



UNIVERSIDAD AUSTRAL

TESIS DE MAESTRÍA

---

**Clasificación de los dígitos escritos  
en los telegramas de las elecciones  
legislativas en Santa Fe mediante  
técnicas de adaptación de dominio.**

---

*Autor:*  
Franco LIANZA

*Supervisor:*  
Dr. Leandro BUGNON

17 de mayo de 2023



UNIVERSIDAD AUSTRAL

*Resumen*

Facultad de Ingeniería

Magister en Explotación de Datos y Gestión del Conocimiento

**Clasificación de los dígitos escritos en los telegramas de las elecciones legislativas en Santa Fe mediante técnicas de adaptación de dominio.**

by Franco LIANZA

TODO: Abstract?



## *Reconocimientos*

TODO: reconocimientos...



# Índice general

<b>Resumen</b>	<b>III</b>
<b>Reconocimientos</b>	<b>v</b>
<b>1. Introducción</b>	<b>1</b>
1.1. Contexto . . . . .	1
1.2. Motivación . . . . .	2
1.3. Objetivos . . . . .	4
1.4. Estructura de la tesis . . . . .	4
<b>2. Marco teórico</b>	<b>7</b>
2.1. Reconocimiento de dígitos . . . . .	7
2.2. Redes Neuronales . . . . .	8
2.2.1. Redes Densas . . . . .	9
2.2.2. Redes Convolucionales . . . . .	10
2.2.3. Arquitecturas . . . . .	13
LeNet . . . . .	14
ResNet . . . . .	14
2.3. Aprendizaje por transferencia . . . . .	15
2.4. Adaptación de Dominio . . . . .	16
2.4.1. Domain Adversarial Neural Network . . . . .	16
2.4.2. Adversarial Discriminative Domain Adaptation . . . . .	18
2.4.3. Batch Spectral Penalization . . . . .	19
2.4.4. Margin Disparity Discrepancy . . . . .	20
2.4.5. Adaptive Feature Norm . . . . .	21
<b>3. Metodología</b>	<b>25</b>
3.1. Descripción de los datos . . . . .	25
3.2. Extracción de dígitos de los telegramas . . . . .	25
3.2.1. Enderezado . . . . .	26
3.2.2. Extracción de la grilla de votos . . . . .	26
3.2.3. Detección de líneas de la grilla de votos . . . . .	27
3.2.4. Segmentación de dígitos . . . . .	29
3.3. Análisis del dataset . . . . .	29
3.4. Diseño experimental . . . . .	32
3.4.1. Metodología de entrenamiento . . . . .	32
3.4.2. Selección y optimización del modelo . . . . .	33
3.5. Métricas de evaluación . . . . .	33
3.5.1. Métricas de clasificación . . . . .	34
Accuracy . . . . .	34

$F_1$ . . . . .	34
Intersección sobre unión . . . . .	35
3.5.2. Métricas de adaptación . . . . .	35
Distancia $\mathcal{A}$ . . . . .	35
Discrepancia de Medias Máxima . . . . .	36
<b>4. Análisis de resultados</b> . . . . .	<b>39</b>
4.1. Análisis de métricas . . . . .	39
4.2. Análisis de los espacios latentes . . . . .	41
4.2.1. LeNet . . . . .	42
4.2.2. ResNet . . . . .	43
4.3. Análisis de errores . . . . .	45
<b>5. Conclusiones</b> . . . . .	<b>49</b>
5.1. Trabajos Futuros . . . . .	50
<b>A. Anexo: Telegramas</b> . . . . .	<b>51</b>
A.1. Ejemplo de telegrama . . . . .	52
A.2. Ejemplo de telegrama mal cargado . . . . .	53
A.3. Ejemplo de telegrama con números sin espacio entre ellos . . . . .	54
A.4. Ejemplo de telegrama con caracteres distintos a números . . . . .	55
A.5. Ejemplo de telegrama ceros a la izquierda de la cantidad de votos . . . . .	56
<b>Bibliografía</b> . . . . .	<b>57</b>

# Índice de figuras

2.1.	Red neuronal densa . . . . .	10
2.2.	Operador <i>convolución</i> . . . . .	11
2.3.	Operador <i>maxpool</i> . . . . .	12
2.4.	Red neuronal convolucional . . . . .	13
2.5.	Arquitectura de una LeNet-5 . . . . .	14
2.6.	Esquema de <i>DANN</i> . . . . .	17
2.7.	Esquema de <i>ADDA</i> . . . . .	19
2.8.	Esquema de <i>DANN+BSP</i> . . . . .	20
2.9.	Esquema de <i>MDD</i> . . . . .	21
2.10.	Representación de características aprendidas de una CNN . . . . .	22
2.11.	Esquema de <i>AFN</i> . . . . .	23
3.1.	Enderezamiento de un telegrama . . . . .	26
3.2.	Grilla de votos detectada . . . . .	27
3.3.	Proyecciones de los ejes de la grilla de votos . . . . .	28
3.4.	Segmentación de la grilla detectada . . . . .	28
3.5.	Primer registro extraído de la grilla de votos . . . . .	29
3.6.	Dígitos detectados en el primer bloque de votos . . . . .	29
3.7.	Histogramas de proporciones mínimas y máximas de píxeles blancos por voto . . . . .	31
3.8.	Conjunto de datos TDS . . . . .	32
3.9.	Ejemplos de distribuciones de dominios . . . . .	36
4.1.	Representación UMAP de las características aprendidas de los modelos LeNet5 . . . . .	42
4.2.	Representación UMAP de las características aprendidas de los modelos ResNet18 . . . . .	44
4.3.	Histogramas de IoU por modelo . . . . .	46



# Índice de cuadros

1.1.	Precisión que se obtiene al aplicar un modelo LeNet-5 entrenado con otros datasets de dígitos. . . . .	3
3.1.	Ejemplo del conjunto de datos generado . . . . .	30
3.2.	Estadísticos descriptivos de los dígitos . . . . .	31
3.3.	Rango de hiperparámetros optimizados . . . . .	33
4.1.	Métricas de los experimentos realizados . . . . .	39
4.2.	IoU y aciertos por modelo . . . . .	40



# Capítulo 1

## Introducción

### 1.1. Contexto

En Argentina se celebran elecciones cada 2 años a excepción de las presidenciales que se realizan cada 4 años. Existen, principalmente, tres tipos de elecciones:

- Elecciones nacionales, para elegir a las autoridades federales del país: el Poder Ejecutivo, constituido por el Presidente y el vicepresidente y el Congreso Nacional, formado por Senadores y Diputados.
- Elecciones provinciales y de la Ciudad de Buenos Aires o locales, para elegir a las autoridades de cada provincia: los poderes ejecutivos de las provincias y sus legislaturas.
- Elecciones municipales, regidas por las leyes y procedimientos de cada provincia.

El proceso para emitir el sufragio en Argentina puede variar según la elección, pero en general se sigue un procedimiento similar en todas ellas<sup>1</sup>. Primero, los ciudadanos habilitados para votar deben presentarse en su lugar de votación el día de las elecciones, que suele ser un domingo, y dirigirse a la mesa correspondiente según su número de documento. Una vez allí, se les solicita que presenten su documento de identidad para que las autoridades de mesa puedan verificar su identidad y comprobar que se encuentran en el padrón electoral. Luego, los votantes reciben una boleta con los candidatos y partidos políticos que participan en la elección. Los ciudadanos deben dirigirse a una cabina de votación para emitir su voto en secreto. Allí, pueden seleccionar la boleta completa de un partido o candidato, o elegir diferentes candidatos de diferentes partidos. Una vez que han realizado su elección, deben doblar la boleta y dirigirse a la urna donde depositarán su voto. Al finalizar la jornada de votación, las autoridades de mesa llevan a cabo el recuento de votos. Esto implica el conteo manual de los votos emitidos y la elaboración de una planilla en la que se detallan los votos obtenidos por cada candidato o partido político. Esta planilla es firmada por todas las autoridades de mesa y se digitalizan para ser enviadas al correo argentino. La mayoría se escanea en las escuelas y el restante lo escanea el propio correo. Una vez en el centro de cómputo, un grupo de personas se encarga de contabilizar los votos

---

<sup>1</sup><https://www.cippec.org/como-se-cuentan-los-votos/>.

utilizando un sistema informático. En primer lugar, se digitaliza la información de las planillas recibidas para que pueda ser procesada por el sistema. Luego, se lleva a cabo un proceso de validación para verificar que los datos sean correctos y no haya errores. Una vez que se ha completado el proceso de digitalización y validación, se obtiene el resultado final de la elección y se anuncia públicamente el candidato o partido político ganador. Es importante destacar que, aunque se utilice la boleta única electrónica, en la mayoría de los procesos electorales siempre se realiza un conteo manual de respaldo para asegurar la transparencia y la integridad de los resultados. De esta manera, se verifica que el conteo realizado por el sistema informático sea consistente con el conteo manual y se evita cualquier posibilidad de fraude o error.

El proceso de conteo de votos en un centro de cómputo requiere de una gran cantidad de personas para poder llevarse a cabo de manera eficaz y rápida. Sin embargo, esta solución resulta altamente ineficiente en términos de tiempo y costos. En las elecciones legislativas del 2021 en Argentina, se destinaron alrededor de \$17.000 millones de pesos para el proceso electoral, de los cuales alrededor de \$4.000 millones de pesos fueron utilizados para el pago de salarios del personal encargado del centro de cómputo<sup>2</sup>.

## 1.2. Motivación

Digitalizar los telegramas de manera automática supondrá un ahorro considerable en el presupuesto de las elecciones, agilizará la obtención de los resultados y aportará transparencia al proceso en general. Es posible entrenar un modelo de clasificación de dígitos a un costo extremadamente menor al actual y utilizarlo al momento de la contabilización de los votos.

Contar con una digitalización automática permitirá bajar los costos debido a que se necesitará un grupo menor de personas en el centro de cómputo. Además, el trabajo a realizar será más simple ya que sólo constará del proceso de validación.

La digitalización automática de los telegramas de votación es una iniciativa que busca mejorar la eficiencia del proceso electoral, reducir los costos y aumentar la transparencia en las elecciones. Actualmente, el proceso de conteo de votos se realiza manualmente y puede ser un proceso largo y laborioso.

Utilizando la tecnología de procesamiento de imágenes para contar automáticamente los votos de los telegramas, es posible reducir la cantidad de personal requerido en el centro de cómputo. Esto podría ahorrar costos en términos de recursos humanos y materiales, ya que el personal necesario para el conteo manual de votos sería menor. Además, se reduciría la posibilidad de error humano, lo que mejoraría la precisión del proceso de conteo de votos.

---

<sup>2</sup><https://www.cronista.com/economia-politica/Elecciones-legislativas-2021-cuanto-mas-se-gastara-por-el-coronavirus-segun-el-Presupuesto-20201004-0006.html>.

Por otro lado, la digitalización automática de los telegramas también aceleraría el proceso de conteo de votos y permitiría que los resultados se obtengan más rápidamente. Esto podría ayudar a reducir la ansiedad y la tensión que a menudo rodean los resultados electorales, así como mejorar la transparencia del proceso en general. Al contar con resultados electorales precisos y rápidos, se puede mejorar la confianza en el sistema electoral y en las instituciones democráticas.

Es importante destacar que este proceso no requiere de una gran inversión. De hecho, es posible entrenar modelos de clasificación de dígitos a un costo extremadamente menor que el actual, utilizando técnicas de aprendizaje automático para identificar y clasificar los números individuales que componen los votos en los telegramas. Esto permitiría la implementación de un sistema de conteo de votos automatizado con un menor costo y una mayor precisión.

La clasificación de números manuscritos es un problema que puede parecer ya resuelto gracias al dataset MNIST y al trabajo pionero de LeCun en 1998. Sin embargo, no se puede subestimar la complejidad de este problema. Cada año, cambian las personas que llenan los telegramas de las elecciones y, por lo tanto, las características de los números escritos a mano pueden variar. Esto puede generar un *corrimiento de dominio* o *data drift*, lo que significa que un modelo entrenado en un conjunto de datos estático como MNIST puede presentar una mala clasificación de los dígitos escritos a mano en las elecciones.

Cuando los dominios de entrenamiento y aplicación son diferentes, se debe a que existe un sesgo en los datos. Las técnicas de entrenamiento tradicionales asumen que, aunque existe un sesgo, este será el mismo entre el origen y el destino. Sin embargo, cuando este supuesto no se cumple, las predicciones del modelo pueden verse afectadas negativamente. El cuadro 1.1 ilustra un ejemplo de cómo la precisión del modelo puede degradarse cuando se aplica a un conjunto de datos diferente al que se entrenó. Incluso en conjuntos de datos similares, como MNIST y USPS, la disminución en la precisión de los modelos es notable.

Entrenamiento \ Validación	MNIST	USPS	SVHN
MNIST			
USPS			
SVHN			

CUADRO 1.1: Precisión que se obtiene al aplicar un modelo LeNet-5 entrenado con otros datasets de dígitos.

En conclusión, debido a que la escritura de los números en los telegramas puede variar de elección en elección y a que los datos de entrenamiento no son suficientes para una clasificación precisa en este contexto, se necesita recurrir a técnicas más complejas de entrenamiento que permitan al modelo adaptarse a diferentes dominios. La adaptación de dominio es una técnica que busca reducir el efecto del sesgo de los datos y hacer que el modelo sea más robusto ante las variaciones en la escritura de los números. En la actualidad, no existen trabajos publicados que comparen diferentes técnicas de adaptación de dominio para la digitalización de telegramas en Argentina, lo que sugiere una oportunidad para investigar y desarrollar modelos más efectivos para este propósito.

### 1.3. Objetivos

El objetivo general de esta tesis es poder clasificar los dígitos de los votos en las elecciones legislativas de la provincia de Santa Fe del año 2021 mediante adaptación de dominio de forma que el modelo generalice y no dependa de un etiquetado de los datos. El enfoque se centrará en explorar y comparar diferentes técnicas de adaptación de dominio con el fin de lograr un modelo robusto que pueda generalizar en diferentes contextos y escenarios electorales. Con esto se contribuye a mejorar la transparencia y eficiencia en los procesos electorales en Argentina y sentar las bases para el desarrollo de soluciones similares para el resto de las provincias y la nación.

Los objetivos específicos para lograr el objetivo planteado son los siguientes:

- Armar el proceso de extracción, transformación y carga (ETL) de los telegramas, con el propósito de segmentar y limpiar de manera efectiva los dígitos. Los datos utilizados serán obtenidos de fuentes oficiales del estado argentino<sup>3</sup>. En el anexo A.1 se adjunta un ejemplo de uno de ellos.
- Entrenar distintas arquitecturas de redes convolucionales mediante técnicas de adaptación de dominio.
- Analizar y evaluar las métricas relevantes para seleccionar el mejor modelo.
- Detectar oportunidades de mejora en el proceso eleccionario respecto a los telegramas.

### 1.4. Estructura de la tesis

Esta tesis se encuentra organizado con la siguiente estructura:

---

<sup>3</sup><https://op.elecciones.gob.ar/telegramas/generales2021/>.

- Capítulo 2: Marco teórico del reconocimiento de dígitos, redes neuronales, aprendizaje por transferencia y adaptación de dominio.
- Capítulo 3: Metodología del trabajo realizado sobre los telegramas de las elecciones de Santa Fe. Proceso de extracción, transformación y limpieza de los mismos. Diseño experimental y métricas de evaluación.
- Capítulo 4: Análisis de resultados. Análisis de métricas, espacios latentes obtenidos y errores observados.
- Capítulo 5: Conclusiones del trabajo, mejoras planteadas y futuras investigaciones.



## Capítulo 2

# Marco teórico

Este capítulo del marco teórico tiene como objetivo presentar una visión general de la bibliografía relevante al objetivo de la tesis. En la primera sección, se explorará la historia del reconocimiento de dígitos por parte de las computadoras y las técnicas desarrolladas. Posteriormente, se brindará un breve resumen de la evolución de las redes neuronales, con especial atención en las redes densas y convolucionales, y sus conceptos básicos. La sección siguiente se enfocará en el aprendizaje por transferencia de las redes neuronales, sus capacidades y limitaciones. Finalmente, se profundizará en la adaptación de dominio y se explicarán las distintas técnicas utilizadas en el trabajo.

### 2.1. Reconocimiento de dígitos

El reconocimiento de caracteres por parte de las computadoras ha sido un campo de investigación activo durante varias décadas. Una de las primeras investigaciones en este campo se centró en cómo las neuronas del cerebro procesan y reconocen información visual.

En 1959, Hubel y Wiesel comenzaron a investigar la percepción visual en el cerebro de los gatos. Descubrieron que las neuronas en la corteza visual primaria del cerebro de los gatos respondían a estímulos visuales específicos, como líneas y bordes. Además, identificaron dos tipos de células en la corteza visual primaria: células simples y células complejas. Las células simples respondían a estímulos visuales específicos, como barras de luz en una dirección particular, mientras que las células complejas respondían a estímulos visuales más complejos, como patrones en movimiento. En 1981, recibieron el Premio Nobel de Fisiología o Medicina por su trabajo en el procesamiento visual del cerebro.

Inspirado por el trabajo de Hubel y Wiesel, en 1980 se desarrolló la red Neocognitron (Fukushima, 1980). La red Neocognitron fue uno de los primeros modelos de aprendizaje profundo utilizados para el reconocimiento de caracteres. La red estaba compuesta por múltiples capas, cada una de las cuales procesaba información visual en diferentes niveles de abstracción.

A pesar de que la red Neocognitron fue un gran avance en el reconocimiento de caracteres, todavía tenía limitaciones. En particular, la red no podía reconocer caracteres escritos a mano que eran muy diferentes de los caracteres que se habían utilizado para entrenar la red.

A fines de la década de 1980, se presentó un enfoque importante para el reconocimiento de dígitos escritos a mano llamado “Sistema Neuronal de Reconocimiento de Dígitos” (LeCun et al., 1989). Este sistema utilizó una red neuronal de varias capas y fue entrenado utilizando el método de retropropagación del error. El modelo tuvo un rendimiento significativamente mejor que los enfoques anteriores y sentó las bases para el desarrollo de sistemas de reconocimiento de dígitos escritos a mano más sofisticados.

En la década de 1990, se desarrollaron otros modelos de aprendizaje profundo que superaron las limitaciones de los enfoques anteriores. LeCun et al., 1995 se menciona por primera vez el término de *red convolucional*.

Desde entonces, los algoritmos de reconocimiento de caracteres han seguido evolucionando y mejorando. Actualmente, los sistemas de reconocimiento de caracteres son capaces de reconocer caracteres escritos a mano con una precisión cercana al 100 %. Además del reconocimiento de caracteres, estas técnicas también se han utilizado para la detección de objetos en imágenes y videos, así como para la traducción automática.

