros para detectar la figura en sus dos orientaciones posibles, pero dos de ellos son rotaciones de los otros dos. En el caso (c) se toma este conocimiento en cuenta, de manera que la detección de features se produce ahora sobre dos orientaciones distintas de la imagen de entrada. Con esto se logra detectar la figura en dos orientaciones posibles, utilizando solamente dos kernels. Como consecuencia de esto, el parameter sharing se extiende al espacio de las posibles rotaciones (en este caso, congruentes a 90^o).

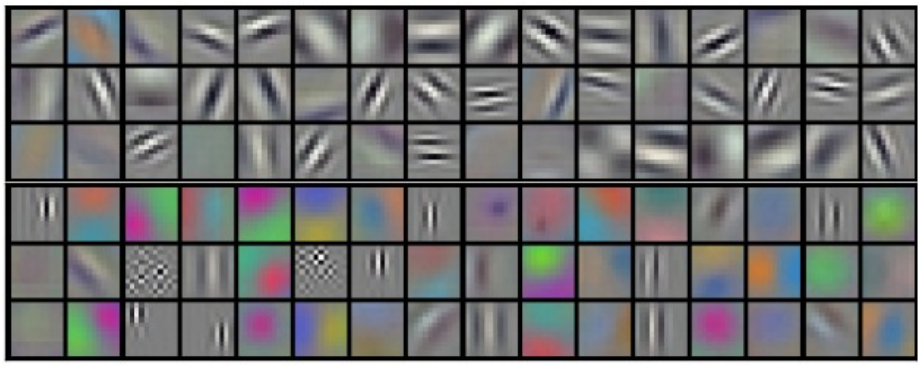

Figura 3.5: Visualización de los pesos de la primer capa convolucional de la arquitectua AlexNet entrenada con el dataset IMAGENET. Se observan grupos de patrones repetidos, generalmente con una diferencia de orientación de 90^o. Fuente: Krizhevsky *et al.* (2012).

En el trabajo Dieleman *et al.* (2016) se introduce un framework basado en principio antedicho. El autor propone cuatro operaciones sobre el flujo de datos (tensores) que pueden ser implementadas como capas adicionales en la arquitectura de la red. Cada operación afecta el tamaño del batch (el tensor 4D que ingresa a la capa), del espacio de características o de ambos. Las operaciones se restringen a rotaciones congruentes a 90^o.

La operación denominada *cyclic-slicing* toma cada feature map del batch y produce cuatro feature maps correspondiente al ciclo de rotaciones congruentes a 90^o (0^o, 90^o, 180^o, 270^o). $S(x) = [x, rx, r^2x, r^3x]$, donde x es el feature map de entrada y se utiliza la notación de vector columna para indicar que la expansión se hace sobre la dimensión del batch. Los feature maps generados son ingresados una a la vez en la capa convolucional subsiguiente, de manera que se produce cuatro *vías* de predicciones en simultáneo.

La operación *cyclic-pooling* permite combinar las predicciones generadas por las distintas vías abiertas por el slice. Para esto se aplica sobre el minibatch alguna función invariante a permutaciones, como por ejemplo el promedio. Formalmente, si $x = [x_0, x_1, x_2, x_3]$ son predicciones generadas por los cuatro caminos, $P(x) = p(x_0, x_1^- 1, x_2^- 2, x_3^- 3)$, donde p es

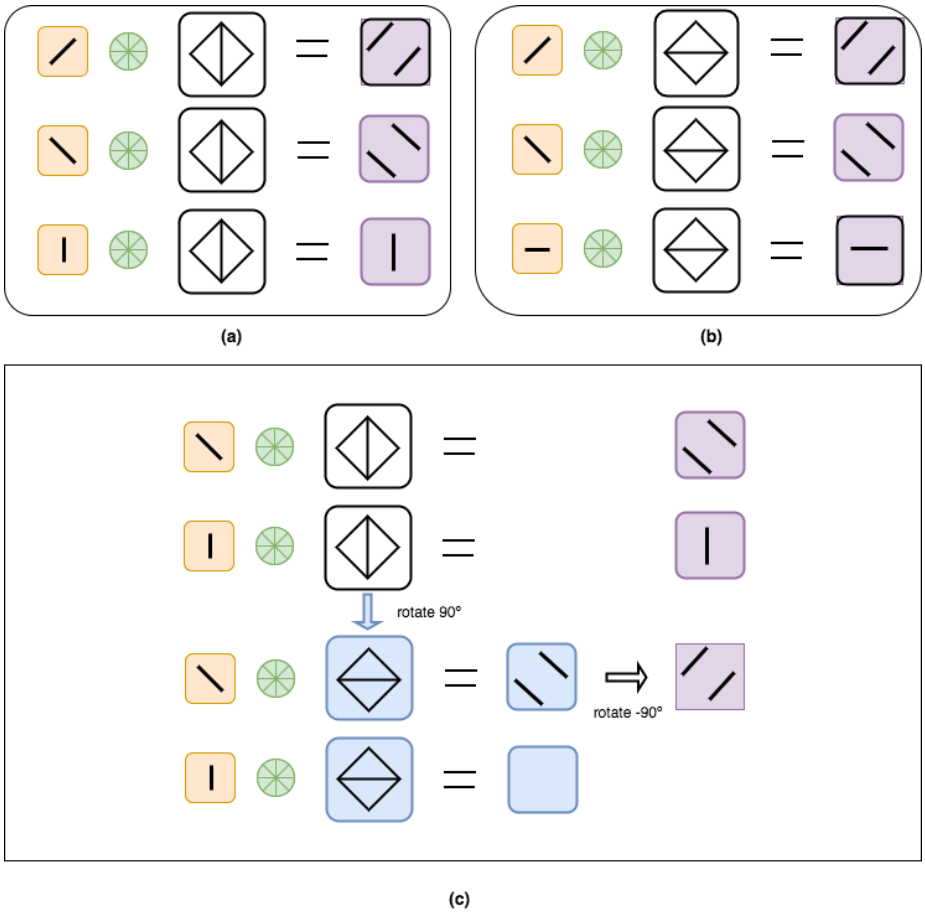

Figura 3.6: Ejemplo conceptual de la explotación de la invarianza rotacional para incrementar el parameter sharing y la cantidad de parámetros.

la operación de pooling elegida. Nótese que la transformación hace la rotación inversa correspondiente sobre los feature maps generados. Este es análogo a lo que ocurre con los feature maps celestes en el caso (c) de la figura 3.5.

Este método, por su diseño, garantiza cierta misma-equivarianza de las representaciones entre grupos de ángulos congruentes módulos 90°, como se puede observar en la figura (casos 0°/90°). 3.8.

Si por otro lado, se omiten las rotaciones inversas en la capa de cyclic-pooling, se obtienen representaciones invariantes a rotaciones cíclicas. Esto se puede hacer incluso colocando una capa de cyclic-pooling luego de una capa densa, donde la representación de salida ya no tiene una estructura espacial.

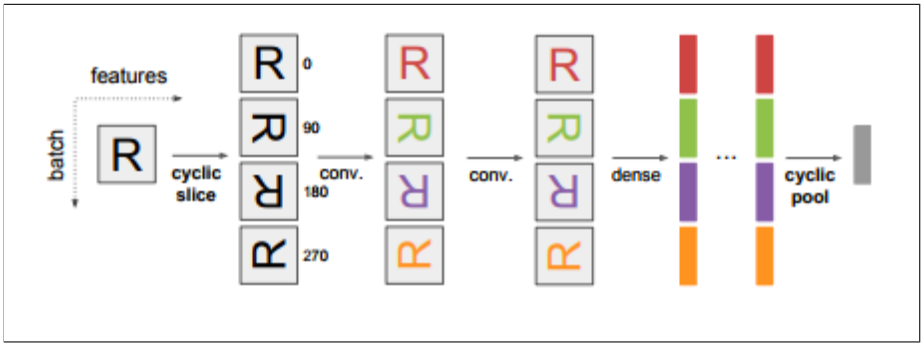

Figura 3.7: Operaciones de cycle-slicing y cycle-pooling. Un feature map ingresa a la capa de cyclic-pooling y es aumentado 4 veces en la dimensión del mini-batch. Esto multiplica los feature maps generados. Cada grupo de éstos ingresa separadamente a una capa densa y a la salida de esta se promedian las activaciones, obteniéndose una representación equivariante respecto de rotaciones cíclicas. Fuente: Reelaboración en base a Dieleman *et al.* (2016)

