



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

21 de marzo 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

Resumen	III
Reconocimientos	v
1. Introducción	1
1.1. Contexto	1
1.2. Motivación	2
1.3. Objetivos	3
1.4. Estructura del trabajo	4
2. Marco teórico	5
2.1. Reconocimiento de dígitos	5
2.2. Redes Neuronales	5
2.2.1. Redes Densas	6
2.2.2. Redes Convolucionales	7
2.2.3. Arquitecturas	9
LeNet	10
ResNet	11
2.3. Aprendizaje por transferencia	12
2.4. Adaptación de Dominio	13
2.4.1. Domain Adversarial Neural Network	13
2.4.2. Adversarial Discriminative Domain Adaptation	14
2.4.3. Batch Spectral Penalization	15
2.4.4. Margin Disparity Discrepancy	16
2.4.5. Adaptive Feature Norm	17
3. Metodología	21
3.1. Descripción de los datos	21
3.2. Extracción de dígitos de los telegramas	21
3.2.1. Enderezado	21
3.2.2. Extracción de la grilla de votos	22
3.2.3. Detección de líneas de la grilla de votos	22
3.2.4. Segmentación de dígitos	24
3.3. Análisis del dataset	24
3.4. Diseño experimental	26
3.4.1. Metodología de entrenamiento	26
3.4.2. Selección y optimización del modelo	27
3.5. Métricas de evaluación	27
3.5.1. Métricas de clasificación	28
Accuracy	28

F_1	28
Intersección sobre unión	29
3.5.2. Métricas de adaptación	29
Distancia \mathcal{A}	29
Discrepancia de Medias Máxima	30
4. Análisis de resultados	33
4.1. Análisis de métricas	33
4.2. Análisis de los espacios latentes	35
4.2.1. LeNet	36
4.2.2. ResNet	38
4.3. Análisis de errores	39
5. Conclusiones	41
5.1. Conclusiones	41
5.2. Trabajos Futuros	41
A. Anexo: Telegramas	43
A.1. Ejemplo de telegrama	44
A.2. Ejemplo de telegrama mal cargado	45
A.3. Ejemplo de telegrama con números sin espacio entre ellos	46
A.4. Ejemplo de telegrama con caracteres distintos a números	47
A.5. Ejemplo de telegrama ceros a la izquierda de la cantidad de votos	48
Bibliografía	49

Índice de figuras

1.1. Proceso eleccionario. TODO: cambiar para que se pueda leer o eliminar directament?	2
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.	7
2.2. Ejemplo de una convolución para una matriz I y un filtro o kernel K , produciendo la matriz $I * K$.	8
2.3. Maxpool 2×2 aplicado al ejemplo de la figura 2.2 $I * K$.	9
2.4. Arquitectura de una red 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} .	10
2.5. Arquitectura de una LeNet-5. Imagen tomada de LeCun et al., 1998.	11
2.6. Esquema de las <i>DANN</i> . 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.	13
2.7. Esquema de las <i>ADDA</i> . 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} .	15
2.8. Esquema de un modelo <i>DANN+BSP</i> .	16
2.9. Esquema de <i>MDD</i> . Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. <i>MDD</i> introduce un 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.	17
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.	18

2.11. Esquema de <i>AFN</i> . 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 <i>backbone</i> \mathcal{G} produce características \mathbf{x} que luego son aplicadas por bloques \mathcal{N} (FC-BN-ReLU-Dropout) produciendo características normalizadas \mathbf{d} . La red clasificadora \mathcal{C} toma las características \mathbf{d}^s y se computa $\mathcal{L}_{\mathcal{C}}$. \mathcal{L}_{AFN} se calcula utilizando \mathbf{d} .	19
3.1. Enderezamiento de un telegrama utilizando OpenCV.	22
3.2. Grilla de votos extraída buscando el contorno de mayor área.	22
3.3. Proyecciones de los ejes de la grilla. En rojo se marca el umbral de corte.	23
3.4. Grilla detectada (en verde) utilizando el umbral por sobre las proyecciones y su posterior agrupamiento con clustering jerárquico.	23
3.5. Primer registro extraído de la grilla de votos.	24
3.6. Dígitos detectados en el primer bloque de votos.	24
3.7. Histogramas de proporciones de píxeles blancos por voto.	25
3.8. Dataset TDS de dígitos.	26
3.9. Ejemplos de distribuciones de dominios.	30
4.1. Distribución de aciertos de cantidad de votos por cada par técnica AD y modelo.	34
4.2. Representaciones UMAP de los espacios latentes de los modelos LeNet. Los puntos naranjas representan observaciones de MNIST y los azules de TDS.	36
4.3. Representaciones UMAP de los espacios latentes de los modelos ResNet. Los puntos azules representan observaciones de TDS y los naranjas de MNIST.	38
4.4. Histogramas de la métrica <i>IoU</i> promedio por telegrama por cada técnica AD y modelo.	39

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 de registros del dataset. Cada fila representa un voto.	24
3.2. Estadísticos descriptivos del mínimo y máximo tamaño de las imágenes de los votos.	25
3.3. Rango de hiperparámetros optimizados. *LR refiere a “learning rate”. **T-O refiere a “trade-off”.	27
4.1. Métricas de los experimentos realizados. Entre paréntesis se encuentra la posición que ocupa dentro del top 3 de la columna.	33
4.2. IoU promedio y cantidad promedio de aciertos al aplicar los modelos a cada telegrama.	34

Capítulo 1

Introducción

La intro la podes pensar en 3 subsecciones: descripción de elecciones y contabilización de las actas hoy, la oportunidad que se presenta con la digitalización (y los desafíos que aparecen) y luego los objetivos. Tiene que quedar claro que problema ves y qué propones hacer a grandes rasgos. También es importante que estos objetivos se vean reflejados en los resultados y conclusiones (es decir, prometer algo que después cumplis al final del doc)

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.

Si bien emitir el sufragio es diferente en cada una de ellas, generalmente consta de ingresar a un cuarto oscuro, elegir el candidato que se desea y depositar el voto en una urna. Al finalizar la jornada, las autoridades de mesas recuentan los votos y llenan una planilla a mano alzada donde se resume la cantidad de votos obtenidos por cada candidato o partido político. Dicha planilla es escaneada y enviada a través de un telegrama del correo argentino al centro de cómputo para su procesamiento. Una vez allí, se contabilizan en un sistema informático a partir de un grupo de personas. Este proceso cuenta con una etapa de digitalización y otra de validación (TODO: esto lo vi en el diagrama de las elecciones de cordoba. no encontre nada oficial al respecto.).



FIGURA 1.1: Proceso eleccionario. TODO: cambiar para que se pueda leer o eliminar directamente?

Para que el proceso sea lo mas rápido y eficaz posible, se requiere a una gran cantidad personas destinadas al centro de cómputo. Tal solución hace que el proceso sea altamente ineficiente en cuanto a tiempos y costos se refiere. En las elecciones legislativas del 2021 se gastaron unos \$17.000 millones de pesos de los cuales \$4.000 millones de pesos fueron destinados a sueldos para el personal¹.

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 mas simple ya que sólo constará del proceso de validación.