## 2.2. Redes Neuronales

McCulloch y Pitts, 1943 estudiaron y propusieron un modelo del comportamiento de la neurona biológica. La neurona artificial que propusieron fue un sistema binario que consiste en que si la suma de entradas excitatorias supera un umbral de activación, y además no hay una entrada inhibitoria, la neurona se activa y emite una respuesta; en caso contrario, la neurona no se activa.

En 1949 se propone que estas redes podían aprender. Este hecho estaba relacionado con la conductividad de la sinapsis. Así, la repetida activación de una neurona por otra, a través de una sinapsis determinada, aumenta su conductividad y la hace más propensa a ser activada sucesivamente, induciendo a la formación de un circuito de neuronas estrechamente conectadas entre sí (Hebb, 1949).

Años más tarde, Rosenblatt, 1958 propone un modelo de neurona llamadas perceptrones basadas en el modelo de McCulloch-Pitts. El mayor logro de Rosenblatt fue que pudo demostrar que, flexibilizando algunas de las reglas de la McCulloch-Pitts (la inhibición absoluta, la contribución igual de todas las entradas, etc), las neuronas artificiales podían aprender de los datos. Y lo que es más importante, ideó un algoritmo de aprendizaje supervisado para este modelo de neurona que permitía averiguar los pesos correctos directamente a partir de los datos de entrenamiento. La sencillez y eficacia de este algoritmo de aprendizaje para problemas linealmente separables son algunas de las razones clave por las que se hizo tan popular a finales de los años cincuenta y principios de los sesenta. Sin embargo, esta popularidad hizo que Rosenblatt exagerara la capacidad de aprendizaje de su perceptrón, dando lugar a expectativas poco realistas en la comunidad científica y/o también difundidas por los medios de comunicación.

En 1969, se publica un libro en el cual se expone la naturaleza lineal de los perceptrones y de sus limitadas capacidades (Minsky y Papert, 1969). A

raíz de este evento es que se inicia el denominado “invierno de la inteligencia artificial” de los 1980s, donde cesó el interés de la comunidad por los modelos conexionistas.

La contribución más importante en la reemergencia del conexionismo en los años ochenta fue la técnica *backpropagation* desarrollada por Rumelhart, Hinton y Williams, 1986. En realidad, esta técnica fue desarrollada inicialmente por Werbos, 1974 y después redescubierta por varios grupos de investigadores (LeCun, 1985; Rumelhart, Hinton y Williams, 1986). Este algoritmo abrió las puertas al uso práctico de las redes neuronales en un sinfín de campos. La idea detrás del *backpropagation* es ajustar los pesos de una red neuronal para minimizar el error de predicción en un conjunto de datos de entrenamiento. Este error se calcula como la diferencia entre la salida real de la red y la salida deseada para una entrada dada. Para ajustar los pesos de la red, se utilizan las derivadas parciales del error con respecto a cada uno de los pesos, lo que permite actualizar los pesos en la dirección que minimiza el error. El entrenamiento de una red utilizando *backpropagation* se divide en dos fases: la propagación hacia adelante y la propagación hacia atrás. En la fase de propagación hacia adelante, se calcula la salida de la red para una entrada dada. La salida de la red se compara con la salida deseada para calcular el error de predicción. En la fase de propagación hacia atrás, se utiliza este error para calcular las derivadas parciales del error con respecto a cada uno de los pesos de la red. Este algoritmo ajusta los pesos en la dirección que minimiza el error de la red, multiplicando la derivada parcial del error por una tasa de aprendizaje y restando el resultado del peso correspondiente.

Las redes neuronales más “simples” entrenadas con *backpropagation*, son las denominadas redes multicapa alimentadas hacia delante (*feed forward*) o perceptrones mutlicapa *MLP*. En 1989, se demuestra teóricamente que una red de este tipo, con una capa oculta y el suficiente número de nodos, puede aproximar cualquier función continua con cualquier grado de precisión (Funahashi, 1989).

### 2.2.1. Redes Densas

Como se detalló anteriormente, los perceptrones son modelos lineales incapaces de resolver problemas que no sean linealmente separables por un hiperplano. Una forma de solventar esta situación es encadenar un conjunto de perceptrones para crear redes neuronales densas (ver figura 2.1). La predicción  $\hat{y}$  de una red es una generalización directa de la predicción de un perceptrón. Primero se calculan las activaciones  $a_i^1$  de los nodos de la capa oculta  $h_i^1$  en función de las entradas  $x_i$  y los pesos de entrada  $w_i^1$ . Luego, se calculan las activaciones  $a_i^2$  de la unidad de salida  $h_i^2$  dadas las activaciones de la capa anterior y los pesos  $w_i^2$ .

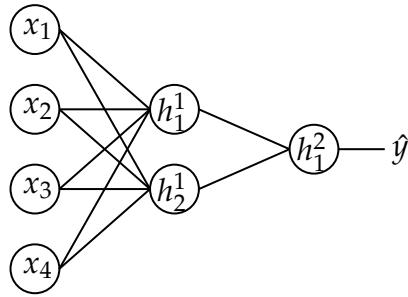


FIGURA 2.1: Red neuronal con una capa de entrada de 4 características, una capa oculta de 2 neuronas y una capa de salida con 1 neurona.

La única diferencia entre este cálculo y el del perceptrón es que las unidades ocultas calculan una función no lineal de sus entradas. Generalmente es llamada como *función de activación* o *función de enlace*. Formalmente, si  $w_{i,d}$  son los pesos que conectan las entradas  $d$  a la unidad oculta  $i$ , entonces la activación de la unidad  $i$  se calcula como:

$$h_i = f(\mathbf{w}_i \cdot \mathbf{x} + b_i) \quad (2.1)$$

Donde  $f$  es la función de activación,  $\mathbf{w}_i$  es el vector de pesos que alimentan el nodo  $i$  y  $b_i$  es el término de *bias*. Existen múltiples funciones de activación, las más comunes son:

- Tangente hiperbólica:  $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Función sigmoidea:  $f(x) = \sigma(x) = \frac{1}{1+exp(-x)}$
- ReLU:  $f(x) = max(0, x)$
- Softmax:  $f(x_i) = \sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \text{ para } i = 1, 2, \dots, K$

### 2.2.2. Redes Convolucionales

Una de las principales limitaciones de las redes densas es su incapacidad para manejar datos que presentan una estructura espacial, como imágenes o audio. Esta limitación se debe principalmente al problema de la invarianza espacial, donde una misma entidad (por ejemplo, el dígito '6') puede tener diferentes ubicaciones en una imagen. Dado que las redes densas tratan cada entrada de manera aislada, ignorando las relaciones espaciales entre ellas, es posible que no detecten patrones o características importantes en los datos, lo que puede reducir su rendimiento.

Para solucionar este problema, surgió la idea de utilizar redes neuronales convolucionales (CNN). Estas redes se basan en utilizar capas de neuronas que comparten pesos y realizan una operación de convolución en los datos de entrada. Esto permite que la red tenga una mayor capacidad para detectar patrones y características en los datos, ya que las neuronas de las capas de

convolución están compartiendo información y pueden detectar patrones a diferentes escalas y en diferentes posiciones de los datos.

Matemáticamente, la operación de convolución se puede representar como:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt \quad (2.2)$$

Donde  $f$  y  $g$  son dos funciones y  $*$  es el operador de convolución. Esta operación consiste en desplazar la función  $g$  a través de la función  $f$  y multiplicar cada valor de  $f$  por el valor de  $g$  en esa posición. El resultado de la operación de convolución es una nueva función que refleja la combinación de los valores de  $f$  y  $g$ .

En la figura 2.2 se ilustra el proceso de cálculo de una convolución para escenarios discretos. Para una matriz  $I$  de tamaño  $6 \times 6$  y un filtro  $K$  de tamaño  $3 \times 3$ , la operación de convolución  $*$  en una CNN aplica el filtro  $K$  deslizándolo sobre la matriz  $I$  para producir  $I * K$ .

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 5 & 2 & 3 \\ 3 & 2 & 4 & 2 \\ 1 & 4 & 2 & 1 \\ 3 & 2 & 1 & 1 \end{pmatrix}$$

$I$                              $K$                              $I * K$

FIGURA 2.2: Ejemplo de una convolución para una matriz  $I$  y un filtro o kernel  $K$ , produciendo la matriz  $I * K$ .

En una CNN, la matriz  $I$  representa la imagen y el filtro  $K$  es el conjunto de parámetros entrenables del modelo, análogo a las neuronas en una red densa. Aunque existen filtros estudiados para detectar ciertas características, como el filtro Sobel para detectar bordes, se permite que la red aprenda de forma autónoma  $K$  con el fin de detectar las características relevantes  $I * K$  para la tarea a través del entrenamiento. Esto permite que la red encuentre características sin el sesgo que se introduciría al utilizar únicamente filtros predefinidos.

Una de las principales ventajas de las CNNs es que son capaces de procesar datos con una gran cantidad de dimensiones, como imágenes con altura, anchura y profundidad (canales de color). Esto se debe a que las capas de convolución son capaces de realizar operaciones de filtrado sobre los datos de entrada, permitiendo a la red aprender características específicas de los datos de manera eficiente. Además, las CNNs tienen la capacidad de generalizar de manera efectiva, lo que significa que son capaces de realizar buenas predicciones en conjuntos de datos desconocidos.

Otra ventaja de las CNN es que cuentan con capas de *pooling* o también llamadas de submuestreo. Su objetivo principal es reducir el tamaño espacial

de la representación de características para disminuir la cantidad de parámetros y de operaciones necesarias en la red, lo que permite reducir el costo computacional y evitar el sobreajuste. Además, al reducir el tamaño de la representación, se logra una cierta invarianza en las características aprendidas por la red, lo que significa que la posición exacta de las características dentro de la imagen ya no importa tanto. La operación de *pooling* en una CNN consiste en dividir la imagen de entrada en subregiones no superpuestas y aplicar una función a cada subregión. En general, existen tres tipos de funciones que se utilizan en *pooling*: *max pooling*, que toma el valor máximo de cada subregión; *average pooling*, que toma el valor promedio de cada subregión; y *min pooling*, que toma el valor mínimo de cada subregión. La figura 2.3 ilustra el cálculo de *max pooling* a la matriz  $I * K$  del ejemplo anterior.

$$\left( \begin{array}{|c|c|} \hline 1 & 5 & 2 & 3 \\ \hline 3 & 2 & 4 & 2 \\ \hline 1 & 4 & 2 & 1 \\ \hline 3 & 2 & 1 & 1 \\ \hline \end{array} \right) \xrightarrow{\text{maxpool}} \left( \begin{array}{|c|c|} \hline 5 & 4 \\ \hline 4 & 2 \\ \hline \end{array} \right)$$

$I * K$                                    $\text{maxpool}(I * K)$

FIGURA 2.3: Maxpool  $2 \times 2$  aplicado al ejemplo de la figura 2.2  
 $I * K$ .

Las redes neuronales convolucionales tienen una gran variedad de aplicaciones prácticas en diferentes campos, algunas de las cuales incluyen:

- Reconocimiento de imágenes: CNN se utilizan ampliamente en tareas de clasificación de imágenes, como el reconocimiento de objetos, el análisis de sentimientos en imágenes, la generación de descripciones automáticas para imágenes, entre otras (Krizhevsky, Sutskever e Hinton, 2017).
- Análisis de videos: CNN también se utilizan en tareas de análisis de videos, como el seguimiento de objetos, la detección de eventos, la generación de subtítulos automáticos para videos, entre otras (Simonyan y Zisserman, 2014).
- Procesamiento de lenguaje natural: CNN también se utilizan en tareas de procesamiento de lenguaje natural, como la clasificación de texto, la generación de texto, el análisis de sentimientos y la traducción automática (Bugnon et al., 2020).
- Medicina: CNN se utilizan en tareas de diagnóstico médico, como la detección de cáncer de mama, la detección de enfermedades cardíacas, la detección de problemas en imágenes médicas, entre otras (Wang et al., 2016).

### 2.2.3. Arquitecturas

Existen distintos tipos de arquitecturas de redes neuronales convolucionales debido a que cada una tiene diferentes fortalezas y debilidades en términos de capacidad de generalización, extracción de características y rendimiento (LeCun, Bengio e Hinton, 2015). Algunos factores que pueden influir en la elección de una arquitectura en particular incluyen (He et al., 2016):

- Tamaño de los datos de entrada: para procesar grandes conjuntos de datos, es posible que se necesite una red con una mayor capacidad de procesamiento y una mayor cantidad de parámetros.
- Nivel de detalle de las características: algunas arquitecturas pueden ser más eficientes para detectar características a diferentes escalas y en diferentes posiciones, mientras que otras pueden tener una mayor capacidad para detectar patrones complejos.
- Nivel de generalización: algunas arquitecturas pueden ser más capaces de generalizar a datos nuevos y menos vistos durante el entrenamiento, mientras que otras pueden ser más propensas a sobreajuste.
- Requerimientos de tiempo y recursos: algunas arquitecturas pueden requerir más tiempo y recursos para entrenar y evaluar, lo que puede ser un factor importante a considerar dependiendo de los objetivos del proyecto.

Una arquitectura simple de CNN para la clasificación de imágenes es la ilustrada en la figura 2.4: una subred convolucional  $\mathcal{G}$  que extrae información y genera características  $f$  a partir de la entrada  $x$ ; y una subred densa  $\mathcal{C}$  que se encarga de clasificar las características aprendidas  $f$  en las etiquetas del problema  $\hat{y}$ .

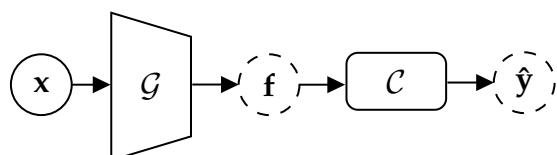


FIGURA 2.4: Arquitectura de una red convolucional simple. Las entradas  $x$  son procesadas por una red convolucional  $\mathcal{G}$  que generan características  $f$  para luego ser clasificadas por una red densa  $\mathcal{C}$  en  $\hat{y}$ .

Tal es el caso de la red LeNet, que fue propuesta por LeCun et al., 1998. Esta red consta de dos capas de convolución seguidas por dos capas densas y una capa de salida, y fue utilizada con éxito para el reconocimiento de números escritos a mano en el conjunto de datos MNIST.

Por otro lado, una arquitectura de red neuronal convolucional más compleja es la red ResNet (abreviación de Residual Network), propuesta por He et al., 2016. Estas redes están diseñadas para resolver el problema de la propagación del gradiente en las redes neuronales profundas, lo que permite entrenar modelos con una gran cantidad de capas.

## LeNet

LeNet es una de las primeras arquitecturas de redes neuronales de aprendizaje profundo desarrolladas para tareas de reconocimiento de caracteres escritos a mano. Aunque la arquitectura original de LeNet es bastante antigua, su diseño sigue siendo relevante hoy en día y se utiliza como una arquitectura básica para tareas similares.

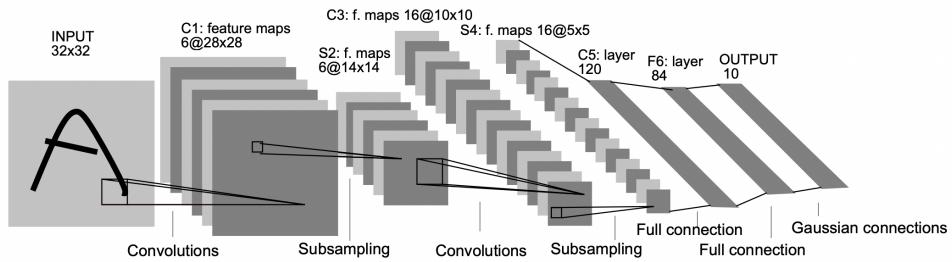


FIGURA 2.5: Arquitectura de una LeNet-5. Imagen tomada de LeCun et al., 1998.

El desarrollo de la LeNet implicó la aparición de las CNNs y sus componentes básicos. No obstante, no fue popular en su momento debido a la falta del hardware necesario para entrenarlas y otros algoritmos como las máquinas de vectores de soporte (SVM por sus siglas en inglés) que podían obtener resultados similares o superadores (LeCun et al., 1998; DeCoste y Schölkopf, 2002; Lauer, Suen y Bloch, 2007).

En 2012, Alex Krizhevsky ganó la competencia de ImageNet con su arquitectura AlexNet (Krizhevsky, Sutskever e Hinton, 2017), basada en la LeNet pero con modificaciones. La principal diferencia entre ambas es la complejidad y profundidad de la arquitectura, con AlexNet teniendo ocho capas y más de 60 millones de parámetros entrenables, mientras que LeNet tiene cinco capas y menos de un millón de parámetros. Además, AlexNet utiliza filtros convolucionales más grandes (11x11 y 5x5) que permiten capturar características de imagen más grandes y complejas, y técnicas de regularización más avanzadas, como el dropout, para prevenir el sobreajuste. En contraste, LeNet utiliza filtros más pequeños (5x5 y 3x3) y técnicas de regularización más simples como la regularización L2.

Desde el éxito de AlexNet, las CNNs se han convertido en la mejor opción para aplicaciones de visión computacional y se han creado muchos tipos diferentes de CNN, como las redes R-CNN utilizada para detección y segmentación de objetos (Girshick et al., 2014). En la actualidad los modelos CNN son bastante diferentes de LeNet, pero todos se desarrollan sobre la base de LeNet.

## ResNet

La idea principal detrás de las redes ResNet es la utilización de *bloques residuales* en lugar de las capas tradicionales de la red neuronal. Un bloque residual se compone de varias capas de una red neuronal tradicional, pero con

una conexión residual que permite que la información de entrada se sume directamente a la salida de las capas internas. Esto permite que la información se propague de manera más eficiente a través de las capas profundas de la red, lo que ayuda a reducir el problema del desvanecimiento de gradientes en la retropropagación del error.

Además, las redes ResNet también utilizan una técnica llamada *bottleneck* que reduce el número de canales de salida en las capas internas del bloque residual, lo que ayuda a reducir el número de parámetros y mejorar la eficiencia del modelo obligando a la red a aprender características más relevantes.

