UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

PROGRAMA MAGÍSTER EN CIENCIAS DE LA COMPUTACIÓN

ADAPTACIÓN DE DOMINIO PROFUNDA SOBRE IMÁGENES ASTRONÓMICAS

Tesis presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al grado académico de Magíster en Ciencias de la Computación

POR CESAR ANDRES BOLIVAR SEVERINO

Profesor Guía: Guillermo Cabrera Vives

25 de abril de 2022

Concepción, Chile

Dedicado a mi padre, que en paz descanse.

AGRADECIMIENTOS

Me encantaría poder decir que este trabajo es de mi autoría y mío solamente, pero sería una vil mentira. Sin la ayuda y el apoyo de muchas personas, este trabajo jamás hubiese sido posible.

En primer lugar quiero agradecer a mi familia, a mi padre por haber sido una inspiración y por haberme inculcado la curiosidad desde pequeño. A mi madre por criarnos a 3 hermanos prácticamente sola y por sacar la familia adelante luego de que mi padre falleciera. A mi hermana por ser el ejemplo más cercano de éxito y auto superación que conozco y por su apoyo económico, y a mi hermano por ser la persona más cercana y ayudarme a mantenerme cuerdo durante la cuarentena.

Gracias a mis amigos y ex compañeros de colegio con los cuales aún comparto noches hablando por discord, a veces estando muertos de la risa, y a veces al borde de la depresión, pero siempre presentes. Gracias a mis amigos y compañeros de universidad, en especial a Daniel Ortega y Martin Pincheira, que más que amigos han sido hermanos para mí. Sin su compañía y apoyo no hubiese podido sobrevivir el pregrado. Gracias a Valentina Hernández, por validar y sugerir el modelo de adaptación de dominio utilizado en este trabajo, sin el cual aún estaría estancado sin resultados.

Gracias a los docentes del Departamento de Ingeniería Informática y Ciencias de la Computación (DIICC) de la Universidad de Concepción, por haberme formado durante tantos años y por ser personas muy inspiradoras. Agradecimientos especiales para Guillermo Cabrera, por ser el mejor profesor guía que podría haber pedido, y a Roberto Asín, por la confianza, el apoyo y las oportunidades que me ha brindado a lo largo de los años.

Gracias a los chicos y chicas de la “Data Science League”, por tener siempre buena disposición para ayudar cuando me enfrentaba a algún problema y no podía avanzar con mi tesis.

Finalmente, este trabajo fue financiado con el apoyo de la Beca de Magíster Nacional año 2020 otorgada por la Agencia Nacional de Investigación y Desarrollo (ANID, ex CONICYT), folio número 22201249. Sin este apoyo económico, 2020 hubiese sido un año aún peor.

Gracias a todos.

ÍNDICE

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

RESUMEN

Los modelos de Aprendizaje Profundo pueden verse afectados negativamente cuando ocurre un cambio de distribuición entre el conjunto de datos de entrenamiento y el conjunto de datos de prueba. Esta baja en desempeño puede acrecentarse aún más cuando se tienen pocos datos etiquetados, pero puede mitigarse empleando técnicas de Apatación de Dominio. En este trabajo, se estudia la clasificación binaria de alertas astronómicas en “real” versus “bogus”, utilizando cuatro conjuntos de datos diferentes: Asteroid Terrestrial-impact Alert System (ATLAS), Dark Energy Survey (DES), High-cadence Transient Survey (HiTS) y Zwicky Transient Facility (ZTF). Se utiliza un modelo de clasificación profundo con un entrenamiento fine tuning y un modelo de adaptación de dominio llamado Minimax Entropy (MME), y se estudia el comportamiento de dichos modelos en diferentes escenarios donde el conjunto de entrenamiento difiere al de prueba, y donde se utiliza pocos elementos etiquetados por clase de los conjuntos de datos objetivo para el entrenamiento. Se muestra que el domain shift está presente en los conjuntos de datos, y que ambos modelos son capaces de mejorar la exactitud de un modelos base, incluso con apenas 1 elemento etiquetado por clase para algunos de los escenarios propuestos.

ABSTRACT

Deep Learning models can be adversely affected when there is a shift in distribution between the training dataset and the test dataset. This drop in accuracy can be further exacerbated when there is little labeled data, but can be mitigated by employing Domain Adaptation techniques. In this work, we study the problem of binary classification of astronomical alerts in “real” versus “bogus”, using four different datasets: Asteroid Terrestrial-impact Alert System (ATLAS), Dark Energy Survey (DES), High-cadence Transient Survey (HiTS) and Zwicky Transient Facility (ZTF). We take a deep model for classification in conjunction with fine tuning training, and a domain adaptation model called Minimax Entropy (MME), and we explore the behavior of these models in different settings where the training set differs from the test set, and where only a few labeled items per class from the target dataset are used in the training process. We show that domain shift is present in these datasets, and that both models are able to improve the balanced accuracy of a base model, even with only 1 labeled element per class for some of the proposed scenarios.

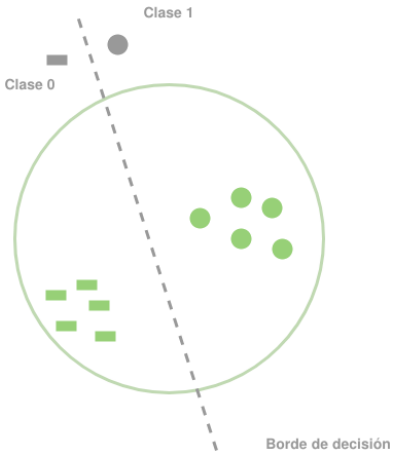
1. Introducción

1.1. Contexto

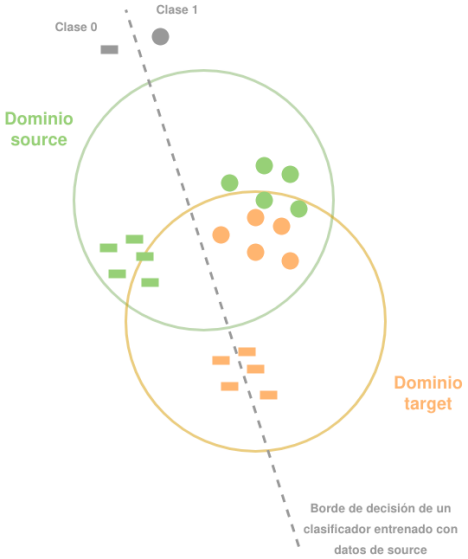
El *Observatorio Vera C. Rubin*[1], previamente conocido como *Large Synoptic Survey Telescope* o *LSST*, es un telescopio actualmente en construcción ubicado en la Zona Norte de Chile. El telescopio contará con una cámara de 3.2 gigapíxeles que se estima que producirá terabytes de datos en fotografías cada noche. Este volumen de datos masivo hace que sea intratable procesar dichos datos manualmente, por lo que el tratamiento automático de dichos datos jugará un papel importante.

Una de las tareas que actualmente se realiza de forma automática en astronomía es la clasificación de *alertas* astronómicas. Una de las formas de en que esto se consigue, es empleando emplear técnicas de *aprendizaje automático*[2], las cuales han sido exitosas en una variedad de tareas de visión computacional[3][4][5]. Dichas técnicas, se basan en ajustar los parámetros de un modelo profundo utilizando los datos de un *conjunto de entrenamiento*, para luego predecir sobre un *conjunto de prueba*.

Sin embargo, los conjunto de entrenamiento y de prueba pueden presentar diferencias de distribución entre ellos. Un ejemplo de este fenómeno se da, por ejemplo, cuando se tienen conjuntos de imágenes de que provienen de telescopios diferentes. Esta diferencia, conocida como *domain shift*, es un problema para los modelos de aprendizaje automático, pues debido a ella, un modelo puede disminuir su exactitud significativamente sobre el conjunto de prueba. La figura 1 muestra una representación visual en 2 dimensiones de dicho fenómeno.



(a) Clasificación sin domain shift.



(b) Clasificación ante la presencia del domain-shift.

Figura 1: Domain shift. A la izquierda, se muestra un clasificador que predice correctamente todos los elementos. A la derecha, se ve que al cambiar de dominio el clasificador comienza a incurrir en errores.

El fenómeno del domain shift ha sido observado en trabajos previos. En[6], cambiar de un dominio de imágenes sacadas de comercio en línea (Amazon) a imágenes capturadas por una cámara web, hace caer la exactitud de un modelo 61 % a 19 %. En [7], cambiar de un dominio de imágenes de supernovas obtenidas mediante espectroscopía a uno obtenido mediante fotometría, hace que 4 modelos diferentes bajen de una exactitud de sobre 90 % a exactitudes que rondan en el 70 %. Es posible que el domain shift también se presente si se intenta clasificar los datos del LSST con un modelo entrenado con datos que provengan de otro telescopio, por lo que el desempeño de modelos existentes como los de [8] podría verse negativamente afectado.

Además del domain shift, existe otro problema, el cual es la disponibilidad de datos etiquetados del LSST. Los modelos de aprendizaje automático supervisado requieren datos etiquetados, es decir, además de la información provista por los datos mismos, se requiere información sobre las etiquetas que se desea predecir. Para el LSST sin embargo, esta información no existirá hasta que ya haya pasado un tiempo desde que inicie su funcionamiento y se haya etiquetado una cantidad suficiente de datos, por lo que no será posible en principio utilizar una cantidad masiva de datos etiquetados de LSST como conjunto de entrenamiento. Sin embargo, sería deseable poder clasificar los datos del LSST desde el día uno, aún con la escasez de datos etiquetados.

Afortunadamente, han surgido las técnicas de *Domain Adaptation* (adaptación de dominio), que abordan el problema de domain shift. El objetivo de la adaptación de dominio es, dados dos conjuntos de datos denominados *fuente* (o *source*) y *objetivo* (o *target*) y un modelo de aprendizaje automático, conseguir que dicho modelo se desempeñe, idealmente tan bien sobre target como en source. Cuando el modelo corresponde a un modelo profundo, como por ejemplo una red neuronal convolucional, se habla de *adaptación de dominio profunda*. El beneficio de estas técnicas es que pueden trabajar con las etiquetas de los datos del dominio source en conjunto con pocas o incluso ninguna etiqueta del dominio target.

Trabajos previos sobre adaptación de dominio han sido exitosos. Se ha realizado bastante investigación sobre adaptación de dominio, pero se ha investigado principalmente sobre conjuntos de datos como lo son los de dígitos (por ejemplo, MNIST [9] o USPS[10]) o los de artículos de oficina (por ejemplo Office-31[11]), pero poco sobre otros conjuntos de datos menos conocidos, como por ejemplo los catálogos astronómicos. Por otra parte, existen bastantes trabajos que tratan el problema de clasificación de datos astronómicos utilizando técnicas de aprendizaje automático, pero con poco foco en adaptación de dominio.

Actualmente, existen diversos conjuntos de datos de imágenes astronómicas provenientes de diferentes fuentes, con miles de alertas etiquetadas cada uno. Algunos ejemplos son el *High-cadence Transient Sur-*

vey (*HiTS*), *Dark Energy Survey* (*DES*), *Zwicky Transient Facility* (*ZTF*) y *Asteroid Terrestrial-impact Last Alert System* (*ATLAS*). Con todos estos datos disponibles, y empleando las técnicas de adaptación de dominio profunda, ¿será posible producir un modelo de aprendizaje profundo que generalice a otro dominio, aún cuando se tengan muy pocos elementos etiquetados por clase en el dominio objetivo?

En este trabajo, se aborda la tarea de clasificación de alertas astronómicas en *real* versus *bogus*. Es decir, clasificar correctamente si una alerta corresponde a un objeto astronómico de interés o no, sin preocuparnos si dicho objeto corresponde a una supernova, estrella variable, asteroide, u otro tipo. Se explora el fenómeno del domain shift y se intenta lograr la adaptación de dominio entre 4 conjuntos de datos de alertas astronómicas: HiTS, DES, ZTF y ATLAS, en ambientes donde se posean pocos elementos etiquetados en el conjunto target (*few-shot learning*).

1.2. Hipótesis

Usar técnicas de adaptación de dominio sobre imágenes astronómicas mejora la exactitud de un modelo de aprendizaje profundo en la clasificación de dichas imágenes en real o bogus, cuando se poseen pocos datos etiquetados en el dominio target.

1.3. Objetivo general

Conseguir la adaptación de dominio para la clasificación binaria de alertas astronómicas entre los conjuntos de datos HiTS, DES, ZTF y/o ATLAS, en ambientes donde se tengan pocas etiquetas en el conjunto de datos target, aplicando las técnicas de fine-tuning y el modelo de adaptación de dominio Minimax Entropy.

1.4. Objetivos específicos

- Desarrollar una pipeline de preprocesado para los diferentes conjuntos disponibles (HiTS, DES, ZTF y ATLAS).
- Escoger uno o varios modelos de aprendizaje profundo y de adaptación de dominio y entrenarlos con los conjuntos de datos, considerando casos con pocas etiquetas.
- Evaluar y comparar el desempeño de los modelos en diferentes escenarios, por ejemplo, utilizar los conjuntos de datos HiTS como source y ZTF como target, luego ZTF como source y HiTS como target, y otras combinaciones.

1.5. Resumen de resultados

Utilizando *fine tuning* y una técnica particular de adaptación de dominio denominada *Minimax Entropy*[12], se logró mejorar la exactitud balanceada de clasificación de un modelo de aprendizaje profundo en todas las combinaciones source y target propuestas, con a lo más 40 elementos etiquetados por clase provenientes del conjunto target. Además, este modelo de adaptación de dominio obtiene iguales o mejores resultados sobre el conjunto de datos target en la mayoría de los escenarios comparado con fine tuning, pero a diferencia de este último, lo hace logrando un desempeño mucho más alto y con menos incertidumbre sobre el conjunto source.

Lo que resta de este documento está estructurado de la siguiente forma: en la siguiente sección se presenta el marco teórico, donde se presentan los conceptos de aprendizaje automático y de astronomía necesarios para comprender el resto del documento. A continuación, se presentan algunos trabajos previos. Posteriormente, se describe en detalle cada uno de los conjuntos de datos utilizados. Luego, se describe la metodología: el preprocesado de datos y los modelos de clasificación y de adaptación de dominio usados. Después, el detalle de los experimentos realizados, partiendo por un modelo base seguido de un experimento fine tuning y de adaptación de dominio, con sus respectivos resultados caso por caso. El documento finaliza con las conclusiones y describiendo posibles extensiones a este trabajo.

2. Marco teórico

2.1. Aprendizaje supervisado

El aprendizaje supervisado es una tarea en donde dados un conjunto de datos X de n elementos y un conjunto Y de etiquetas, donde cada etiqueta $y_i \in Y$ está asociada a un elemento $x_i \in X$, se desea producir una aproximación de la función f que recibe como entrada un elemento del conjunto de datos y entregue como resultado la etiqueta correcta, es decir tal que $f(x_i) = y_i$. Para un problema de clasificación con 2 clases, $y_i \in \{0, 1\}$, y se habla de *clasificación binaria*.