La clasificación de números es un problema que, si bien parece resuelto con LeCun et al., 1998 y la creación del dataset MNIST, no debe ser tomada a la ligera. No existe una única forma de escribir y año a año cambian las personas que son los jefes de mesa encargados de completar los telegramas. Las características de los números escritos a mano difiere entre cada elección. Cuando la distribución de los datos de entrenamiento difiere a la de los datos de aplicación, se está ante un *corrimiento de dominio* (*domain shift* o *data drift* en inglés). Esto implica que un modelo que se entrene en algún dataset estático como el MNIST que fue creado en el 1998, hará malas clasificaciones los dígitos de las elecciones.

Cuando los dominios de entrenamiento (origen) y aplicación (destino) son distintos, se debe a que hay un *sesgo* en los datos. Las técnicas de entrenamiento clásicas suponen que, si bien existe el sesgo existe, éste será el mismo entre origen y destino. Cuando el supuesto no se cumple, las predicciones del modelo se ven afectadas negativamente. El cuadro 1.1 muestra un

¹Fuente: [El cronista](#)

ejemplo de la degradación de la precisión cuando un modelo se aplica a un conjunto de datos que es diferente al cual se entrenó.

	MNIST	Testing USPS	SVHN
Entrenamiento	  	  	  
MNIST	 	99.17 %	78.08 %
USPS	  	57.10 %	95.42 %
SVHN	  	61.92 %	64.28 %

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

Es por esto que se debe recurrir a técnicas de entrenamiento más complejas donde se intenta que el modelo aprenda a sin el sesgo de los datos. Es decir, que pueda aprender a *adaptarse* de un dominio a otro. En la actualidad no existen trabajos publicados relacionados a la digitalización de telegramas en Argentina que compare cuál es la mejor técnica de adaptación de dominio.

1.3. Objetivos

La presente tesis enfocará en el desarrollo de un modelo que permita digitalizar los telegramas de las elecciones en Argentina utilizando las legislativas de la provincia de Santa Fe del año 2021 como comparativa. Los telegramas son públicos y se encuentran subidos en la [página oficial del estado argentino](#). En el anexo A.1 se adjunta un ejemplo de uno de ellos.

Para poder llevarlo a cabo, se procede a:

- Armar el proceso de ETL de los telegramas que permita limpiar y extraer los dígitos.
- Determinar un conjunto de datos etiquetado a ser utilizado como dominio de origen.
- Entrenar distintos arquitecturas de redes convolucionales mediante técnicas de adaptación de dominio.
- Analizar y evaluar las métricas que permitan seleccionar el mejor modelo.
- Seleccionar el mejor par modelo - técnica de adaptación de dominio.
- Detectar oportunidades de mejora en el proceso eleccionario respecto a los telegramas.

1.4. Estructura del trabajo

Este trabajo se encuentra organizado con la siguiente estructura:

- 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.
- Capítulo 5: Conclusiones del trabajo, mejoras planteadas y futuras investigaciones.

Capítulo 2

Marco teórico

En este capítulo se presenta una visión general de la bibliografía relacionada al objetivo del presente trabajo. En la primera sección se detalla una revisión de trabajos relacionados. En ... TODO: Completar

2.1. Reconocimiento de dígitos

El reconocimiento de dígitos escritos a mano por parte de las computadoras es una tarea que ha fascinado a la comunidad de aprendizaje automático durante muchos años. Aunque el reconocimiento de dígitos escritos a mano es una tarea relativamente sencilla para los humanos, ha sido un desafío para las computadoras debido a la gran variabilidad en la forma en que los dígitos son escritos por diferentes personas.

A mediados 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.

TODO: agregar mas. en la tesis de lamagna hay mas referencias.

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 conexiónistas.

La contribución más importante en la reemergencia del conexiónismo 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. Las más “simples” en la práctica son las llamadas redes multicapa alimentadas hacia delante (*feed forward*) entrenadas con *backpropagation*, también referidas como perceptrones multicapa MLP.

En 1989, se demuestra teóricamente que una red de este tipo, con una capa oculta, el suficiente número de nodos y entrenada con *backpropagation*, 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_1^2 dadas las activaciones de la capa anterior y los pesos w_i^2 .

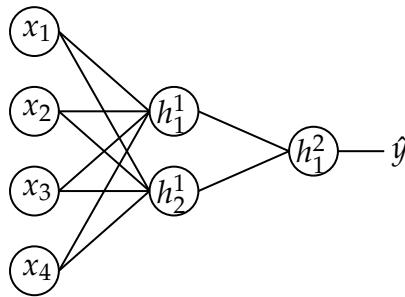


FIGURA 2.1: Red neuronal con una capa de entrada de 4 características, una capa oculta de 2 neuronas y una capa de salida con 1 neurona.

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: $f(x) = max(0, x)$
- Sofmax: $f(x_i) = \sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad \text{para } i = 1, 2, \dots, K$

2.2.2. Redes Convolucionales

Una de las principales limitaciones de las redes densas es que no son muy buenas para trabajar con datos que tienen una estructura espacial, como imágenes o audio. Esto se debe a que las redes densas tratan a todas las entradas de manera aislada, sin tener en cuenta las relaciones entre ellas. Esto puede llevar a que la red no sea capaz de detectar patrones o características importantes en los datos, lo que puede limitar su rendimiento.

Para solucionar este problema, surgió la idea de utilizar redes neuronales convolucionales (CNN). Estas redes se basan en la idea de 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 .

A modo de ejemplo, si se posee una matriz de $6 \times 6 I$ y un filtro $3 \times 3 K$, la operación de convolución $*$ que produce $I * K$ en una CNN puede visualizarse como:

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

I K $I * K$

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

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 también cuentan con capas de pooling, que se encargan de seleccionar las características más importantes de los datos y reducir su tamaño, lo que permite que la red tenga un mejor rendimiento y sea más eficiente. Existen distintos tipos de pooling: *max pooling*, el cual calcula el máximo valor de la submatriz; *min pooling*, el cual calcula el mínimo valor de la submatriz; y *average pooling*, el cual calcula el valor promedio de la submatriz.

$$\begin{pmatrix}
 \begin{array}{|c|c|} \hline 1 & 5 \\ \hline 3 & 2 \\ \hline \end{array} &
 \begin{array}{|c|c|} \hline 2 & 3 \\ \hline 4 & 2 \\ \hline \end{array} \\
 \hline
 \begin{array}{|c|c|} \hline 1 & 4 \\ \hline 3 & 2 \\ \hline \end{array} &
 \begin{array}{|c|c|} \hline 2 & 1 \\ \hline 1 & 1 \\ \hline \end{array} \\
 \hline
 \end{pmatrix} \xrightarrow{\text{maxpool}} \begin{pmatrix} 5 \\ 4 \\ 4 \\ 2 \end{pmatrix}$$

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

FIGURA 2.3: Maxpool 2×2 aplicado al ejemplo de la figura 2.2

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

Hay 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 presenta: una subred convolucional \mathcal{G} que extraen información y generan características f a partir de la entrada x ; y una subred densa \mathcal{C} que se encargan de clasificar las características aprendidas f en las etiquetas del problema \hat{y} .

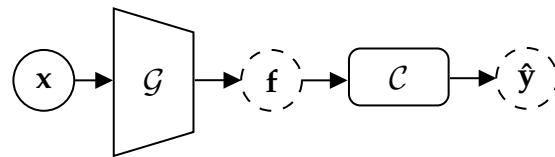


FIGURA 2.4: Arquitectura de una red simple. Las entradas x son procesadas por una red convolucional \mathcal{G} que generan características f para luego ser clasificadas por una red densa \mathcal{C} en \hat{y} .

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.