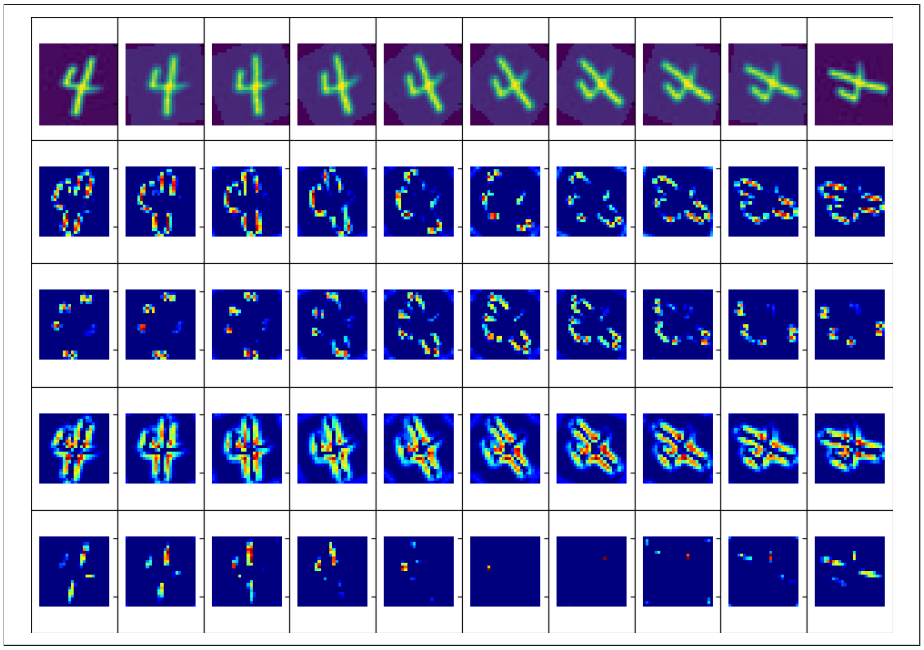

Figura 3.8: Equivarianza utilizando el método de explotación de invarianza cíclica. La equivarianza está garantizada para todos los grupos de ángulos congruentes módulo 90°.

3.6 Spatial Transformer Networks

Otra forma de lograr cierta invarianza a las rotaciones es lograr que la red aprenda a transformar la imagen de entrada antes de ingresarla a la primer capa convolucional. Las Spatial Transformer Networks (Jaderberg *et al.* (2015)) son un tipo de red convolucional que incorpora en su arquitectura un mecanismo de *attention*. Este concepto se encuentra inspirado en el funcionamiento de la visión humana para reconocer un determinado patrón. Por ejemplo, en el proceso de reconocimiento de un rostro dentro de una escena, primero tiene lugar un proceso "rápido" mediante el cual el sistema de visión detecta que efectivamente hay un rostro y cual es su posición y orientación. Una vez ubicada la atención sobre ese objeto, tiene lugar la tarea de reconocimiento fino sobre el mismo.

Siguiendo esta idea, las DCNN del tipo *Spatial Transformer Network* incorporan uno o más módulos de atención denominados *módulos de transformación espacial* (*Spatial Transformer Modules*). La figura 3.9 ilustra una arquitectura STL que incorpora uno de estos módulos.

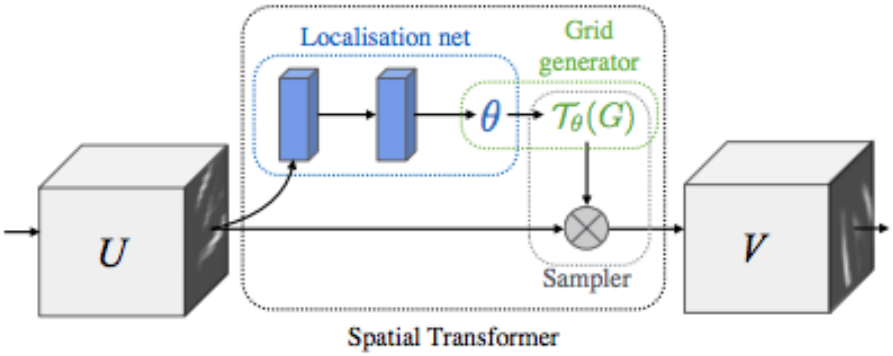

Figura 3.9: Ejemplo de arquitectura con STL. Fuente: Jaderberg *et al.* (2015)

Los módulos de transformación espacial pueden insertarse en cualquier punto de la red. Desde el punto de vista estrictamente funcional, estos constituyen una capa que se encarga de transformar la imagen (o feature map) de entrada en otra imagen conteniendo solamente la región importante para efectuar el reconocimiento en las capas subsiguientes. En el caso de attention, por ejemplo, la salida del módulo sería un recorte de la imagen original. Internamente, estos módulos se componen de una red de localización, un generador de grilla de muestreo y un sampler. La red de localización, que puede constituir una red neuronal en si misma, genera en su salida los parámetros θ de la transformación a efectuar en función de las características de la imagen de entrada U. La forma de generar estos parámetros se encuentra codificada en los pesos de la red de localización y los de las capas anteriores al módulo. Los parámetros θ obtenidos en cada caso son utilizados para generar una grilla de muestreo, aplicando T_θ sobre la grilla identidad G. La grilla resultante es utilizada por el sampler para muestrear la imagen original de entrada (fig. 3.10), lo que constituye

la salida del módulo. Se pueden diseñar módulos que apliquen cualquier tipo de transformación. Particularmente, para el caso de una transformación afin 2D $\boldsymbol{A_\theta}$, la grilla de muestreo puede calcularse punto a punto de la siguiente manera:

$$\begin{bmatrix} \boldsymbol{x_i^s} \\ \boldsymbol{y_i^s} \end{bmatrix} = \boldsymbol{T_\theta(G_i)} = \boldsymbol{A_\theta} \begin{bmatrix} \boldsymbol{x_i^t} \\ \boldsymbol{y_i^t} \\ \boldsymbol{1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta_1 1} & \boldsymbol{\theta_1 2} & \boldsymbol{\theta_1 3} \\ \boldsymbol{\theta_2 1} & \boldsymbol{\theta_2 2} & \boldsymbol{\theta_2 3} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_i^t} \\ \boldsymbol{y_i^t} \\ \boldsymbol{1} \end{bmatrix}$$

donde $\boldsymbol{(x_t, y_t)}$ son las coordenadas normales del feature map de salida, $\boldsymbol{(x_s, y_s)}$ son las coordenadas de muestreo de la imagen o feature map de entrada y $\boldsymbol{A_\theta}$ es la matriz de transformación instanciada en los parámetros de salida de la red de localización.

La única restricción respecto a las transformaciones junto con las operaciones llevadas a cabo por el sampler es que las mismas deben ser diferenciables. Esto es importante debido a que dichas operaciones forman parte de la cadena de back propagation, lo cual hace posible que la red de localización pueda aprender sus parámetros ajustables en base a la optimización de la función de costo de la red principal.

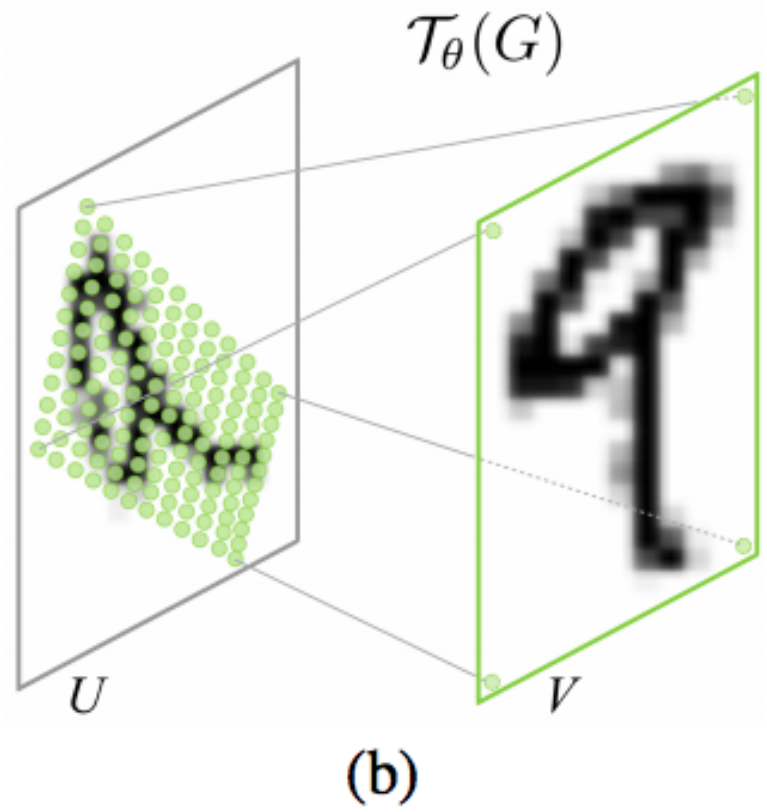

Figura 3.10: Aplicación de una grilla de resampling sobre una imagen de entrada. La grilla se genera utilizando los parámetros que genera la red de localización cuando a su entrada se ingresa la misma imagen a transformar.Fuente: Jaderberg *et al.* (2015)

3.7 Rotational Transformer Layers

En el contexto de esta tesis, se propone estudiar una STL con una transformación restringida para explotar solamente la invarianza rotacional. Si se asume que los objetos se encuentran centrados y correctamente escalados dentro de la imagen, los parámetros requeridos para la transformación se reducen a 1 en lugar de 6.

$$\boldsymbol{R_\theta} = \begin{bmatrix} \boldsymbol{cos(\theta)} & \boldsymbol{-sin(\theta)} & \mathbf{0} \\ \boldsymbol{sin(\theta)} & \boldsymbol{cos(\theta)} & \mathbf{0} \end{bmatrix}$$

En este caso, la grilla de resampling corresponde a una ventana cuadrada fija, que gira en torno al centro de la imagen original. El único parámetro $\boldsymbol{\theta}$ corresponde al ángulo de giro. Respecto a las STL generales, esta matriz permite tener un mayor control sobre el área efectiva de pixeles originales que quedan comprendidos dentro del recorte. En el contexto de este trabajo, una capa STL que utilice $\boldsymbol{R_\theta}$ como su matriz de transformación se denominará *Rotational Transformer Layer*(RTL)

Para estudiar la factibilidad de este método, se implementó el módulo RTL en Keras como una adaptación de una implementación de STL para Keras (ver capítulo "Métodos y herramientas de hardware y software") y seguidamente se lo insertó en una red de dos capas convolucionales entre la entrada y la primer capa convolucional. La red fue entrenada con el dataset MNIST rotado.