Se presupone que la función f puede ser representada por un *modelo*, el cual engloba una variedad de funciones matemáticas diferentes que pueden obtenerse al variar un conjunto de *parámetros* de dicho modelo. Entonces, se utiliza el conjunto de datos para ajustar los parámetros de dicho modelo (esto se conoce como *entrenar* el modelo), de tal forma de producir una función específica $\hat{f}$ que aproxime la función ideal f.

El objetivo acá no es producir una función que simplemente memorice las entradas y entregue las salidas que correspondan, si no que se quiere que la función pueda generalizar a datos nunca antes vistos. Es por esto que al ajustar los parámetros de la función se utiliza un *conjunto de entrenamiento* $X_{\text{train}} \subset X$, mientras que para evaluar el desempeño de dicho modelo se utiliza un *conjunto de prueba* $X_{\text{test}} \subset X$, ambos disjuntos ($X_{\text{train}} \cap X_{\text{test}} = \emptyset$).

El proceso de ajuste de parámetros se realiza mediante un *algoritmo de optimización*, el cual optimizará una *función de costo* $\mathcal{L}$, con tal de encontrar una aproximación suficientemente buena para f. La función de costo, usualmente depende de las etiquetas reales del conjunto de entrenamiento y las etiquetas predichas por el modelo. Para evaluar el desempeño del modelo sobre datos no antes vistos, se utiliza alguna *métrica*, que dependerá de las etiquetas predichas por el modelo para los datos del conjunto de prueba y sus respectivas etiquetas reales asociadas.

Un ejemplo de modelo es la regresión logística. Si suponemos que nuestra función se puede representar como $\hat{f}(x) = \sigma(w^T x) = \hat{y}$, donde $x \in X$, $\hat{y}$ es la etiqueta predicha, w es un vector de parámetros desconocido y $\sigma(z) = 1/(1 + e^{-z})$ (conocida como función *sigmoide*), entonces el ajustar este modelo significa encontrar los parámetros desconocidos de dicho modelo, es decir, w.

Para optimizar los parámetros w se puede utilizar el algoritmo del *descenso estocástico del gradiente* (SGD)[13]. Este es un método de optimización local iterativo, que trabaja muestreando *mini-batches*, k pares (x_i, y_i) del conjunto de entrenamiento, y haciendo una actualización a los parámetros del modelo

en cada paso, según la fórmula:

$$w \leftarrow w - \eta g$$

Donde w son los parámetros del modelo, η es un *hiper parámetro* llamado *learning rate* que controla la magnitud del cambio en cada iteración, y g es un estimado del gradiente de la función de costo con respecto a los parámetros. Una variante de SGD es *momentum*[14], que actualiza los parámetros del modelo pero tomando en cuenta las actualizaciones realizadas previamente, según la siguiente regla:

$$v \leftarrow \alpha v - \eta g$$

$$w \leftarrow w + v$$

donde v es un término que acumula los gradientes de las iteraciones previas, y α es otro hiper parámetro llamado *momentum*.

Una función de costo que se suele utilizar en problemas de clasificación es la *entropía cruzada* definida por

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} -(y_i \log \hat{y_i} + (1 - y_i) \log (1 - \hat{y_i}))$$

.

Donde $y = (y_1, y_2, \ldots y_n)$ corresponde a etiquetas reales e $\hat{y} = (\hat{y_1}, \hat{y_2}, \ldots \hat{y_n})$ a etiquetas predichas por el modelo.

Una métrica muy utilizada para comparar el desempeño de los modelos es la *exactitud* o *accuracy*. Para una clasificación binaria, la exactitud es:

$$ACC = \frac{TP + TN}{TP + FN + TN + FP}$$

donde

- TP: son los *verdaderos positivos*, es decir, los elementos correspondientes a la clase 1 predichos como clase 1 por el modelo.
- TN: son los *verdaderos negativos*, es decir, los elementos correspondientes a las clase 0 predichos como clase 0 por el modelo.
- FP: son los *falsos positivos*, es decir, los elementos correspondientes a la clase 0 predichos como

clase 1 por el modelo.

- FN: son los *falsos negativos*, es decir, los elementos correspondientes a la clase 1 predichos como clase 0 por el modelo.

Otra métrica más balanceada cuando las clases contienen una cantidad distinta de elementos (y también, la utilizada en este trabajo), es la *exactitud balanceada* o *Balanced Accuracy* (en adelante *BACC*), definida como:

$$BACC = \frac{\frac{TN}{TN+FP} + \frac{TP}{TP+FN}}{2}$$

en otras palabras, el promedio entre la tasa de verdaderos positivos y la tasa de verdaderos negativos.

Dada la multitud de arquitectura de modelos, los algoritmos de optimización, las funciones de costo y los datos posibles, existe una gran diversidad de combinaciones al momento de entrenar un modelo, y escoger una combinación en particular al realizar experimentos de aprendizaje automático implica tomar varias decisiones.

2.2. Redes neuronales profundas

Una *red neuronal*[15] es un un modelo que describe una función como un proceso en múltiples pasos. Es decir, se tiene varias operaciones o *capas* con parámetros ajustables, y una relación de dependencia entre dichas capas, donde algunas de ellas reciben como entrada la salida de otras. Las redes neuronales que contienen una cantidad importante de capas se les denomina *profundas*. Dependiendo del tipo de operación que realice cada capa, las redes neuronales pueden recibir un nombre especial. La combinación específica de las capas de una red neuronal se denomina su *arquitectura*. En la figura 2, se muestra un ejemplo de una red neuronal pequeña.

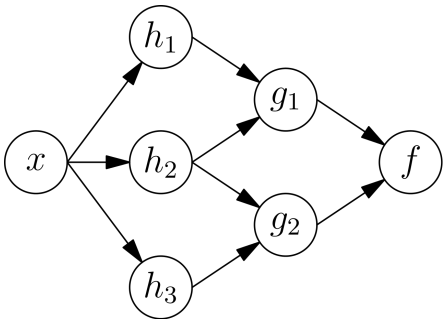


Figura 2: Ejemplo de una red neuronal con 3 capas. Los nodos en la primera capa (h_1, h_2 y h_3), tienen como input un dato de entrada x y como salida funciones de x, $h_1(x)$, $h_2(x)$ y $h_3(x)$. Los nodos en la segunda capa (g_1 y g_2) tienen como entrada resultados de la primera capa, y como salidas $g_1(h_1(x), h_2(x))$ y $g_2(h_2(x), h_3(x))$. La tercera capa tiene 1 solo nodo, que tiene como entradas las salidas de la segunda capa, y como salida $f(g_1(h_1(x), h_2(x)), g_2(h_2(x), h_3(x)))$. Figura extraída de [16].

A continuación se describen algunos tipos de capa:

Lineales (o densas): Estas capas reciben como entrada un vector x d-dimensional, tienen como parámetros una matriz W de $k \times d$ y un vector b k-dimensional (llamado *sesgo* o *bias*) y entregan como salida el vector k-dimensional $Wx + b$. Una red neuronal que sólo posee capas lineales y capas de activación se conoce también como *perceptrón multi capa* o MLP.

Funciones de activación: Una composición de funciones que solamente son lineales es lineal, y dichos modelos son limitados. Una forma de conseguir modelos con no linealidades es mediante capas de funciones de activación. Estas capas reciben un tensor de entrada, de cualquier dimensión y entregan como resultado otro tensor del mismo tamaño al cual se le ha aplicado una función componente a componente. Algunos ejemplos son:

- *ReLU* o *Rectifier Linear Unit*, definida como $f(x) = \max(0, x)$
- *Sigmoide*, descrita anteriormente, definida como $\sigma(x) = 1/(1 + e^{-x})$
- *Softmax*, Está definida para cada elemento de entrada como $\sigma(x_i) = e^{x_i} / \sum_{j=1}^{n} e^{x_j})$

Convolucionales: Estas capas permiten capturar propiedades espaciales de los datos y son particularmente aptas para imágenes. Estas capas reciben una imagen con n canales, poseen $m \times n$ matrices de parámetros llamados *filtros* o *kernels*, un vector de m parámetros llamados *sesgos* o *bias*, y entregan como salida una imagen con m canales. La forma y el orden en que se combinan los diferentes canales de entrada con los diferentes filtros para producir los diferentes canales de salida es similar a una multiplicación de matrices (imaginando que cada canal de entrada pertenece a un vector E, cada filtro pertenece a una matriz K, y que en vez de multiplicar escalares se realiza una convolución). En la figura 4 se muestra un ejemplo.

$$
\begin{aligned}
S_1 &= E_1 \circledast K_{1,1} + E_2 \circledast K_{1,2} \ldots + E_n \circledast K_{1,n} + B_1 \\
S_2 &= E_1 \circledast K_{2,1} + E_2 \circledast K_{2,2} \ldots + E_n \circledast K_{2,n} + B_2 \\
&\vdots \\
S_m &= E_1 \circledast K_{m,1} + E_2 \circledast K_{m,2} \ldots + E_n \circledast K_{m,n} + B_m
\end{aligned}
$$

Figura 3: Ejemplo de capa convolucional. La imagen de entrada posee n canales: $E_1, E_2, \ldots E_n$, la imagen de salida m canales, $S_1, S_2, \ldots S_m$. $B_1, \ldots B_m$ son los bias, y los términos $K_{i,j}$ son los kernels o filtros. La operación $\circledast$ representa una convolución.

Al operar un canal con un filtro, se utiliza una operación llamada *convolución*, de ahí el nombre "convolucional". En esta operación, se multiplica componente a componente el filtro con una submatriz del canal, y luego estos productos son sumados para producir un elemento de una matriz de salida. Los demás elementos son calculados con otras submatrices del canal de entrada, las cuales se obtienen al desplazar vertical y horizontalmente la región seleccionada.

Las capas convolucionales además pueden tener otros parámetros:

- *Tamaño del kernel*: ancho y alto K_w, K_h de los filtros.
- *Stride*: desplazamiento horizontal y vertical S_w, S_h para obtener las siguientes matrices al realizar la convolución.
- *Padding*: relleno horizontal y vertical P_w, P_hcon el fin de poder aplicar las convoluciones en los bordes de las imágenes.
- *Dilation*: Separación horizontal y vertical D_w, D_h de los elementos de la submatriz del canal de entrada.

Así, si el ancho y alto de los canales de entrada es W_{in} y H_{in} respectivamente, el tamaño de los canales de salida está dado por:

$$W_{out} = \frac{\lfloor (W_{in} + 2P_w - D_w(K_w - 1) - 1)}{S_w} + 1 \rfloor$$
$$H_{out} = \frac{\lfloor (H_{in} + 2P_h - D_h(K_h - 1) - 1)}{S_h} + 1 \rfloor$$

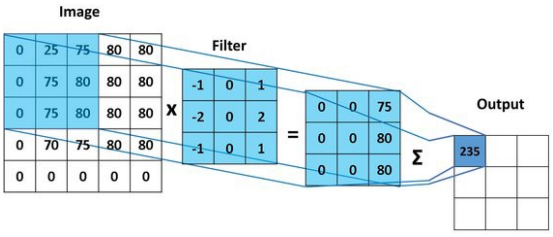


Figura 4: Ejemplo de operación de convolución con 1 imagen de entrada y 1 de salida. Una submatriz de 3×3 de la imagen de entrada es multiplicada componente a componente con el filtro de 3×3, y dichos productos sumados para producir 1 escalar. Al repetir dicho proceso con las demás submatrices se obtiene una matriz de 3×3.

Una red neuronal con capas convoluciones se dice que es una *red neuronal convolucional*, abreviado CNN[17], por su sigla en inglés (Convolutional Neural Network).

Max pooling: Una capa de *max pooling*[18] es una operación que entrega como resultado los mayores elementos de varias subregiones de los elementos de entrada. La versión 2D de max pooling que se suele usar junto con las capas convolucionales en las CNN, recibe una matriz como entrada, y entrega otra matriz como salida. Por ejemplo, si la matriz de entrada es de 6×6 y el pooling es de 2×2, la matriz de salida es de 3×3, donde cada elemento es el máximo de una subregión disjunta de 2×2 de la matriz de entrada.

Batch normalization: Las capas de *batch normalization*[19] actúan como una corrección de las medias y varianzas de las salidas. Estas capas tienen 2 parámetros γ y β. La versión 1D de batch normalization recibe como entrada un vector, mientras que la versión 2D recibe como entrada una matriz, y en ambos casos la salida es $\gamma\hat{x} + \beta$, donde $\hat{x} = (x - \mu_B)/\sqrt{\sigma_B^2}$, siendo μ_B la media del batch y σ_B^2 la varianza del mismo.

Dropout: Las capas de *dropout*[20] reciben una entrada y entregan una salida del mismo tamaño, pero con la salvedad de que cada elemento de entrada puede ser anulado con una probabilidad p. Esto fuerza a que no sean sólo una pocas entradas las que el modelo utilice para realizar predecir su salida correctamente, si no que todas ellas sean importantes, pues no se sabe de antemano cuales serán desactivadas por la capa, y además cambian durante el entrenamiento. Estas capas existen en versión 2D también.

En general, para las redes neuronales profundas, es intratable obtener una fórmula explícita que minimice la función de costo. En su lugar, se utiliza algoritmos iterativos como el SGD o sus variantes, optimizando típicamente la función de costo de entropía cruzada. Estos algoritmos requieren las derivadas de la función a optimizar con respecto a los parámetros, pues estas indican la dirección en la que la función de costo varía más rápido. Afortunadamente, el algoritmo de *retropropagación*[21] provee una forma recursiva de calcular estas derivadas, permitiendo calcular todas estas derivadas en un solo barrido lineal desde las últimas capas de la red hacia la primera.

Oversampling *Oversampling*[22] es una forma de lidiar con clases desbalanceadas, es decir cuando las clases del conjunto de datos no poseen la misma cantidad de datos. Las clases desbalanceadas son un problema, pues pueden producir *sobre ajuste* (*overfitting*), de la clase mayoritaria. Para una clasificación binaria, oversampling funciona muestreando los elementos de la clase minoritaria más de una vez, de tal forma de balancear ambas clases.

Early Stopping *Early Stopping* es otra técnica para reducir el overfitting. Este método requiere un conjunto de *validación*, diferente al conjunto de entrenamiento y de prueba. Early stopping consiste en

detener el entrenamiento cuando la métrica de evaluación aplicada sobre el conjunto de validación no mejore en una cierta cantidad de iteraciones (hiper parámetro llamado *paciencia*), incluso cuando la misma métrica sobre el conjunto de entrenamiento pueda seguir mejorando. Así, el modelo se detiene antes de que su capacidad de generalizar a datos no vistos previamente se vea perjudicada.

2.3. Adaptación de Dominio

Dados un conjunto de datos $\mathcal{D}^s$ denominado *dominio source* y otro conjunto de datos $\mathcal{D}^t$ denominado *dominio target*, el domain shift se presenta cuando las distribuciones $p(x \in \mathcal{D}^t)$ y $p(x \in \mathcal{D}^s)$ son distintas. La adaptación de dominio es el desafío de conseguir que un modelo entrenado principalmente con datos de source prediga correctamente sobre el dominio target.

Dependiendo de la disponibilidad de datos etiquetados de target, se pueden tener los siguientes escenarios:

Semi supervisado: en esta configuración, se poseen pocos datos etiquetados de target, y posiblemente muchos datos no etiquetados.

No supervisado: en esta configuración, solo se poseen datos no etiquetados del dominio target.