Las redes neuronales ResNet se han desarrollado en diferentes arquitecturas, cada una de las cuales tiene su propio conjunto de características y aplicaciones. Las mismas son:

- ResNet18 y ResNet34 (He et al., 2016): son arquitecturas de ResNet con un número moderado de capas (18 y 34 respectivamente) que se utilizan comúnmente en tareas de clasificación de imágenes y reconocimiento de objetos.
- ResNet50, ResNet101 y ResNet152 (He et al., 2016): son arquitecturas de ResNet con un mayor número de capas (50, 101 y 152 respectivamente) que se utilizan comúnmente en tareas de detección de objetos y segmentación de imágenes. Estas arquitecturas tienen una mayor capacidad representativa y son más precisas en las tareas de aprendizaje profundo complejas, pero requieren más tiempo de entrenamiento y recursos.
- ResNet-S y ResNet-M (Sandler et al., 2018): son arquitecturas de ResNet que se utilizan en dispositivos móviles y sistemas de baja potencia. Estas arquitecturas son más ligeras en términos de parámetros y requisitos de cálculo que las arquitecturas anteriores.
- ResNeXt (Xie et al., 2017): es una variante de ResNet que utiliza una técnica llamada *agrupación cardinal* para mejorar la capacidad representativa de la red. Esta técnica utiliza varios grupos de canales en lugar de uno solo para mejorar la capacidad representativa del modelo y lograr mejores resultados en tareas de clasificación y reconocimiento.

## 2.3. Aprendizaje por transferencia

El aprendizaje por transferencia es una técnica que consiste en utilizar lo que se ha aprendido en una tarea para mejorar el rendimiento en otra tarea relacionada (Thrun y Pratt, 1998). Esta técnica se aplica comúnmente en el contexto de redes neuronales, especialmente en tareas de imágenes, y se basa en la idea de que algunos conocimientos adquiridos en una tarea pueden ser reutilizados en otra tarea, lo que permite que la red aprenda de manera más eficiente y con menos datos. Ha ganado popularidad en los últimos años debido a la creciente cantidad de datos disponibles y a la necesidad de entrenar modelos más complejos y de mayor rendimiento.

En el contexto de las redes neuronales, el aprendizaje por transferencia se puede llevar a cabo de varias maneras, como el *pre-entrenamiento* y el *fine-tuning*. El pre-entrenamiento es una técnica de aprendizaje por transferencia en la que se entrena una red en una tarea y luego se utiliza como punto de partida para entrenar otra red en una tarea diferente (Erhan et al., 2010). Esto permite que la red aprenda de manera más rápida y con menos datos, ya que ya ha adquirido algunos conocimientos en la primera tarea. El pre-entrenamiento se puede llevar a cabo de varias maneras, como por ejemplo entrenando una red en un conjunto de datos de gran tamaño y luego utilizándola como inicialización para una tarea específica de menor tamaño. Esto se ha utilizado con éxito en varias tareas de aprendizaje automático, como la clasificación de imágenes (Chen et al., 2021) y el procesamiento del lenguaje natural (Liu y Lapata, 2019). También se ha utilizado con éxito en el contexto de redes neuronales profundas, donde ha demostrado ser una técnica efectiva para mejorar el rendimiento y la capacidad de generalización de las redes (Girshick et al., 2014).

Por otro lado, el fine-tuning es una técnica en la que se utilizan las capas de una red pre-entrenada en una tarea como capas fijas en otra red que se entrena en una tarea diferente (Yosinski et al., 2014). Esto permite que la red aprenda de manera más rápida y con menos datos (Howard y Ruder, 2018).

## 2.4. Adaptación de Dominio

El *pre-entrenamiento* y el *fine-tuning* han mejorado considerablemente el estado del arte para diversos problemas y aplicaciones del machine learning, incluso las redes profundas pre-entrenadas pueden adaptarse fácilmente a tareas donde se cuenta con una pequeña cantidad de datos etiquetados. Sin embargo, en muchos escenarios prácticos, no hay datos de entrenamiento etiquetados y, por lo tanto, existe la necesidad de transferir el aprendizaje de una red profunda desde un dominio de origen en el que se dispone de datos de entrenamiento etiquetados a un dominio de destino en el que solo existen datos sin etiquetar (Glorot, Bordes y Bengio, 2011). En esta situación, los modelos profundos siguen sufriendo degradaciones de rendimiento debido al cambio de distribución (Quinonero-Candela et al., 2008). Por lo tanto, se propone la *adaptación de dominio* para reducir el cambio de distribución entre los dominios de entrenamiento y de prueba (Jiang et al., 2022).

Se han propuesto muchos métodos para la adaptación de dominios, ya sea ponderando o seleccionando muestras del dominio de origen (Sugiyama et al., 2007) o buscando una transformación de la distribución de origen a la distribución de destino (Gong, Grauman y Sha, 2013). Esta tesis analizará algunos de los métodos más comunes del ultimo caso.

### 2.4.1. Domain Adversarial Neural Network

Un gran hito al momento de modelar distribuciones es la Red Generativa Adversaria GAN (Goodfellow et al., 2020). Inspirada en estas arquitecturas

GAN, la Red Neural Adversaria de Dominio *DANN* (Ganin et al., 2016) de tres componentes clave: la generadora de características  $\mathcal{G}$ , la discriminadora de dominios  $\mathcal{D}$  y la red clasificadora  $\mathcal{C}$  como se ilustra en la figura 2.6. En primer lugar, la generadora de características  $\mathcal{G}$  se entrena con el objetivo de confundir a la discriminadora  $\mathcal{D}$  y al mismo tiempo ayudar a la red clasificadora  $\mathcal{C}$ . La idea principal es que  $\mathcal{G}$  aprenda a generar características que sean similares en ambos dominios, de manera que la discriminadora  $\mathcal{D}$  se equivoque al distinguir entre las características de origen y destino. Esta estrategia de engañar a  $\mathcal{D}$  ayuda a  $\mathcal{C}$  a clasificar correctamente las características entre ambos dominios. Por otro lado, la discriminadora de dominios  $\mathcal{D}$  es entrenada para diferenciar entre las características de origen y las de destino. Su función es detectar las diferencias inherentes a cada uno de ellos. Al entrenar a  $\mathcal{D}$  en la discriminación de dominios, se impulsa a  $\mathcal{G}$  a generar características que sean indistinguibles para  $\mathcal{D}$ , fomentando así la adaptación de dominio.

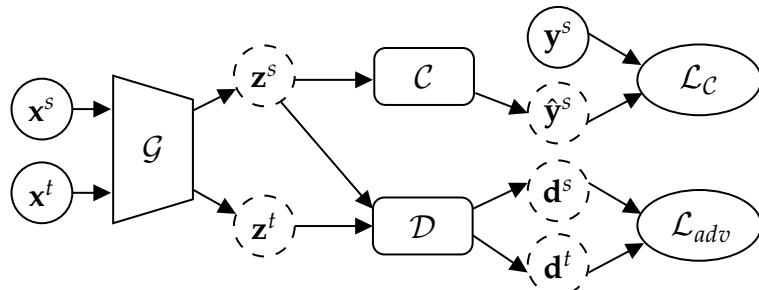


FIGURA 2.6: Esquema de las *DANN*. Los supra índices  $s$  y  $t$  indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos.

El objetivo de *DANN* consiste en minimizar función de pérdida que depende de  $\mathcal{L}_C$  y  $\mathcal{L}_{adv}$ .  $\mathcal{L}_C$  mide el error que posee la red en la clasificación de los datos de origen y viene dada por el *cross-entropy*  $\mathcal{L}_{CE}$  aplicado a la salida de la red  $\mathcal{C}$ . El objetivo de  $\mathcal{D}$  será la descripta en la ecuación 2.3:

$$\begin{aligned}
 \min_{\mathcal{C}} \mathcal{L}_C(\mathbf{x}^s, \mathbf{y}^s) &= \mathcal{L}_{CE}(C(\mathcal{G}(\mathbf{x}^s)), \mathbf{y}^s) \\
 &= \mathcal{L}_{CE}(C(\mathbf{z}^s), \mathbf{y}^s) \\
 &= \mathcal{L}_{CE}(\hat{\mathbf{y}}^s, \mathbf{y}^s)
 \end{aligned} \tag{2.3}$$

La función de pérdida de  $\mathcal{D}$  viene dada por la ecuación 2.4, donde  $\mathbb{E}_{\mathbf{x}^s \sim \hat{\mathcal{S}}}$  y  $\mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}}$  representan la proporción esperada de datos de origen  $\hat{\mathcal{S}}$  y destino  $\hat{\mathcal{T}}$  respectivamente.

$$\begin{aligned}
 \max_{\mathcal{D}} \mathcal{L}_{adv}(\mathbf{x}^s, \mathbf{x}^t) &= \mathbb{E}_{\mathbf{x}^s \sim \hat{\mathcal{S}}} \log[\mathcal{D}(\mathcal{G}(\mathbf{x}^s))] + \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \log[1 - \mathcal{D}(\mathcal{G}(\mathbf{x}^t))] \\
 &= \mathbb{E}_{\mathbf{x}^s \sim \hat{\mathcal{S}}} \log[\mathcal{D}(\mathbf{z}^s)] + \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \log[1 - \mathcal{D}(\mathbf{z}^t)] \\
 &= \mathbb{E}_{\mathbf{x}^s \sim \hat{\mathcal{S}}} \log[\mathbf{d}^s] + \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \log[1 - \mathbf{d}^t]
 \end{aligned} \tag{2.4}$$

Por lo tanto, el objetivo de un modelo *DANN* viene dado por la ecuación 2.5. Donde  $\lambda$  es un hiper-parámetro a optimizar que regula el trade-off entre la clasificación y la discriminación adversaria. La minimización de  $\mathcal{L}_C$  dará lugar a representaciones discriminables, mientras que la disminución de  $\mathcal{L}_{\perp \sqsubseteq}$  dará lugar a representaciones transferibles.

$$\min_{\mathcal{G}, \mathcal{C}} \mathcal{L}_C(\mathbf{x}^s, \mathbf{y}^s) + \lambda \mathcal{L}_{adv}(\mathbf{x}^s, \mathbf{x}^t) \quad (2.5)$$

#### 2.4.2. Adversarial Discriminative Domain Adaptation

Las redes Adversarial Discriminative Domain Adaptation *ADDA* (Tzeng et al., 2017) surgen a partir de las *DANN* como propuesta para mitigar el desvanecimiento de gradiente, un problema importante en el entrenamiento de GANs. Por ejemplo, cuando el espacio latente de destino generado  $\mathbf{z}^t$  es extremadamente distingible del de origen tal que  $\mathcal{D}(\mathbf{z}^t) = 0$ , el gradiente es pequeño y vice versa. Esto provoca que la optimización de  $\mathcal{G}$  sea extremadamente difícil.

Las redes *ADDA* separan la optimización de  $\mathcal{G}$  y de  $\mathcal{D}$  en dos objetivos independientes como se muestra en la figura 2.7. El primero es el pre-entrenamiento de  $\mathcal{G}_s$  y  $\mathcal{C}$  en los datos de origen y el otro la adaptación adversaria de una  $\mathcal{G}_t$  utilizando a  $\mathcal{G}_s$  pre-entrenada y  $\mathcal{D}$ .

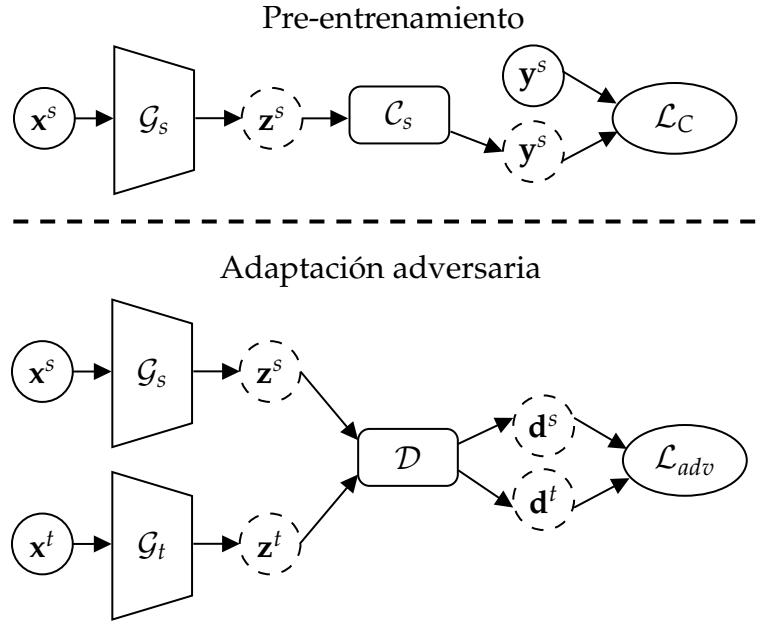


FIGURA 2.7: Esquema de las ADDA. Los supra indices  $s$  y  $t$  indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. La primera fase consta de un pre-entrenamiento con una red generadora  $\mathcal{G}_s$  con los datos origen y la segunda fase de adaptación consta de otra red generadora  $\mathcal{G}_t$  que debe aprender a generar el mismo espacio latente que  $\mathcal{G}_s$  con los datos de destino para confundir a  $\mathcal{D}$ .

No obstante del cambio, los objetivos de ADDA son similares a los de las DANN a excepción de  $\mathcal{G}_t$ . La ecuación 2.6 contiene  $\mathcal{L}_C$  que es análoga a 2.3. La ecuación 2.7 contiene  $\mathcal{L}_{adv}$  que es análoga a 2.4. Finalmente, la ecuación 2.8 contiene el objetivo a optimizar, que asigna gradientes pequeños a registros de destino que sean similares a los de origen y gradientes mas grandes para los otros registros de destino.

$$\min_{\mathcal{G}_s, \mathcal{C}} \mathcal{L}_C(\mathbf{x}^s, \mathbf{y}^s) = \mathcal{L}_{CE}(\mathcal{C}_s(\mathcal{G}_s(\mathbf{x}^s)), \mathbf{y}^s) \quad (2.6)$$

$$\max_{\mathcal{D}} \mathcal{L}_{adv}(\mathbf{x}^s, \mathbf{x}^t) = \mathbb{E}_{\mathbf{x}^s \sim \mathcal{S}} \log[\mathcal{D}(\mathcal{G}_s(\mathbf{x}^s))] + \mathbb{E}_{\mathbf{x}^t \sim \mathcal{T}} \log[1 - \mathcal{D}(\mathcal{G}_t(\mathbf{x}^t))] \quad (2.7)$$

$$\min_{\mathcal{G}_t} -\mathbb{E}_{\mathbf{x}^t \sim \mathcal{T}} \log[\mathcal{D}(\mathcal{G}_t(\mathbf{x}^t))] \quad (2.8)$$

### 2.4.3. Batch Spectral Penalization

Aunque los métodos de adaptación de dominio adversarios mejoran la *transferibilidad* de las características aprendidas; es decir, la capacidad de que las representaciones puedan superar las discrepancias entre los dominios, lo hacen a costa de la *discriminabilidad*; es decir, la facilidad de separar categorías sobre las representaciones de ambos dominios (Chen et al., 2019). Al

aplicar descomposición en valores singulares (SVD en inglés) para analizar las propiedades espectrales de las representaciones aprendidas en batches, se confirma que los autovectores con valores singulares más altos dominan la *transferibilidad* mientras que los autovectores con valores singulares más pequeños se encuentran penalizados, lo que provoca una *discriminabilidad* deficiente. *Batch Spectral Penalization BSP* (Chen et al., 2019) penaliza el mayor valor singular para que los demás autovectores puedan mejorar la *discriminabilidad*.  $\mathcal{L}_{BSP}$  es propuesto como un término de regularización sobre los  $k$  mayores valores singulares:

$$\mathcal{L}_{BSP}(\mathbf{z}) = \sum_{i=0}^k (\sigma_{s,i}^2 + \sigma_{t,i}^2), \quad (2.9)$$

donde  $\sigma_{s,i}$  y  $\sigma_{t,i}$  corresponden al  $i$ -ésimo mayor valor singular de  $\Sigma_s$  y  $\Sigma_t$  respectivamente. En la presente tesis se utilizará el valor por defecto propuesto por Chen et al. de  $k = 1$ .

*BSP* puede integrarse a cualquiera de los esquemas mencionados anteriormente, por ejemplo a una *DANN* como se muestra en la figura 2.8.

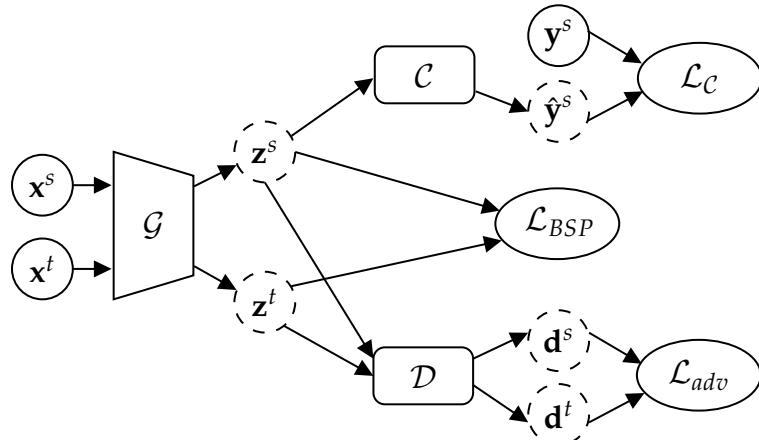


FIGURA 2.8: Esquema de un modelo *DANN+BSP*.

Por lo tanto, un modelo *DANN+BSP* presenta como objetivos:

$$\min_{\mathcal{G}, \mathcal{C}} \mathcal{L}_C(x^s, y^s) + \lambda \mathcal{L}_{adv}(x^s, x^t) + \beta \mathcal{L}_{BSP}(\mathcal{G}(x)) \quad (2.10)$$

$$\max_{\mathcal{D}} \mathcal{L}_{adv}(x^s, x^t), \quad (2.11)$$

donde  $\beta$  es el parámetro que regula el trade-off de  $\mathcal{L}_{BSP}$ .

#### 2.4.4. Margin Disparity Discrepancy

Las técnicas detalladas hasta el momento implican algoritmos de optimización minimax, que funcionan correctamente en los métodos basados en el aprendizaje adversario. Sin embargo, estos algoritmos utilizan funciones de scoreo que carecen de garantías teóricas ya que se estudiaron funciones de

pérdida 0-1. Es por esto que se introducen métodos que apuntan a reducir la brecha que existe entre la teoría y el algoritmo, como Margin Disparity Discrepancy *MDD* (Zhang et al., 2019) esquematizado en la figura 2.9. La idea general de *MDD* es introducir una segunda red clasificadora  $\mathcal{C}'$  y relacionar su salida con la de  $\mathcal{C}$ .