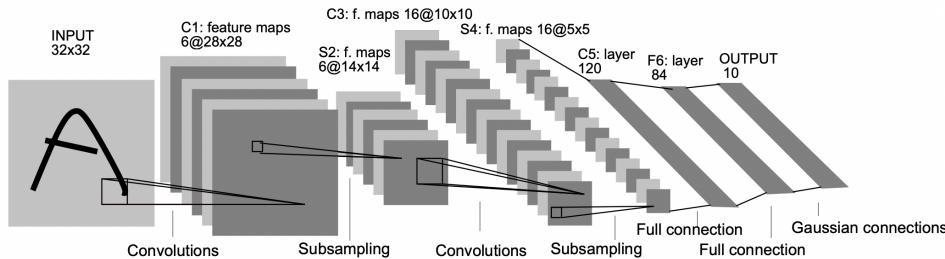


FIGURA 2.5: Arquitectura de una LeNet-5. Imagen tomada de LeCun et al., 1998.

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).

Desde el éxito de AlexNet (Krizhevsky, Sutskever e Hinton, 2017), las CNNs se han convertido en la mejor opción para aplicaciones de *computer vision* 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 de la propagación del gradiente.

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.

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, lo que las hace ideales para la implementación en dispositivos móviles y sistemas de baja potencia.
- 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 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 y el procesamiento del lenguaje natural (Howard et al., 2017). 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 actuales 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) consta de dos redes en la adaptación del dominio (figura 2.6). La primera es la discriminadora de dominios \mathcal{D} entrenada para distinguir las características de origen de las de destino y la segunda es una generadora de características \mathcal{G} que se entrena para confundir a \mathcal{D} y simultáneamente ayudar a la red clasificadora \mathcal{C} .

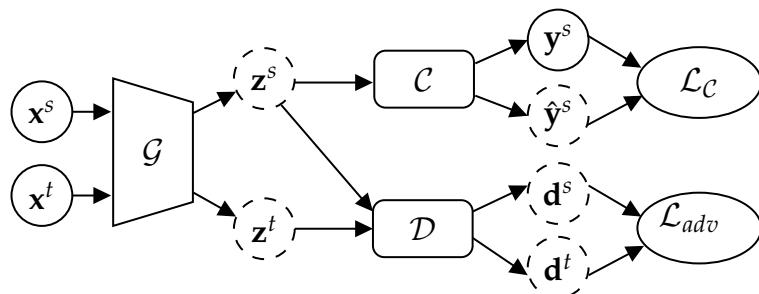


FIGURA 2.6: Esquema de las *DANN*. Los supra indices s y t indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos.

El 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:

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

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.

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

Por lo tanto, el objetivo de un modelo *DANN* viene dado por la ecuación 2.5. Donde λ 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}_{adv} 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* debido a que su estrategia de optimización podría no funcionar bien en la práctica debido a la desvanecimiento del gradiente, que también es 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 separados (figura 2.7) siendo el primero el pre-entrenamiento de \mathcal{G}_s y \mathcal{C} en los datos de origen y el otro siendo la adaptación adversaria de una \mathcal{G}_t utilizando a \mathcal{G}_s pre-entrenada y \mathcal{D} .

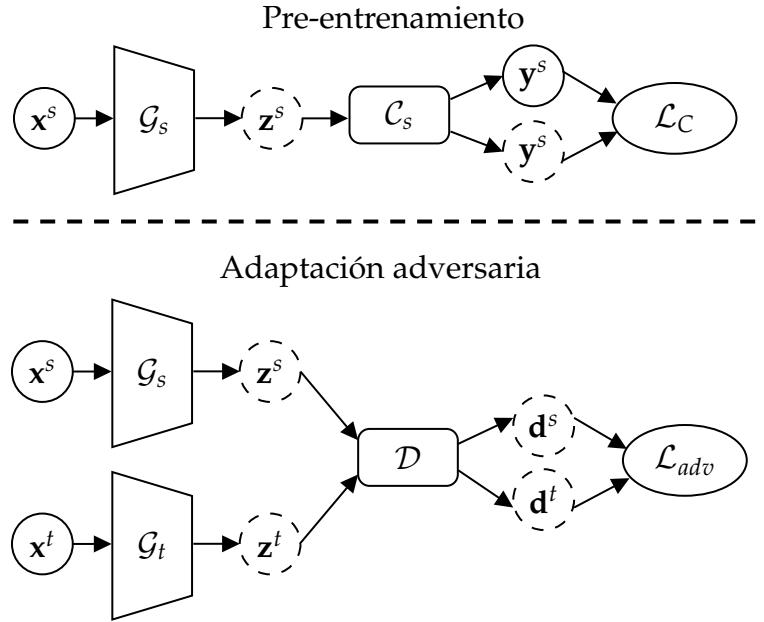


FIGURA 2.7: Esquema de las ADDA. Los supra índices s y t indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. La primera fase consta de un pre-entrenamiento con una red 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 son similares a ADDA 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.

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

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

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

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 eigenvectores con valores singulares más altos dominan la *transferibilidad* mientras que los eigenvectores 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 eigenvectores 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 Σ_s y Σ_t respectivamente. En la presente tesis se utilizará $k = 1$.

BSP puede integrarse a cualquiera de los esquemas mencionados anteriormente, por ejemplo a una *DANN* (figura 2.8).

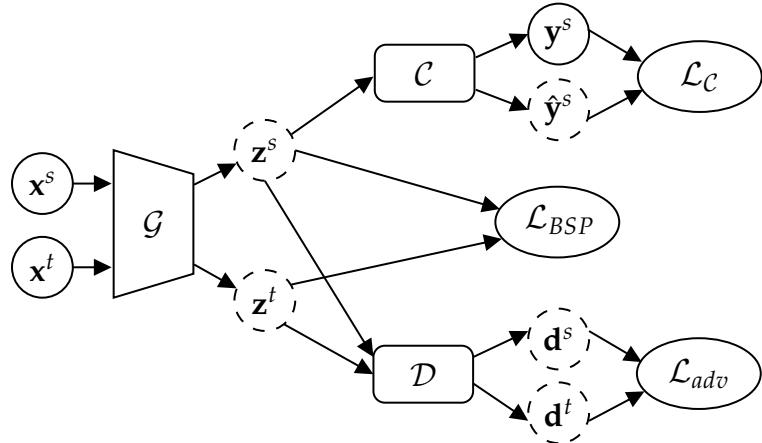


FIGURA 2.8: Esquema de un modelo *DANN+BSP*.

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

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

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

Donde β 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).

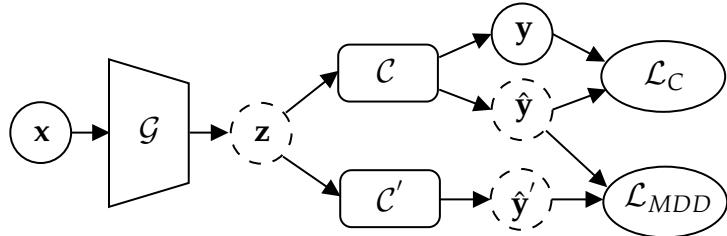


FIGURA 2.9: Esquema de *MDD*. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. *MDD* introduce un clasificador clasificador adversario \mathcal{C}' para maximizar la discrepancia y entrena el generador de características \mathcal{G}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

\mathcal{L}_C continúa 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* ρ que se obtiene introduciendo el parámetro $\gamma \triangleq \exp \rho$ en la ecuación 2.13.