Uno de los enfoques de la adaptación de dominio son los métodos basados en *entrenamiento adversario*. Estos métodos utilizan una función objetivo que involucra, por una parte minimizar una función de costo y al mismo tiempo maximizar parte de la misma función de costo, alternadamente. Este proceso de maximización y minimización en ocasiones se puede simular utilizando una función de costo que contenga todos los términos involucrados, pero donde algunos de los términos han pasado por una capa de *inversión de gradiente* o *GRL*.

GRL: Gradient Reversal Layer [23] Este es un tipo especial de capa, cuya única función es entregar la misma entrada como salida, pero al momento de entregar el gradiente lo hacen con signo opuesto. Si un término $\mathcal{L}$ de la función de costo se quiere minimizar y dicho término ha sido calculado pasando por exactamente 1 capa GRL, los parámetros de todas las capas entre la GRL y la función de costo serán actualizados minimizando $\mathcal{L}$, como es usual. Sin embargo, los parámetros entre la entrada inicial de la red y la capa GRL recibirán un gradiente con signo opuesto, por lo que se actualizarán maximizando $\mathcal{L}$. Así, es posible construir un modelo que realice un entrenamiento adversario utilizando una sola arquitectura y una sola función de costo, siempre y cuando la ubicación de las capas GRL sea la adecuada. Es posible tener además varios caminos, para calcular diferentes términos de la función de costo, algunos que pasen por capas GRL y otros que no.

2.4. Astronomía

2.4.1. Surveys

Existe una cantidad de telescopios a lo largo del mundo que están sistemáticamente y constantemente monitoreando el cielo. La *Dark Energy Camera* o *DeCam*[24], es una cámara montada en el telescopio Victor M. Blanco ubicado en el Cerro Tololo, cerca de La Serena, Chile. Esta es responsable del *Dark Energy Survey*, de donde proviene el conjunto de datos denominado DES en este trabajo. La *Zwicky Transient Facility* o *ZTF*[25] es una survey proveniente del telescopio Schmidt del observatorio Palomar, ubicado en San Diego, California. De esta survey proviene el conjunto denominado ZTF en este trabajo.

2.4.2. Taxonomía de alertas

En este trabajo se utiliza la nomenclatura y taxonomía descrita en [26], donde las alertas son clasificadas en *real* versus *bogus*. Una alerta etiquetada como "bogus" corresponde a una alerta que es o bien una falsa alarma o un *artifacto*[1], mientras que una "real" puede (dependiendo del conjunto de datos) sub clasificarse en las siguientes categorías:

Transientes: Objetos tales como supernovas tipo Ia, Ib, Ic, II.

Periódicos: Objetos tales como estrellas cefeidas, RR lyrae y binarias eclipsantes.

Estocásticos: Objetos tales como núcleos galácticos activos (AGN) blazars y cuásars.

Para este trabajo, se aborda la clasificación *binaria* de alertas en real versus bogus, sin preocuparse de la subclasificación de reales, dado que no todos los conjuntos de datos utilizados en este trabajo[2] contienen toda la información asociada a las subcategorías.

[1] Un artefacto es un efecto visual causado por rayos cósmicos, halos estelares, fluctuaciones del fondo, píxeles defectuosos, astrometría imprecisa, entre otros

[2] véase sección 4

3. Trabajos previos

3.1. Adaptación de Dominio

El trabajo de [27] recopila trabajos de *transfer learning* (TL), clasificando la adaptación como un caso especial de transfer learning. Se define la adaptación de dominio como aprender la misma tarea pero aplicada a dominios distintos *source* y *target*, en contraste con el caso general de transfer learning donde la tarea a aprender no necesariamente es la misma. El trabajo de [28] presenta una recopilación de trabajos previos específicamente de adaptación de dominio para aplicaciones visuales. En él, se propone una taxonomía de métodos, clasificándolos en métodos basados en *discrepancia*, métodos basados en *adversarios* y métodos basados en *reconstrucción*. Por otra parte, estos métodos se pueden clasificar según la disponibilidad de etiquetas en el dominio target, siendo *supervisados* si es que todas las etiquetas son conocidas, *semi supervisados* cuando solo se tienen algunas, y *no supervisados* cuando no hay ninguna.

Los métodos basados en discrepancia, en general, realizan *fine tuning* (esto es, realizar un entrenamiento a partir de un modelo pre entrenado) para disminuir el shift entre los dominios. El criterio a usar para determinar que tan bien alineados están ambos dominios da a lugar a distintos métodos. Como ejemplos de criterios se puede mencionar Maximum Mean Discrepancy (MMD[29]) o (deep) Correlation Alignment (CORAL[30][31]).

Los métodos basados en adversario, pueden ser modelos generativos, o no generativos. Los primeros se basan en las GAN[4], mientras que los segundos definen una función de costo que intenta confundir los dominios, mediante entrenamiento adversario.

Finalmente, los modelos basados en reconstrucción se valen de modelos para reconstrucción para confundir los dominios. Estos pueden ser basados en autoencoders, o en adversarios, por ejemplo, usando cycleGAN[32].

En [23] se propuso la DANN, una red que incorpora el entrenamiento adversario en su arquitectura sin necesidad de tener una red separada, mediante una capa de inversión de gradiente o *GRL*.

En [33] se propone un método basado en las CGAN, aplicado a clasificación de artículos de oficina y dígitos. En [34] se propone Adversarial Discriminative Domain Adaptation, pero se aplica a datos de dígitos. El trabajo de [35] combina una red pre entrenada sobre ImageNet[36] con Minimax Entropy (MMD) y Domain Specific Batch Normalization.

En [12] se propone un método que funciona con few-shot, aplicado sobre DomainNet y Office-Home. Por otra parte, [37] sugiere usar un enfoque encoder-decoder, y proponen un método que funciona en el

caso supervisado y una extensión para el caso semi supervisado.

En [38] se utiliza un método que adopta una métrica que minimiza la distancia intra clase mientras maximiza la distancia inter clase en el espacio latente de los datos. Este método funciona incluso en los casos semi supervisados, sin embargo, es aplicado a datos de dígitos, artículos de oficina y el dataset VISDA-C[39], un dataset que contiene imágenes reales y modelos 3D de diferentes objetos.

El trabajo de [40] propone una técnica novedosa que utiliza consistencia de ciclo, dando muy buenos resultados en clasificación de dígitos y segmentación de imágenes.

Existen generalizaciones de adaptación de dominio. *Partial Domain Adaptation*[41], es una configuración en la cual el espacio de etiquetas del dominio target es un subconjunto del espacio de etiquetas del dominio source. *Open Set Domain Adaptation*[42], es más general aún, pues se trata del caso cuando los espacios de etiquetas se intersectan en solo algunos elementos, pudiendo haber etiquetas exclusivas de source y de target. *Universal Domain Adaptation* va más allá, pues en esta configuración el espacio de etiquetas es totalmente desconocido. Sin embargo, estos casos están fuera del alcance de este trabajo.

3.2. Astronomía

Trabajos de clasificación real versus bogus usando aprendijaze automático en astronomía datan de 2007[43]. Este primer trabajo utilizó técnicas de aprendijaze automático como las Support Vector Machines y Boosted Decision Trees, y fue aplicado a datos de la Nearby Supernova Factory. Otros trabajos realizan la tarea de clasificación usado datasets distintos: Para Pan-STARRS1[44] se han usado redes neuronales, Support Vector Machines y Random Forests, para la *Dark Energy Survey* (DES)[45] se ha usado random forest, para *Zwicky Transient Facility* (ZTF) se ha usado redes neuronales convolucionales[46] y para el dataset *High-cadence Transient Survey* (HiTS)[47][48] también se usan redes neuronales convolucionales. Para este último dataset, se han empleado redes convolucionales recurrentes (RCNN) para clasificar secuencias de imágenes ya no sólo en real y bogus, si no que en más clases: Supernovas, RR Lyrae, Cefeidas, Binarias Eclipsantes, Objetos no variables, Galaxias y Asteroides[49].

Existen también algunos trabajos de adaptación de dominio para Astronomía. En[50], en el cual se trabaja con datasets Magellanic Cloud, Small Magellanic Cloud, y M33, la clasificación corresponde a identificar subclases de estrellas cefeidas, en vez de llevar a cabo la clasificación binaria real vs bogus. En [7] se utilizan los datos del Supernova Photometric Classification Challenge: el dominio source corresponde a datos espectroscópicos y el target corresponde a datos fotométricos. Sin embargo, la tarea corresponde específicamente a identificar Supernovas del tipo Ia. En [51], se ataca el problema de distinguir las supernovas tipo Ia de las Ib y Ic. Sin embargo, se ocupan datos simulados, aproximados a los

del conjunto de datos DES.

En [52] se aplica adaptación de dominio, pero para la clasificación de objetos astronómicos en galaxias esferoidales, discoidales o irregulares, en fuentes puntiformes y en artefactos de la imagen. En [53], se propone un modelo de adaptación de dominio y se aplica a los conjuntos de datos CANDELS y CLASH, pero nuevamente estos corresponden a clasificación morfológica de galaxias.

Buscando exhaustivamente en la literatura científica, no se encontraron trabajos previos que se enfoquen en adaptación de dominio propiamente tal para los conjuntos de datos HiTS, DES, ZTF y ATLAS en conjunto, ni para entornos supervisados ni semi supervisados.

4. Conjuntos de Datos

Para este trabajo, se utilizaron 4 conjuntos de datos. Cada uno de ellos contiene miles de alertas, donde cada alerta corresponde a una *tripleta* de imágenes con 1 solo canal de color cada una (monocromáticas). Cada tripleta viene acompañada de una etiqueta de clase, las cuales son reasignadas a "real" o "bogus". Las 3 imágenes de cada tripleta son referidas en este trabajo con los siguientes nombres:

Template: Imagen del cielo en un instante de tiempo de referencia.

Science: Imagen de cómo se ve un objeto en el cielo en un instante de tiempo posterior.

Difference: Imagen que contiene la diferencia entre *Science* y *Template*.

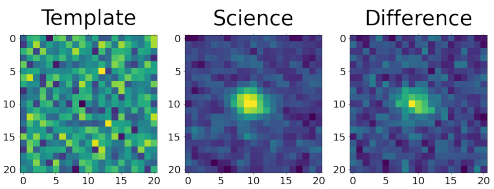
A continuación, se describe cada conjunto de datos en detalle.

4.1. HiTS: High-cadence Transient Survey

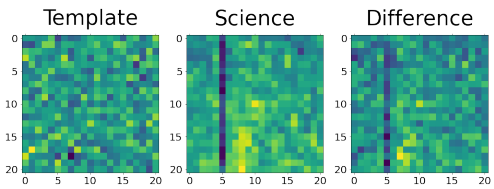
Este conjunto de datos corresponde a un catálogo proveniente de la Dark Energy Camera [24]. Los datos crudos vienen en formato FITS[54], y contienen 4 imágenes por cada alerta: "Template" (llamado `temp_images` originalmente) "Science" (`sci_images`), "Difference" (`diff_images`) y "Ratio Señal/Ruido". Esta cuarta imagen, al no estar presente explícitamente en los conjuntos DES y ZTF o ATLAS, es ignorada en este trabajo.

Las imágenes Template, Science y Difference de cada tripleta de HiTS poseen un tamaño de 21×21 píxeles, almacenadas en valores de punto flotante. El conjunto de datos posee 718.842 tripletas etiquetadas como "real" y 718.842 tripletas etiquetadas como bogus. Así, este conjunto de datos tiene sus clases completamente balanceadas, y posee un total de 1.437.684 tripletas etiquetadas.

En la figura 5a se ve un ejemplo de una tripleta "real" de HiTS, y en la figura 5b un ejemplo de una tripleta "bogus" de HiTS, ambas en su forma original. Los datos originales se pueden obtener en el enlace al pié de página[3].



(a) Ejemplo real de HiTS.



(b) Ejemplo bogus de HiTS.

Figura 5: Muestras del conjunto de datos HiTS.

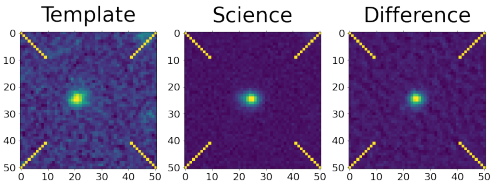
[3]HiTS: `http://archiving.cmm.uchile.cl/pub/DeepHits/all_chunks.tar.gz`

4.2. DES: Dark Energy Survey

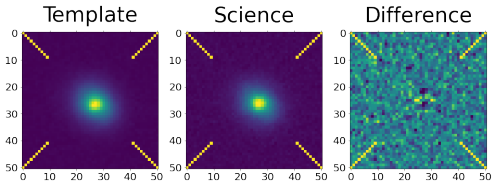
Este conjunto de datos corresponde a otro catálogo proveniente de la Dark Energy Camera. Los datos crudos vienen en formato FITS y también en formato GIF[55] (no animado). Contienen 3 imágenes por cada alerta: "Template" (llamado `temp` originalmente) "Science" (`srch`) y "Difference" (`diff`).

Las imágenes Template, Science y Difference de cada tripleta de DES poseen un tamaño de 51×51 píxeles, almacenadas en valores de punto flotante en los archivos FITS y almacenadas en valores de tipo entero en los archivos GIF. El conjunto de datos posee 454.092 tripletas etiquetadas como "real" y 444.871 tripletas etiquetadas como bogus. Así, este conjunto de datos tiene sus clases casi balanceadas, y posee un total de 898.963 tripletas etiquetadas.

En la figura 6a se ve un ejemplo de una tripleta "real" de DES, y en la figura 6b un ejemplo de una tripleta "bogus" de DES, ambas en su forma original. Los datos originales se pueden obtener en el enlace al pié de página[4].



(a) Ejemplo real de DES.



(b) Ejemplo bogus de DES.

Figura 6: Muestras del conjunto de datos DES.

4.3. ZTF: Zwicky Transient Facility

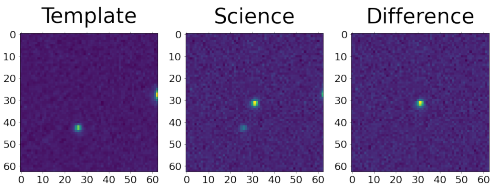
Este conjunto de datos corresponde a una muestra de un catálogo proveniente de la Zwicky Transient Facility. Los datos crudos vienen en formato numpy array[56]. Contienen 3 imágenes por cada alerta: "Template" (llamado `temp` originalmente) "Science" (`srch`) y "Difference" (`diff`).

Las imágenes Template, Science y Difference de cada tripleta de ZTF poseen un tamaño de 63×63 píxeles en su mayoría, pero existen 467 tripletas que poseen tamaño menor. Todas las imágenes vienen almacenadas en valores de punto flotante, y posiblemente pueden contener valores inválidos (NaN) en uno o más píxeles. El conjunto de datos posee 9.996 tripletas etiquetadas como "AGN (Núcleo Galáctico Activo)", 1.079 tripletas etiquetados como "SN (Supernova)", 9.938 tripletas etiquetadas como "VS (Estrella Variable)", 9.899 tripletas etiquetadas como "asteroid (Asteroide)" y 5.350 tripletas etiquetadas como "bogus". Para este trabajo se agrupa AGN, SN, VS y asteroid como "real", dando un total de 30.912

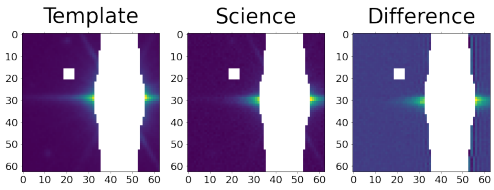
[4]DES: `https://portal.nersc.gov/project/dessn/autoscan/`

tripletas etiquetadas como real y las 5.350 tripletas etiquetadas como bogus originales. Así, este conjunto de datos tiene sus clases desbalanceadas, y posee un total de 36.262 tripletas etiquetadas.