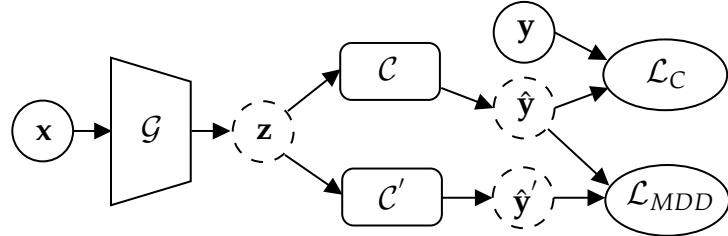


FIGURA 2.9: Esquema de *MDD*. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. *MDD* introduce un clasificador clasificador adversario  $\mathcal{C}'$  para maximizar la discrepancia y entrena el generador de características  $\mathcal{G}$  para minimizar el error de origen como tambien la discrepancia.

$\mathcal{L}_C$  continua siendo la función *cross-entropy*  $\mathcal{L}_{CE}$  aplicado a la salida de la red  $\mathcal{C}$ :

$$\min_{\mathcal{C}} \mathcal{L}_C(\mathbf{x}^s, \mathbf{y}^s) = \mathcal{L}_{CE}(C(\mathcal{G}(\mathbf{x}^s)), \mathbf{y}^s) \quad (2.12)$$

Por otro lado,  $\mathcal{L}_{MDD}$  contiene un *margen*  $\rho$  que se obtiene introduciendo el parámetro  $\gamma \triangleq \exp \rho$  en la ecuación 2.13.

$$\begin{aligned} \max_{\mathcal{C}'} \mathcal{L}_{MDD}(\mathbf{x}^s, \mathbf{x}^t) = & \gamma \mathbb{E}_{\mathbf{x}^s \sim \hat{\mathcal{S}}} \log[\sigma(\mathcal{C}'(\mathcal{G}(\mathbf{x}^s)))] + \\ & \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \log[1 - \sigma(\mathcal{C}'(\mathcal{G}(\mathbf{x}^t)))] \end{aligned} \quad (2.13)$$

donde  $\sigma$  es la función *softmax*. Por lo tanto, el objetivo de optimización viene dado por la ecuación 2.14, donde  $\lambda$  es el trade-off entre la clasificación y la discriminación adversaria.

$$\min_{\mathcal{G}, \mathcal{C}} \mathcal{L}_C(\mathbf{x}^s, \mathbf{x}^t) + \lambda \mathcal{L}_{MDD}(\mathbf{x}^s, \mathbf{x}^t) \quad (2.14)$$

#### 2.4.5. Adaptive Feature Norm

El mayor problema que surge de la adaptación de dominio es la degradación del modelo en cuanto a la tasa de acierto en la clasificación (Yosinski et al., 2014). Las divergencias estadísticas existentes pueden no representar con

precisión el cambio de dominio y subsanar tales discrepancias puede no garantizar la transferencia entre dominios. Tales discrepancias se ejemplifican en la figura 2.10.

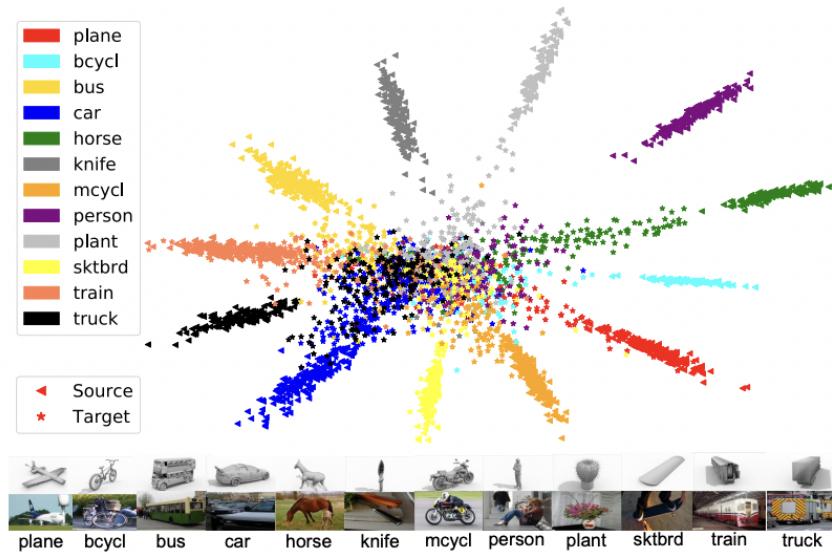


FIGURA 2.10: Visualización de las características aprendidas para el origen y destino utilizando un modelo entrenado utilizando muestras del origen. Imagen tomada de Xu et al., 2019.

Esto sugiere que las normas excesivamente pequeñas de las características de destino con respecto a las de origen explican el deterioro en la clasificación. No obstante, pueden plantearse dos hipótesis:

- Normas de las características desalineadas: el desplazamiento entre los dominios de origen y destino se explica en la desalineación de los valores esperados de las normas de sus características. La coincidencia de las normas de ambos dominios a un escalar arbitrariamente elegido supondrá una mejora en la transferencias.
- Norma de las características más pequeña: el desplazamiento entre los dominios se explica debido a las características menos informativas con normas más pequeñas del dominio de destino. Adaptar las características de destino a regiones alejadas de las normas pequeñas supondrá una mejora en la transferencia.

Xu et al., 2019 propone un método llamado *Adaptive Feature Norm AFN*, ilustrado en la figura 2.11, que consiste en normalizar las normas de las características aprendidas por la red  $\mathcal{G}$  mediante  $n$  bloques  $\mathcal{N}$  (compuestos por FC-BN-ReLU-Dropout) y mediante  $\mathcal{L}_{AFN}$ .

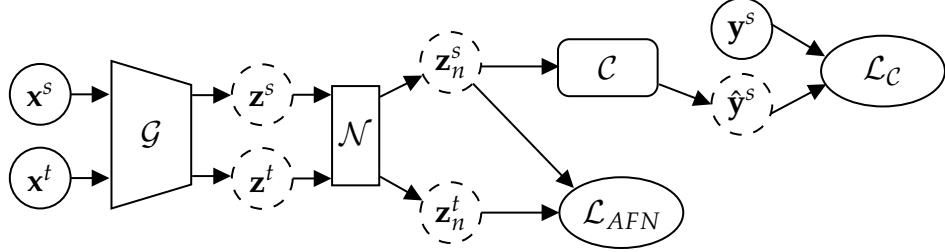


FIGURA 2.11: Esquema de AFN. Los supra indices  $s$  y  $t$  indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. El modelo *backbone*  $\mathcal{G}$  produce características  $\mathbf{z}$  que luego son aplicadas por bloques  $\mathcal{N}$  (FC-BN-ReLU-Dropout) produciendo características normalizadas  $\mathbf{z}_n$ . La red clasificadora  $\mathcal{C}$  toma las características  $\mathbf{z}_n^s$ , predice  $\hat{\mathbf{y}}^s$  y se calcula  $\mathcal{L}_{\mathcal{C}}$ . Finalmente,  $\mathcal{L}_{AFN}$  se calcula utilizando  $\mathbf{z}_n$ .

En su variante iterativa,  $\mathcal{L}_{AFN}$  se define como:

$$\begin{aligned} \min_{\mathcal{G}} \mathcal{L}_{AFN}(\mathbf{x}^s, \mathbf{x}^t) = & \mathbb{E}_{\mathbf{x}^s \sim \mathcal{S}} \mathbb{E}[(\|\mathcal{N}(\mathcal{G}(\mathbf{x}^s))\| - \Delta_r)^2] \\ & + \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \mathbb{E}[(\|\mathcal{N}(\mathcal{G}(\mathbf{x}^t))\| - \Delta_r)^2], \end{aligned} \quad (2.15)$$

donde  $\Delta_r$  corresponde al escalar que controla la amplitud de la norma. Los autores afirman que puede agregarse un término a la ecuación 2.15 respecto a la entropía  $\mathcal{H}$  de la función softmax  $\sigma$  para facilitar la discriminabilidad mediante un factor  $\beta$ :

$$\begin{aligned} \min_{\mathcal{G}} \mathcal{L}_{AFN}(\mathbf{x}^s, \mathbf{x}^t) = & \mathbb{E}_{\mathbf{x}^s \sim \mathcal{S}} \mathbb{E}[(\|\mathcal{N}(\mathcal{G}(\mathbf{x}^s))\| - \Delta_r)^2] \\ & + \mathbb{E}_{\mathbf{x}^t \sim \hat{\mathcal{T}}} \mathbb{E}[(\|\mathcal{N}(\mathcal{G}(\mathbf{x}^t))\| - \Delta_r)^2] \\ & + \beta \mathcal{H}(\sigma(\mathcal{N}(\mathcal{G}(\mathbf{x}^s)))) \end{aligned} \quad (2.16)$$

Por lo tanto, el objetivo de optimización viene dado por la ecuación 2.17, donde  $\lambda$  es el trade-off entre la clasificación y normalización de normas.

$$\min_{\mathcal{G}, \mathcal{C}} \mathcal{L}_{\mathcal{C}}(\mathbf{x}^s, \mathbf{x}^t) + \lambda \mathcal{L}_{AFN}(\mathbf{x}^s, \mathbf{x}^t) \quad (2.17)$$



## Capítulo 3

# Metodología

En este capítulo se detallarán los procesos abordados para la extracción y clasificación de los dígitos escritos en los telegramas de las elecciones. Primero se describirá el proceso de obtención, limpieza y transformación de los datos a fin de obtener un dataset con el cual entrenar. Luego, se describirán los procesos a ser llevados a cabo para el diseño experimental de los modelos.

### 3.1. Descripción de los datos

Los telegramas fueron descargados desde la [página oficial del estado argentino](#) en formato TIFF. Los mismos presentan un formato estándar en forma de grilla, en la que se encuentran los partidos políticos y los votos obtenidos para diputados y senadores en cada renglón. En la página también se puede descargar un archivo en formato CSV donde se encuentra el identificador de cada telegrama y los valores digitalizados oficialmente.

### 3.2. Extracción de dígitos de los telegramas

Durante las elecciones en Santa Fe, los telegramas de los votos presentaron un formato tabular en el que cada fila representaba a un partido político y los votos obtenidos se encontraban divididos por senadores y diputados. Para poder utilizar estos datos y entrenar modelos de aprendizaje automático, fue necesario llevar a cabo un proceso de extracción, transformación y carga (ETL) de los mismos.

Este proceso de ETL implicó la ejecución de múltiples pasos para extraer la información relevante de los telegramas y transformarla en un formato adecuado para su uso en modelos de aprendizaje automático. En primer lugar, se utilizó la librería OpenCV (Bradski, 2000) para manipular las imágenes de los telegramas y poder llevar a cabo el proceso de ETL.

A continuación, se realizó la extracción de la información tabular de los telegramas, identificando las filas correspondientes a cada partido político y los votos obtenidos por senadores y diputados. Este proceso también incluyó la limpieza y validación de los datos para asegurar su calidad y confiabilidad.

Posteriormente, se transformaron los datos extraídos en un formato adecuado para su uso en modelos de aprendizaje automático. Esto implicó la

normalización de los datos, la codificación de variables categóricas y la creación de variables adicionales para mejorar la calidad de los datos. Finalmente, se cargó el conjunto de datos procesados en un dataset que fue utilizado para entrenar los modelos.

### 3.2.1. Enderezado

Cuando se escanean telegramas a mano, es común que estos no estén perfectamente alineados y presenten una ligera inclinación en su posición. Por lo tanto, el primer paso en el procesamiento de los telegramas es enderezarlos para facilitar su lectura y análisis.

Para lograr esto, se emplea un algoritmo que busca el rectángulo de mayor área en la imagen del telegrama. Este rectángulo representa la región que abarca todo el contenido del telegrama y se utiliza para calcular el ángulo de rotación necesario para alinear la imagen.

Una vez que se ha determinado el ángulo de rotación, se aplica una transformación geométrica a la imagen completa para enderezarla. Para realizar esta transformación, se utiliza la función `getRotationMatrix2D` de la biblioteca de procesamiento de imágenes OpenCV. Esta función recibe como argumentos el ángulo de rotación calculado y las dimensiones de la imagen, y devuelve una matriz de transformación que se aplica a la imagen para enderezarla. La figura 3.1 muestra un ejemplo del resultado que se obtiene al enderezar los telegramas de esta manera.

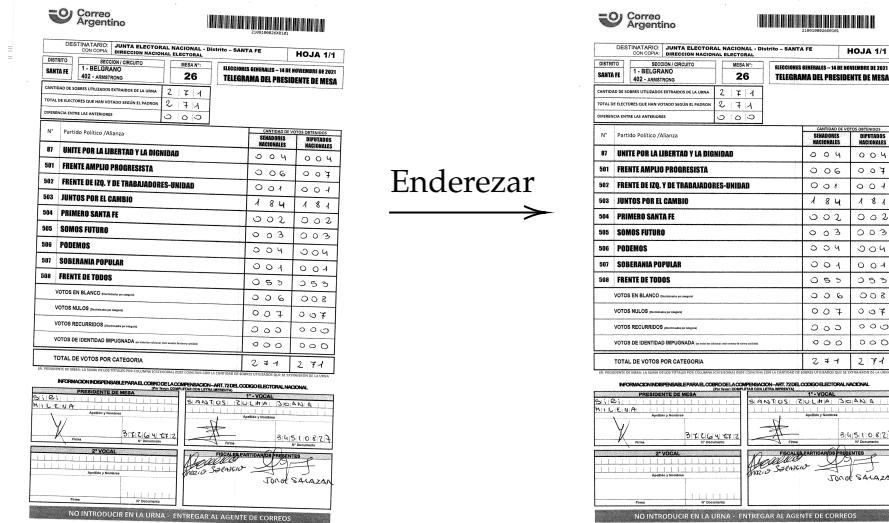


FIGURA 3.1: Proceso de enderezamiento de un telegrama mediante el uso de OpenCV.

### 3.2.2. Extracción de la grilla de votos

El siguiente paso es extraer la grilla de votos para poder trabajar con ella de forma aislada. Para lograr esto, se utiliza la función `getContours` de la biblioteca de procesamiento de imágenes OpenCV.

Esta función permite identificar y seleccionar los contornos presentes en la imagen, que son los bordes que forman los objetos presentes en la misma. Seleccionando el contorno de mayor área, se puede obtener la grilla de votos como un objeto independiente.

Una vez identificado el contorno de mayor área, se procede a extraer la grilla de votos y a trabajar en ella de forma separada del resto de la imagen. De esta manera, se simplifica la tarea de procesamiento de los datos y se puede trabajar de forma más efectiva en la extracción y análisis de los votos presentes en la grilla. La figura 3.2 muestra la grilla que se logra extraer con este proceso.

Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS	
		SEÑADORES MUNICIPALES	DIPUTADOS MUNICIPALES
507	UNITE POR LA LIBERTAD Y LA DIGNIDAD	0 0 4	0 0 4
501	FRENTE ANGULU PROGRESISTA	0 0 6	0 0 7
502	FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD	0 0 1	0 0 1
503	JUNTOS POR EL CAMBIO	1 8 4	1 8 1
504	PRIMERO SANTA FE	0 0 2	0 0 2
505	SOMOS FUTURO	0 0 3	0 0 3
506	PODEMOS	0 0 4	0 0 4
507	SODERANIA POPULAR	0 0 1	0 0 1
508	FRENTE DE TODOS	0 5 5	0 5 5
VOTOS EN BLANCO (descartados por voto nulo)		0 0 6	0 0 8
VOTOS NULOS (descartados por voto nulo)		0 0 7	0 0 7
VOTOS RECURRIDOS (descartados por voto nulo)		0 0 0	0 0 0
VOTOS DE IDENTIDAD IMPUGNADA (se votó la elección pero no se realizó la votación)		0 0 0	0 0 0

FIGURA 3.2: Grilla de votos extraída buscando el contorno de mayor área.

### 3.2.3. Detección de líneas de la grilla de votos

Teniendo la grilla de votos separada del telegrama, se procede a detectar las líneas para poder extraer cada registro de la misma. Si bien OpenCV posee funciones para la detección de líneas, se optó por utilizar un enfoque simplificado similar al utilizado en (Lamagna, 2016). Debido a que la grilla se encuentra enderezada, se pueden utilizar las proyecciones de los colores sobre los ejes  $x$  e  $y$  para detectarlas. Al sumar todos los colores por eje, se pueden observar picos de valores donde se encuentran las líneas horizontales y verticales. Posteriormente, se selecciona un umbral de corte por cada eje, siendo el mismo el promedio de cada eje menos dos desvíos estándar. La figura 3.3 muestra las proyecciones obtenidas con los umbrales por cada eje.

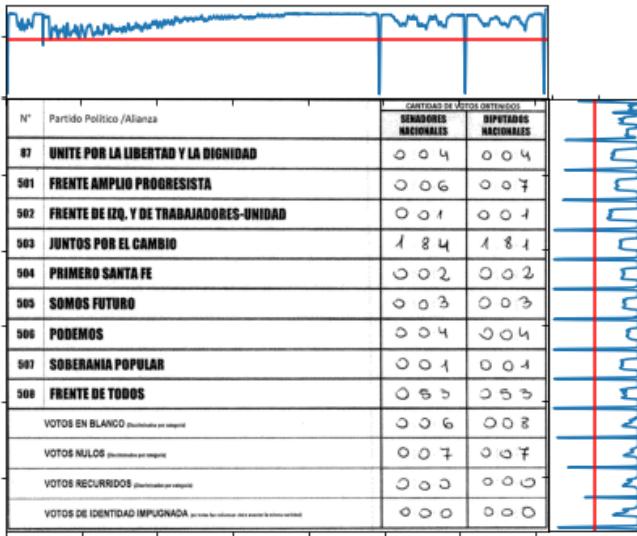


FIGURA 3.3: Proyecciones de los ejes de la grilla de votos. En rojo se marca el umbral de corte.

Sin embargo, aunque se seleccionen aquellos píxeles que se encuentren por debajo del umbral, existe mas de un píxel por cada pico. Siguiendo con el ejemplo, en el eje  $x$  se obtienen los siguientes píxeles que se encuentran en los “picos”:

```
array([    3,     4,     5,     6,     7,   100,   119,   120,   988,
       989,   990,   991,   992,  1215,  1216,  1217,  1218,  1219,
      1423,  1424,  1425,  1426,  1427])
```

Con la estrategia del umbral no se obtienen las 4 líneas que se desean detectar sobre el eje  $x$ . No obstante, se puede aplicar un clustering jerárquico sobre los índices obtenidos y luego calcular el índice promedio de cada cluster para tomarlo como único representante. El proceso se repite de la misma manera sobre el eje  $y$ . Esta última modificación sobre el trabajo realizado en (Lamagna, 2016) asegura tener un único píxel por línea. La figura 3.4 muestra la segmentación detectada mediante el proceso descripto.