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

Donde σ es la función *softmax*. Por lo tanto, el objetivo de optimización viene dado por la ecuación 2.14, donde λ 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 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.

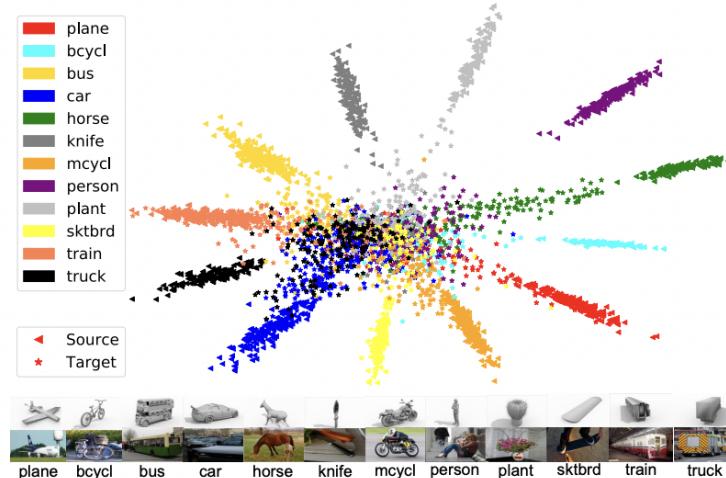


FIGURA 2.10: Visualización de las características aprendidas para el origen y destino utilizando un modelo entrenado utilizando muestras del origen. Imagen tomada de Xu et al., 2019.

Esto sugiere que las normas excesivamente pequeñas de las muestras 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* 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} .

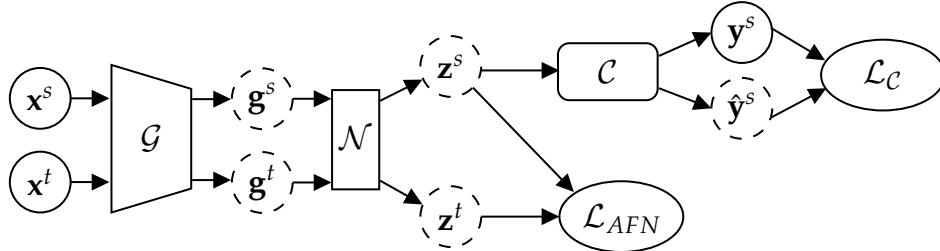


FIGURA 2.11: Esquema de AFN. Los supra indices s y t indican si los datos son del origen o destino respectivamente. Los círculos con líneas rayadas son salidas de los modelos mientras que los círculos de líneas sólidas son datos conocidos. El modelo *backbone* \mathcal{G} produce características \mathbf{x}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

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

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

Donde Δ_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* σ para facilitar la discriminabilidad mediante un factor β :

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

Por lo tanto, el objetivo de optimización viene dado por la ecuación 2.17, donde λ 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](#). Presentan un formato estándar en forma de grilla donde en cada renglón se encuentra el partido político y los votos obtenidos para diputados y senadores. En la página también se puede descargar un CSV donde se encuentra el *id* de cada telegrama y los valores digitalizados oficialmente.

3.2. Extracción de dígitos de los telegramas

Los telegramas de las elecciones de Santa Fe presentan un formato tabular, donde cada fila representa un partido político y los votos obtenidos abiertos por senadores y diputados. Se debieron ejecutar múltiples pasos de extracción, transformación y carga (ETL por sus siglas) de los mismos antes de poder armar un dataset con el cual entrenar los modelos. Se utilizó la librería OpenCV (Bradski, 2000) para manipular las imágenes y poder llevar a cabo el proceso de ETL.

3.2.1. Enderezado

Como los telegramas son escaneados a mano, el primer paso consiste en enderezarlos. Este proceso puede realizarse buscando el rectángulo de mayor área, calculando el ángulo de rotación y rotando la imagen completa con la función `getRotationMatrix2D` de OpenCV.

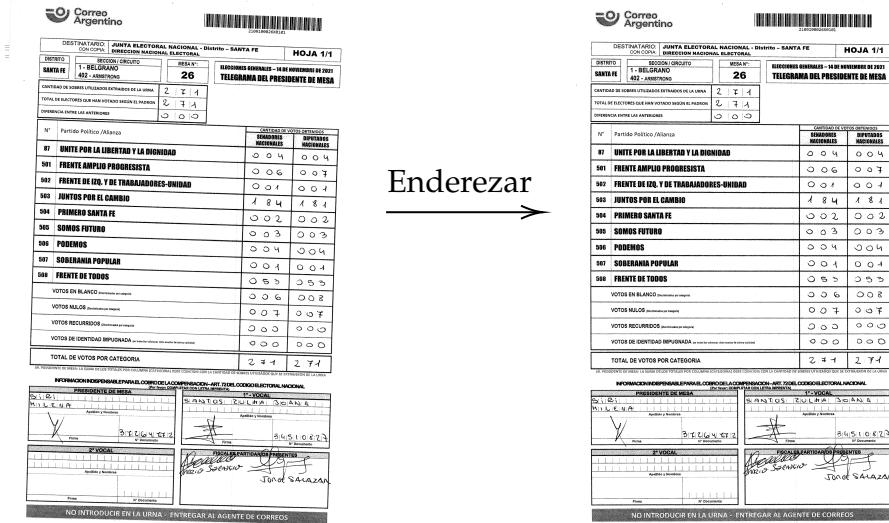


FIGURA 3.1: Enderezamiento de un telegrama utilizando OpenCV.

3.2.2. Extracción de la grilla de votos

El siguiente paso consiste en poder seleccionar la grilla de los votos y poder extraerla para seguir trabajando en ella. Esto puede realizarse utilizando la función getContours de OpenCV y quedándose con el contorno de mayor área.

Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS	
		SEÑALADORES NACIONALES	SEÑALADORES MUNICIPALES
07	UNITE POR LA LIBERTAD Y LA DIGNIDAD	0 0 4	0 0 4
501	FRENTE AMPLIO 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 4
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	SOBERANIA POPULAR	0 0 1	0 0 1
508	FRENTE DE TODOS	0 5 5	0 5 5
VOTOS EN BLANCO (señalados por completo)			
VOTOS NULOS (señalados por completo)			
VOTOS RECURRIDOS (señalados por completo)			
VOTOS DE IDENTIDAD IMPUGNADA (señalados por completo)			
TOTAL DE VOTOS POR CATEGORÍA		2 4 1	2 7 4

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. El mismo es similar al utilizado en (Lamagna, 2016). Debido a que la grilla se encuentra enderezada, se puede utilizar proyecciones de los colores sobre los ejes x e y para detectarlas. Al sumar todos los colores por eje, se pueden encontrar los picos donde se encuentran las líneas horizontales y verticales. Posteriormente, se selecciona un umbral de corte por cada

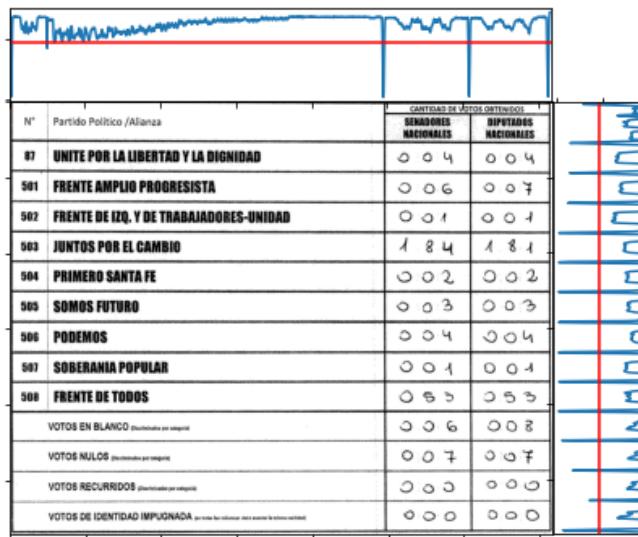


FIGURA 3.3: Proyecciones de los ejes de la grilla. En rojo se marca el umbral de corte.

eje, siendo el promedio menos dos desvíos estándar. La figura 3.3 muestra las proyecciones obtenidas con los umbrales por cada eje.