En la figura 7a se ve un ejemplo de una tripleta "real" de ZTF, y en la figura 7b un ejemplo de una tripleta "bogus" de ZTF, ambas en su forma original. Los datos originales se pueden obtener en el enlace al pié de página[5].



(a) Ejemplo real de ZTF.



(b) Ejemplo bogus de ZTF.

Figura 7: Muestras del conjunto de datos ZTF.

4.4. ATLAS: Asteroid Terrestrial-impact Last Alert System

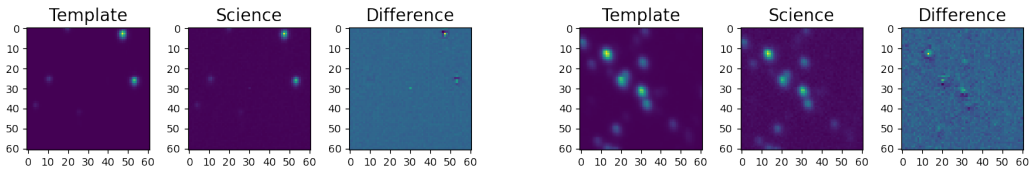
Este conjunto de datos corresponde a un catálogo proveniente del Asteroid Terrestrial-impact Last Alert System. Los datos se obtuvieron crudos y vienen en formato fits exportados mediante la librería pickele de Python. Este conjunto de datos contenía originalmente 2 imágenes por cada alerta: "Science" y "Difference". La imagen "Template" se calculó restando Diff a Science.

Las imágenes Science y Difference de cada tripleta de ATLAS poseen unn tamaño de 61×61 píxeles. El conjunto de datos posee 500 imágenes etiquetadas como "burn", 500 imágenes etiquetadas como "cr", 500 imágenes etiquetadas como "kast", 500 imágenes etiquetadas como "noise", 500 imágenes etiquetadas como "spike" y 678 imágenes etiquetadas como "streak". Para este trabajo, se agrupan cr y kast como "real", dando un total de 1.000 imágenes etiquetadas como real mientras que las 2.678 imágenes restantes son tratadas como bogus. Así, este conjunto de datos tiene sus clases desbalanceadas, y posee un total de 3.678 imágenes etiquetadas.

En la figura 8a se ve un ejemplo de una tripleta "real" de ATLAS, y en la figura 8b un ejemplo de una tripleta "bogus" de ATLAS, ambas en su forma original. Los datos originales se pueden obtener en el enlace al pié de página[6].

Si bien los cuatro conjuntos de datos tienen varias características en común, es necesario preprocesarlos para poder trabajar con todos ellos en una representación común.

[5] ZTF: `https://drive.google.com/drive/folders/1MHaSJu8GrhLu5dsabwjN98nMItaKrsV8`
[6] ATLAS: `https://droppy.alerce.online/#/training_sets/ATLAS`



(a) Ejemplo real de ATLAS.

(b) Ejemplo bogus de ATLAS.

Figura 8: Muestras del conjunto de datos ATLAS.

5. Metodología

5.1. Experimentos

Los experimentos realizados siguen la secuencia de pasos descrita a continuación.

1. **Preprocesamiento:** Dejar todas las alertas de cada uno de los catálogos en una representación común, donde cada alerta contenga la misma cantidad de imágenes y en la misma resolución, y con una etiqueta binaria asociada.

2. **Particionado:** Particionar cada uno de los conjuntos de datos en conjuntos de entrenamiento, conjuntos de validación y conjuntos de prueba, aleatoriamente. El conjunto de entrenamiento de cada catálogo utiliza un 80 % de sus alertas, el conjunto de validación un 10 % y el de prueba el 10 % restante, y ningún par de estos conjuntos tiene etiquetas en común.

3. **Experimento *baseline*:** Este experimento tiene como propósito verificar que el fenómeno del domain shift está presente cuando se entrena un modelo de aprendizaje profundo usando uno de los 4 conjuntos de datos presentados, pero se evalúa dicho modelo sobre conjunto de datos. Se toman 4 instancias de un modelo de aprendizaje profundo, y se entrena cada uno con el conjunto de entrenamiento de un conjunto "source" distinto: uno usando HiTS, otro usando DES, otro usando ZTF y otro usando ATLAS. Luego, para cada uno de los modelos ajustados resultantes, se evalúa su desempeño sobre el conjunto de prueba de cada uno de los 4 conjuntos de datos "target" disponibles: uno usando HiTS, otro usando DES, otro usando ZTF y otro usando ATLAS (16 evaluaciones en total).

4.a **Experimento *fine tuning few-shot*:** En este experimento se evalúa si es que un modelo es capaz de clasificar correctamente en un conjunto de datos target, usando pocos datos. Para esto, se toma

un modelo previamente entrenado con un conjunto de datos source del experimento "baseline", y se continúa entrenando dicho modelo, pero usando una cantidad n de elementos etiquetados o *shots* por cada clase. Estos n elementos son muestreados del conjunto de entrenamiento de target. Luego se evalúa el comportamiento de este nuevo modelo, sobre los conjuntos de prueba de source y de target. Esto se realiza para cada combinación source-target posible (12 pares source-target en total). Por ejemplo, para el caso source HiTS y target ZTF y 1 shot, se toma el modelo que ya fue entrenado con el conjunto de entrenamiento de HiTS, y se sigue entrenando usando 1 tripleta real con 1 tripleta bogus del conjunto de entrenamiento de ZTF, y para el modelo resultante se evalúa su desempeño sobre los conjuntos de prueba de HiTS y ZTF.

4.b **Experimento de adaptación de dominio con MME:** Este experimento es similar a fine tuning, con la diferencia de que el modelo usado es uno de adaptación de dominio profunda. La forma de entrenar este modelo es diferente, pues trabaja con tripletas etiquetadas de source, n tripletas etiquetadas de target por clase, y tripletas no etiquetadas de target. Se prueba también todas las combinaciones posibles source y target.

La razón por la que tanto en fine tuning como en MME se mide separadamente el BACC sobre un conjunto de prueba proveniente de source y otro proveniente de target, es que se desea medir no solamente el desempeño sobre target utilizando pocos datos etiquetados para el entrenamiento, si no que además se quiere evaluar cuanto afectan estos entrenamientos sobre el desempeño en el conjunto de datos source. En otras palabras, se quiere analizar si es que continuar entrenando los pesos obtenidos por el modelo baseline sobre source empeora al realizar fine tuning o entrenamiento adversario con MME.

Con el propósito de evaluar la significancia estadística de nuestro enfoque, cada uno de los pasos descritos anteriormente se realiza 10 veces (excepto el preprocesamiento, que se realiza solamente 1 vez). Esto da a lugar a 10 instancias de cada experimento por cada par source-target para baseline, y 10 instancias de cada experimento por cada tripleta source-target-shots para fine tuning y MME. Para fine tuning y MME y con el fin de analizar su comportamiento en función de la cantidad de elementos etiquetados de target, se utiliza 1, 5, 10, 20 y 40 shots por cada par source-target.

El total de instancias del modelo baseline es (10 particionados) $\times$ (4 conjuntos source posibles) = (40 instancias), y cada modelo es evaluado en los 4 conjuntos de datos posibles, dando un total de $40 \times 4 =$ 160 evaluaciones.

El total de instancias del modelo fine tuning es (10 particionados) $\times$ (4 conjuntos source posibles) $\times$ (3 conjuntos target posibles) $\times$ (5 cantidades de shots posibles) = (720 instancias), y cada modelo es

evaluado en sus respectivos conjuntos source y target, dando a lugar a 720 × 2 = (1.440 evaluaciones).

El total de instancias del modelos MME es (10 particionados) × (4 conjuntos source posibles) × (3 conjuntos target posibles) × (5 cantidades de shots posibles) = (720 instancias), y cada modelo es evaluado en sus respectivos conjuntos source y target, dando a lugar a 720 × 2 = (1.440 evaluaciones).

5.2. Preprocesamiento

Para dejar las imágenes de todos los conjuntos de datos con las mismas características, se aplicó la siguiente secuencia de preprocesamientos (en el siguiente orden):

1. **Filtro**: Si bien la mayoría de los datos son imágenes cuadradas, algunas tienen menos filas o columnas. Las tripletas con al menos 1 imagen no cuadrada fueron descartadas. Para el conjunto de datos ZTF, esto resultó en eliminar 467 tripletas, bajando de las 36.262 originales a 35.795. Para DES y ZTF este paso no tuvo efecto, pues todas sus tripletas ya eran cuadradas.

2. **Recorte**: Se recortaron los 21 × 21 píxeles centrales de cada imagen, descartando los restantes. Para el conjunto de datos HiTS, este paso no tiene efecto, pues sus imágenes ya vienen en esa dimensión.

3. **Normalización**: Se desplazaron y escalaron las intensidades de los píxeles de las imágenes, de tal manera que cada canal (template, science y difference) de cada imagen tenga media 0 y desviación estándar 1. Para esto sólo se usaron los píxeles válidos (no NaN o inf) de cada canal.

4. **Rellenado**: Se completaron los píxeles inválidos de cada canal con la *mediana* de los píxeles válidos de dicho canal. En el caso de que luego de todos los pasos anteriores no haya quedado ni un solo píxel válido, para algún canal, se rellena con cero (ver figura 10 para un ejemplo de este caso extremo).

De esta forma, cada alerta después del procesamiento es un par imagen-etiqueta, donde la imagen tiene 3 canales (Template, Science y Difference) y resolución de 21 × 21 píxeles, y la etiqueta es un valor 0 ó 1. La secuencia completa de preprocesado sobre una muestra bogus del dataset ZTF se puede ver en la figura 9.

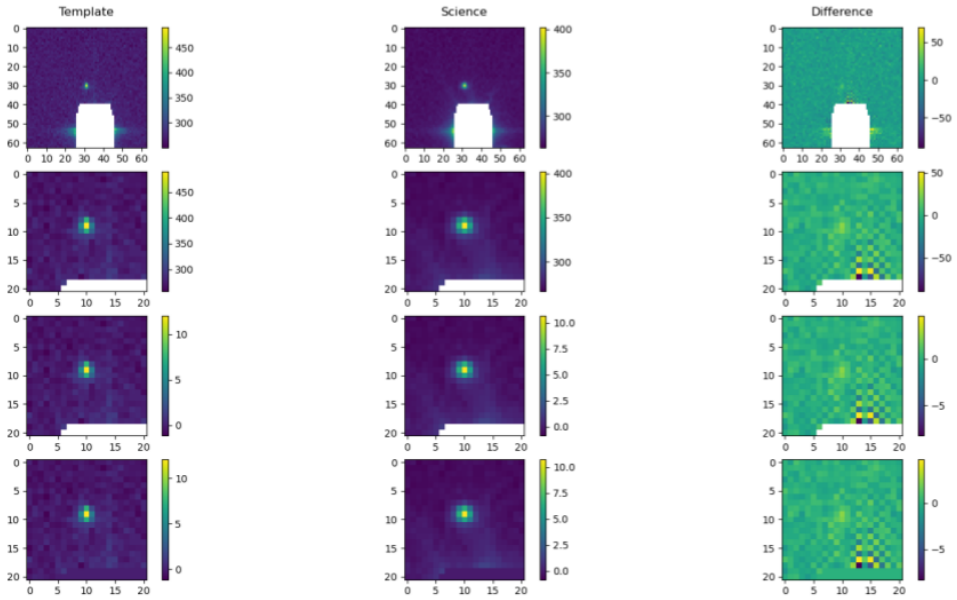


Figura 9: Preprocesamiento de datos. De arriba hacia abajo: Tripleta de imágenes original, tripleta recortada, tripleta normalizada, tripleta rellenada.

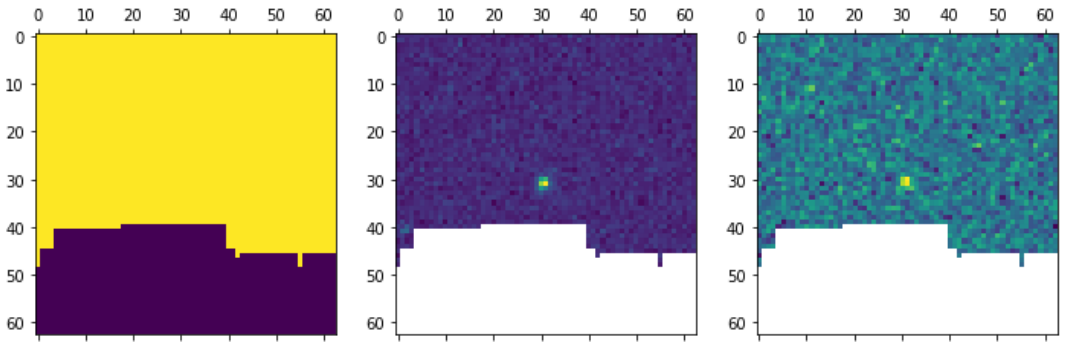


Figura 10: Caso extremo correspondiente a un real de ZTF (de izquierda a derecha: Template, Science y Difference). Los píxeles amarillos corresponden a valores casi infinitos (-3.4028235e+38). En el canal de más a la izquierda, el recorte central de la imagen deja ninguno o muy pocos píxeles validos, que en el paso de rellenado se convierten en regiones planas grandes. Hay 72 imágenes de ZTF con este problema, 38 real y 34 bogus. HiTS, DES y ATLAS no presentan este problema.

5.3. Modelos

5.3.1. Modelo base

El modelo base para la clasificación es una red neuronal convolucional siguiendo la arquitectura descrita en[23].

El modelo consta de las siguientes capas:

1. Convolución 2D, 3 canales de entrada, 64 canales de salida, tamaño de kernel de 5×5.
2. Batch normalization.
3. MaxPool2d(tamaño de kernel de 2×2).
4. Función de activación ReLU.
5. Convolución 2D, 64 canales de entrada, 50 canales de salida, tamaño de kernel de 5×5,
6. Batch Normalization.
7. Dropout 2D, con probabilidad 0.5.
8. Max Pool 2D, tamaño de kernel de 2×2.
9. Lineal, 200 neuronas de entrada, 100 neuronas de salida.
10. Batch normalization 1D
11. Función de activación ReLU.
12. Dropout 2D, con probabilidad 0.5.
13. Lineal, 100 neuronas de entrada, 100 neuronas de salida.
14. Batch Normalization 1D.
15. Función de activación ReLU.
16. Lineal 100 neuronas de entrada, 2 neuronas de salida.
17. Función de activación softmax.