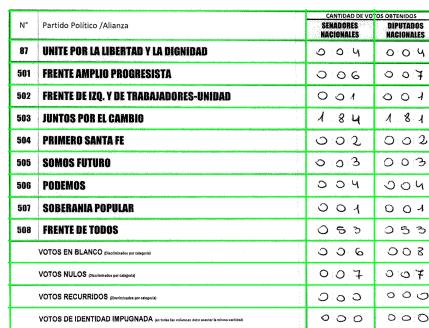


FIGURA 3.4: Grilla detectada (en verde) utilizando el umbral por sobre las proyecciones y su posterior agrupamiento con clustering jerárquico.

### 3.2.4. Segmentación de dígitos

Una vez obtenida la grilla, se itera por cada registro obteniendo los rectángulos que contienen los votos, ignorando el primer renglón que contiene los títulos de la grilla. La figura 3.5 muestra un registro de la grilla obtenida.

87	UNITE POR LA LIBERTAD Y LA DIGNIDAD	0 0 4	0 0 4
----	-------------------------------------	-------	-------

FIGURA 3.5: Primer registro extraído de la grilla de votos

Una vez obtenida la grilla de votos, se busca separar cada dígito de forma individual para poder procesarlos de forma independiente. Para lograr esto, se utiliza un análisis de componentes conectados mediante la función `connectedComponents` de OpenCV.

Esta función permite identificar los distintos componentes (regiones conectadas) presentes en una imagen binaria. En este caso, los componentes corresponden a los dígitos presentes en la grilla de votos. Luego, se establece un rango de áreas permitido para seleccionar los componentes que corresponden a los dígitos.

Para determinar el rango de áreas permitido, se realizó una experimentación previa y se establecieron como límites el 5 % y el 70 % del total de píxeles de la imagen. De esta manera, se descartan aquellos contornos que representan ruido o elementos no deseados que puedan haber sido detectados por el análisis de componentes conectados.

Una vez que se han identificado los componentes correspondientes a los dígitos, se procede a recortarlos individualmente para obtener una imagen de cada dígito por separado. De esta manera, se facilita el procesamiento y análisis de los votos de forma individual. La figura 3.6 muestra los dígitos de un voto determinado obtenidos de esta manera.



FIGURA 3.6: Dígitos detectados en el primer bloque de votos

Luego de haber realizado la segmentación de los dígitos, se procede a aplicar un proceso de transformación para que todas las imágenes de dígitos tengan una dimensión cuadrada.

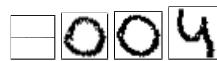
Luego, se guardan en un dataset que también incluye la información correspondiente al partido político y al tipo de voto (senadores o diputados). De esta manera, se construye un conjunto de datos que servirá para entrenar y validar los modelos que permitan analizar los resultados electorales.

## 3.3. Análisis del dataset

Una vez que se ha completado el proceso de extracción y segmentación de los dígitos, se procede a realizar un análisis exploratorio de los datos. Este análisis tiene como objetivo detectar posibles errores o inconsistencias en los

datos extraídos, que puedan afectar la calidad de los modelos de aprendizaje automático que se construirán a partir de ellos.

La tabla 3.1 muestra algunos registros del dataset, que incluye información como el número del telegrama, el partido político, el tipo de voto (senadores o diputados) y los dígitos correspondientes a cada candidato.

Telegrama	Partido	Tipo	Dígitos	# Dígitos
2100100026X	Unite	Diputados		3
2100100026X	Unite	Senadores		4
2100100067X	Frente de Todos	Diputados		4
2100100067X	Frente de Todos	Senadores		3

CUADRO 3.1: Ejemplo de registros del conjunto de datos. Cada fila representa un voto.

Se pueden detectar errores evidentes en la tabla presentada. En primer lugar, se puede observar que en la columna donde solamente deben aparecer las imágenes de los dígitos, en el segundo registro existe una imagen de una línea horizontal, lo cual no es correcto. Además, también se detectan dígitos mal escritos o enmendados, como los que se encuentran en los dos últimos registros.

Para corregir estas extracciones incorrectas, se calculan dos columnas adicionales que contienen las proporciones mínimas y máximas de píxeles blancos en las imágenes para cada voto. Estas proporciones permiten establecer un umbral de confianza para las extracciones, de tal manera que aquellas imágenes con proporciones de píxeles blancos fuera de los límites establecidos son descartadas.

Para visualizar de manera más clara la distribución de estas proporciones, se generan histogramas que muestran la frecuencia de cada valor dentro del rango establecido. La figura 3.7 presenta estos histogramas para las proporciones mínimas y máximas de píxeles blancos en las imágenes del dataset.

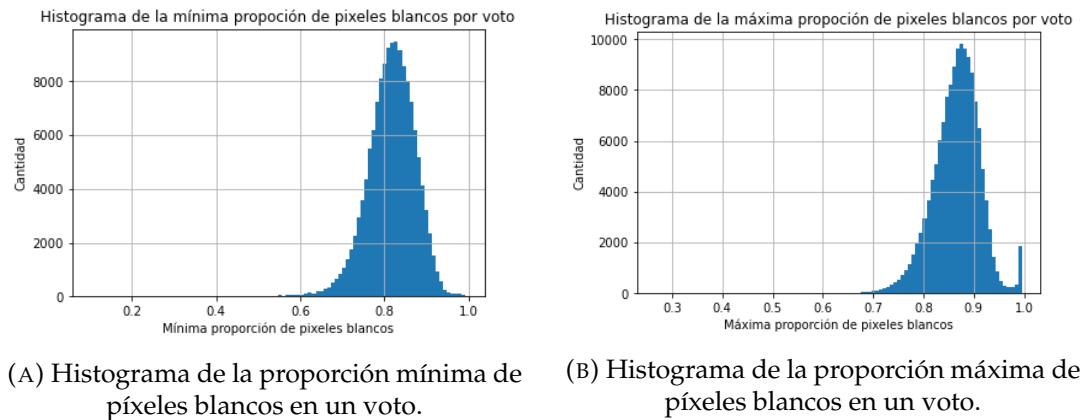


FIGURA 3.7: Histogramas de proporciones de píxeles blancos por voto.

La cola derecha de la distribución de [3.7b](#) presenta un pico anormal en el extremo cercano a 1. Al eliminar aquellas imágenes de dígitos que posean mas de un 95 % de píxeles blancos, se logran descartar los casos de líneas como la vista en el cuadro [3.1](#). Por otro lado, al eliminar aquellas imágenes que posean menos del 50 % de píxeles blancos, se logran descartar los casos de escritura incorrecta.

El siguiente punto a analizar es el tamaño de las imágenes. Como se muestra en el cuadro [3.1](#), las imágenes no fueron estandarizadas en cuanto a tamaño. De manera análoga a la variable de proporción de píxeles blancos, se calculan las variables mínimo y máximo tamaño por voto. La tabla [3.2](#) muestra los estadísticos descriptivos de las mismas.

Variable	Promedio	Desvío	Mínimo	Mediana	Máximo
Tamaño mínimo	38.35	9.61	13	37	621
Tamaño máximo	44.80	10.73	18	43	621

CUADRO 3.2: Estadísticos descriptivos del mínimo y máximo tamaño de las imágenes de los votos.

Al analizar aquellos registros que poseen los valores extremos en cuanto a mínimo y máximo de tamaño, se encuentran telegramas que fueron cargados de forma incorrecta (ver anexo [A.2](#)) o con una caligrafía en la cual los dígitos no están separados entre si, provocando que la segmentación por el análisis de componentes conectados falle (ver anexo [A.3](#)). Se eliminan estos casos filtrando por aquellos registros que se encuentren por fuera de la media mas/menos cuatro desvíos estándar.

Una vez finalizada la etapa de análisis y limpieza, se exportan los dígitos extraídos en un formato similar al conocido dataset *MNIST* junto con la etiqueta correspondiente. Este conjunto de datos será denominado *TDS* (dataset de telegramas) a partir de este momento. El conjunto de datos *TDS* consta de un total de 170.718 imágenes de dígitos cuadradas con sus dimensiones originales, es decir, no se han estandarizado en una dimensión específica. Esta

decisión se tomó intencionalmente, ya que podría permitir una mayor flexibilidad en el perfeccionamiento, procesamiento y análisis de los mismos en trabajos futuros. La figura 3.8 muestra una previsualización del conjunto de datos TDS.



FIGURA 3.8: Muestra de 100 dígitos del conjunto de datos TDS

## 3.4. Diseño experimental

Se llevaron a cabo experimentos con el fin de evaluar el desempeño de dos redes neuronales convolucionales, la ResNet-18 (He et al., 2016) y la LeNet-8 (LeCun et al., 1998), para cada una de las técnicas de adaptación de dominio que fueron descritas en el capítulo 2. Además, como modelo de referencia, se entrenaron ambas redes sin ninguna técnica de adaptación. De esta manera, se busca comparar el rendimiento de las redes con y sin técnicas de adaptación de dominio, para determinar cuál técnica es la más adecuada para el problema en cuestión.

Las tecnologías utilizadas son:

- Python como lenguaje de programación general.
- PyTorch (Paszke et al., 2019) para la implementación de los modelos.
- PyTorch Lightning (Falcon y The PyTorch Lightning team, 2019) para simplificar el entrenamiento de los modelos.
- TLLIB (Junguang Jiang, Chen y Fu, 2020) para aplicar las técnicas de adaptación de dominio.

### 3.4.1. Metodología de entrenamiento

Para entrenar modelos mediante adaptación de dominio, se necesitan dos conjuntos de datos: uno etiquetado, llamado origen, y otro sin etiquetar, llamado destino. En los experimentos llevados a cabo, se utilizó el conjunto de

datos MNIST como origen y el conjunto de datos TDS como destino. Durante cada iteración de entrenamiento, se proporcionó un lote de 1024 imágenes del conjunto de datos de origen y otro del conjunto de datos de destino. Se realizaron tres particiones para cada conjunto de datos:

- Entrenamiento (70 % de cada dataset): utilizado para entrenar la LeNet-8 y ResNet-18 en cada época.
- Validación (15 % de cada dataset): utilizado para calcular los mejores hiperparámetros del modelo.
- Test (15 % de cada dataset): utilizado para calcular las métricas y gráficos finales del modelo entrenado.

Los modelos se entrenaron utilizando Stochastic Gradient Descent (SGD) (Sutskever et al., 2013) utilizando nesterov, 0.9 como momento y 0.001 de weight decay.

### 3.4.2. Selección y optimización del modelo

La optimización de hiperparámetros se realizó utilizando la librería Optuna (Akiba et al., 2019). Se ejecutaron 30 rondas de optimización con 10 épocas de entrenamiento por cada experimento. En cada uno de los experimentos, se optimizaron los hiperparámetros para minimizar el loss de cada modelo. El cuadro 3.3 muestra los rangos búsquedas.

	<i>DANN</i>	<i>ADDA</i>	<i>DANN+BSP</i>	<i>MDD</i>	<i>AFN</i>	<i>Sin AD</i>
$\eta_0$	[1e-4, 0.1]	[1e-4, 0.1]	[1e-4, 0.1]	[1e-4, 0.1]	[1e-4, 0.1]	[1e-4, 0.1]
$\lambda$	[0.5, 2]	[0.5, 2]	[0.5, 2]	[0.5, 2]	[0.001, 0.1]	-
$\beta$	-	-	[1e-5, 0.1]	-	[0.0, 0.1]	-
$\gamma$	-	-	-	[1, 10]	-	-
$\Delta_r$	-	-	-	-	[0.01, 5]	-
# Blocks	-	-	-	-	[1, 4]	-
Dropout	-	-	-	-	[0.3, 0.7]	-

CUADRO 3.3: Rango de hiperparámetros optimizados. \*LR refiere a “learning rate”. \*\*T-O refiere a “trade-off”.

Se implementó una disminución del learning rate en cada iteración de cada época, mediante la fórmula 3.1. Se mantuvieron constantes los valores de  $\gamma = 0,001$  y  $\theta = 0,25$ .

$$\eta(t) = \eta_0 * (1 + \gamma * t)^{-\theta} \quad (3.1)$$

## 3.5. Métricas de evaluación

Los modelos optimizados serán evaluados con distintas métricas en tiempo de test, descriptas en las siguientes sub-secciones del capítulo. Las mismas

pretenden evaluar qué tan buenos son los modelos respecto a la tarea de clasificación y qué capacidad de adaptación de dominio poseen.

### 3.5.1. Métricas de clasificación

#### Accuracy

La métrica de *accuracy* permite identificar qué tan cerca o lejos un conjunto de observaciones se encuentra respecto a los valores reales. Es el ratio de predicciones correctas sobre las totales.

$$\text{Accuracy}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^{n-1} 1(\hat{y}_i = y_i) \quad (3.2)$$

donde:

- $n$ : es la cantidad de observaciones totales.
- $y_i$ : es el valor de la  $i$ -ésima observación correspondiente al real.
- $\hat{y}_i$ : es el valor predicho para la  $i$ -ésima observación.
- $1(x)$ : es la función indicador.

#### $F_1$

La métrica  $F_1$  es una media armónica de otras dos: *precisión* y *recall*. De manera simplificada, la primera muestra la capacidad del modelo de no etiquetar como positivo una obsevación que es negativa y la segunda muestra la capacidad del modelo de encontrar todas las observaciones positivas.

La *precisión* consiste en calcular el ratio de predicciones positivas correctas de la clase respecto al total de predicciones positivas de la clase.

$$\text{Precision}(y, \hat{y}) = \frac{TP}{TP + FP} \quad (3.3)$$

donde:

- $TP$ : es la cantidad de verdaderos positivos.
- $FP$ : es la cantidad de falsos positivos.

Por otro lado, el *recall* consiste en calcular el ratio de predicciones positivas correctas de la clase respecto al total de predicciones correctas de la clase.

$$\text{Recall}(y, \hat{y}) = \frac{TP}{TP + FN} \quad (3.4)$$

donde:

- $TP$ : es la cantidad de verdaderos positivos.
- $FN$ : es la cantidad de falsos negativos.

Las dos métricas mencionadas anteriormente pueden ser combinadas en una sola denominada  $F_\beta$ . El valor de  $\beta$  permite asignar un peso distinto a la precisión o al recall dentro del promedio armónico. Cuando  $\beta = 1$ , ambas poseen el mismo peso.

$$F_\beta(y, \hat{y}) = (1 + \beta^2) \times \frac{\text{precision}(y, \hat{y}) \times \text{recall}(y, \hat{y})}{\beta^2 \times \text{precision}(y, \hat{y}) + \text{recall}(y, \hat{y})} \quad (3.5)$$

El rango de valores es de  $[0, 1]$ , donde 1 corresponde a un clasificador que funciona sin errores.

### Intersección sobre unión

Otra forma de medir el clasificador es mediante la *intersección sobre unión* (IoU por sus siglas en inglés). Consta de calcular la cantidad de dígitos únicos predichos por sobre la cantidad de dígitos únicos reales. Por ejemplo, para un telegrama y un partido el clasificador predice 189 votos cuando lo real es 180. Entonces, el *IoU* viene dado por:

$$\begin{aligned} \text{IoU}(\{1, 8, 0\}, \{1, 8, 9\}) &= \frac{|\{1, 8, 0\} \cap \{1, 8, 9\}|}{|\{1, 8, 0\} \cup \{1, 8, 9\}|} \\ &= \frac{|\{1, 8\}|}{|\{1, 8, 0, 9\}|} \\ &= \frac{2}{4} \\ &= 0,5 \end{aligned} \quad (3.6)$$

De forma general: sea  $y_i$  el dígito  $i$ -ésimo del voto real  $y$  de longitud  $n$ ,  $\hat{y}_j$  el dígito  $j$ -ésimo del voto predicho por el modelo  $\hat{y}$  de longitud  $m$ , entonces la métrica viene dada por:

$$\text{IoU}(y, \hat{y}) = \frac{|\{y_i\} \cap \{\hat{y}_j\}|}{|\{y_i\} \cup \{\hat{y}_j\}|} \quad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, m\} \quad (3.7)$$

### 3.5.2. Métricas de adaptación

#### Distancia $\mathcal{A}$

La distancia  $\mathcal{A}$  mide la similaridad entre dos distribuciones. Puede ser utilizada para analizar los espacios latentes de clasificadores en problemas de adaptación de dominio (Ben-David et al., 2006). La métrica viene dada por:

$$\text{dist}_{\mathcal{A}} = 2(1 - 2\epsilon), \quad (3.8)$$

donde  $\epsilon$  es el error en test de un clasificador entrenado para discriminar el dominio de origen del de destino. Cuando el error de clasificación es bajo,

significa que hay diferencias significativas entre las dos distribuciones, haciendo que  $dist_{\mathcal{A}}$  sea grande y vice versa.

En la figura 3.9 se muestran posibles casos de distribuciones de dominios. En la sub-figura 3.9a las distribuciones de dominios son significativamente diferentes entre si, por lo tanto  $dist_{\mathcal{A}} \approx 2$ . Por el contrario, en la sub-figura 3.9b las distribuciones de dominios son similares entre si, por lo que se espera que  $dist_{\mathcal{A}} \approx 0$ .

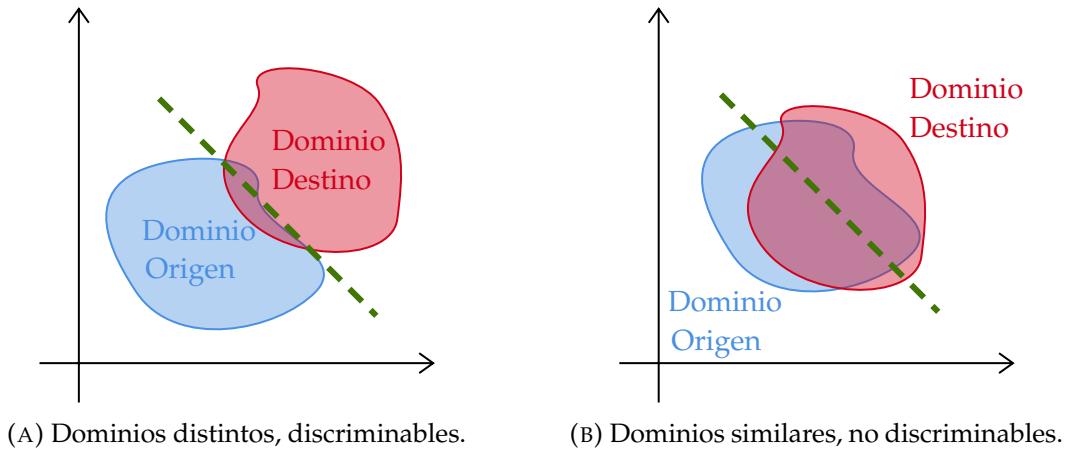


FIGURA 3.9: Ejemplos de distribuciones de dominios.