Si se visualiza la salida de la capa RTL, se puede confirmar que al ingresarse imágenes de dígitos en distintos ángulos, las mismas son rotadas de manera que el ángulo de salida tiende a estar comprendido en un rango más acotado. En el ejemplo de la figura 3.11 puede apreciarse este fenómeno visualmente.

Para contrastar el resultado visual, se realizó un análisis cuantitativo de los ángulos de salida, de manera de compararlos con los ángulos de entrada. Los gráficos de la figura 3.12 corresponden a la distribución de ángulos de la salida de la capa RTL respecto a 10 ángulos de entrada en el rango 0-90°, para el dígito 4. Puede observarse en ambos casos que la mayoría de los ángulos de salida se agrupan en un rango pequeño.

El agrupamiento de las imágenes de entrada en ángulos similares permite que las representaciones sean *aproximadamente* invariantes respecto a la rotación y por ende equivariantes. Las propiedades antedichas no pueden garantizarse, dado que dependen de un modelo probabilístico.

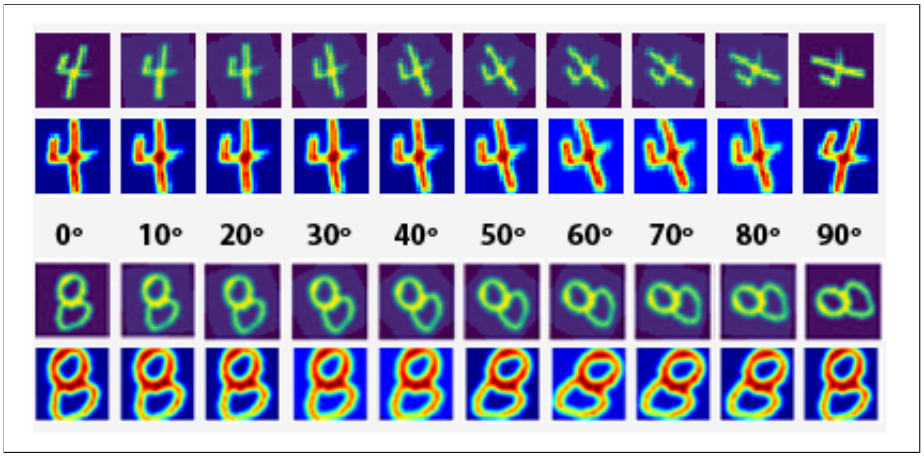

Figura 3.11: Rotaciones efectuadas por la capa RTL sobre la imagen de los dígitos 4 y 8 rotados progresivamente en 10 ángulos distintos [0-90]. Se puede observar como las orientaciones de salida tienden a agruparse.

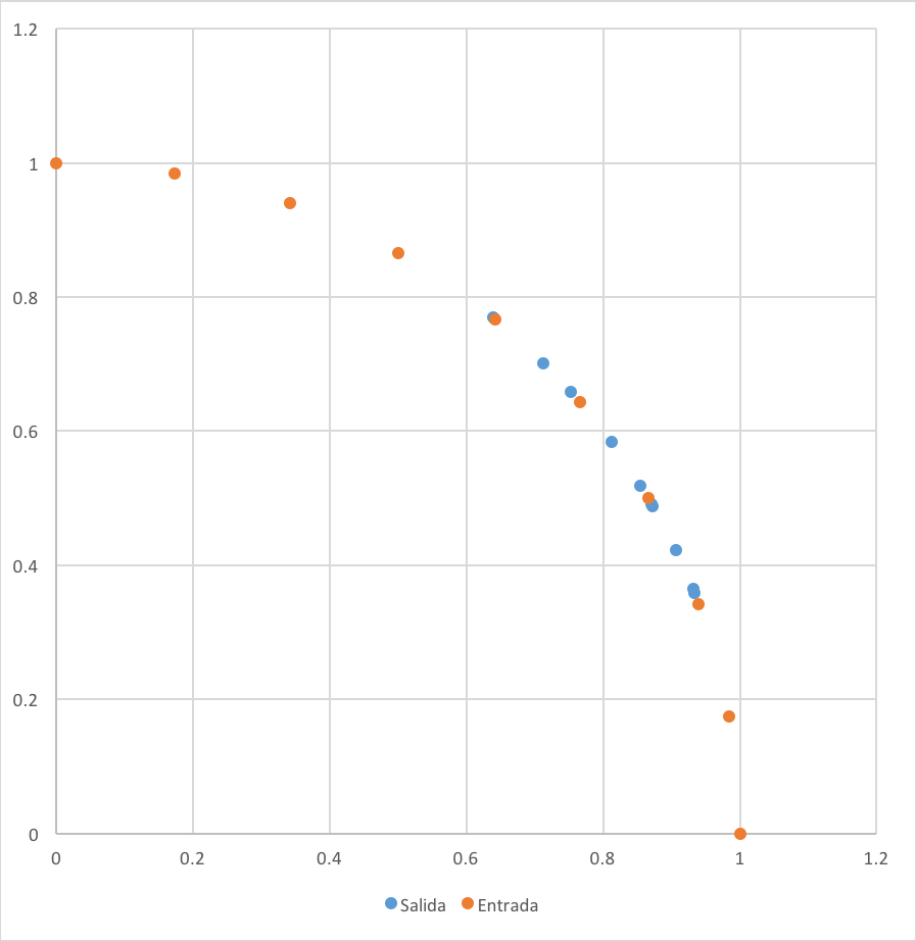

Figura 3.12: Distribución de ángulos de rotación de entrada y de salida

Ángulo de entrada (grados)	Corrección (grados)	Ángulo de salida (grados)
0	21.04	21.04
10	11.36	21.36
20	5.01	25.01
30	-0.78	29.22
40	-10.57	29.42
50	-18.74	31.25
60	-24.32	44.52
70	-25.47	50.32
90	-29.67	41.23

Tabla 3.1: Ángulos de entrada, corrección (salida de la capa de localización) y ángulo final de salida.

4 Clasificación de Galaxias

Las galaxias son entidades celestiales unidas gravitacionalmente, compuestas de polvo, gas y billones de estrellas (además de la reciente evidencia de a presencia de materia oscura). La morfología de las mismas, esencialmente su figura y apariencia general, proporcionan a los astrónomos información sobre su formación, composición y evolución. La clasificación morfológica de galaxias es de vital importancia, debido a que frecuentemente los astrónomos confeccionan o trabajan con catálogos que les permiten contrastar teorías existentes o formar nuevas conjeturas sobre los procesos físicos que gobiernan las galaxias o la formación de estrellas.

4.1 Esquemas de clasificación de galaxias

El primer esquema de clasificación y el más conocido es el denominado "Hubble tuning Fork" figura(4.1), introducido por Edwin Hubble en 1926. Este esquema divide principalmente las galaxias en dos grandes grupos: Elípticas y Espirales. A su vez, dentro de las espirales define dos subgrupos: Las que contienen una barra en su núcleo (barradas) y las que no o. Las galaxias dentro del grupo de las elípticas se clasifican basicamente por su elipticidad, mientras que las espirales por que tan cerrados o abiertos se encuentran su brazos.

Al haberse considerado el sistema Hubble Fork como demasiado simple, se han propuesto otros esquemas de clasificación basados en principios similares, como por ejemplo el esquema Hubble-de Vaucouleurs, en el cual está basado el catálogo EFIGI (Baillard, A. *et al.* (2011)), utilizado como base de los experimentos realizados en este trabajo. De Vaucouleurs sontenía que el subesquema bidimensional de galaxias espirales definido originalmente por Hubble no representaba el rango completo de galaxias observadas. Particularmente, sostenía que los "anillos" y "lentes" eran componentes estructurales importantes para este tipo de galaxias. Para complementar el esquema original, De Vaucouleurs introdujo una clasificación más detallada de las galaxias espirales, basada en las siguientes características:

- Barras: Además de la clasificación original SB denotando la presencia de barras nucleares, se agregó la categoría complementaria SA que indica la ausencia total de las mismas. A su vez, se introdujo una categoría intermedia SAB, indicando galaxias con barras no tan pronunciadas. Por otro lado, se introdujeron las galaxias lenticulares SA0 (sin barra) o SB0 (con barra), dejandose la notación S0 para galaxias donde este atributo no se puede determinar (por ejemplo, en los casos donde las galaxias se presentan de perfil (Figura 4.2).
- Anillos: Las galaxias son divididas entre las que poseen una estructura anular (denotadas r) y las que no (denotadas s). También existe una categoría de transición denotada rs.
- Brazos de la espiral: Al igual que en el esquema original de Hubble, se clasifican los brazos de acuerdo a su apertura. El esquema de De Vaucouleurs extiende la clasificación de los brazos, añadiendo así varias categorías de galaxias espirales.