Una representación visual de este modelo, omitiendo funciones de activación, dropout y batch normalization se puede ver en la figura 11. Este modelo al hacer inferencia recibe un tensor de tamaño $N \times 3 \times 21 \times 21$ y entrega otro de tamaño $N \times 2$, con las probabilidades predichas para las clases bogus y real de cada tripleta de entrada.

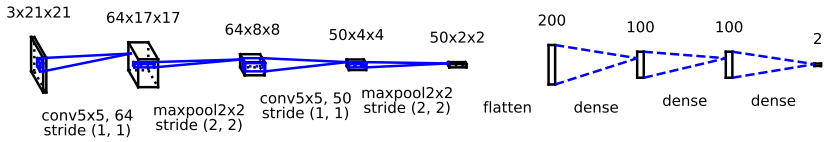


Figura 11: Arquitectura de la red convolucional. Figura dibujada utilizando la herramienta convnet-drawer[57]

5.3.2. Fine tuning

La arquitectura base descrita en la subsección anterior es la misma que se utiliza para el entrenamiento fine tuning. La diferencia entre "baseline" y fine tuning yace en la forma en que se entrenan ambos modelos. El modelo baseline se entrena desde cero utilizando un conjunto de datos source. Fine tuning, por su parte, si bien utiliza la misma arquitectura de red neuronal que el modelo base, comienza con sus parámetros inicializados con los valores resultantes del modelo baseline para algún conjunto source, y continúa ajustando dichos parámetros utilizando una cantidad limitada de datos etiquetados del conjunto target. En la figura 12 se muestra una descripción visual de este proceso.

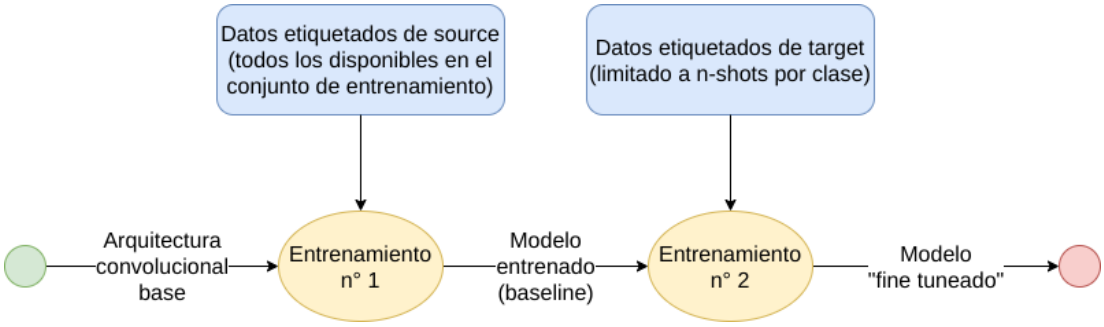


Figura 12: Modelo fine tuning.

5.3.3. MME

El modelo de adaptación de dominio usado en este trabajo es el propuesto en [12]. Este modelo es en realidad un meta-modelo, que puede utilizar diferentes modelos por debajo. Al entrenar este modelo, los datos pasan por el extractor de características, que puede constar de cualquier cantidad y combinación de capas. En este punto, los datos pueden o no pasar por una capa de inversión de gradiente.

Luego, los datos son normalizados, pasados por una capa lineal, que es dividida por un parámetro de *temperatura* T, es decir, son transformados de acuerdo a la siguiente fórmula:

$$v_{out} = \frac{W\left(\frac{v_{in}}{\|v_{in}\|_2}\right)}{T}$$

donde v_{in} es la salida del extractor de características para una entrada x (vector D dimensional), W es la matriz de pesos de la capa lineal (dimensión $2 \times D$), y T es un escalar llamado temperatura. La idea de esta operación es llevar la representación latente de cada entrada del modelo a una esfera unitaria de D dimensiones, y que en la matriz W existan *vectores representantes* de cada una de las 2 clases. Al hacer el producto punto entre el vector v_{in} con cada fila de W se está calculando una similitud entre dichos operandos basada en su dirección: mientras más similar más positivo será el resultado para dicha clase, y mientras más se encuentre en una dirección opuesta, más negativo será el resultado del producto punto. El parámetro de temperatura entonces, cuando su valor es < 1 tiene la función de exagerar la separación entre ambas similitudes.

Finalmente, los vectores v_{out} son pasados por una capa softmax, con el fin de extraer valores válidos de probabilidad para cada clase.

La función de costo que se utiliza con esta arquitectura cuenta con 2 términos: un término supervisado y otro no supervisado. El término supervisado corresponde a una entropía cruzada, que se computa utilizando batches compuestos en su mitad por datos etiquetados de source y en su otra mitad por datos etiquetados de target. Por otra parte, el término no supervisado corresponde a una entropía, la cual es computada utilizando un batch compuesto solamente de datos de target no etiquetados.

La función de costo final es:

$$\mathcal{L} = H(y_s, \hat{y_s}) - \lambda H(\hat{y_u})$$

donde $H(y_s, \hat{y_s})$ es la entropía cruzada entre las etiquetas reales del batch supervisado y las etiquetas predichas por el modelo, $H(\hat{y_u})$ es la entropía del batch no supervisado y λ es un hiper parámetro que controla la importancia de la entropía en esta función de costo.

El batch supervisado no pasa por la capa GRL, mientras que el batch no supervisado si lo hace. Esto quiere decir, que este modelo minimiza la entropía *cruzada* en todo momento, con lo cual se intenta minimizar el error de clasificación. Por otra parte, la entropía es minimizada solo en extractor de características (con el fin de acercar los vectores en el espacio de características a los vectores representantes) y maximizada en el clasificador del modelo (con el fin de ajustar los vectores representantes a valores mas centrados dentro de su clase). De esta forma, al usar la función de costo $\mathcal{L}$ en conjunto con la capa GRL estamos efectivamente realizando un entrenamiento adversario, en una sola red neuronal.

La forma de entrenar este modelo es alternando batches supervisados y no supervisados. En la figura 13 se muestra una representación visual de este modelo.

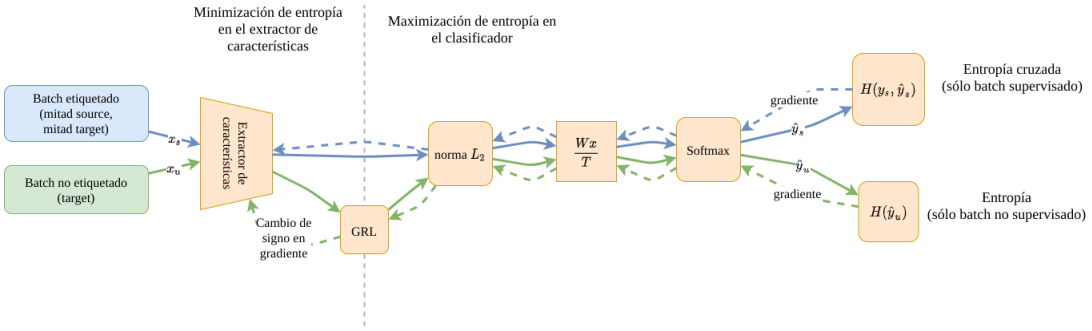


Figura 13: Arquitectura MME.

El extractor de características de este modelo consta de las siguientes capas:

1. Convolución 2D, 3 canales de entrada, 64 canales de salida, tamaño de kernel de 5×5.
2. Batch normalization.
3. MaxPool2d, tamaño de kernel de 2×2.
4. Función de activación ReLU.
5. Convolución 2D, 64 canales de entrada, 50 canales de salida, tamaño de kernel de 5×5,
6. Batch Normalization.
7. Dropout 2D, con probabilidad 0.5.
8. Max Pool 2D, tamaño de kernel de 2×2.
9. Lineal, 200 neuronas de entrada, 100 neuronas de salida.
10. Batch normalization 1D
11. Función de activación ReLU.
12. Dropout 2D, con probabilidad 0.5.
13. Lineal, 100 neuronas de entrada, 100 neuronas de salida.
14. Batch Normalization 1D.
15. Función de activación ReLU.

que es equivalente al modelo para el baseline sin la última capa lineal, la cual es ubicada después de la normalización L2.

Este modelo puede realizar el entrenamiento desde cero o comenzar pre entrenado. Siguiendo la metodología de [12], se utilizaron los parámetros del modelo pre entrenado de baseline para inicializar los parámetros de las capas del extractor de características de MME, mientras que la capa lineal después de la normalización se inicializó aleatoriamente.

Este modelo al hacer inferencia recibe un tensor de tamaño $N \times 3 \times 21 \times 21$ y entrega otro de tamaño $N \times 2$, con las probabilidades predichas para las clases bogus y real de cada tripleta de entrada.

6. Resultados

A menos que se especifique lo contrario, todos los experimentos descritos a continuación se entrenaron en igualdad de condiciones. Se utilizó un learning rate de 0.01, un tamaño de batch de 50, y una paciencia de 100 épocas para el early stopping. Se utilizó el algoritmo de descenso de gradiente estocástico con momentum 0.9, y para MME, se utilizó un parámetro de temperatura fijo de 0.05 para todos los experimentos. La función de costo para fine tuning fue la entropía cruzada, mientras que para MME fue la descrita en la sección 5.3.3.

Para balancear la cantidad de datos observadas por cada modelo independiente de la cantidad de elementos del conjunto de datos, se definió una época como 10.000 elementos etiquetados muestreados aleatoriamente (posiblemente con repetición) de cada conjunto de entrenamiento. En los casos donde ZTF o ATLAS es el conjunto de entrenamiento, se utilizó oversampling, para que estos 10.000 elementos muestreados tuviesen cantidades de reales y bogus balanceadas.

Siguiendo los resultados obtenidos en [47], se hizo decrecer el learning rate a la mitad cada 100.000 iteraciones. El parámetro lambda para MME se mantuvo fijo a lo largo de un entrenamiento, pero se optimizó para cada combinación source target.

Todos los experimentos fueron ejecutados en máquinas provistas por Amazon Web Services, usando una instancia de tipo *p2.xlarge*. El tiempo de ejecución fue de aproximadamente 3 semanas en total.

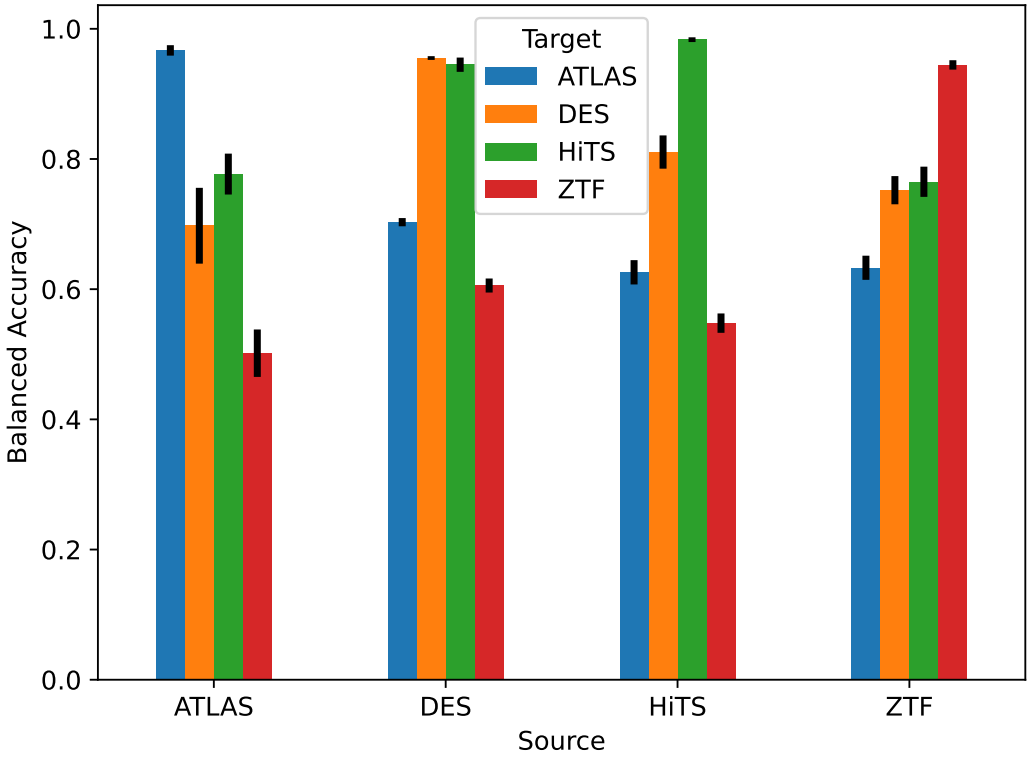


Figura 14: Resultados baseline. Promedio y desviación estándar del BACC de 10 particionados distintos por cada par de conjuntos de datos.

6.1. Baseline

source/target	ATLAS	DES	HiTS	ZTF
ATLAS	0.967 ± 0.008	0.697 ± 0.058	0.777 ± 0.031	0.502 ± 0.036
DES	0.703 ± 0.006	0.955 ± 0.003	0.945 ± 0.011	0.606 ± 0.011
HiTS	0.626 ± 0.019	0.811 ± 0.026	0.983 ± 0.004	0.548 ± 0.015
ZTF	0.633 ± 0.019	0.752 ± 0.022	0.765 ± 0.023	0.945 ± 0.007

Cuadro 1: Resultados baseline. Promedio y desviación estándar del balanced accuracy de 10 particionados distintos. En las filas, el conjunto de datos usado para entrenamiento, en las columnas el conjunto de datos evaluado.

En la figura 14 y en el cuadro 1 se muestra el resultado del experimento baseline. Cada barra en la figura y cada celda en el cuadro muestran la media y desviación estándar de 10 instancias con un particionamiento aleatorio diferente cada una.

Se ve que cuando source y target vienen del mismo conjunto de datos, las exactitudes son altas, llegando a un balanced accuracy sobre el 90 % en los 4 casos. Sin embargo, en los casos donde source y target son distintos, la exactitud disminuye significativamente, sobre todo en los experimentos que involucran a ZTF como target. Se puede observar el fenómeno del “domain shift” en todos los casos, pues todos

los modelos bajan su exactitud al ser probados sobre un dominio diferente de aquel con el que fueron entrenados.

Se observa además una asimetría en los conjuntos de datos. Los modelos entrenados con HiTS y evaluados en DES obtienen un BACC promedio menor que los modelos entrenados con DES y evaluados en HiTS. Los modelos entrenados con HiTS y evaluados en ZTF y los entrenados en DES y evaluados en ZTF obtienen un BACC promedio mucho menor que cuando son entrenados en ZTF, y evaluados en HiTS y DES, disminuyendo hasta un 20 % su BACC. Esto se puede explicar por el tipo de objeto que posee cada catálogo: HiTS y DES contienen supernovas y utilizan la misma cámara, mientras que ZTF, además de supernovas, contiene AGN, asteroides y otros, por lo que es entendible que para HiTS y DES sea difícil generalizar a tipos de objetos no presentes en su catálogo, a pesar de que contengan decenas o cientos de veces más datos que la cantidad de datos de ZTF.

6.2. Fine tuning y MME

Para los experimentos con MME, se utilizó el primero de los particionamientos para determinar el parámetro óptimo λ para cada combinación source-target. Se definieron diferentes valores posibles para λ: 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1 y 10. Se procedió entonces a evaluar el BACC sobre los conjuntos de validación y de entrenamiento, en función de la cantidad de shots.