Sin embargo, aunque se seleccione aquellos píxeles que se encuentren por debajo del umbral, por cada pico existe mas de un pixel. 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 pixel por línea.

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

Nº	Partido Político /Alianza	CANTIDAD DE VOTOS DETERMINADOS	
		SEÑADORES NACIONALES	DIPUTADOS NACIONALES
07	UNITE POR LA LIBERTAD Y LA DIGNIDAD	0 0 4	0 0 4
501	FRENTE AMPLIO 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 4
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	SOBERANIA POPULAR	0 0 1	0 0 4
508	FRENTE DE TODOS	0 5 5	0 5 5
	VOTOS EN BLANCO (descuentados por voto nulo)	0 0 6	0 0 8
	VOTOS NULOS (descuentados por voto nulo)	0 0 7	0 0 7
	VOTOS RECURRIDOS (descuentados por voto nulo)	0 0 0	0 0 0
	VOTOS DE IDENTIDAD IMPUGNADA (en votos que no se consideran válidos para el escrutinio)	0 0 0	0 0 0

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.



FIGURA 3.5: Primer registro extraído de la grilla de votos.

Para separar cada dígito en una única imagen, se utiliza un análisis de componentes conectados (connectedComponents de OpenCV) y se recortan aquellos contornos que posean de área entre el 5 % y el 70 % de los píxeles totales de la imagen. Los valores fueron obtenidos mediante experimentación y sirven para descartar ruido que pueda detectar el análisis de componentes conectados.



FIGURA 3.6: Dígitos detectados en el primer bloque de votos.

Por último, los dígitos son encuadrados y se guardan junto al partido político y al tipo de voto (senadores o diputados) en un dataset, cuyo análisis será abordado en la siguiente sección.

3.3. Análisis del dataset

Con la extracción descripta en la sección anterior, se procede a hacer un análisis exploratorio de los datos en búsqueda de posibles errores. La tabla 3.1 muestra algunos registros del dataset que vincula los dígitos extraídos de los telegramas con los partidos políticos.

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 dataset. Cada fila representa un voto.

El lector podrá darse cuenta que existen errores en el cuadro anterior. En primer lugar, en la columna donde sólamente deben haber las imágenes de

los dígitos existe una con una línea en el segundo registro. Por otro lado, también existen dígitos mal escritos como los que se encuentran en los dos últimos registros. Para limpiar estas extracciones incorrectas, se calculan dos columnas las cuales contienen las proporciones mínimas y máximas de pixeles blancos en las imágenes por cada voto. La figura 3.7 muestra los histogramas generales de las mismas.

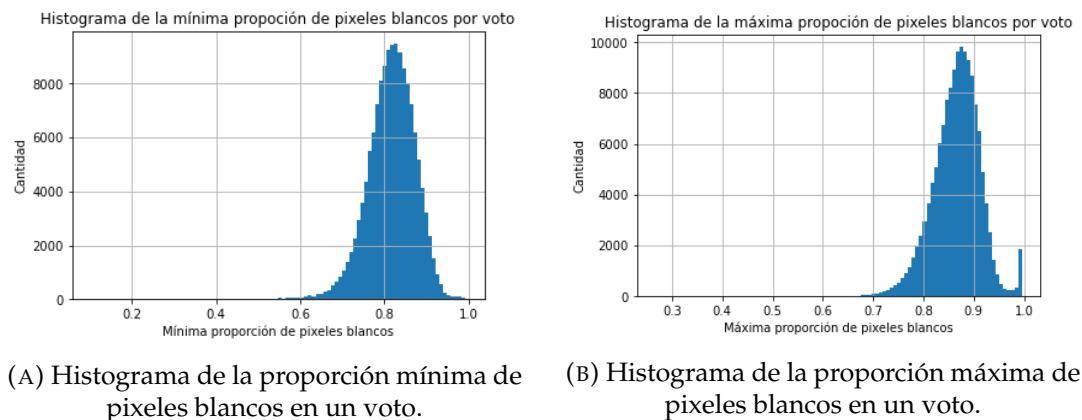


FIGURA 3.7: Histogramas de proporciones de pixeles blancos por voto.

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 pixeles blancos, se logran descartar los casos de líneas como la vista en el cuadro 3.1. Por el contrario, al eliminar aquellas imágenes que posean menos del 50 % de pixeles blancos, se logran descartar los casos de escritura incorrecta.

El siguiente punto a analizar es el tamaño de las imágenes. Como fue mostrado en el cuadro 3.1, las imágenes no fueron estandarizadas en un tamaño estándar. De manera análoga a la variable de proporción de pixeles 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 los valores extremos, se encuentran telegramas que fueron cargados de forma incorrecta (ver anexo A.2) o con caligrafía en la cual los números se encuentran unidos y no pueden ser separados por el análisis de componentes conectados de OpenCV (ver anexo A.3). Se eliminan estos casos filtrando por aquellos registros que se encuentren por fuera de la media ± 4 desvíos estándar.

Concluida la etapa de análisis y limpieza, se exportan los dígitos en un formato similar al dataset *MNIST* con la etiqueta que oficialmente fue registrada en las elecciones del 2021. El mismo será nombrado como *TDS* (dataset de telegramas) de aquí en adelante. El mismo posee un total de 170.718 imágenes de dígitos cuadradas con sus dimensiones orginales, es decir, no se encuentran estandarizados en una dimensión específica. Esto último es adrede ya que puede servir para mejorar el dataset en futuros trabajos.



FIGURA 3.8: Dataset TDS de dígitos.

3.4. Diseño experimental

Los experimentos fueron programados en Python mediante la librería PyTorch (Paszke et al., 2019) y TLLIB (Junguang Jiang, Chen y Fu, 2020) para aplicar las técnicas de adaptación de dominio.

Se entrenaron redes ResNet-50 (He et al., 2016) y LeNet-8 (LeCun et al., 1998) por cada una de las técnicas de adaptación de dominio descriptas en el capítulo 2. Como modelo baseline, se entrenaron ambas redes sin ninguna técnica de adaptación.

3.4.1. Metodología de entrenamiento

Para entrenar modelos mediante adaptación de dominio, se precisan dos datasets: el de origen que se encuentra etiquetado y el de destino sin etiquetar. Los experimentos llevados a cabo utilizan el *MNIST* como origen y el *TDS* como destino. En cada iteración de entrenamiento, se provee de un batch de 1024 imágenes del dataset origen y otro del de destino. Se realizaron 3 particiones de cada uno:

- 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.