Durante el proyecto se utiliza como clasificador una regresión logística a fin de calcular la distancia  $\mathcal{A}$ .

### Discrepancia de Medias Máxima

La discrepancia de medias máxima (MMD por sus siglas en inglés) es una prueba estadística basada en *kernels* que se utiliza para determinar si dos distribuciones dadas son iguales midiendo la distancia entre ellas (Gretton et al., 2012). Dadas las distribuciones  $P$  y  $Q$  sobre un conjunto  $\mathcal{X}$  y un mapa de características  $\phi : \mathcal{X} \rightarrow \mathcal{H}$  donde  $\mathcal{H}$  es un conjunto de destino, MMD se define como:

$$MMD^2(P, Q) = \|\mathbb{E}_{\mathbf{x} \sim P}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{y} \sim Q}[\phi(\mathbf{y})]\|_{\mathcal{H}}^2 \quad (3.9)$$

Por ejemplo, si el mapa de características es  $\phi(x) = x$  y  $\mathcal{X} = \mathcal{H} = \mathbb{R}^d$ , MMD se calcula como:

$$\begin{aligned} MMD^2(P, Q) &= \|\mathbb{E}_{\mathbf{x} \sim P}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{y} \sim Q}[\phi(\mathbf{y})]\|_{\mathcal{H}}^2 \\ &= \|\mathbb{E}_{\mathbf{x} \sim P}[\mathbf{x}] - \mathbb{E}_{\mathbf{y} \sim Q}[\mathbf{y}]\|_{\mathbb{R}^d}^2 \\ &= \|\mu_P - \mu_Q\|_{\mathbb{R}^d}^2, \end{aligned} \quad (3.10)$$

de forma que MMD es la distancia entre las medias de las distribuciones. No obstante, evaluarlas de esta manera puede generar que distribuciones

con distinta varianza, kurtosis, etc sean consideradas como iguales. Se puede aplicar el *kernel trick* para calcular MMD con distancias más complejas. Dado  $k(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{H}}$ , MMD se calcula como:

$$\begin{aligned} MMD^2(P, Q) &= \|\mathbb{E}_{\mathbf{x} \sim P}[\phi(\mathbf{x})] - \mathbb{E}_{\mathbf{y} \sim Q}[\phi(\mathbf{y})]\|_{\mathcal{H}}^2 \\ &= \mathbb{E}_{\mathbf{x}, \mathbf{x}' \sim P} k(\mathbf{x}, \mathbf{x}') - \mathbb{E}_{\mathbf{y}, \mathbf{y}' \sim Q} k(\mathbf{y}, \mathbf{y}') - 2\mathbb{E}_{\mathbf{x} \sim P, \mathbf{y} \sim Q} k(\mathbf{x}, \mathbf{y}) \end{aligned} \quad (3.11)$$

Se utilizó la librería TorchDrift (Viehmann, Antiga y Marcolini, 2021) para implementarla en la evaluación de los modelos utilizando como *kernel* RBF.



# Capítulo 4

## Análisis de resultados

A lo largo del presente capítulo, se analizarán los resultados obtenidos a partir de los experimentos planteados. Se detallarán las métricas obtenidas, se visualizarán los espacios latentes  $\mathbf{z}$  generados por la parte convolucional de las redes y se analizarán los errores que surgen de aplicar los modelos a los telegramas.

### 4.1. Análisis de métricas

Los experimentos realizados arrojaron los resultados mostrados en el cuadro 4.1. Las métricas referidas a la capacidad de clasificación (Accuracy,  $F_1$ ) son evaluadas sobre la partición de test del dataset de origen donde se conocen las etiquetas a ciencia cierta (MNIST). Por otro lado, las métricas de adaptación ( $MMD$ , Dist.  $\mathcal{A}$ ) son evaluadas sobre los espacios latentes que los modelos generaron para las particiones de test de ambos datasets.

Es importante destacar que el mejor modelo es aquel que tiene tanto la mejor capacidad de clasificación como la mejor adaptación a nuevos conjuntos de datos. Las métricas de clasificación deben ser maximizadas y las de adaptación deben ser minimizadas.

AD	Modelo	Acc.	$F_1$	$MMD$	Dist. $\mathcal{A}$
-	ResNet	(2) 0.9885	(2) 0.9883	0.0632	1.9306
	LeNet	0.9811	0.9810	0.0469	1.9515
DANN	ResNet	<b>(1) 0.9890</b>	<b>(1) 0.9890</b>	0.1379	1.9776
	LeNet	0.9822	0.9821	(3) 0.0090	1.6774
ADDA	ResNet	0.9476	0.9485	0.0165	1.8495
	LeNet	0.9191	0.9184	0.0399	1.8495
DANN+BSP	ResNet	0.9780	0.9777	0.0409	1.7888
	LeNet	0.9859	0.9857	(2) 0.0051	(3) 1.6369
MDD	ResNet	(3) 0.9864	(3) 0.9863	0.0615	1.8987
	LeNet	0.9856	0.9854	0.0399	1.7468
AFN	ResNet	0.9829	0.9827	<b>(1) 0.0040</b>	<b>(1) 1.0886</b>
	LeNet	0.9862	0.9859	0.0117	(2) 1.5747

CUADRO 4.1: Métricas de los experimentos realizados. Entre paréntesis se encuentra la posición que ocupa dentro del top 3 de la columna.

Se observa que los modelos entrenados sin adaptación de dominio presentan las mejores métricas de clasificación, como se esperaba. Sin embargo, estos modelos tienen los peores valores de adaptación, lo que hace que no puedan generalizarse a nuevos conjuntos de datos, como TDS.

El modelo ResNet18, utilizando AFN, demuestra ser el modelo que combina mejor las métricas de clasificación y adaptación. A pesar de esto, se destaca que los modelos LeNet entrenados con DANN y AFN obtuvieron buenas métricas en general, lo que sugiere que en ocasiones, un modelo más complejo no necesariamente garantiza un mejor rendimiento.

Los modelos se aplicaron a todos los telegramas, y se calculó el promedio de *IoU* por telegrama utilizando el método descrito en el capítulo 3. Además, se determinó la cantidad promedio de aciertos por telegrama utilizando las etiquetas transcritas en el centro de cómputo, asumiendo que hay pocos errores en ellas. El cuadro 4.2 muestra el *IoU* promedio y la cantidad de aciertos promedios obtenidos para los telegramas de prueba.

AD	Modelo	<i>IoU</i> prom.	# aciertos prom.	% aciertos prom.
-	ResNet	0.4494	4	22 %
	LeNet	0.4715	6	33 %
DANN	ResNet	0.6941	12	67 %
	LeNet	0.7024	12	67 %
ADDA	ResNet	0.6763	11	61 %
	LeNet	0.6406	10	56 %
DANN+BSP	LeNet	0.6695	11	61 %
	ResNet	0.6515	11	61 %
MDD	ResNet	0.5451	8	44 %
	LeNet	0.5801	9	50 %
AFN	ResNet	<b>0.7486</b>	<b>13</b>	<b>72 %</b>
	LeNet	0.6493	11	61 %

CUADRO 4.2: *IoU* promedio y cantidad promedio de aciertos al aplicar los modelos a cada telegrama.

Analizando el cuadro anterior, se confirma la elección del mejor modelo ResNet18 con AFN. Es importante destacar que independientemente de la técnica de adaptación utilizada, todos los modelos presentan mejores porcentajes de aciertos promedio que aquellos modelos entrenados únicamente con MNIST.

Esto sugiere que la adaptación de dominio es fundamental para obtener un mejor rendimiento en la tarea de clasificación de telegramas. Los modelos entrenados con técnicas de adaptación logran generalizar mejor y son más efectivos en la tarea de reconocimiento de los patrones presentes en los telegramas, incluso en presencia de ruido y variaciones en los datos de entrada.

## 4.2. Análisis de los espacios latentes

La capacidad de adaptación de dominio es una medida que se utiliza para evaluar el rendimiento de un modelo de aprendizaje automático en un dominio de prueba diferente al dominio en el que se entrenó. Uno de los factores clave que influyen en la capacidad de adaptación de dominio es la calidad de los espacios latentes que genera el modelo durante la etapa de entrenamiento.

El espacio latente en el contexto de las redes convolucionales se refiere a una representación de los datos de entrada que se ha aprendido durante el entrenamiento del modelo. Durante dicho proceso, la red aprende a extraer características relevantes de las imágenes de entrada a través de la etapa convolucional.

El espacio latente  $\mathbf{z}$  se genera después de la etapa convolucional y antes de la etapa densa de la red. En este espacio latente, cada dimensión representa una característica específica de la imagen de entrada. Por ejemplo, en el caso de una red convolucional entrenada para clasificar imágenes de gatos y perros, una dimensión podría representar la forma de las orejas y otra dimensión podría representar el color de la nariz.

Para evaluar la capacidad de adaptación de dominio de un modelo, se pueden comparar las distribuciones de los espacios latentes generados para diferentes conjuntos de datos. En general, se espera que los espacios latentes generados para diferentes conjuntos de datos sean similares, ya que esto indica que el modelo es capaz de generalizar.

Si los espacios latentes son iguales o similares para diferentes conjuntos de datos, la etapa densa de la red que se encarga de la clasificación puede realizar predicciones precisas para el dominio de destino. Por otro lado, si los espacios latentes son muy diferentes para diferentes conjuntos de datos, esto puede indicar que el modelo no es capaz de adaptarse a nuevos conjuntos de datos y, por lo tanto, su capacidad de generalización es limitada.

Una forma común de visualizarlos es mediante una técnica de reducción de dimensionalidad, como Uniform Manifold Approximation and Projection (McInnes, Healy y Melville, 2018). UMAP es una técnica no lineal que se utiliza para visualizar estructuras de datos complejas en un espacio de menor dimensión.

Al visualizar los espacios latentes utilizando UMAP, se pueden observar patrones y relaciones en los datos que de otra manera serían difíciles de detectar. Al comparar las distribuciones de los espacios latentes generados por un modelo para diferentes conjuntos de datos, se pueden identificar las similitudes y diferencias entre los dominios, lo que puede ser útil para evaluar la capacidad de adaptación de dominio del modelo.

En las siguientes subsecciones se analizarán las representaciones UMAP de los espacios latentes obtenidos por cada modelo entrenado.

### 4.2.1. LeNet

La figura 4.1 a continuación contiene las representaciones UMAP obtenidas de los espacios latentes generados por todos los modelos LeNet.

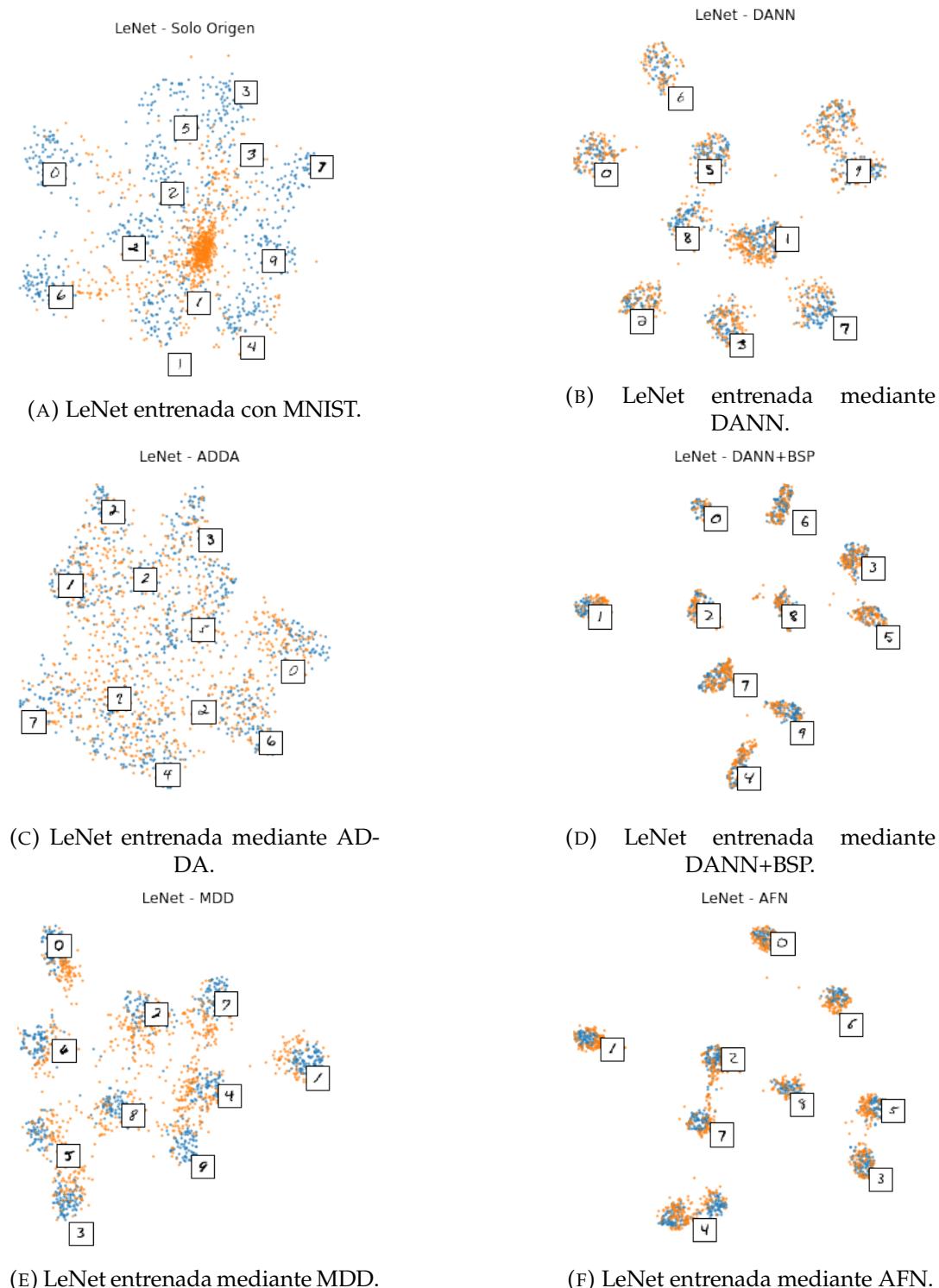


FIGURA 4.1: Representación UMAP de las características aprendidas de los modelos LeNet5. Los puntos azules representan observaciones de MNIST y los naranjas de TDS.

Se pueden observar los siguientes resultados:

- Entrenar el modelo solo con MNIST no permite una buena generalización para TDS, como se muestra en la figura 4.1a.
- Con el entrenamiento mediante ADDA o MDD se comienzan a generar agrupaciones en MNIST, pero no logran generarse de la misma manera en TDS, como se puede apreciar en las figuras 4.1c y 4.1e.
- Al entrenar el modelo mediante DANN, se generan agrupaciones similares para ambos conjuntos de datos, como se muestra en la figura 4.1b.
- Finalmente, al utilizar la técnica de entrenamiento DANN con penalización BSP y AFN, se generan agrupaciones similares y más densas para ambos conjuntos de datos, tal como se muestra en las figuras 4.1d y 4.1f.

En conclusión, se puede afirmar que la técnica de adaptación de dominio utilizada tiene un gran impacto en el rendimiento de un modelo LeNet debido a su estructura simple. Entre las diferentes técnicas evaluadas, se puede observar que DANN+BSP y AFN son las que proporcionan una mejor capacidad de adaptación al modelo LeNet. En general, estos resultados resaltan la importancia de seleccionar cuidadosamente la técnica de adaptación de dominio adecuada para un modelo dado a fin de lograr un rendimiento óptimo en diferentes escenarios de aplicación.

#### 4.2.2. ResNet

La figura 4.2 a continuación contiene las representaciones UMAP obtenidas de los espacios latentes generados por todos los modelos ResNet.

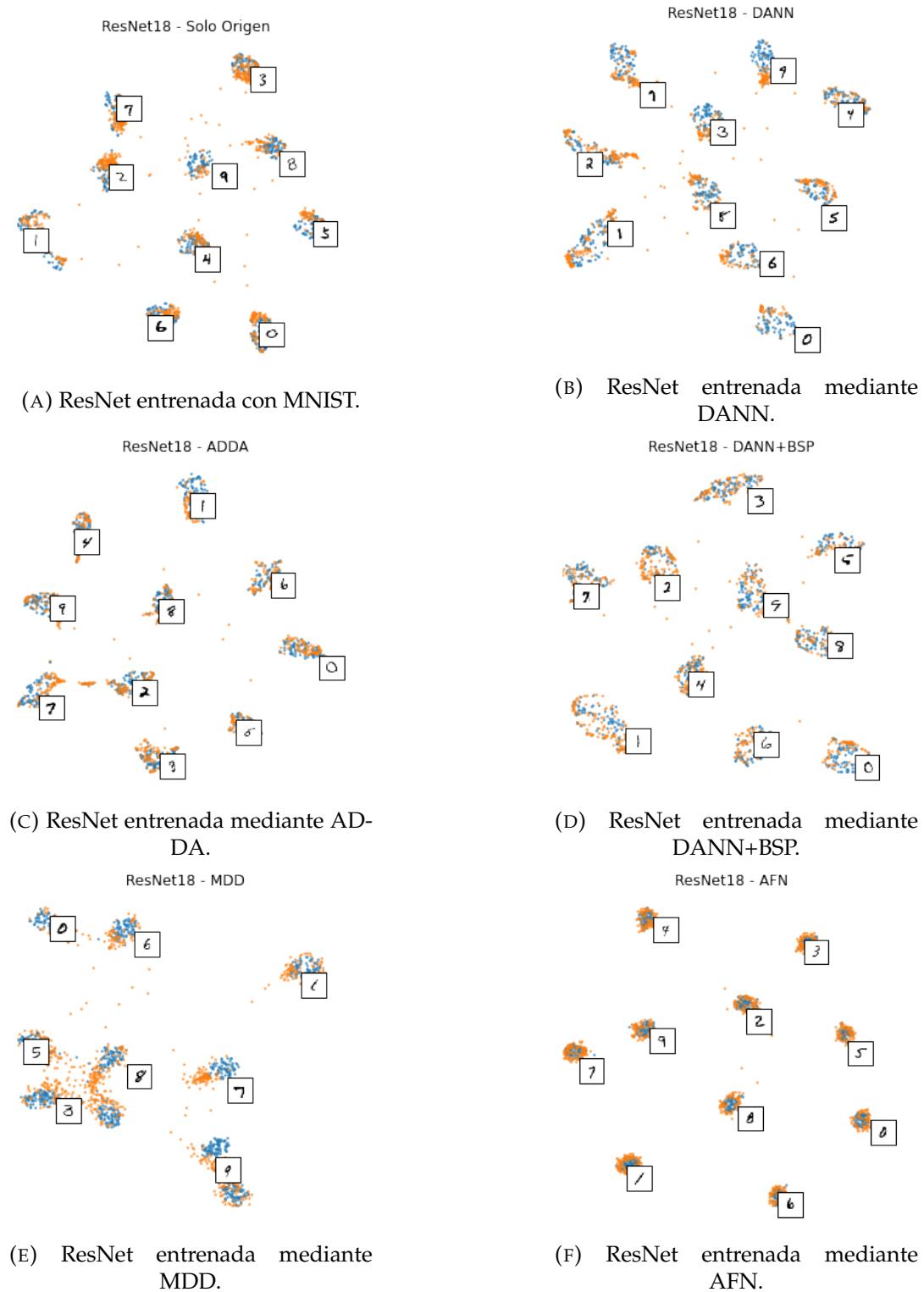


FIGURA 4.2: Representaciones UMAP de los espacios latentes de los modelos ResNet18. Los puntos azules representan observaciones de TDS y los naranjas de MNIST.