En la figura 15 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios ATLAS→DES y DES→ATLAS. De acuerdo al comportamiento sobre el conjunto de validación (no el de prueba), se determinaron los valores $\lambda = 0.2$ para ATLAS→DES y $\lambda = 0.5$ para DES→ATLAS.

En la figura 16 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios ATLAS→HiTS y HiTS→ATLAS. De acuerdo al comportamiento sobre el conjunto de validación, se determinaron los valores $\lambda = 0.5$ para ATLAS→HiTS y $\lambda = 0.5$ para HiTS→ATLAS.

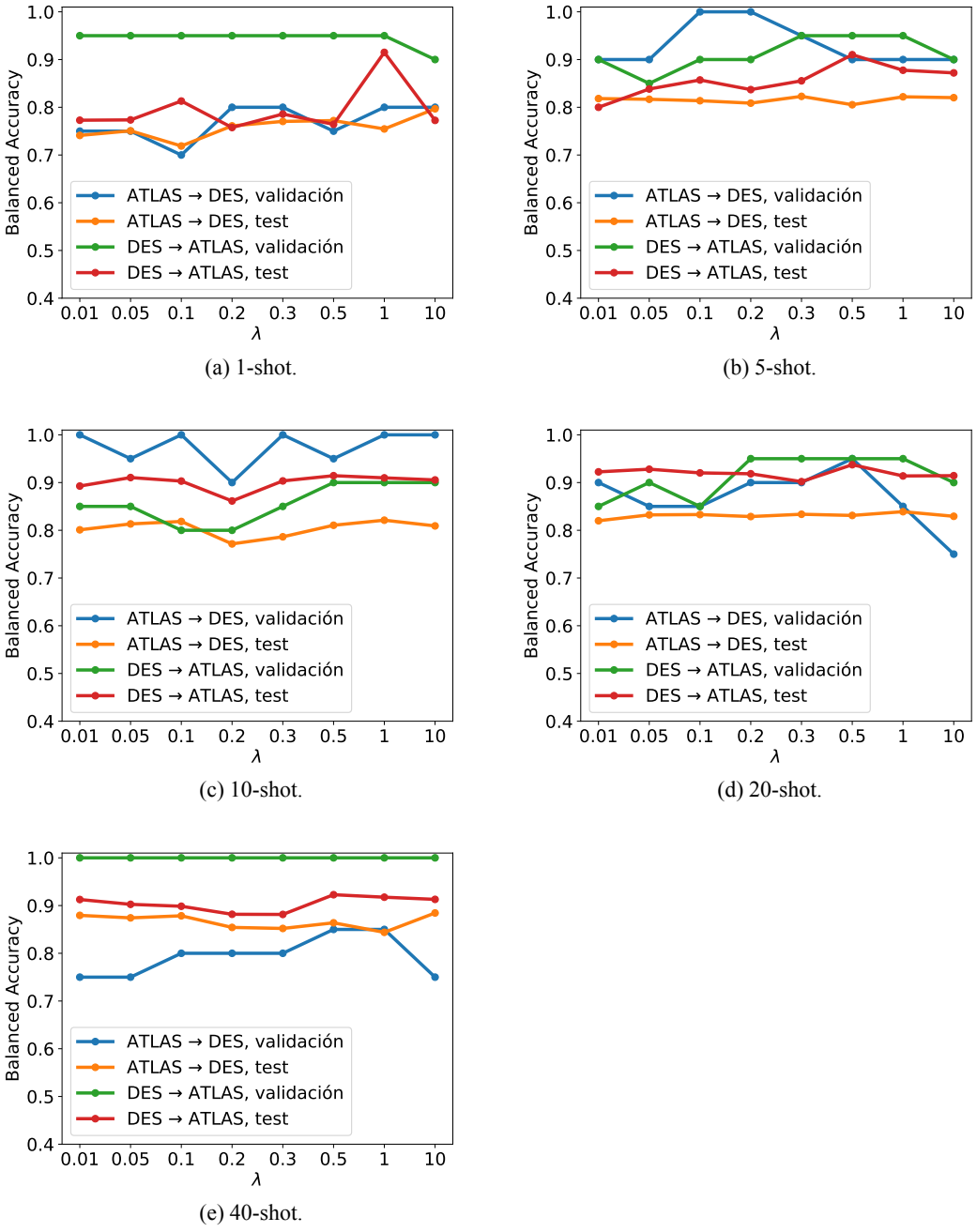
En la figura 17 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios ATLAS→ZTF y ZTF→ATLAS. De acuerdo al comportamiento sobre el conjunto de validación, se determinaron los valores $\lambda = 0.3$ para ATLAS→ZTF y $\lambda = 0.3$ para ZTF→ATLAS.

En la figura 18 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios HiTS→DES y DES→HiTS. De acuerdo al comportamiento para pocos shots sobre el conjunto de validación, se determinaron los valores $\lambda = 0.5$ para HiTS→DES y $\lambda = 0.3$ para DES→HiTS.

En la figura 19 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios DES→ZTF y ZTF→DES. De acuerdo al comportamiento sobre el conjunto de validación, se determi-

naron los valores $\lambda = 0.5$ para DES→ZTF y $\lambda = 0.3$ para ZTF→DES.

En la figura 20 se muestran las curvas de BACC en función de la cantidad de shots para los escenarios HiTS→ZTF y ZTF→HiTS. De acuerdo al comportamiento sobre el conjunto de validación, se determinaron los valores $\lambda = 0.1$ para HiTS→ZTF y $\lambda = 0.1$ para ZTF→HiTS.



(a) 1-shot.

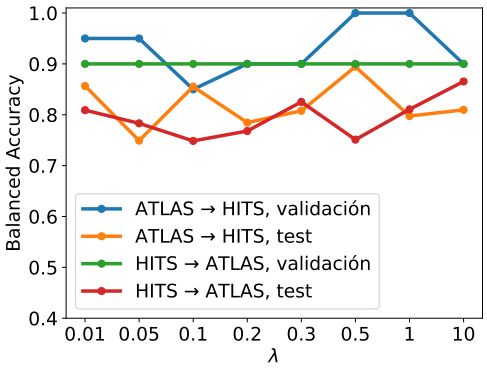
(b) 5-shot.

(c) 10-shot.

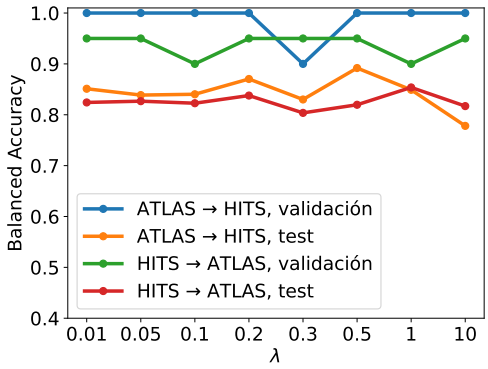
(d) 20-shot.

(e) 40-shot.

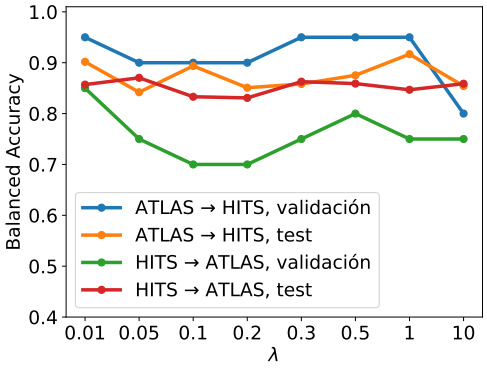
Figura 15: Lambda versus BACC casos ATLAS→DES y DES→ATLAS.



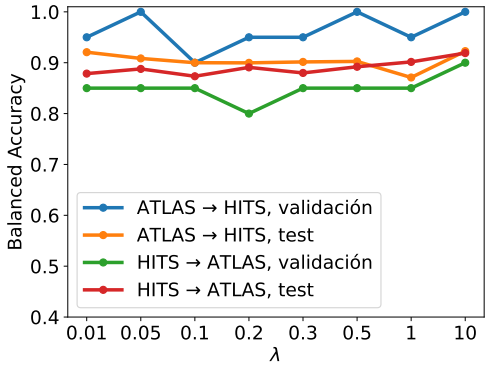
(a) 1-shot.



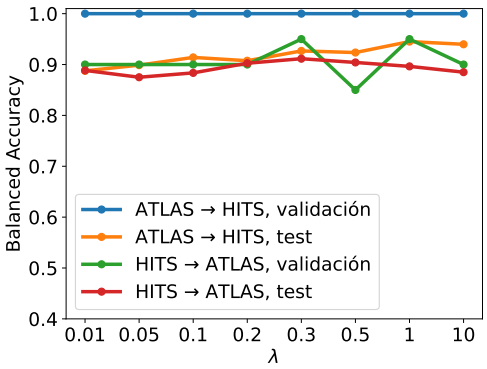
(b) 5-shot.



(c) 10-shot.

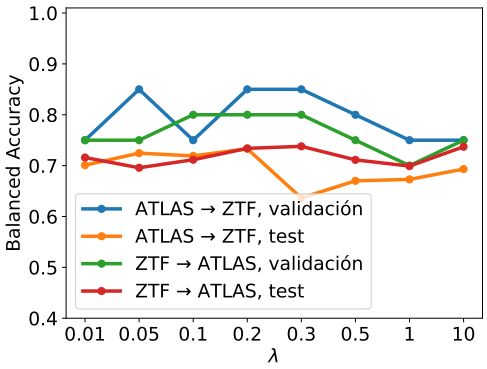


(d) 20-shot.

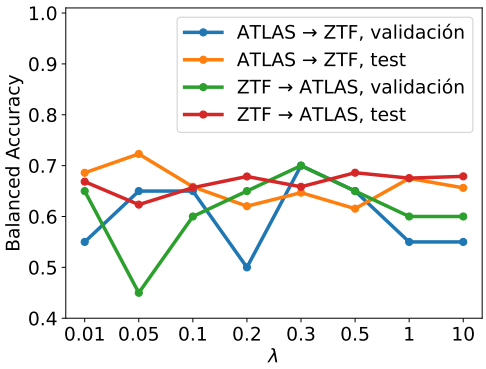


(e) 40-shot.

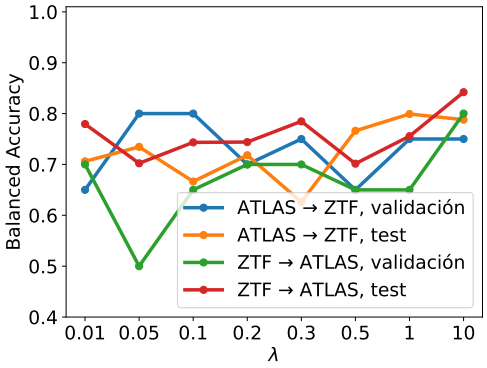
Figura 16: Lambda versus BACC casos ATLAS→HiTS y HiTS→ATLAS.



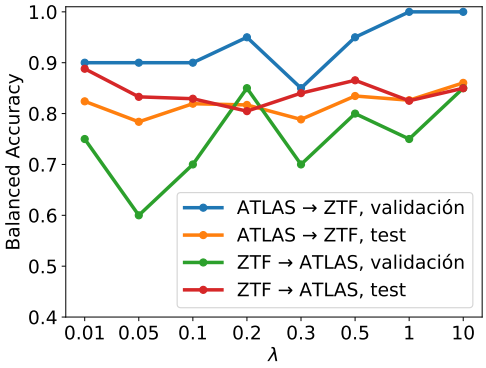
(a) 1-shot.



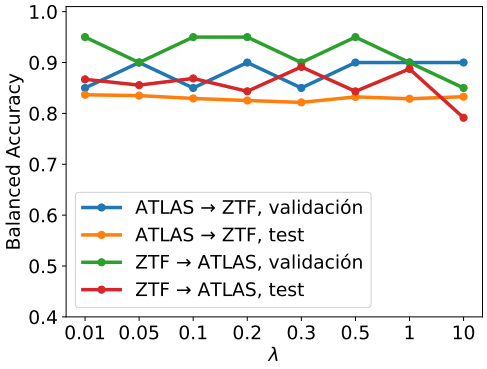
(b) 5-shot.



(c) 10-shot.

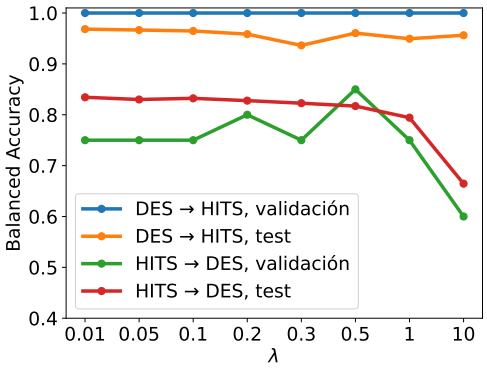


(d) 20-shot.

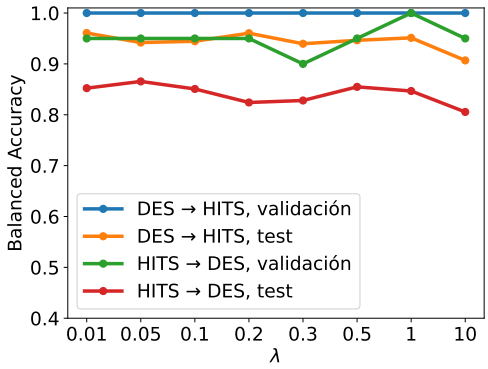


(e) 40-shot.

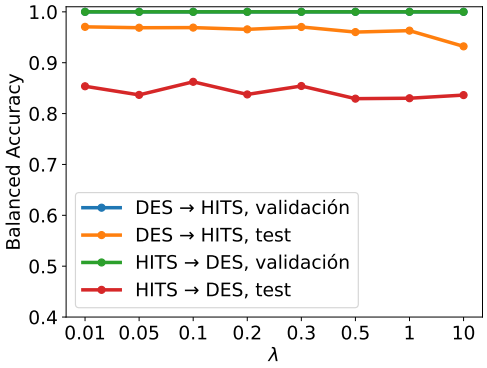
Figura 17: Lambda versus BACC casos ATLAS→ZTF y ZTF→ATLAS.



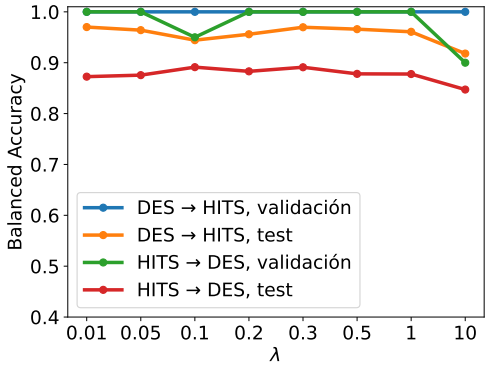
(a) 1-shot.



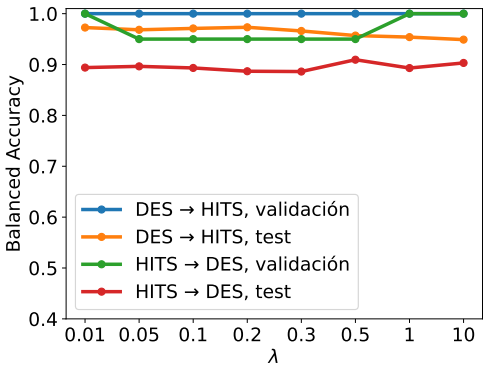
(b) 5-shot.