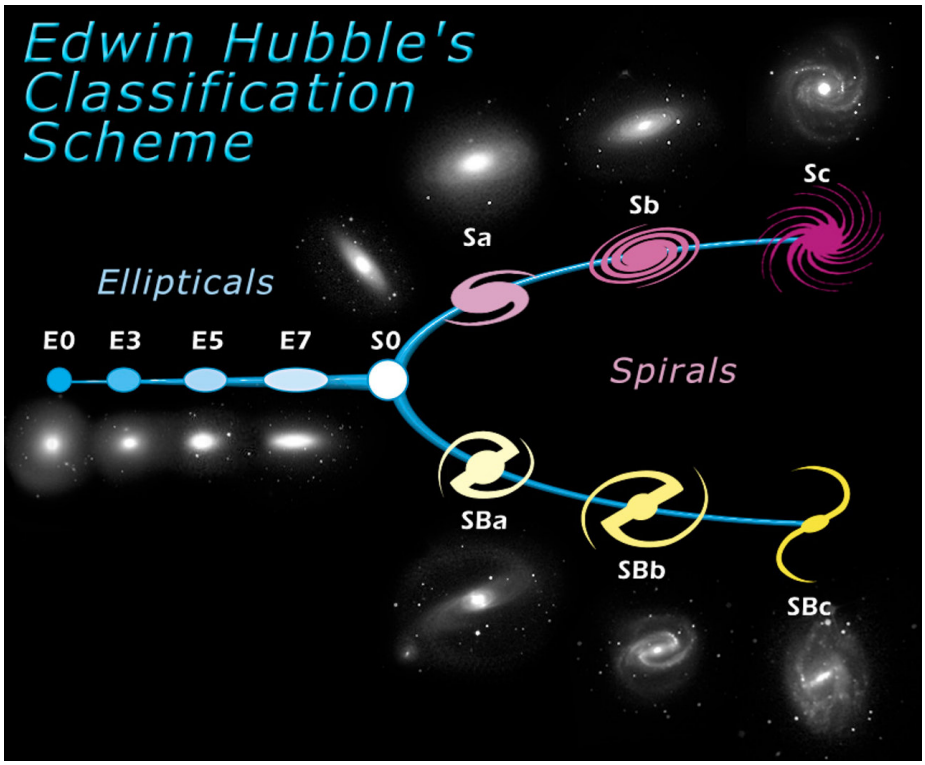

Figura 4.1: Esquema de clasificación "Hubble tuning Fork". Fuente: www.spacetelescope.org.

Figura 4.2: Galaxia vista de perfil. Fuente: National Astronomical Observatory of Japan.

- Sd (SBd), brazos rotos o difusos, formados por cúmulos individuales de estrellas y nebulosas. Bulbo ténue.
- Sm (SBm), de apariencia irregular, sin bulbo
- Im, altamente irregular

La mayoría de las galaxias pertenecientes a estas clases serían clasificadas como irregulares en el esquema original de Hubble. Adicionalmente, la clase Sd contiene algunas instancias clasificadas como Sc en el esquema original. Las galaxias en las clases Sm e Im suelen denominarse "Espirales Magallánicas" e "irregulares", respectivamente, las primeras debido a las nubes de Magallanes. La gran nube de Magallanes es de tipo SBm mientras que la pequeña es de tipo irregular (Im).

De Vaucouleurs asignó también valores numéricos a cada clase de su esquema. Los valores de esta escala, denominada Escala Hubble T, van desde -6 hasta +10, donde los números negativos corresponden a tipos de galaxias "tempranas" (elípticas y lenticulares, por su ubicación en el esquema) mientras que los números positivos corresponden a tipos tardíos (espirales e irregulares) [1].

[1] Los términos "early type" y "late type" fueron originalmente concebidos por Edwin Hubble, quien había postulado que las galaxias elípticas podían evolucionar hacia galaxias espirales, lo cual fue refutado posteriormente

Hubble Stage	-6	-5	-4	-3	-2	-1
de Vaucouleurs class	cE	E	$\boldsymbol{E^+}$	$\boldsymbol{S0^-}$	$\boldsymbol{S0^0}$	$\boldsymbol{S0^+}$
approximate Hubble class	E			S0		

Hubble Stage	0	1	2	3	4	5	6	7	8	9	10
de Vaucouleurs	S0a	Sa	Sab	Sb	Sbc	Sc	Scd	Sd	Sdm	Sm	Im
Hubble	S0a	Sa	Sa-b	Sb	Sb-c	Sc			Sc-Irr	Irr I	

4.2 Clasificación automática de galaxias

El proceso de clasificación de galaxias puede ser realizado manualmente por expertos mediante la inspección visual de fotografías. Sin embargo, esta tarea es lenta y ciertamente no vale la pena emplear tiempo de un astrónomo para llevarla a cabo. Recientemente, se ha producido una gran explosión de datos astronómicos, generados por programas como la Sloan Digital Sky Survey (SDSS), el cual se espera que produzca 50 millones de imágenes de galaxias. Esto último ha incrementado el interés por el desarrollo de métodos de clasificación automática.

Desde el punto de vista del procesamiento de imágenes, clasificar galaxias constituye un caso de estudio interesante debido a que se trata de objetos invariantes respecto a la rotación. Esto quiere decir que los objetos a clasificar pueden presentarse en las imágenes que los contienen en cualquier ángulo no conocido de antemano, lo cual añade una complejidad extra a cualquier clasificador.

4.3 Trabajo previo

La clasificación morfológica de galaxias es una de las aplicaciones más frecuentes de las redes neuronales en el campo de la astronomía. La mayoría de estos trabajos previos se basan en el preprocesamiento previo de las imágenes de manera de extraer ciertas features manualmente que luego tienen cierto valor discriminativo para determinar la morfología, como por ejemplo la elipticidad, concentración, brillo de la superficio y radios. De especial interes para esta tesis son los modelos que intentan extraer features directamente desde los pixeles de las imágenes.

- Polsterer *et al.* (2012) proponen un clasificador del catálogo EFIGI para ciertas clases morfológicas basado en la aplicación SVM directamente sobre los pixeles crudos de la imagen de entrada. También proponen utilizar el mismo modelo para la extracción de features que luego sirvan como entrada para clasificadores basados en redes neuronales o árboles de decisión.
- Polsterer *et al.* (2016) utilizan un algorítmo no supervisado basado en mapas de Kohonen. Para lograr la invarianza rotacional, diseñaron una función de similaridad que examina un rango de rotaciones posibles de la imagen de entrada, de manera de seleccionar la orientación que maximiza la función. Este proceso tiene la ventaja de ser facilmente paralelizable.
- Dieleman *et al.* (2015) utilizan una arquitectura DCNN que logra cierta invarianza rotacional extendiendo el parameter sharing de los

filtros a un conjunto fijo de 16 transformaciones de la imagen de entrada. Esta técnica se encuentra explicada en detalle en la sección 3.5. El modelo implementa una función de regresión y es entrenado con datos del catálogo GalazyZoo.

5 Experimentos con modelos de clasificadores de galaxias

El objetivo de los experimentos descritos en esta sección fue evaluar el desempeño de distintos tipos de arquitecturas de redes convolucionales para la tarea de reconocimiento de galaxias. Debido a la invarianza rotacional de los datos a clasificar, se propuso evaluar el potencial de los métodos de explotación de invarianza cíclica (cyclic-simmetry) y rotation transformer layers (RTL) descritos en la sección 3. Las arquitecturas evaluadas extienden una arquitectura convolucional base. La arquitectura base también es evaluda, de manera que en total se comparan tres modelos. El formato de los experimentos esta basado en Polsterer *et al.* (2012), con el fin de contar con un estándar de comparación.

5.1 Descripción del Dataset

El dataset elegido para realizar los experimentos fue el catálogo EFIGI (Baillard, A. *et al.* (2011)). Este catálogo cuenta con 4458 imágenes de galaxias cercanas, las cuales han sido clasificadas por once expertos independientes, siguiendo un esquema similar al de de Vaucouleur. El catálogo proporciona además otras métricas que pueden ser explotadas en clasificadores futuros, así como también imágenes en diferentes bandas (además de RGB). La desventaja obvia es el tamaño del dataset, lo cual obliga a implementar diferentes técnicas con el fin de reducir la posibilidad de incurrir en overfitting. Para los experimentos descritos en esta tesis se han utilizado las imágenes RGB junto con el atributo "EFIGI Morphology Type", el cual define 18 clases morfológicas.