	DANN	ADDA	DANN+BSP	MDD	AFN	Sin AD
η_0	[1e-4, 0.1]	[1e-4, 0.1]				
λ	[0.5, 2]	[0.5, 2]	[0.5, 2]	[0.5, 2]	[0.001, 0.1]	-
β	-	-	[1e-5, 0.1]	-	[0.0, 0.1]	-
γ	-	-	-	[1, 10]	-	-
Δ_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 mencionada anteriormente pueden ser combinadas en una sola denominada F_β . El valor de β 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:

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

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 ϵ 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 $\text{dist}_{\mathcal{A}}$ 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 $\text{dist}_{\mathcal{A}} \approx 2$. Por el contrario, en la sub-figura

3.9b las distribuciones de dominios son similares entre si, por lo que se espera que $dist_{\mathcal{A}} \approx 0$.

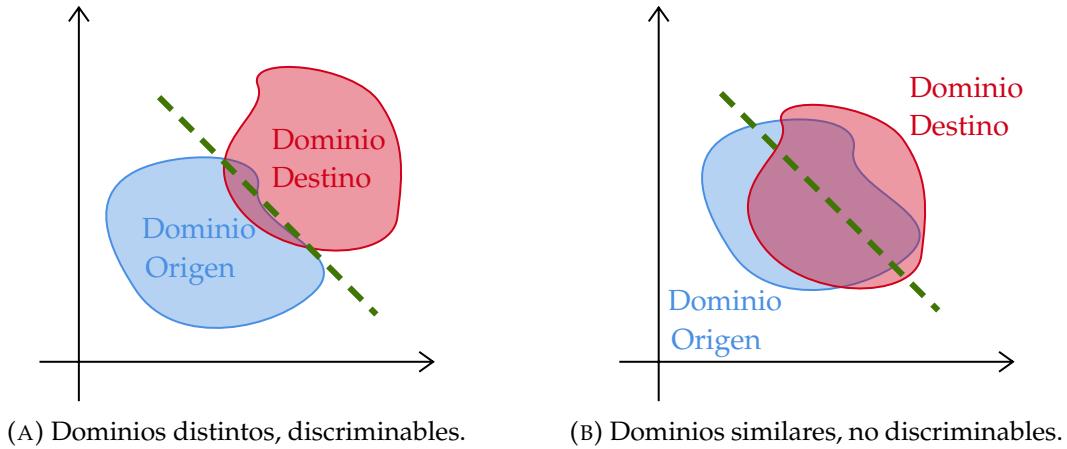


FIGURA 3.9: Ejemplos de distribuciones de dominios.

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. El mejor modelo es el que posea mejor capacidad de clasificación y de adaptación.

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	(1) 0.9890	(1) 0.9890	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	(1) 0.0040	(1) 1.0886
	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.

Como era de esperarse, los modelos que fueron entrenados sin adaptación de dominio son los que mejores métricas de clasificación poseen. No obstante, sus valores de adaptación son los peores provocando que no puedan generalizar a *TDS*.

El modelo que mejor combinación de clasificación y adaptación es la ResNet utilizando AFN. Cabe destacar que los modelos LeNet entrenados con DANN y AFN obtuvieron buenas métricas en general, lo que deja en evidencia que *no siempre un modelo más complejo es mejor*.

Es posible aplicar los modelos sobre todos los telegramas y calcular el *IoU* promedio por telegrama con el cálculo descripto en el capítulo 3 y la cantidad de aciertos promedio por telegrama utilizando como etiquetas lo transcripto en el centro de cómputo suponiendo que existen pocos errores en ellos.

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	0.7486	13	72 %
	LeNet	0.6493	11	61 %

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

El cuadro 4.2 confirma la elección del mejor modelo realizada anteriormente. Independientemente de qué técnica de adaptación se utilice, todas presentan mejores porcentajes de aciertos promedio que los modelos que fueron entrenados únicamente con *MNIST*.

TODO: Actualizar para agregar DANN + BSP. TODO: buscar una mejor forma de mostrar los histogramas para que se vea las distintas distribuciones de aciertos dependiendo de cada modelo y AD.

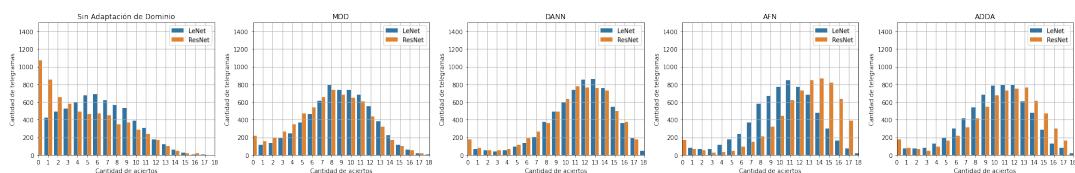


FIGURA 4.1: Distribución de aciertos de cantidad de votos por cada par técnica AD y modelo.