(c) 10-shot (curvas de validación solapadas).

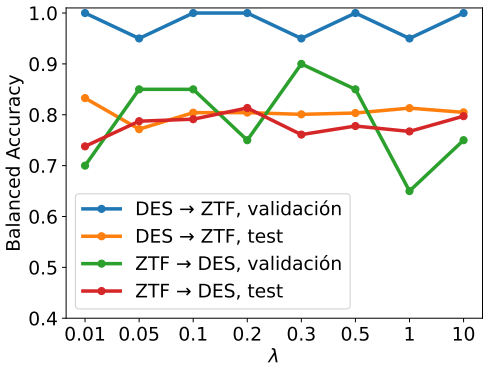


(d) 20-shot.

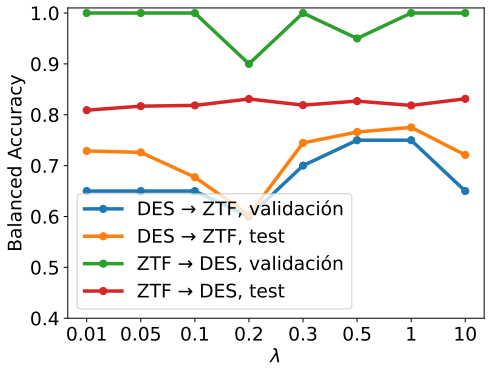


(e) 40-shot.

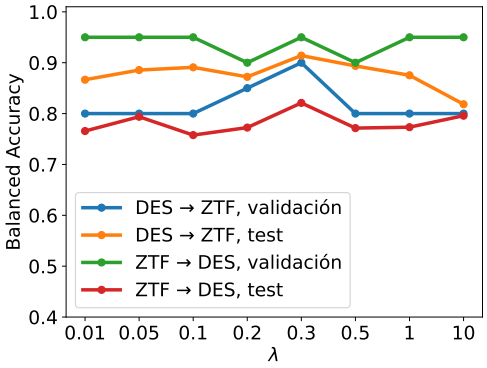
Figura 18: Lambda versus BACC casos HiTS→DES y DES→HiTS.



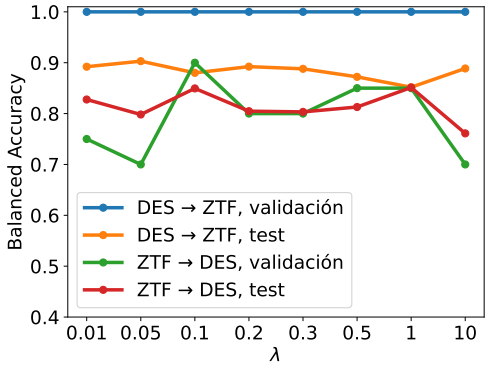
(a) 1-shot.



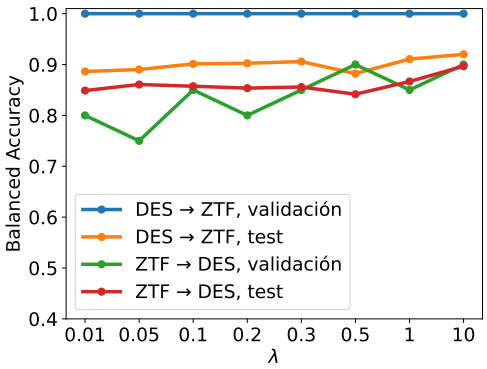
(b) 5-shot.



(c) 10-shot.

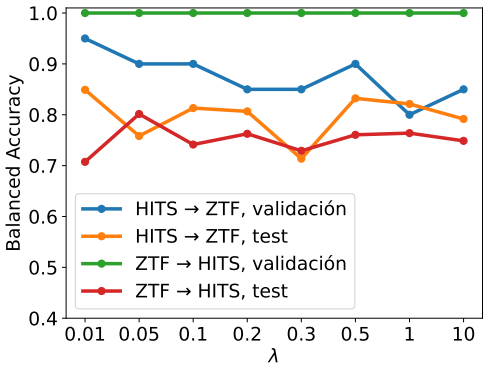


(d) 20-shot.

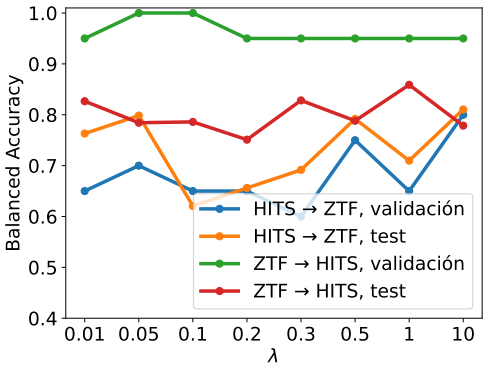


(e) 40-shot.

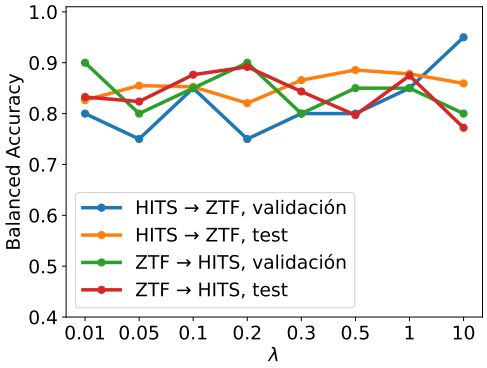
Figura 19: Lambda versus BACC casos DES→ZTF y ZTF→DES.



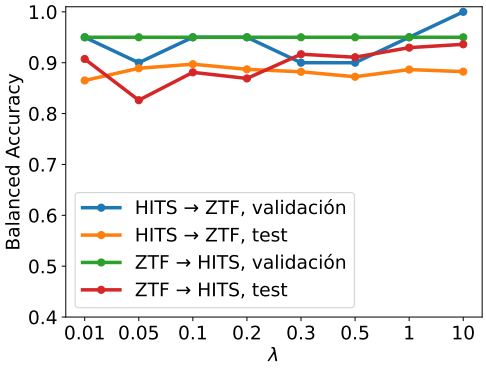
(a) 1-shot.



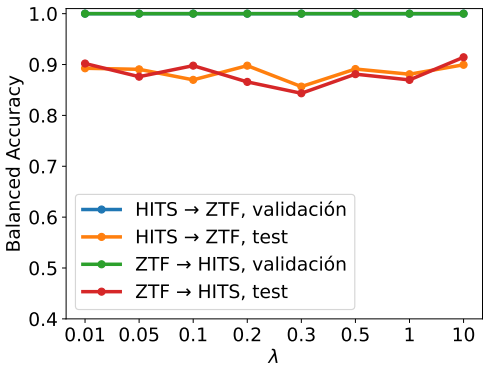
(b) 5-shot.



(c) 10-shot.



(d) 20-shot.



(e) 40-shot (curvas de validación solapadas).

Figura 20: Lambda versus BACC casos HiTS→ZTF y ZTF→HiTS.

A continuación, se analiza caso por caso el comportamiento de los experimentos fine tuning y MME, por combinación source-target.

6.2.1. Source ATLAS, Target DES

En el cuadro 2 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ATLAS), mientras que en el cuadro 3 se muestran los mismos modelos evaluados sobre el conjunto target (DES).

shots	Fine tuning	MME
1	0.725 ± 0.162	0.972 ± 0.009
5	0.785 ± 0.059	0.972 ± 0.008
10	0.867 ± 0.061	0.974 ± 0.007
20	0.840 ± 0.098	0.973 ± 0.006
40	0.840 ± 0.076	0.971 ± 0.007

Cuadro 2: BACC de modelos ATLAS→DES evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

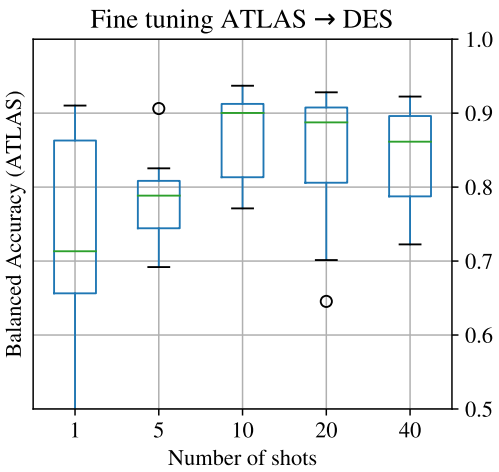
En la figura 21 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento [7]

shots	Fine tuning	MME
1	0.675 ± 0.061	0.727 ± 0.039
5	0.771 ± 0.030	0.782 ± 0.033
10	0.796 ± 0.033	0.806 ± 0.018
20	0.837 ± 0.019	0.836 ± 0.015
40	0.862 ± 0.010	0.857 ± 0.016

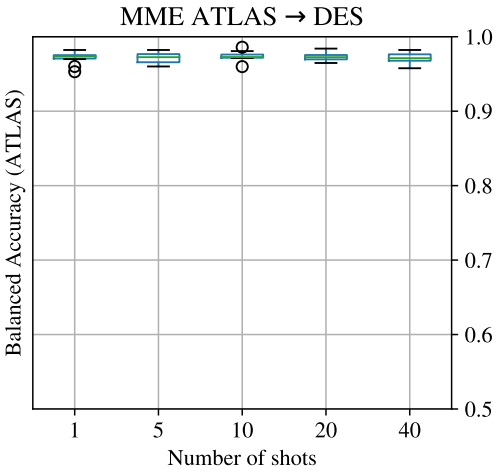
Cuadro 3: BACC de modelos ATLAS→DES evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

Para estos experimentos, no se observa mucha variación entre fine tuning y MME a partir de 5-shot en el conjunto target. Para 10-shot, fine tuning presenta outliers en ambas direcciones, para 1-shot, fine tuning presenta más incertidumbre en el accuracy, pero para el resto presenta distribuciones similares. A partir de 5-shot, tanto fine tuning como MME son capaces de superar el BACC promedio del baseline para cada uno de los 10 particionados. Ambos modelos mejoran su desempeño a medida que se aumenta la cantidad de shots. Sin embargo, aún en 40-shots, ni fine tuning ni MME logran superar el umbral de 0.9, aunque quizás se podría lograr si se siguiera usando más elementos etiquetados. Cuando comparamos

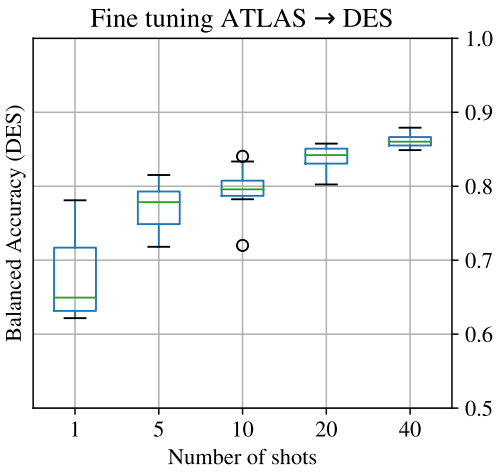
[7] Las cajas representan la distribución de las diferentes iteraciones para cada cantidad de shots. Estas cajas se extienden desde el primer cuartil Q_1 hasta el tercer cuartil Q_3. La linea que atraviesa cada caja indica el BACC mediano (Q_2) y las barras indican el rango para los valores que no son outliers. Los círculos en cada figura indican outliers, de acuerdo al comportamiento por defecto de la librería pandas del lenguaje de programación Python. Estos se definen como puntos que están a una distancia superior a $1.5(Q_3 - Q_1)$ desde los bordes de cada caja [58] .



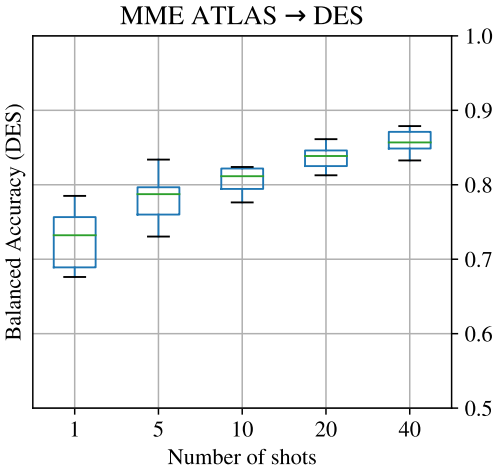
(a) Fine-tuning ATLAS→DES evaluado en source



(b) MME ATLAS→DES evaluado en source.



(c) Fine-tuning ATLAS→DES evaluado en target.



(d) MME ATLAS→DES evaluado en target.

Figura 21: Distribución del BACC para modelos ATLAS→DES

los modelos en source, se observa el notable desempeño de MME. Este modelo no reduce su exactitud en source, ni si quiera cuando se ocupa 1-shot, llegando por sobre el .0.9 en todos los casos, mientras que fine tuning, si bien logra superar dicho umbral en el mejor de los casos para cualquier cantidad de shots, presenta mucha más incertidumbre, pudiendo llegar hasta bajo el 0.6 en el peor de los casos.