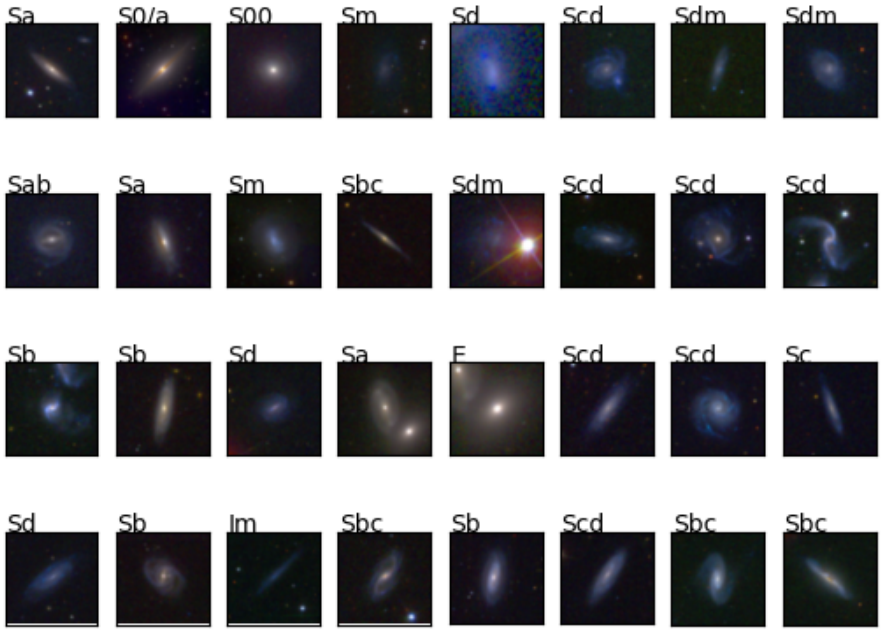

Figura 5.1: Galaxias del catálogo EFIGI junto con su clasificación

Otra característica del dataset EFIGI que se tuvo en cuenta es que la cantidad de muestras de cada clase se encuentra fuertemente desbalanceada. Particularmente, la cantidad de muestras que contienen galaxias de los tipos espiral conforman el 70% del dataset (ver tabla 5.1). Para normalizar esta situación, se utilizó el procedimiento descrito en el pseudocódigo 5.1:

El tamaño original de las imágenes del catálogo es 255x255 pixels. Debido a que esto representa un costo computacional elevado y una enorme cantidad de parámetros ajustables, se escalaron las imágenes al tamaño 69x69, de manera que cada imagen RGB representa un tensor de tamaño 69x69x3. También se normalizaron los valores de los pixels de manera que se encuentren en el rango $[\mathbf{0-1}]$.

Type	Elípticas			Lenticulares			Espirales										Im	dE
Label	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11
EMT	cE	E	cD	$\boldsymbol{S0^-}$	$\boldsymbol{S0}$	$\boldsymbol{S0^+}$	$\boldsymbol{S0/a}$	Sa	Sab	Sb	Sbc	Sc	Scd	Sd	Sdm	Sm	Im	dE
Qty	18	227	44	189	196	152	196	257	219	517	472	303	448	285	355	263	248	69

Tabla 5.1: Las 18 clasificaciones posibles del catálogo EFIGI junto con la cantidad de miembros de cada una, agrupadas en cinco supraclases

```
def get_balanced_dataset(class_map)
    # Entrada:
    #     class_map: Mapeo que agrupe las clases originales y
    #      las que se quieren utilizar para clasificar
    #      Ejemplo:  {0:0,1:0,2;0,3:1,4:1, ...}
    # Salida: Imagenes y labels aumentados

    #   Obtener imagenes y labels
    X, y = load_data()
    # Reasignar labels
    y2 = reassign (y, class_map)
    # Obtener cantidad de elementos de la clase con mas miembros
    z = max (bincount (y2))
    # Aumentar el resto de las clases para que equiparen
    # la cantidad de miembros replicando elementos seleccionados
    # en forma aleatoria
    y_bal = y2
    X_bal = X
    for c in class_map.keys:
        X_bal, y_bal = inflate_class(X_bal, y_bal, c, z)

    shuffle (X_bal, y_bal)
        return X_bal, y_bal
```

Listing 5.1: Pseudocódigo de balanceo del dataset

5.2 Prevención de overfitting

En todos los modelos, dado el tamaño relativamente pequeño del dataset, se puso especial énfasis en la utilización de técnicas para evitar el overfitting, las cuales incluyeron:

- Aumentación de datos en tiempo real: En cada iteración, las imágenes de cada minibatch se generan eligiendo un conjunto aleatorio de imágenes de entrenamiento, al cual se le aplican en forma aleatoria una o más de las siguientes perturbaciones:
  - Rotación aleatoria en el rango [-45°,45°]
  - Flip horizontal o vertical
  - Rotaciones rectangulares [90°,180°,270°]
  - Zoom en el rango [1.1,-1.3]
  - Inclinación [-10°,10°]
  - Descarte aleatorio de información de color (pasar a escala de grises)
  - Variaciones de brillo [90%,110%]

  De esta manera, durante el entrenamiento se le presentan a la red siempre imágenes únicas. Cabe aclarar que la aumentación con rotaciones aleatorias tiene como fin adicional inducir en el modelo los

patrones en distintos ángulos de orientación, independientemente de como éste los codifique.

- Dropout: Las capas de dropout se utilizaron fuertemente, acopladas a la mayoría de las capas de la red. Para el caso de las capas fully-connected se utilizó un drop-factor de 0.5, mientas que para las convolucionales se utilizó 0.2 (aunque, según Nielsen (2015) esto podría no ser necesario).

5.3 Métricas y detalles de entrenamiento

Se llevaron a cabo pruebas con dos optimizadores:

- SGD con learning rate variable programado 0.4,0.04 y 0.004 en modo similar a Dieleman *et al.* (2015).
- Adaptive Learning Rate Method (ADADELTA) (Zeiler (2012))

La métrica utilizada para evaluar la precisión del modelo es la denominada *categorical accuracy*, la cual se mide sobre predicciones que toman la forma de un vector de categorías (como una salida softmax). Dado un vector de vectores de categorias $\boldsymbol{y_{true}}$ conteniendo los valores reales y un tensor $\boldsymbol{y_{pred}}$ conteniendo las predicciones correspondientes, la métrica se calcula de la siguiente manera:

$$\boldsymbol{ca} = \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{n}} \boldsymbol{equal}(\boldsymbol{argmax}(\boldsymbol{y_{true}}[\boldsymbol{n}]), \boldsymbol{argmax}(\boldsymbol{y_{pred}}[\boldsymbol{n}]))$$

donde argmax(b) toma un vector de categorías b y devuelve la posición que contiene el mayor valor y la función equal devuelve 1 en caso de que sus argumentos sean iguales o 0 en caso contrario. Los valores reportados en este trabajo fueron convertidos a porcentaje.

Se dividió el dataset en dos conjuntos, empleando el 60% de los datos para entrenamiento y el 40% para validación. Esta elección es similar a la división en partes iguales propuesta por Polsterer *et al.* (2012). Parece razonable contar con un test set proporcionalmente grande, debido a que algunas clases de galaxias tienen una representación muy baja en el dataset. Se utilizó 5-fold cross validation durante el entrenamiento.

El tamaño de minibatch utilizado es 32. La utilización de tamaños mayores parece estancar el entrenamiento tempranamente. Se ejecutan 100 pasos por cada epoch. En todos los experimentos se entrenó la red mostrándole un máximo de 8.000.000 imágenes. Sin embargo, se observó que en la práctica el modelo llega a su precisión de validación máxima (antes de comenzar a producirse overfitting) aproximadamente en la epoch 1500, lo cual corresponde a 4.800.000 imágenes.

5.4 Descripción de experimentos

5.4.1 Clasificación binaria Elípticas/Lenticulares y Espirales

Este formato de experimento fue el utilizado como métrica para comparar la performance de los modelos propuestos. El clasificador distingue dos

categorias de galaxias: Elípticas y lenticulares como una clase y espirales como la otra (de acuerdo a la tabla 5.1). Las galaxias de tipo irregular y magallánicas quedaron excluidas de este experimento. De esta manera, el tamaño del dataset se reduce a 4141 imágenes.

5.4.2 Detección de barras nucleares

Además de la tarea de clasificación de galaxias, se utilizan las mismas arquitecturas para la tarea de predecir si una galaxia tiene una barra o no en su núcleo. A diferencia de la tarea de clasificación binaria, en este caso el dataset se encuentra balanceado, por lo que no es necesario utilizar el algoritmo de balanceo. Se hizo un mapeo del atributo "bar_length" del catálogo, el cual reporta la longitud estimada de la barra en caso de ser visible (o 0 si no lo es). Se consideraron galaxias con barra a aquellas que tienen un valor mayor a 0 para dicho atributo. Se excluyeron las galaxias que no tienen este atributo determinado. Esto resulta en un dataset con 1992 galaxias del tipo "con barra" y 1886 del tipo "sin barra".