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

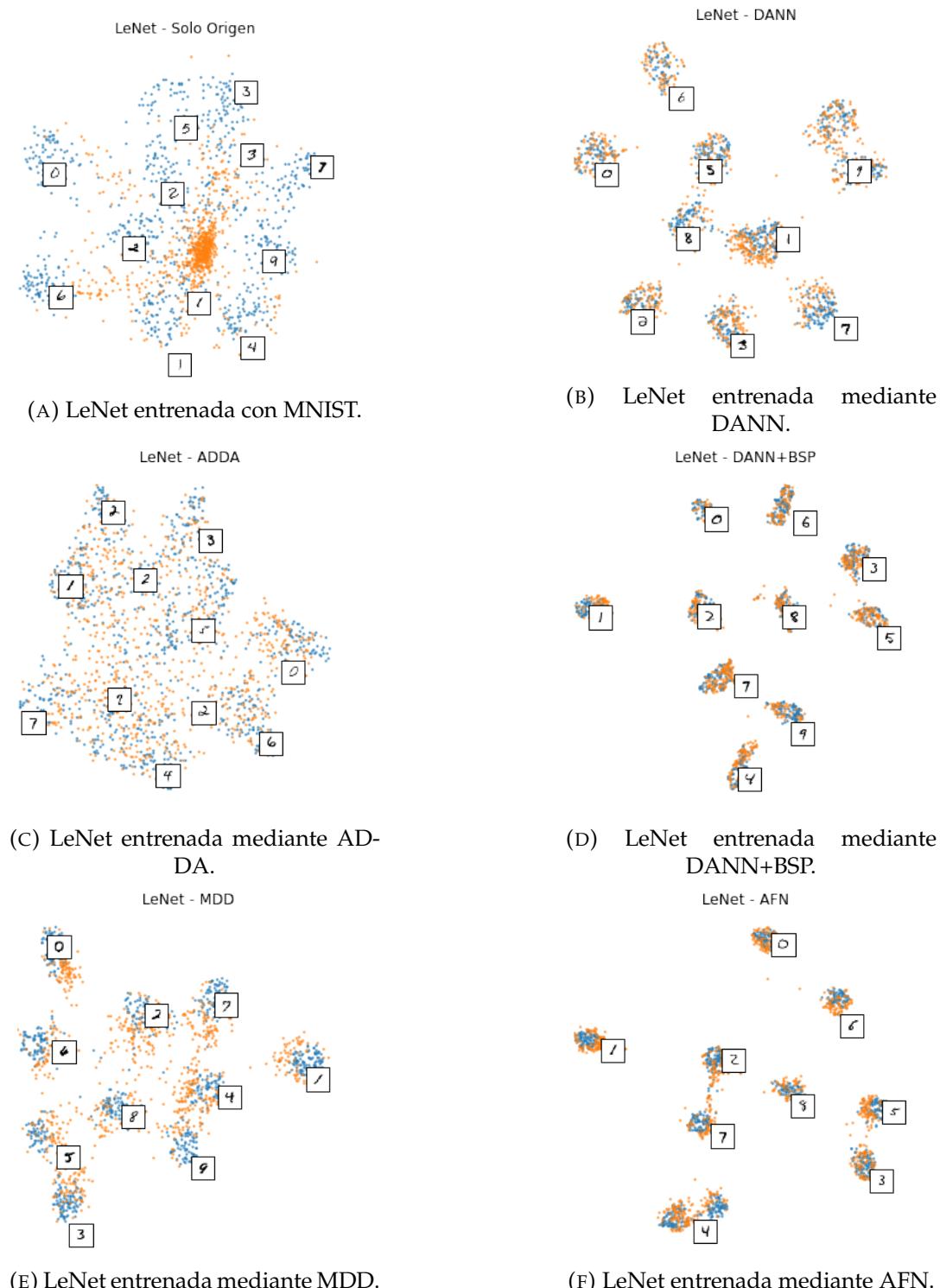


FIGURA 4.2: Representaciones UMAP de los espacios latentes de los modelos LeNet. Los puntos naranjas representan observaciones de MNIST y los azules de TDS.

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 4.2a.
- 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 4.2c y 4.2e.
- Al entrenar el modelo mediante DANN, se generan agrupaciones similares para ambos conjuntos de datos, como se muestra en la figura 4.2b.
- 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 4.2d y 4.2f.

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

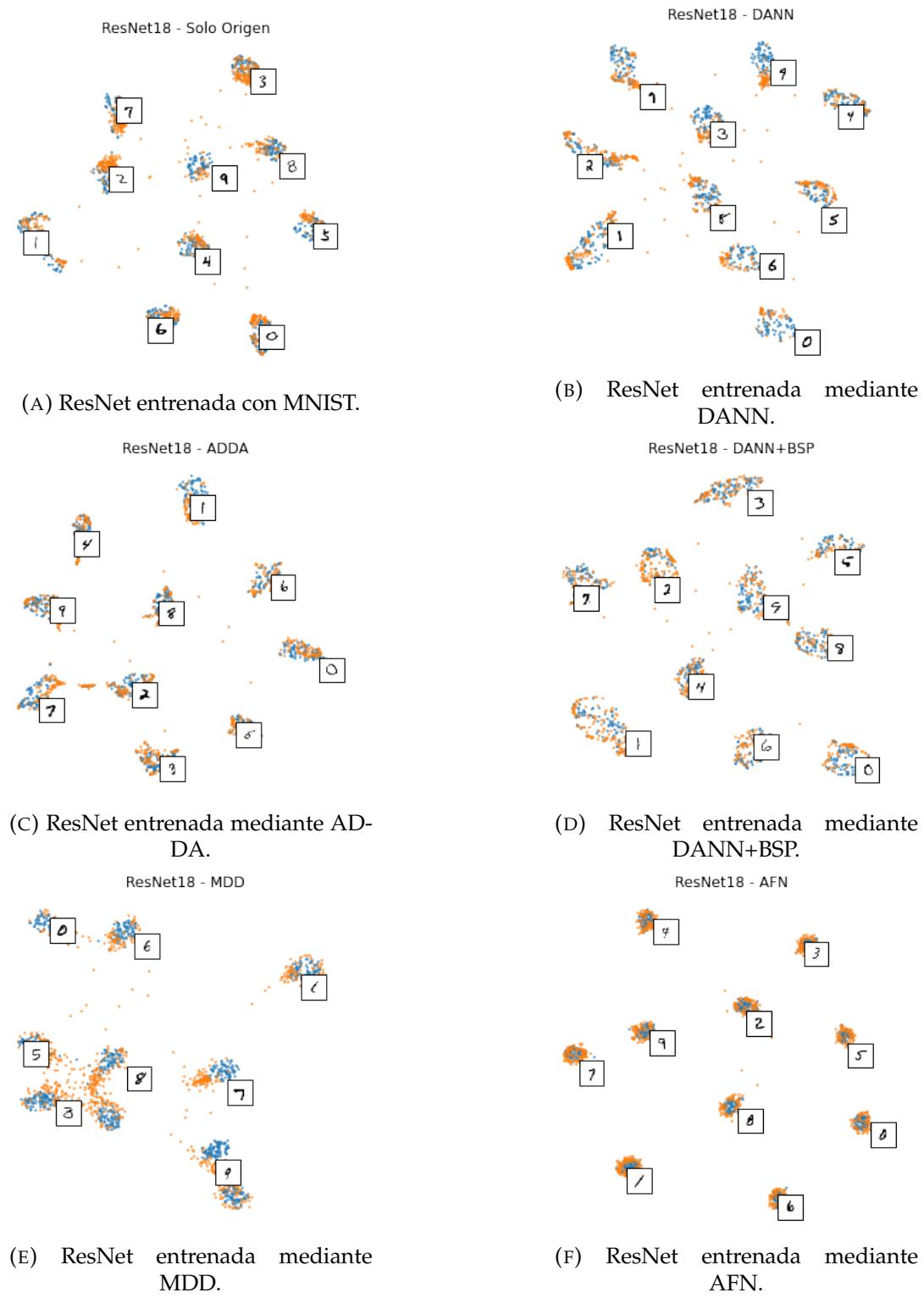


FIGURA 4.3: Representaciones UMAP de los espacios latentes de los modelos ResNet. Los puntos azules representan observaciones de TDS y los naranjas de MNIST.

4.3. Análisis de errores

Los errores de predicción de los modelos pueden reflejarse en las distribución de la métrica IoU para cada uno de ellos.