Al analizar las representaciones UMAP de los espacios latentes obtenidos por los modelos ResNet, se pueden observar los siguientes resultados:

- En contraste con los modelos LeNet, la ResNet es más eficaz en la extracción de características *out of the box*, lo que le permite generalizar mejor sin necesidad de técnicas de adaptación de dominio (figura 4.2a).
- La aplicación de técnicas de adaptación de dominio como DANN, DANN con penalización BSP y ADDA mejora la capacidad de generalización, pero no de forma significativa en comparación con las mejoras obtenidas con LeNet (figuras 4.2b, 4.2d y 4.2c).
- El uso de MDD parece disminuir la capacidad de generalización (figura 4.2e).
- La aplicación de AFN potencia la generalización de la ResNet, obteniendo representaciones de los espacios latentes prácticamente idénticas para ambos conjuntos de datos (figura 4.2f).

En general, estas representaciones permiten visualizar los resultados obtenidos a partir de las métricas de adaptación descritas en el cuadro 4.1 de la sección anterior.

### 4.3. Análisis de errores

Los errores de predicción de los modelos pueden ser analizados mediante los histogramas de la métrica *IoU* que se obtienen de aplicar los modelos a los telegramas.

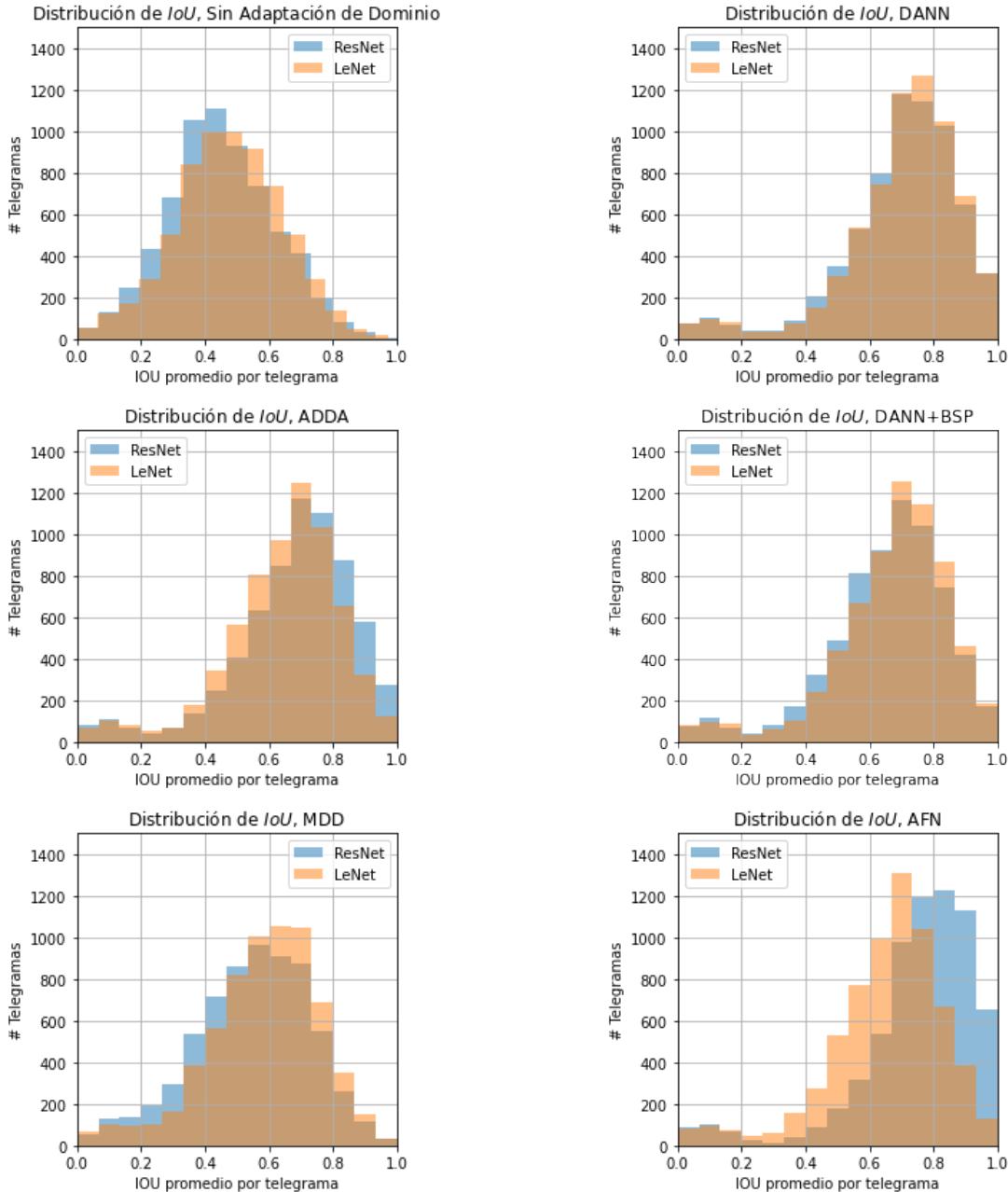


FIGURA 4.3: Histogramas de la métrica  $IoU$  promedio por telegrama por cada técnica AD y modelo.

Resulta interesante mencionar existen telegramas que son más difíciles de analizar que otros. Esto puede evidenciarse en los histogramas de la figura 4.3 donde se pueden observar un conjunto de observaciones que contienen valores entre [0, 0.2] en todos los experimentos realizados. Luego de analizar cada uno de estos casos, se detectaron las siguientes situaciones:

- Telegramas cargados de forma errónea: ver ejemplo en el anexo A.2.
- Telegramas correctos pero por alguna cuestión la lógica de extracción de dígitos no funciona correctamente: ver ejemplo en el anexo A.3.

- Telegramas donde existen otros caracteres distintos a números: al ser un cuadro de texto libre sin formato, los jefes de mesa pueden escribir lo que deseen. Ver ejemplo en el anexo A.4 donde se representa el 0 a la izquierda con X.
- Telegramas de mesas donde la mayor cantidad de votos se las lleva un único partido y completan los votos a la izquierda con 0: al agregar los ceros a la izquierda, aumenta la probabilidad de que el modelo se equivoque con esos ceros que no aportan al número final. Ver ejemplo en el anexo A.5.

En los primeros dos puntos se describen problemas que fueron detectados en el proceso de ETL del capítulo 3. Estandarizar los telegramas agregando un casillero por cada dígito junto a mejorar el proceso de extracción de los mismos, supondrá una mejora considerable en las capacidades predictivas de los modelos.

El tercer punto presenta un problema dentro de la adaptación de dominio. La misma supone que, si bien los datasets de origen y destino son diferentes pero representan lo mismo, debe existir la misma cantidad de clases entre origen y destino. Al agregar uno o varios caracteres adicionales en TDS (como es el ejemplo donde representaban el 0 con una X), se está incumpliendo este supuesto.

El cuarto punto aumenta la probabilidad de error en los modelos debido a que el 0 a la izquierda no aporta significado alguno al número de la cantidad de votos que se desea predecir.

Estandarizar la carga de los telegramas por parte de los jefes de mesa mediante alguna capacitación permitiría reducir los errores de los puntos tres y cuatro en elecciones futuras.



# Capítulo 5

## Conclusiones

La presente tesis se centró en la utilización de técnicas de adaptación de dominio para mejorar la digitalización de redes neuronales entrenados con distintos algoritmos de adaptación de dominio. En primera instancia, se desarrolló un proceso de Extracción, Transformación y Carga (ETL) de los telegramas que permitió segmentar cada uno de los dígitos y crear un conjunto de datos adecuado para la tarea de digitalización. Este proceso de ETL estandarizó los datos para que pudieran ser procesados y analizados de manera más efectiva.

En una siguiente etapa, tras realizar múltiples experimentos, se pudo determinar que la red ResNet18 entrenada mediante AFN es el modelo que tiene el mejor rendimiento en la tarea de digitalización de telegramas. Este resultado indica que el uso de modelos más complejos y profundos puede ser beneficioso en este tipo de tarea. Sin embargo, también se pudo comprobar que una red más simple, como la LeNet5 con AFN, tiene un rendimiento satisfactorio, lo que sugiere que no siempre es necesario utilizar modelos complejos para resolver tareas como ésta.

En cuanto a la precisión obtenida al aplicar el mejor modelo a los telegramas de prueba, se alcanzó una precisión promedio del 73 %, sin haber estandarizado la casilla donde se escriben los dígitos. No obstante, se podría mejorar aún más el resultado si se dispusiera de un casillero por número a escribir, lo que permitiría evitar agregar ceros a la izquierda y, en consecuencia, aumentar la calidad del proceso de ETL de segmentación de dígitos. En este sentido, se espera que la implementación de esta mejora contribuya a mejorar la precisión de las predicciones.

Es importante destacar que la utilización de técnicas de adaptación de dominio en la tarea de digitalización de telegramas de elecciones es un campo de investigación en desarrollo, y aún existen diversas limitaciones que deben ser abordadas. No obstante, los resultados obtenidos en esta tesis son prometedores y sugieren que la adaptación de dominio puede ser una técnica útil para mejorar la digitalización de telegramas de elecciones en contextos de alta variabilidad.

Cabe señalar que los resultados obtenidos en esta tesis pueden tener implicaciones importantes en la automatización de procesos electorales, lo que puede mejorar la eficiencia y la transparencia de las elecciones. Con la digitalización de telegramas de manera efectiva, es posible mejorar la velocidad y la precisión del conteo de votos, lo que puede reducir la posibilidad de errores humanos y aumentar la confianza en los procesos electorales.

## 5.1. Trabajos Futuros

Existen diversas posibilidades de mejora y trabajos futuros que pueden llevarse a cabo para ampliar el alcance y la calidad de los resultados. A continuación, se detallan algunas posibles líneas de investigación futura:

- Mejora del proceso ETL: Una de las limitaciones de la presente investigación es la calidad del conjunto de datos TDS, ya que presenta ciertas limitaciones en términos de segmentación de los dígitos en los telegramas. Por lo tanto, una posible línea de investigación futura es la mejora del proceso ETL para mejorar la segmentación de los dígitos y mejorar la calidad del conjunto de datos.
- Mejorar el proceso de post-procesamiento de las predicciones: En la implementación actual, no se realizó ningún tipo de post-procesamiento en las predicciones obtenidas por los modelos de aprendizaje automático. Sería útil explorar distintas opciones de post-procesamiento, como la validación de que la cantidad total de votos predichos por telegrama no supere el total que debe haber por mesa, o la validación de que la cantidad de votos por partido no supere el total de la mesa. También se podría considerar la utilización de otras técnicas de post-procesamiento, como la corrección de errores en las predicciones mediante la incorporación de información contextual adicional.
- Mejora de la estandarización de los telegramas: Como se mencionó en las conclusiones, la estandarización en la sección donde se escriben los dígitos podría mejorar aún más la precisión de los modelos de ya que mejoraría la calidad de TDS.
- Evaluación de otras técnicas de adaptación de dominio: En esta tesis se evaluaron algunas técnicas de adaptación de dominio, pero existen muchas otras técnicas que podrían ser útiles para mejorar la precisión de la digitalización de telegramas. Por lo tanto, una posible línea de investigación futura es la evaluación de otras técnicas de adaptación de dominio para comparar su efectividad con las técnicas utilizadas en esta investigación.
- Evaluación de modelos híbridos: Otra posible línea de investigación futura es la evaluación de modelos híbridos que combinen técnicas de adaptación de dominio con otros métodos de mejora de la precisión en la tarea de digitalización de telegramas. Por ejemplo, se podría explorar la combinación de redes neuronales con técnicas de procesamiento de imágenes para mejorar la calidad de los datos.

En conclusión, existen varias líneas de investigación desprendidas de la tesis actual que pueden ser exploradas para mejorar la precisión de los modelos que deben aplicarse en un conjunto de datos con características distintas que del que se entrenó. Estas mejoras pueden ser importantes para aumentar la confianza en los procesos electorales y mejorar la transparencia y la eficiencia en el conteo de votos.



# Apéndice A

## Anexo: Telegramas

## A.1. Ejemplo de telegrama

Correo Argentino

210010001X0101

DESTINATARIO: CON COPIA:	JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE DIRECCION NACIONAL ELECTORAL		HOJA 1/1
DISTRITO <b>SANTA FE</b>	SECCION / CIRCUITO <b>1 - BELGRANO 402 - ARMSTRONG</b>	MESA N°: <b>1</b>	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 <b>TELEGRAMA DEL PRESIDENTE DE MESA</b>
CANTIDAD DE SOBRES UTILIZADOS EXTRAIDOS DE LA URNA		Z   8   3	
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON		Z   8   3	
DIFERENCIA ENTRE LAS ANTERIORES		O   0   0	
			CANTIDAD DE VOTOS OBTENIDOS SENADORES NACIONALES   DIPUTADOS NACIONALES
Nº	Partido Político /Alianza		
87	<b>UNITE POR LA LIBERTAD Y LA DIGNIDAD</b>		4   5
501	<b>FRENTE AMPLIO PROGRESISTA</b>		15   16
502	<b>FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD</b>		8   8
503	<b>JUNTOS POR EL CAMBIO</b>		135   135
504	<b>PRIMERO SANTA FE</b>		2   2
505	<b>SOMOS FUTURO</b>		4   4
506	<b>PODEMOS</b>		1   1
507	<b>SOBERANIA POPULAR</b>		6   6
508	<b>FRENTE DE TODOS</b>		94   92
VOTOS EN BLANCO (desglosados por categoria)			5   5
VOTOS NULOS (desglosados por categoria)			6   6
VOTOS RECURRIDOS (desglosados por categoria)			—   —
VOTOS DE IDENTIDAD IMPUGNADA (en todos los columnas debe asentar la misma cantidad)			—   —
TOTAL DE VOTOS POR CATEGORIA			283   283

S.R. PRESIDENTE DE MESA: LA SUMA DE LOS TOTALES POR COLUMNA (CATEGORÍA) DEBE COINCIDIR CON LA CANTIDAD DE SOBRES UTILIZADOS QUE SE EXTRAERON DE LA URNA.

**NO INTRODUCIR EN LA URNA - ENTREGAR AL AGENTE DE CORREOS**

## A.2. Ejemplo de telegrama mal cargado

### A.3. Ejemplo de telegrama con números sin espacio entre ellos



2106200153X0101

DESTINATARIO: CON COPIA:	JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE DIRECCION NACIONAL ELECTORAL		HOJA 1/1																																																															
DISTRITO <b>SANTA FE</b>	SECCION / CIRCUITO <b>2 - CASEROS</b> 367 - BIGAND	MESA N°: <b>153</b>	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 TELEGRAMA DEL PRESIDENTE DE MESA																																																															
CANTIDAD DE SOBRES UTILIZADOS EXTRAIDOS DE LA URNA	<b>262</b>																																																																	
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON	<b>262</b>																																																																	
DIFFERENCIA ENTRE LAS ANTERIORES	<b>000</b>																																																																	
<table border="1"> <thead> <tr> <th>Nº</th> <th>Partido Político /Alianza</th> <th colspan="2">CANTIDAD DE VOTOS OBTENIDOS</th> </tr> <tr> <th></th> <th></th> <th>SENADORES NACIONALES</th> <th>DIPUTADOS NACIONALES</th> </tr> </thead> <tbody> <tr> <td><b>87</b></td> <td><b>UNITE POR LA LIBERTAD Y LA DIGNIDAD</b></td> <td><b>006</b></td> <td><b>005</b></td> </tr> <tr> <td><b>501</b></td> <td><b>FRENTE AMPLIO PROGRESISTA</b></td> <td><b>027</b></td> <td><b>027</b></td> </tr> <tr> <td><b>502</b></td> <td><b>FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD</b></td> <td><b>003</b></td> <td><b>003</b></td> </tr> <tr> <td><b>503</b></td> <td><b>JUNTOS POR EL CAMBIO</b></td> <td><b>100</b></td> <td><b>101.</b></td> </tr> <tr> <td><b>504</b></td> <td><b>PRIMERO SANTA FE</b></td> <td><b>008</b></td> <td><b>008</b></td> </tr> <tr> <td><b>505</b></td> <td><b>SOMOS FUTURO</b></td> <td><b>005</b></td> <td><b>005</b></td> </tr> <tr> <td><b>506</b></td> <td><b>PODEMOS</b></td> <td><b>001</b></td> <td><b>001</b></td> </tr> <tr> <td><b>507</b></td> <td><b>SOBERANIA POPULAR</b></td> <td><b>011</b></td> <td><b>011</b></td> </tr> <tr> <td><b>508</b></td> <td><b>FRENTE DE TODOS</b></td> <td><b>087</b></td> <td><b>086</b></td> </tr> <tr> <td colspan="2">VOTOS EN BLANCO (discriminados por categoría)</td> <td><b>009</b></td> <td><b>009</b></td> </tr> <tr> <td colspan="2">VOTOS NULOS (discriminados por categoría)</td> <td><b>011</b></td> <td><b>012</b></td> </tr> <tr> <td colspan="2">VOTOS RECURRIDOS (discriminados por categoría)</td> <td><b>000</b></td> <td><b>000</b></td> </tr> <tr> <td colspan="2">VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe aparecer la misma cantidad)</td> <td><b>000</b></td> <td><b>000</b></td> </tr> <tr> <td colspan="2">TOTAL DE VOTOS POR CATEGORIA</td> <td><b>262</b></td> <td><b>262</b></td> </tr> </tbody> </table>		Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS				SENADORES NACIONALES	DIPUTADOS NACIONALES	<b>87</b>	<b>UNITE POR LA LIBERTAD Y LA DIGNIDAD</b>	<b>006</b>	<b>005</b>	<b>501</b>	<b>FRENTE AMPLIO PROGRESISTA</b>	<b>027</b>	<b>027</b>	<b>502</b>	<b>FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD</b>	<b>003</b>	<b>003</b>	<b>503</b>	<b>JUNTOS POR EL CAMBIO</b>	<b>100</b>	<b>101.</b>	<b>504</b>	<b>PRIMERO SANTA FE</b>	<b>008</b>	<b>008</b>	<b>505</b>	<b>SOMOS FUTURO</b>	<b>005</b>	<b>005</b>	<b>506</b>	<b>PODEMOS</b>	<b>001</b>	<b>001</b>	<b>507</b>	<b>SOBERANIA POPULAR</b>	<b>011</b>	<b>011</b>	<b>508</b>	<b>FRENTE DE TODOS</b>	<b>087</b>	<b>086</b>	VOTOS EN BLANCO (discriminados por categoría)		<b>009</b>	<b>009</b>	VOTOS NULOS (discriminados por categoría)		<b>011</b>	<b>012</b>	VOTOS RECURRIDOS (discriminados por categoría)		<b>000</b>	<b>000</b>	VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe aparecer la misma cantidad)		<b>000</b>	<b>000</b>	TOTAL DE VOTOS POR CATEGORIA		<b>262</b>	<b>262</b>	
Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS																																																																
		SENADORES NACIONALES	DIPUTADOS NACIONALES																																																															
<b>87</b>	<b>UNITE POR LA LIBERTAD Y LA DIGNIDAD</b>	<b>006</b>	<b>005</b>																																																															
<b>501</b>	<b>FRENTE AMPLIO PROGRESISTA</b>	<b>027</b>	<b>027</b>																																																															
<b>502</b>	<b>FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD</b>	<b>003</b>	<b>003</b>																																																															
<b>503</b>	<b>JUNTOS POR EL CAMBIO</b>	<b>100</b>	<b>101.</b>																																																															
<b>504</b>	<b>PRIMERO SANTA FE</b>	<b>008</b>	<b>008</b>																																																															
<b>505</b>	<b>SOMOS FUTURO</b>	<b>005</b>	<b>005</b>																																																															
<b>506</b>	<b>PODEMOS</b>	<b>001</b>	<b>001</b>																																																															
<b>507</b>	<b>SOBERANIA POPULAR</b>	<b>011</b>	<b>011</b>																																																															
<b>508</b>	<b>FRENTE DE TODOS</b>	<b>087</b>	<b>086</b>																																																															
VOTOS EN BLANCO (discriminados por categoría)		<b>009</b>	<b>009</b>																																																															
VOTOS NULOS (discriminados por categoría)		<b>011</b>	<b>012</b>																																																															
VOTOS RECURRIDOS (discriminados por categoría)		<b>000</b>	<b>000</b>																																																															
VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe aparecer la misma cantidad)		<b>000</b>	<b>000</b>																																																															
TOTAL DE VOTOS POR CATEGORIA		<b>262</b>	<b>262</b>																																																															