5.5 Modelo base

5.5.1 Descripción

El denominado modelo base (figura 5.2) conforma una arquitectura de 4 capas convolucionales, 2 fully-connected y una salida softmax de un número de vías variables según la tarea de clasificación. En la entrada del modelo ingresan tensores de imágenes de tamaño (B,69,69,3), donde B representa el tamaño del minibatch. Los parámetros de la red fueron ajustados manualmente mediante la experimentación, tomando como referencia el trabajo Dieleman *et al.* (2015). Se busco evitar la utilización de funciones de activación como sigmoid y tangente hiperbólica, debido a que en las primeras versiones del modelo estancaban el aprendizaje por el problema de *vanishing gradient*.

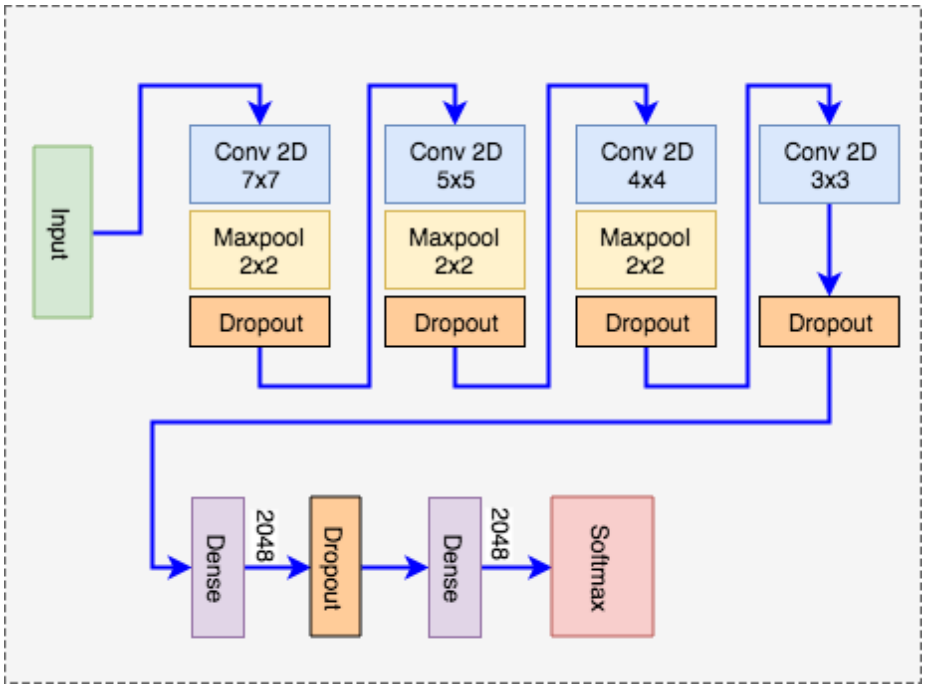

Figura 5.2: Arquitectura convolucional base

Capa	Tipo	Tamaño	Activación
Conv 1	Convolucional	32x7x7	ELU
Conv 2	Convolucional	64x5x5	ELU
Conv 3	Convolucional	128x4x4	ELU
Conv 4	Convolucional	256x3x3	ELU
Dense 1	FC	2048	RELU
Dense 1	FC	2048	Linear
Output 1	FC	2	Softmax

Tabla 5.2: Detalle de la configuración de las capas de la arquitectura base

5.5.2 Performance

El modelo base, para la tarea de clasificación binaria, alcanza una precisión en validación promedio del $\mathbf{94.00 \pm 0.2\%}$. La tabla 5.3 detalla el porcentaje de errores para cada clase

Un análisis más detallado de las clases originales correspondientes a las galaxias mal clasificadas, revela que la mayoría de los errores se producen para los tipos de galaxias intermedios que se encuentran en la frontera entre los dos grupos (tabla 5.3). Esto tiene sentido si se considera que el esquema de clasificación define una progresión de un grupo al otro.

True \ Pred	E+L	S
E+L	1475 (92.1%)	125(7.9%)
S	79(4.9%)	1521(95.1%)

Tabla 5.3: Matriz de confusión correspondiente a la clasificación de 3200 galaxias (Elípticas y Lenticulares vs. Espirales). Los datos corresponden al set de validación balanceado.

cE	E	cD	$S0^-$	$S0$	$S0^+$	$S0/a$	Sa	Sab	Sb	Sbc	Sc	Scd	Sd	Sdm	Sm
0	2	0	1	11	11	35	22	2	1	0	0	0	0	1	0

Tabla 5.4: Clases verdaderas correspondientes a ejemplos mal predichos.

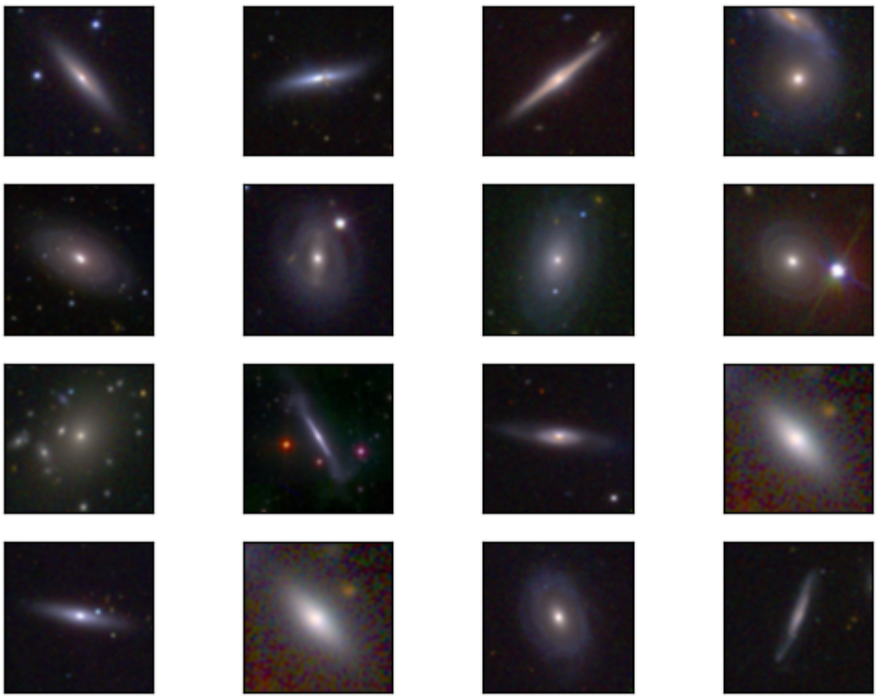

Figura 5.3: Algunos ejemplos de galaxias mal clasificadas

La figura 5.3 ilustra algunos ejemplos de galaxias que fueron mal clasificadas.

5.5.3 Análisis de invarianza respecto a la rotación

Se llevó a cabo un análisis para determinar que tan robusto a la invarianza rotacional es este modelo. Para todas las galaxias distintas del test de validación, se analizaron las predicciones para la secuencia de 19 ángulos de rotación con incrementos de 10° [0°=180°]. El análisis reveló que el modelo cambia su predicción para 103 de l534 galaxías (aproximadamente un 7%) de acuerdo al ángulo con que se presentan las misma. Si se analizan los patrones con que las predicciones cambian en función del ángulo, se encuentran en la mayoría de los casos "ráfagas" pero también existen casos donde la clasificación falla para un ángulo puntual. Estas inconsistencias están correlacionadas con la inprecisión de las predicciones. Si se analizan las galaxias mal clasificadas por el modelo, aproximadamente un 70% corresponde al grupo de las que tienen predicción inconsistente. Esto supone un margen de mejora de aproximadamente 3%, si se conociera de antemano el ángulo adecuado de la imagen a clasificar. La figura 5.4 presenta algunos de estos casos.

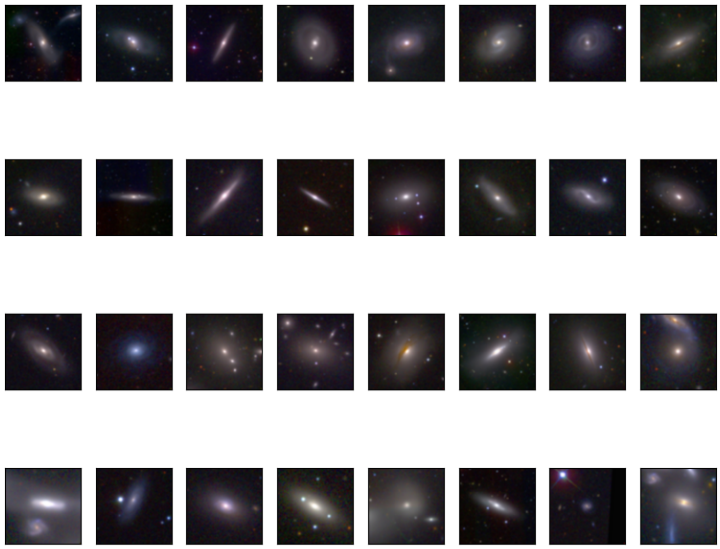

Figura 5.4: Imágenes de galaxias cuya clasificación varía según el ángulo con que se presentan al clasificador

5.6 Modelo Cyclic Simmetry

5.6.1 Descripción

Con el fin de establecer una comparación con el modelo base, se implementaron en Keras las capas de *cyclic-slicing* y *cyclic-pooling* de acuerdo a Dieleman *et al.* (2016). Estas capas fueron insertadas en el modelo base como se muestra en la figura 5.5. La capa de cyclic-pooling envía a la primer capa fully-connected una representación equivariante de la imagen de entrada. También se probaron modelos con la capa de cyclic-pooling colocada a la salida de la primer capa densa (de manera que la representación tiene invarianza cíclica) pero el primer modelo parece arrojar la mejor performance. Aparte del agregado de las capas en si, no se realizaron modificaciones del resto de los parámetros del modelo.