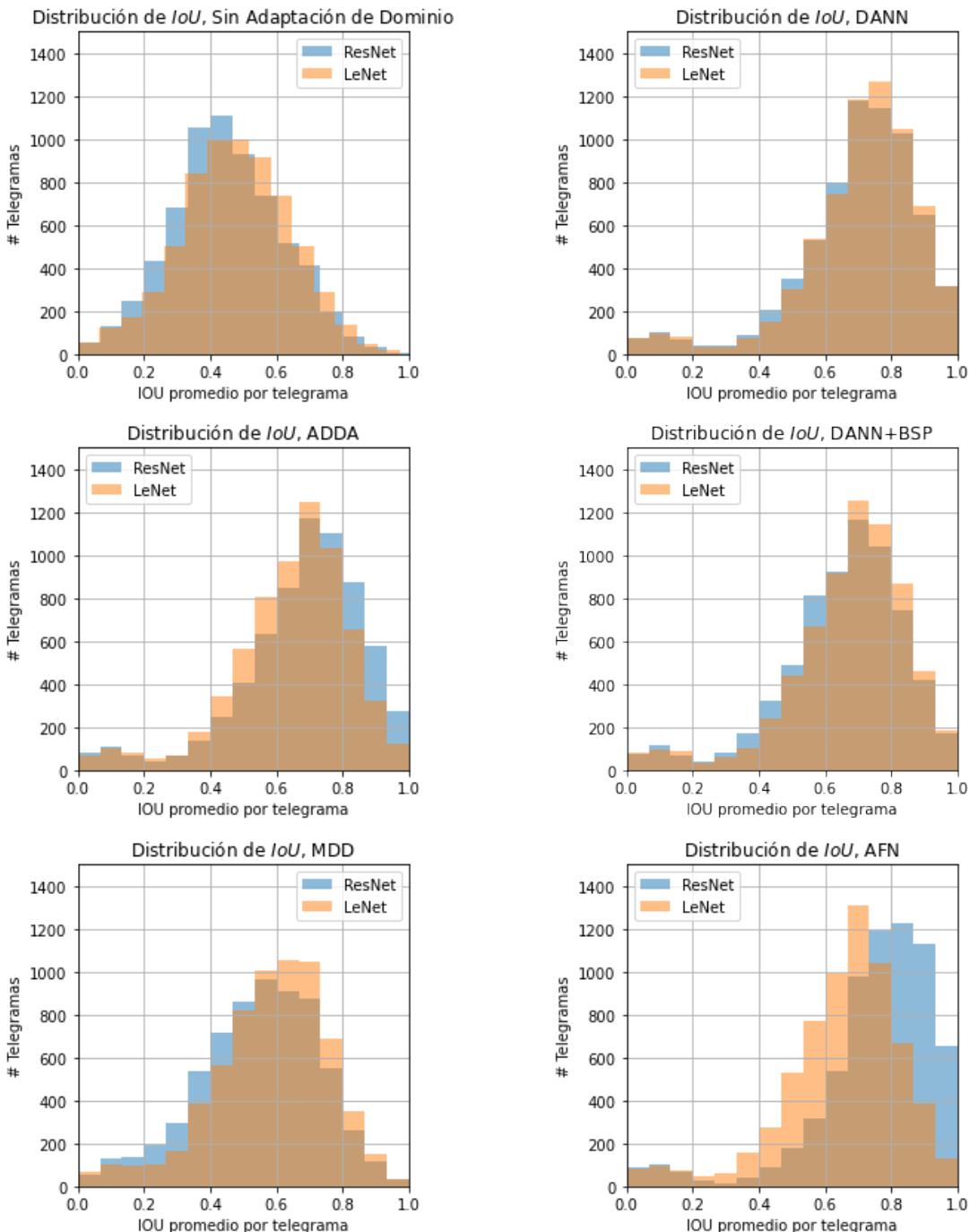


FIGURA 4.4: Histogramas de la métrica IoU promedio por telegrama por cada técnica AD y modelo.

Resulta interesante mencionar existen telegramas que son más difíciles de analizar que otros. Esto puede evidenciarse en los histogramas de la figura 4.4 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

5.1. Conclusiones

TODO: completar. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

5.2. Trabajos Futuros

TODO: completar. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Apéndice A

Anexo: Telegramas

A.1. Ejemplo de telegrama

Correo Argentino

DESTINATARIO:
CON COPIA: JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE
DIRECCION NACIONAL ELECTORAL

HOJA 1/1

DISTRITO SANTA FE	SECCION / CIRCUITO 1 - BELGRANO 402 - ARMSTRONG	MESA N°: 1	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 TELEGRAMA DEL PRESIDENTE DE MESA
CANTIDAD DE SOBRES UTILIZADOS EXTRAIOS DE LA URNA		2 8 3	
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON		2 8 3	
DIFERENCIA ENTRE LAS ANTERIORES		0 0 0	

Nº	Partido Político /Alianza	CANTIDAD DE VOTOS OBTENIDOS	
		SENADORES NACIONALES	DIPUTADOS NACIONALES
87	UNITE POR LA LIBERTAD Y LA DIGNIDAD	4	5
501	FRENTE AMPLIO PROGRESISTA	15	16
502	FRENTE DE IZQ. Y DE TRABAJADORES-UNIDAD	8	8
503	JUNTOS POR EL CAMBIO	135	135
504	PRIMERO SANTA FE	2	2
505	SOMOS FUTURO	7	7
506	PODEMOS	1	1
507	SOBERANIA POPULAR	6	6
508	FRENTE DE TODOS	94	92
VOTOS EN BLANCO (determinados por categoria)		5	5
VOTOS NULOS (determinados por categoria)		6	6
VOTOS RECURRIDOS (determinados por categoria)		—	—
VOTOS DE IDENTIDAD IMPUGNADA (en todas las columnas debe asentar la misma cantidad)		—	—
TOTAL DE VOTOS POR CATEGORIA		283	283

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		1º - VOCAL	
Apellidos y Nombres		Apellido y Nombres	
	3 8 0 8 7 4 6 1 8		3 7 2 2 8 1 2 8
Firma	Nº Documento	Firma	Nº Documento
2º VOCAL		FISCALES PARTIDARIOS PRESENTES	
Apellidos y Nombres		Apellidos y Nombres	
	4 2 3 2 8 1 1 7		3 0 2 5 0 4 9 5
Firma	Nº Documento	Firma	Nº Documento

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



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

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

Apellido y Nombres	
<i>LIUTI YANINA</i>	
Firma	Nº Documento
<i>Liuti Yanina</i>	923841120

1^{er} VOCAL

Apellido y Nombres	
<i>FRAGO LETTI YANNA PAOLA</i>	
Firma	Nº Documento
<i>Frago Letti Yanina Paola</i>	32454035

2^{do} VOCAL

Apellido y Nombres	
<i>GONZALEZ MARINA</i>	
Firma	Nº Documento
<i>Gonzalez Marina</i>	95869553

FICALES PARTIDARIOS PRESENTES

Apellido y Nombres	
<i>JUAN M. LUSAADI 11326427 JUNTOS POR EL CAMBIO</i>	
Firma	Nº Documento
<i>Juan M. Lusaadi</i>	
<i>Silvietta P. Pachoud Phivita. FRENTE DE TODOS.</i>	

NO INTRODUCIR EN LA URNA - ENTREGAR AL AGENTE DE CORREOS

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



DESTINATARIO: CON COPIA:		JUNTA ELECTORAL NACIONAL - Distrito - SANTA FE DIRECCION NACIONAL ELECTORAL	HOJA 1/1
DISTRITO SANTA FE	SECCION / CIRCUITO 9 - LA CAPITAL 20A - SANTO TOME	MESA N°: 3008	ELECCIONES GENERALES - 14 DE NOVIEMBRE DE 2021 TELEGRAMA DEL PRESIDENTE DE MESA
CANTIDAD DE SOBRES UTILIZADOS EXTRADIDOS DE LA URNA		2 1 6	
TOTAL DE ELECTORES QUE HAN VOTADO SEGUN EL PADRON		2 1 6	
DIFERENCIA ENTRE LAS ANTERIORES		1 X X	
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		2 0 2	2 0 1

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

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 PALAD AGUSTINA		1º - VOCAL	
Apellido y Nombres 37295734 Firma N° Documento		Apellido y Nombres Firma N° Documento	
2º VOCAL		ESCALES PARTIDARIOS PRESENTES	
Apellido y Nombres Paula Alcoba 35570523 Firma N° Documento			
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, 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.
- 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*, pág. 770-778.
- Hebb, DO (1949). «The organization of behavior. A neuropsychological theory». En.
- Howard, Andrew G et al. (2017). «Mobilennets: Efficient convolutional neural networks for mobile vision applications». En: *arXiv preprint arXiv:1704.04861*.
- Howard, Jeremy y Sebastian Ruder (2018). «Universal language model fine-tuning for text classification». En: *arXiv preprint arXiv:1801.06146*.
- 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, pág. 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 Assymetric 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. (1998). «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11, págs. 2278-2324.
- 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). «Perceptrons.» En.
- 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.