SR. PRESIDENTE DE MESA: LA SUMA DE LOS totales POR COLUMNA [CATEGORIA] DEBE COINCIDIR CON LA CANTIDAD DE SOBRES UTILIZADOS QUE SE EXTRAJERON DE LA URNA.

## INFORMACION INDISPENSABLE PARA EL CORPODE DE LA COMPENSACION-ART.72 DEL CODIGO ELECTORAL NACIONAL

(Por favor: COMPLETAR CON LETRA IMPRESA)

PRESIDENTE DE MESA		1 <sup>er</sup> VOCAL	
LIUTI YANINA		FRAGO LETTI YANNA PAOLA	
Apellido y Nombres	Firma	Apellido y Nombres	Firma
2 <sup>do</sup> VOCAL		FISCALES PARTIDARIOS PRESENTES	
GONZALEZ MARINA		JUAN M. LUSAADI 11326427 JUNTOS POR EL CAMBIO	
Apellido y Nombres	Firma	Apellido y Nombres	Firma
Mariangela		Silvietta Frente de todos.	
NO INTRODUCIR EN LA URNA - ENTREGAR AL AGENTE DE CORREOS			

## A.4. Ejemplo de telegrama con caracteres distintos a números



2108903008X0101

DESTINATARIO: CON COPIA:	JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE DIRECCION NACIONAL ELECTORAL		HOJA 1/1
DISTRITO <b>SANTA FE</b>	SECCION / CIRCUITO <b>9 - LA CAPITAL 20A - SANTO TOME</b>	MESA N°: <b>3008</b>	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 <b>TELEGRAMA DEL PRESIDENTE DE MESA</b>
CANTIDAD DE SOBRES UTILIZADOS EXTRADIDOS DE LA URNA		<b>2   1   6</b>	
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON		<b>2   1   6</b>	
DIFERENCIA ENTRE LAS ANTERIORES		<b>1   X   X</b>	
Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS	
		SENADORES NACIONALES	DIPUTADOS NACIONALES
		X   2   2	X   2   2
		X   3   3	X   2   9
		XX   5	XX   5
		X   7   9	X   8   1
		X   X   X	X   X   X
		X   X   6	X   X   6
		XX   1	XX   1
		X   5   1	X   5   0
VOTOS EN BLANCO (Descontados por categoria)			
X   X   6			
VOTOS NULOS (Descontados por categoria)			
X   X   8			
VOTOS RECURRIDOS (Descontados por categoria)			
X   X   7			
VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe asentar la misma cantidad)			
X   X   X			
TOTAL DE VOTOS POR CATEGORIA		<b>2   0   2</b>	<b>2   0   1</b>

SR. PRESIDENTE DE MESA: LA SUMA DE LOS TOTALES POR COLUMNA (CATEGORIA) DEBE COINCIDIR CON LA CANTIDAD DE SOBRES UTILIZADOS QUE SE EXTRAJERON DE LA URNA.

INFORMACION INDISPENSABLE PARA EL COBRO DE LA COMPENSACION-ART. 72 DEL CODIGO ELECTORAL NACIONAL  
(Por favor: COMPLETAR CON LETRA IMPRENTA)

PRESIDENTE DE MESA		1º - VOCAL	
ARMANDO LUCILA BELGEM		ANCIO LINDNER ISOL	
Apellido y Nombres		Apellido y Nombres	
Firma	3154681160	Firma	4112561217
Nº Documento		Nº Documento	
2º VOCAL		FISCALES PARTIDARIOS PRESENTES	
ARMANDO LUCILA BELGEM			
Apellido y Nombres			
Firma	3810141438	Nº Documento	
NO INTRODUCIR EN LA URNA - ENTREGAR AL AGENTE DE CORREOS			

## A.5. Ejemplo de telegrama ceros a la izquierda de la cantidad de votos



DESTINATARIO: CON COPIA:	JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE DIRECCION NACIONAL ELECTORAL		HOJA 1/1
DISTRITO <b>SANTA FE</b>	SECCION / CIRCUITO <b>13 - ROSARIO 319 - ROSARIO SEC. 2A</b>	MESA N°: <b>4086</b>	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 <b>TELEGRAMA DEL PRESIDENTE DE MESA</b>
CANTIDAD DE SOBRES UTILIZADOS EXTRAIDOS DE LA URNA	<b>2 618</b>		
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON	<b>2 618</b>		
DIFERENCIA ENTRE LAS ANTERIORES	<b>0 0 0 0</b>		
Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS	
		SENADORES NACIONALES	DIPUTADOS NACIONALES
<b>87</b>	<b>UNITE POR LA LIBERTAD Y LA DIGNIDAD</b>	<b>0 0 5</b>	<b>0 0 7</b>
<b>501</b>	<b>FRENTE AMPLIO PROGRESISTA</b>	<b>0 5 3</b>	<b>0 4 6</b>
<b>502</b>	<b>FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD</b>	<b>0 0 2</b>	<b>0 0 2</b>
<b>503</b>	<b>JUNTOS POR EL CAMBIO</b>	<b>1 4 4</b>	<b>1 4 6</b>
<b>504</b>	<b>PRIMERO SANTA FE</b>	<b>0 0 6</b>	<b>0 0 3</b>
<b>505</b>	<b>SOMOS FUTURO</b>	<b>0 0 0</b>	<b>0 0 0</b>
<b>506</b>	<b>PODEMOS</b>	<b>0 0 1</b>	<b>0 0 2</b>
<b>507</b>	<b>SOBERANIA POPULAR</b>	<b>0 0 7</b>	<b>0 1 9</b>
<b>508</b>	<b>FRENTE DE TODOS</b>	<b>0 3 7</b>	<b>0 3 0</b>
VOTOS EN BLANCO (describidos por categoria)		<b>0 0 7</b>	<b>0 0 7</b>
VOTOS NULOS (describidos por categoria)		<b>0 0 6</b>	<b>0 0 6</b>
VOTOS RECURRIDOS (describidos por categoria)		<b>0 0 0</b>	<b>0 0 0</b>
VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe asentar la misma cantidad)		<b>0 0 0</b>	<b>0 0 0</b>
TOTAL DE VOTOS POR CATEGORIA		<b>2 6 8</b>	<b>2 6 8</b>

SR. PRESIDENTE DE MESA: LA SUMA DE LOS TOTALES POR COLUMNA (CATEGORIA) DEBE COINCIDIR CON LA CANTIDAD DE SOBRES UTILIZADOS QUE SE EXTRAJERON DE LA URNA.

INFORMACION INDISPENSABLE PARA EL COBRO DE LA COMPENSACION-ART. 72 DEL CODIGO ELECTORAL NACIONAL (Por favor: COMPLETAR CON LETRA IMPRESA)			
PRESIDENTE DE MESA <b>PALAD AGUSTINA</b>		1º - VOCAL	
Apellido y Nombres 		Apellido y Nombres 	
Firma	37295734	Firma	Nº Documento
2º VOCAL		ESCALES PARTIDARIOS PRESENTES	
Apellido y Nombres 		Apellido y Nombres 	
Firma	Nº Documento	35570523	
NO INTRODUCIR EN LA URNA - ENTREGAR AL AGENTE DE CORREOS			

# Bibliografía

- Akiba, Takuya et al. (2019). «Optuna: A Next-generation Hyperparameter Optimization Framework». En: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Ben-David, Shai et al. (2006). «Analysis of Representations for Domain Adaptation». En: *Advances in Neural Information Processing Systems*. Ed. por B. Schölkopf, J. Platt y T. Hoffman. Vol. 19. MIT Press. URL: <https://proceedings.neurips.cc/paper/2006/file/b1b0432ceafb0ce714426e9114852ac7-Paper.pdf>.
- Bradski, G. (2000). «The OpenCV Library». En: *Dr. Dobb's Journal of Software Tools*.
- Bugnon, Leandro A et al. (2020). «DL4papers: a deep learning approach for the automatic interpretation of scientific articles». En: *Bioinformatics* 36.11, págs. 3499-3506.
- Chen, Hanting et al. (2021). «Pre-Trained Image Processing Transformer». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, págs. 12299-12310.
- Chen, Xinyang et al. (2019). «Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation». En: *International conference on machine learning*. PMLR, págs. 1081-1090.
- DeCoste, Dennis y Bernhard Schölkopf (2002). «Training invariant support vector machines». En: *Machine learning* 46.1, págs. 161-190.
- Erhan, Dumitru et al. (2010). «Why does unsupervised pre-training help deep learning?». En: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop y Conference Proceedings, págs. 201-208.
- Falcon, William y The PyTorch Lightning team (mar. de 2019). *PyTorch Lightning*. Ver. 1.4. DOI: [10.5281/zenodo.3828935](https://doi.org/10.5281/zenodo.3828935). URL: <https://github.com/Lightning-AI/lightning>.
- Fukushima, Kunihiko (1980). «Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position». En: *Biological cybernetics* 36.4, págs. 193-202.
- Funahashi, Ken-Ichi (1989). «On the approximate realization of continuous mappings by neural networks». En: *Neural networks* 2.3, págs. 183-192.
- Ganin, Yaroslav et al. (2016). «Domain-adversarial training of neural networks». En: *The journal of machine learning research* 17.1, págs. 2096-2030.
- Girshick, Ross et al. (2014). «Rich feature hierarchies for accurate object detection and semantic segmentation». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 580-587.

- Glorot, Xavier, Antoine Bordes y Yoshua Bengio (2011). «Domain adaptation for large-scale sentiment classification: A deep learning approach». En: *ICML*.
- Gong, Boqing, Kristen Grauman y Fei Sha (2013). «Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation». En: *International conference on machine learning*. PMLR, págs. 222-230.
- Goodfellow, Ian et al. (2020). «Generative adversarial networks». En: *Communications of the ACM* 63.11, págs. 139-144.
- Gretton, Arthur et al. (2012). «A kernel two-sample test». En: *The Journal of Machine Learning Research* 13.1, págs. 723-773.
- He, Kaiming et al. (2016). «Deep residual learning for image recognition». En: *Proceedings of the IEEE conference on computer vision and pattern recognition*, págs. 770-778.
- Hebb, DO (1949). «The organization of behavior. A neuropsychological theory». En.
- Howard, Jeremy y Sebastian Ruder (2018). «Universal language model fine-tuning for text classification». En: *arXiv preprint arXiv:1801.06146*.
- Hubel, David H y Torsten N Wiesel (1959). «Receptive fields of single neurons in the cat's striate cortex». En: *The Journal of physiology* 148.3, pág. 574.
- Jiang, Junguang et al. (2022). *Transferability in Deep Learning: A Survey*. DOI: 10 . 48550 / ARXIV . 2201 . 05867. URL: <https://arxiv.org/abs/2201.05867>.
- Junguang Jiang, Baixu, Bo Chen y Mingsheng Long Fu (2020). *Transfer Learning Library*. <https://github.com/thuml/Transfer-Learning-Library>.
- Krizhevsky, Alex, Ilya Sutskever y Geoffrey E Hinton (2017). «Imagenet classification with deep convolutional neural networks». En: *Communications of the ACM* 60.6, págs. 84-90.
- Lamagna, Walter Marcelo (2016). «Lectura artificial de números manuscritos en datos abiertos de elecciones legislativas en la Ciudad de Buenos Aires». Tesis doct. Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales.
- Lauer, Fabien, Ching Y Suen y Gérard Bloch (2007). «A trainable feature extractor for handwritten digit recognition». En: *Pattern Recognition* 40.6, págs. 1816-1824.
- LeCun, Yann (1985). «A Learning Procedure for Assymmetric Threshold Network». En: *Proceedings of Cognitiva* 85, págs. 599-604.
- LeCun, Yann, Yoshua Bengio y Geoffrey Hinton (2015). «Deep learning». En: *nature* 521.7553, págs. 436-444.
- LeCun, Yann et al. (1989). «Backpropagation applied to handwritten zip code recognition». En: *Neural computation* 1.4, págs. 541-551.
- LeCun, Yann et al. (1995). «Learning algorithms for classification: A comparison on handwritten digit recognition». En: *Neural networks: the statistical mechanics perspective* 261.276, pág. 2.
- LeCun, Yann et al. (1998). «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11, págs. 2278-2324.

- Liu, Yang y Mirella Lapata (2019). «Text summarization with pretrained encoders». En: *arXiv preprint arXiv:1908.08345*.
- McCulloch, Warren S y Walter Pitts (1943). «A logical calculus of the ideas immanent in nervous activity». En: *The bulletin of mathematical biophysics* 5.4, págs. 115-133.
- McInnes, Leland, John Healy y James Melville (2018). «UMAP: Uniform manifold approximation and projection for dimension reduction». En: *arXiv preprint arXiv:1802.03426*.
- Minsky, Marvin y Seymour Papert (1969). «An introduction to computational geometry». En: *Cambridge tiass., HIT* 479, pág. 480.
- Paszke, Adam et al. (2019). «PyTorch: An Imperative Style, High-Performance Deep Learning Library». En: *Advances in Neural Information Processing Systems* 32. Ed. por H. Wallach et al. Curran Associates, Inc., págs. 8024-8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- Quinonero-Candela, Joaquin et al. (2008). *Dataset shift in machine learning*. Mit Press.
- Rosenblatt, Frank (1958). «The perceptron: a probabilistic model for information storage and organization in the brain.» En: *Psychological review* 65.6, pág. 386.
- Rumelhart, David E, Geoffrey E Hinton y Ronald J Williams (1986). «Learning representations by back-propagating errors». En: *nature* 323.6088, págs. 533-536.
- Sandler, Mark et al. (2018). «MobileNetV2: Inverted Residuals and Linear Bottlenecks». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Simonyan, Karen y Andrew Zisserman (2014). «Two-Stream Convolutional Networks for Action Recognition in Videos». En: *Advances in Neural Information Processing Systems*. Ed. por Z. Ghahramani et al. Vol. 27. Curran Associates, Inc. URL: <https://proceedings.neurips.cc/paper/2014/file/00ec53c4682d36f5c4359f4ae7bd7ba1-Paper.pdf>.
- Sugiyama, Masashi et al. (2007). «Direct importance estimation with model selection and its application to covariate shift adaptation». En: *Advances in neural information processing systems* 20.
- Sutskever, Ilya et al. (2013). «On the importance of initialization and momentum in deep learning». En: *Proceedings of the 30th International Conference on Machine Learning*. Ed. por Sanjoy Dasgupta y David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, págs. 1139-1147. URL: <https://proceedings.mlr.press/v28/sutskever13.html>.
- Thrun, Sebastian y Lorien Pratt (1998). *Learning to learn*.
- Tzeng, Eric et al. (2017). «Adversarial Discriminative Domain Adaptation». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Viehmann, Thomas, Luca Antiga y Alessia Marcolini (2021). *TorchDrift*. <https://github.com/torchdrift/torchdrift/>.

- Wang, Dayong et al. (2016). «Deep learning for identifying metastatic breast cancer». En: *arXiv preprint arXiv:1606.05718*.
- Werbos, Paul (1974). «Beyond regression: new tools for prediction and analysis in the behavioral sciences». En: *Ph. D. dissertation, Harvard University*.
- Xie, Saining et al. (2017). «Aggregated Residual Transformations for Deep Neural Networks». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Xu, Ruijia et al. (2019). «Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation». En: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, págs. 1426-1435.
- Yosinski, Jason et al. (2014). «How transferable are features in deep neural networks?» En: *Advances in neural information processing systems* 27.
- Zhang, Yuchen et al. (2019). «Bridging theory and algorithm for domain adaptation». En: *International Conference on Machine Learning*. PMLR, págs. 7404-7413.