5.6.2 Performance e Invarianza rotacional

La performance de este modelo para la tarea de clasificación binaria no mejoró de una forma apreciable respecto del modelo base, obteniéndose una precisión promedio de $\mathbf{95.3 \pm 0.3\%}$. Sin embargo, si obtuvo una mejora en la consistencia de la clasificación en función de la rotación, encontrándose solamente 47 casos inconsistentes, es decir, el 46% respecto al modelo base.

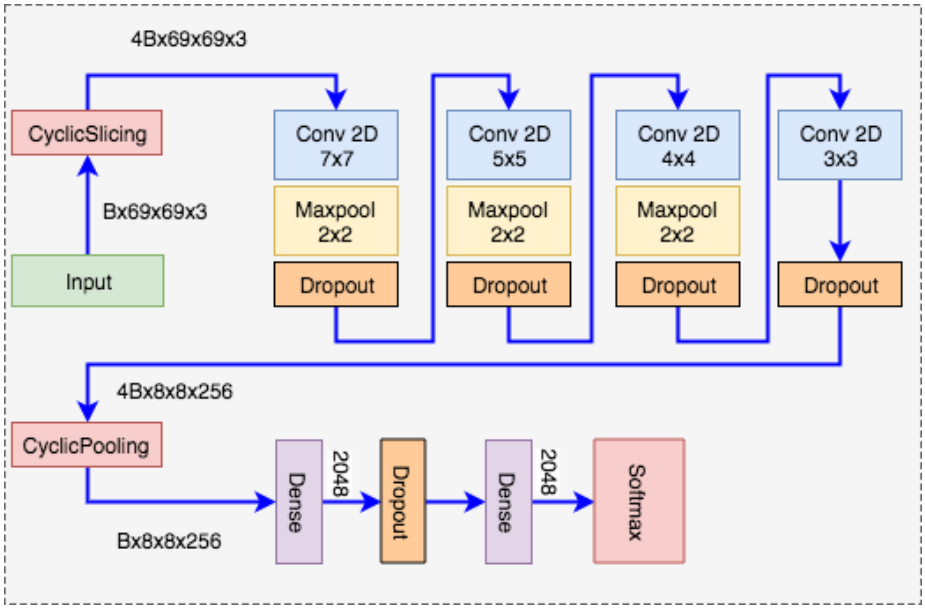

Figura 5.5: Modelo de 4 capas convolucionales con explotación de simetría cíclica

5.7 Modelo RTL

5.7.1 Descripción

Este modelo consta de una arquitectura similar a la del modelo base, a la que se le antepuso una capa RTL entre la entrada del modelo y la primer capa convolucional. El esquema del modelo puedo visualizarse en la figura 5.7. El minibatch 4d de imágenes ingresa a la red donde se generan dos flujos (vías) de datos. La primera vía corresponde a la subred de localización. Las imágenes ingresan primero a una capa de *grayscale* que tiene como fin eliminar la información de color de las mismas, de manera que el tensor con dimensiones (B,97,97,3) pasa a tener dimensión (B,97,97,1). La dimensión de los canales es colapsada mediante la operación promedio. Seguidamente, se submuestrean las imágenes utilizando una capa MaxPool2D, de manera que la dimensión final del minibatch en esta vía es (B,24,24,1). La motivación de utilizar estas dos transformaciones es evitar el exceso de información en la entrada de la capa de localización, dado que el principio de su funcionamiento se basa en realizar un reconocimiento básico de la figura del objeto y no de sus detalles. Por otro lado, esto reduce la cantidad de parámetros utilizados por la red. Una vez que el tensor ingresa a la capa de localización, esta genera como salida un vector de ángulos de rotación $\boldsymbol{\Theta}$ de tamaño B. Esta salida es enviada a la capa de rotación, la cual también recibe directamente el tensor de la imagen de entrada. La capa de rotación realiza el muestreo de acuerdo a la grilla determinada por el ángulo de rotación. Dicha grilla es de dimensiones menores respecto a la imagen original, por un factor de 0.7. Esto genera un tensor de imágenes de salida de dimensiones (B,69,69,3), el mismo tamaño que recibe el modelo base. La decisión de utilizar una grilla de muestreo más pequeña tiene como fin evitar la aparición de esquinas negras en los bordes de la imagen de salida (figura 5.7). La salida de la capa de rotación es enviada a una sucesión de cuatro capas convolucionales, con una configuración similar a la arquitectura base. La incorporación de la capa RTL fue la principal motivación por la que se decidió eliminar las activaciones del tipo tanh y sigmoid. Los primeros modelos evaluados utilizaban una versión de la RTL que hacía la regresión del $\boldsymbol{cos(\theta)}$ en lugar de $\boldsymbol{\theta}$. Esto forzaba que la salida de la capa de localización tuviera que estar en el rango $\boldsymbol{[-1, 1]}$, lo cual restringía la elección de la función de activación a los dos casos antes citados. Como resultado, la precisión del modelo se atascaba en valores inferiores al modelo actual y a su vez la capa de rotación quedaba anulada, dado que la salida de la red de localización siempre saturaba a 1. Como consecuencia, solo se utilizaron en el modelo actual funciones de activación del tipo lineal, exponencial (elu) y relu.

5.7.2 Performance e Invarianza rotacional

No se detectaron mejoras significativas en la performance de clasificación para la tarea de clasificación binaria. Llamativamente, a diferencia de los resultados logrados con MNIST, la capa RTL no introduce en este caso correcciones al ángulo de entrada de las imágenes en forma consistente. Para corroborar esto, se analizó la performance del modelo haciendo infer-

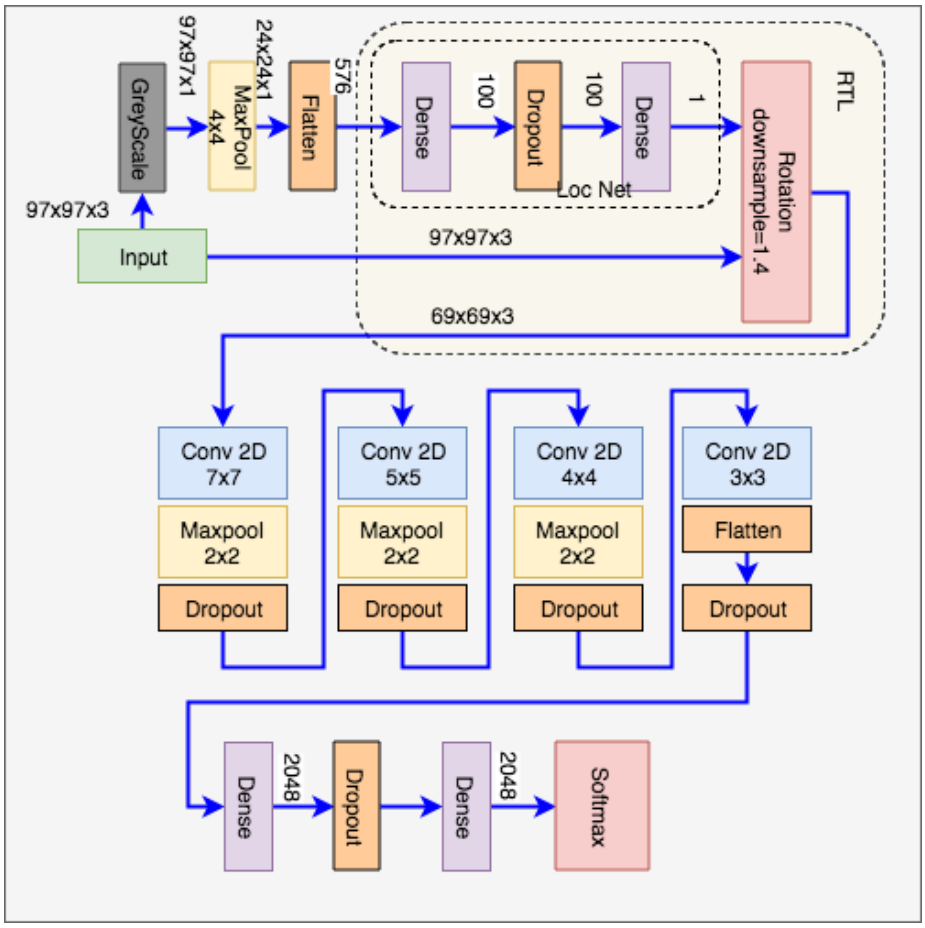

Figura 5.6: Modelo de 4 capas convolucionales con RTL utilizado como clasificador

encias con la capa de rotación activada y desactivada (forzando el ángulo $\boldsymbol{\theta}$ a 0^{o} independientemente de la salida generada por la red de localización). En ambos casos el error de clasificación es similar. Tampoco se detectaron diferencias significativas en la consistencia respecto a la rotación, encontrándose en ambos casos un número similar de galaxias con clasificación variable según el ángulo de entrada. No se ha podido establecer la razón del comportamiento aparentemente aleatorio de la capa de localización. Incluso el intento de simplificar la complejidad de los datos de entrada a la misma (mediante el descarte de la información de color y el subsampling) no generó mejoras notables.

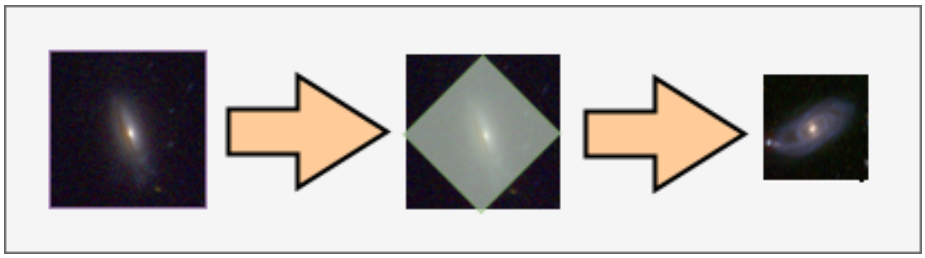

Figura 5.7: Recorte a 45 grados llevado a cabo por la capa de rotación. El tamaño más pequeño de la grilla de muestreo permite evitar la introducción de esquinas en blanco en la imagen de salida.

5.8 Experimento de detección de barras

Los modelos para detección de barras alcanzaron una precisión del 78% para el modelo base, 80% para el modelo cyclic simmetry y 78.5% para el modelo RTL. La cantidad de galaxias cuya clasificación es inconsistente según el ángulo de rotación de la imagen correspondiente resultó mayor respecto a la tarea de clasificación por tipo de galaxia. Esto podría deberse a la geometría del tipo de característica que se está detectando.

5.9 Análisis

Luego de evaluar los 3 modelos, se obtuvieron los resultados sintetizados en la tabla 5.5, donde se incluye también la performance obtenida por el método basado en SVM (Polsterer *et al.* (2012)). La precisión alcanzada resultó relativamente pareja, con una leve ventaja del modelo basado en cyclic simmetry respecto a los otros dos, que incluye una reducción de los casos con clasificación inconsistente según la rotación. Particularmente el método basado en RTL no demostró una mejora en ninguna de las dos métricas, debido a que no logra una equivarianza aproximada con este dataset.

Se incluyó el experimento de detección de barras debido a que permite la comparación con dos trabajos distintos. Las tres arquitecturas logran una performance superior a la del método SVM, pero muy inferior a la del estado del arte, que es del orden del 90%(Dieleman *et al.* (2015)). Sin embargo, debe destacarse la diferencia en la abundancia de datos de entrenamiento, que es del orden de 10 veces menor a la del dataset utilizado en dicho trabajo. Por otro lado, el citado trabajo predice una distribución de múltiples features correlacionadas entre si con una función de pérdida condicionada, por lo que para realizarse una comparación directa debería efectuarse un experimento similar.

╲	Prec (Class)	Inc(Class)	Prec(Bar)	Inc(Bar)
Base	94 %	103	78%	535
CyclicS	95.3%	47	80%	443
RTL	94%	51	78.5%	535
SVM	88%	-	60%	-

Tabla 5.5: Resumen de performance obtenida para la tarea de clasificación y para detección de barra. Se indica para cada caso precisión promedio y cantidad de casos con clasificación inconsistente respecto al ángulo de entrada.

6 Conclusiones

En este trabajo se evaluaron dos tipos posibles de arquitecturas convolucionales con propiedades de equivarianza respecto a la rotación, en contraposición con las arquitecturas convolucionales de base entrenadas con aumentación de datos. Una arquitectura basada en el framework propuesto por Dieleman *et al.* (2016) y una modificación de las Spatial Transformer Layers Jaderberg *et al.* (2015). Para llevar a cabo esta evaluación, se implementaron las dos arquitecturas como capas del framework Keras, las cuales podrán ser utilizadas en trabajos futuros. Los experimentos de la sección 3 demostraron el potencial de las propiedades de misma-equivarianza e invarianza tanto de de la solución RTL como la solución Cyclic Simmetry. Esto motivó la experimentación con estas arquitecturas para la tarea de clasificación morfológica de galaxias. Aunque existen antecedentes de la utilización exitosa de la solución Cyclic Simmetry para esta tarea, se incluyó la misma en la evaluación con el fin de enriquecer la comparación, obteniéndose con la misma los mejores resultados. Los experimentos arrojaron que las redes con el módulo RTL no se comportan en forma equivariante (aproximada) respecto a la rotación cuando son entrenadas y utilizadas con imágenes del dataset EFIGI, obteniéndose la misma performance que el modelo base homólogo. Como trabajo futuro se propone lo siguiente:

- Investigar más a fondo las causas posibles del comportamiento de las capas RTL sobre este tipo de imágenes, intentando buscar diferencias con respecto a los datos utilizados en los experimentos del capitulo 3.
- Repetir los experimentos con un dataset más grande como Galaxy-Zoo, con el fin de determinar si la relativa paridad en la performance de los 3 métodos se deba a un problema de falta de datos de entrenamiento, incluso aunque estos sean aumentados.
- Evaluar arquitecturas donde la capa RTL no se encuentre en la entrada de la red sino entre capas intermedias del modelo.
- En este trabajo se estudiaron las implicaciones de las propiedades de equivarianza de las arquitecturas como fin para mejorar la performance de clasificación de objetos presentados en distintas orientaciones. Sin embargo, se requiere un estudio más amplio de las implicancias de la reducción de parámetros mediante la eliminación

de redundancia de los filtros. Esto posiblemente requiera utilizar un dataset más grande y trabajar con visualización de filtros.

Bibliografía

Baillard, A., Bertin, E., de Lapparent, V., Fouqué, P., Arnouts, S., Mellier, Y., Pelló, R., Leborgne, J.-F., Prugniel, P., Makarov, D., Makarova, L., McCracken, H. J., Bijaoui, A. & Tasca, L. (2011). The efigi catalogue of 4458 nearby galaxies with detailed morphology. *A&A* **532**, A74. URL `https://doi.org/10.1051/0004-6361/201016423`.

Chollet, F. *et al.* (2015). Keras. `https://github.com/fchollet/keras`.

Cohen, T. S. & Welling, M. (2016). Group equivariant convolutional networks. *CoRR* **abs/1602.07576**. URL `http://arxiv.org/abs/1602.07576`.

Dieleman, S., De Fauw, J. & Kavukcuoglu, K. (2016). Exploiting cyclic symmetry in convolutional neural networks. In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16. JMLR.org. URL `http://dl.acm.org/citation.cfm?id=3045390.3045590`.

Dieleman, S., Willett, K. W. & Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society* **450**(2), 1441–1459. URL `+http://dx.doi.org/10.1093/mnras/stv632`.

Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR* **abs/1207.0580**. URL `http://arxiv.org/abs/1207.0580`.

Jaderberg, M., Simonyan, K., Zisserman, A. & Kavukcuoglu, K. (2015). Spatial transformer networks. *CoRR* **abs/1506.02025**. URL `http://arxiv.org/abs/1506.02025`.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**(6), 84–90. URL `http://doi.acm.org/10.1145/3065386`.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**(4), 541–551. URL `http://dx.doi.org/10.1162/neco.1989.1.4.541`.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press. `http://neuralnetworksanddeeplearning.com/`.

Polsterer, K., Gieseke, F., Igel, C., Doser, B. & Gianniotis, N. (2016). *Parallelized rotation and flipping INvariant Kohonen maps (PINK) on GPUs*. i6doc.com, pp. 405–410.

Polsterer, K., Gieseke, F. & Kramer, O. (2012). *Galaxy classification without feature extraction*. Astronomical Society of the Pacific, pp. 561–564.

Schmidt, U. (2012). Learning rotation-aware features: From invariant priors to equivariant descriptors. In: *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12. Washington, DC, USA: IEEE Computer Society. URL `http://dl.acm.org/citation.cfm?id=2354409.2355022`.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* **abs/1605.02688**. URL `http://arxiv.org/abs/1605.02688`.

Zeiler, M. D. (2012). ADADELTA: an adaptive learning rate method. *CoRR* **abs/1212.5701**. URL `http://arxiv.org/abs/1212.5701`.