6.2.2. Source ATLAS, Target HiTS

En el cuadro 4 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ATLAS), mientras que en el cuadro 5 se muestran los mismos modelos evaluados sobre el conjunto target (HiTS).

shots	Fine tuning	MME
1	0.877 ± 0.092	0.975 ± 0.010
5	0.904 ± 0.094	0.968 ± 0.011
10	0.915 ± 0.046	0.973 ± 0.008
20	0.869 ± 0.067	0.969 ± 0.009
40	0.816 ± 0.086	0.969 ± 0.014

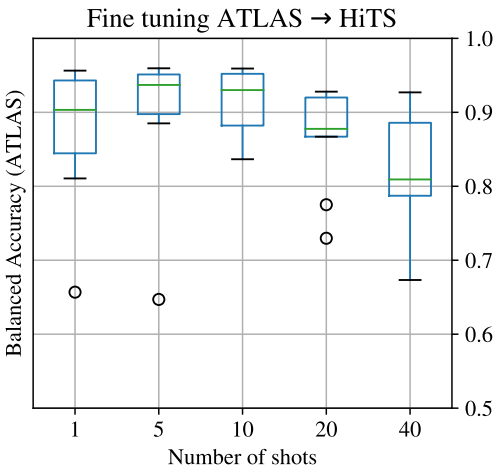
Cuadro 4: BACC de modelos ATLAS→HiTS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 22 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

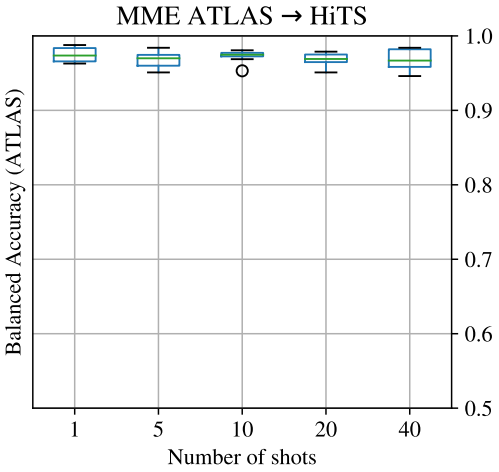
shots	Fine tuning	MME
1	0.796 ± 0.097	0.825 ± 0.040
5	0.871 ± 0.078	0.868 ± 0.028
10	0.899 ± 0.031	0.878 ± 0.026
20	0.930 ± 0.012	0.901 ± 0.019
40	0.935 ± 0.017	0.909 ± 0.015

Cuadro 5: BACC de modelos ATLAS→HiTS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

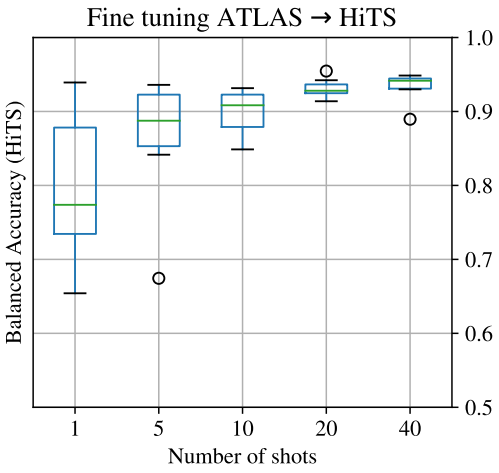
Para estos experimentos, se observa que tanto fine tuning como MME son capaces de superar el BACC en target promedio del baseline para cada uno de los 10 particionados a partir de 5-shots. Ambos modelos mejoran su desempeño a medida que se aumenta la cantidad de shots. Fine tuning consigue superar a MME a partir de 20-shot, y hasta los 10-shot, ambos presentan rangos solapados en el BACC de target. Fine tuning muestra más incertidumbre, llegando más alto que el mejor 1-shot de MME, pero más bajo que el peor 1-shot de MME. Cuando comparamos los modelos en source, se observa nuevamente que MME mantiene su BACC estable y por sobre el 0.9 para cada uno de las 10 instancias ejecutadas y para cada cantidad de shots, mientras que fine tuning a pesar de llegar sobre el 0.9 en el mejor de los casos para cada cantidad de shots, presenta barras de error más grandes. Incluso se puede observar una tendencia a bajar el desempeño en source a partir de 20-shot.



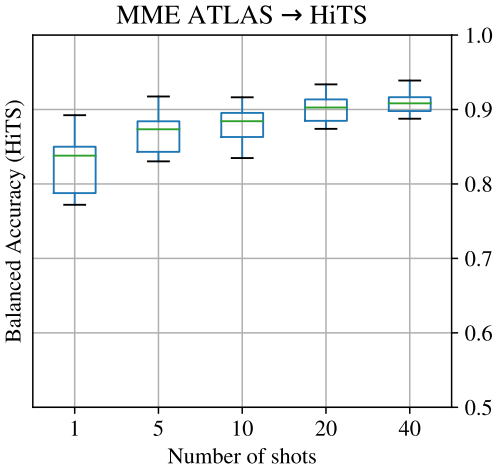
(a) Fine-tuning ATLAS→HiTS evaluado en source.



(b) MME ATLAS→HiTS evaluado en source.



(c) Fine-tuning ATLAS→HiTS evaluado en target.



(d) MME ATLAS→HiTS evaluado en target.

Figura 22: Distribución del BACC para modelos ATLAS→HiTS.

6.2.3. Source ATLAS, Target ZTF

En el cuadro 6 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ATLAS), mientras que en el cuadro 7 se muestran los mismos modelos evaluados sobre el conjunto target (ZTF).

shots	Fine tuning	MME
1	0.815 ± 0.108	0.970 ± 0.007
5	0.788 ± 0.100	0.973 ± 0.005
10	0.702 ± 0.104	0.972 ± 0.008
20	0.703 ± 0.038	0.968 ± 0.010
40	0.712 ± 0.064	0.962 ± 0.009

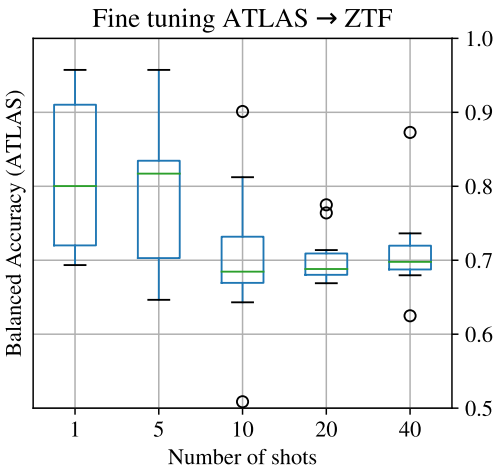
Cuadro 6: BACC de modelos ATLAS→ZTF evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 23 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

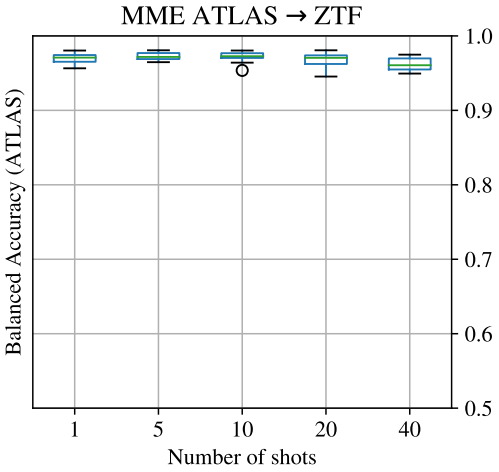
shots	Fine tuning	MME
1	0.626 ± 0.113	0.613 ± 0.044
5	0.743 ± 0.088	0.665 ± 0.057
10	0.818 ± 0.045	0.754 ± 0.028
20	0.872 ± 0.026	0.775 ± 0.016
40	0.882 ± 0.029	0.817 ± 0.036

Cuadro 7: BACC de modelos ATLAS→ZTF evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

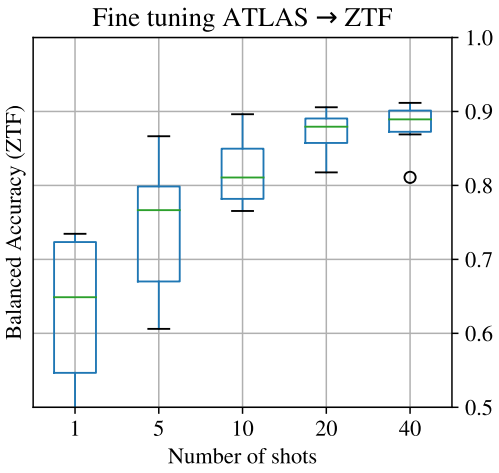
Para estos experimentos, se observa que fine tuning obtiene un accuracy más alto que MME a partir de 5-shots en el conjunto target. Para 1-shot, ambos modelos se encuentran en rangos solapados con promedios bastante bajos (0.6), aunque fine-tuning presenta mucha más incertidumbre para este caso. Ambos modelos consiguen mejorar el baseline a partir de 1-shot, lo cual no dice mucho, puesto que la combinación source ATLAS y target ZTF presenta el accuracy promedio en baseline más bajo. Ambos modelos, fine tuning y MME aumentan su desempeño a medida que se aumenta la cantidad de shots, con fine-tuning llegando a un BACC sobre 0.9 para algunas instancias de 40-shots solamente, mientras que MME nunca supera dicho umbral. Cuando comparamos el desempeño en source, sin embargo, MME mantiene todos sus BACC para toda cantidad de shots sobre el 0.9, relativamente constante al variar la cantidad de shots y con muy poca desviación. Por otra parte, fine tuning en target muestra barras de error notablemente grandes, y una tendencia a empeorar a medida que se aumenta la cantidad de shots.



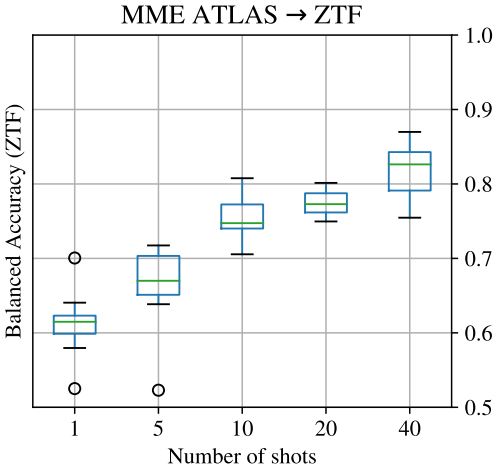
(a) Fine-tuning ATLAS→ZTF evaluado en source.



(b) MME ATLAS→ZTF evaluado en source.



(c) Fine-tuning ATLAS→ZTF evaluado en target.



(d) MME ATLAS→ZTF evaluado en target.

Figura 23: Distribución del BACC para modelos ATLAS→ZTF.

6.2.4. Source DES, Target ATLAS

En el cuadro 8 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (DES), mientras que en el cuadro 9 se muestran los mismos modelos evaluados sobre el conjunto target (ATLAS).

shots	Fine tuning	MME
1	0.899 ± 0.042	0.941 ± 0.012
5	0.893 ± 0.044	0.931 ± 0.020
10	0.893 ± 0.062	0.929 ± 0.014
20	0.910 ± 0.017	0.939 ± 0.013
40	0.882 ± 0.020	0.943 ± 0.008

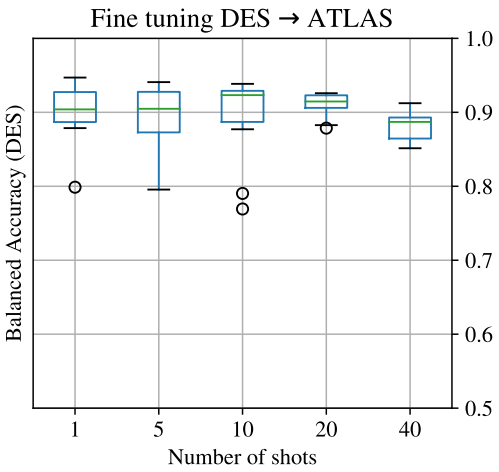
Cuadro 8: BACC de modelos DES→ATLAS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 24 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

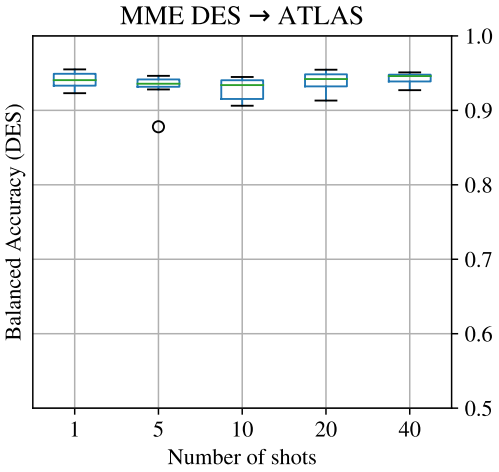
shots	Fine tuning	MME
1	0.768 ± 0.085	0.797 ± 0.088
5	0.832 ± 0.087	0.848 ± 0.081
10	0.891 ± 0.037	0.895 ± 0.024
20	0.914 ± 0.015	0.909 ± 0.012
40	0.927 ± 0.016	0.924 ± 0.013

Cuadro 9: BACC de modelos DES→ATLAS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

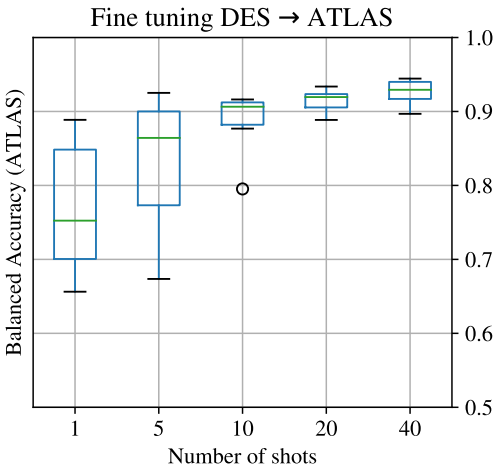
Para estos experimentos, no se observa mucha variación entre fine tuning y MME a partir en el conjunto target. Para 1-shot y 5-shot, fine tuning y MME presentan barras de error grandes, las cuales se reducen a partir de los 10-shot. Tanto fine tuning como MME consiguen superar un BACC de 0.9 para la mayoría de las instancias a partir de 20-shot. Ambos modelos muestran una alza en el BACC para target a medida que se aumenta la cantidad de shots, y ambos consiguen superar el baseline en promedio a partir de 1-shot. En source, ambos modelos consiguen superar el umbral del 0.9 de BACC para la mayoría de las instancias y para cualquier cantidad de shots, excepto fine tuning para 1 y 40-shot. A pesar de que fine tuning presenta barras de error más grandes para 1, 5 y 10-shot, el rango de BACC se solapa con el de MME, aunque se puede ver una ligera disminución de fine tuning en 40-shot.



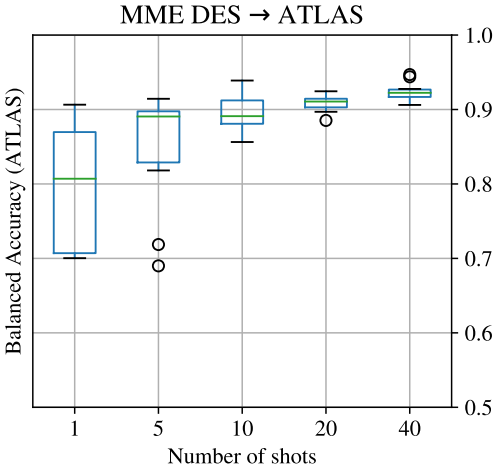
(a) Fine-tuning DES→ATLAS evaluado en source.



(b) MME DES→ATLAS evaluado en source.



(c) Fine-tuning DES→ATLAS evaluado en target.



(d) MME DES→ATLAS evaluado en target.

Figura 24: Distribución del BACC para modelos DES→ATLAS.

6.2.5. Source DES, Target HiTS

En el cuadro 10 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (DES), mientras que en el cuadro 11 se muestran los mismos modelos evaluados sobre el conjunto target (HiTS).

shots	Fine tuning	MME
1	0.858 ± 0.136	0.944 ± 0.007
5	0.908 ± 0.045	0.943 ± 0.009
10	0.902 ± 0.057	0.948 ± 0.004
20	0.874 ± 0.062	0.940 ± 0.010
40	0.921 ± 0.014	0.945 ± 0.006

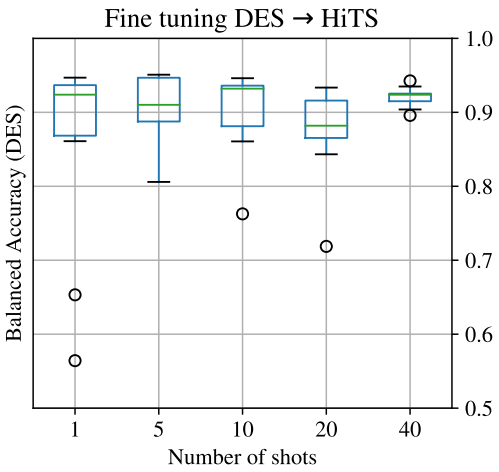
Cuadro 10: BACC de modelos DES→HiTS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 25 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

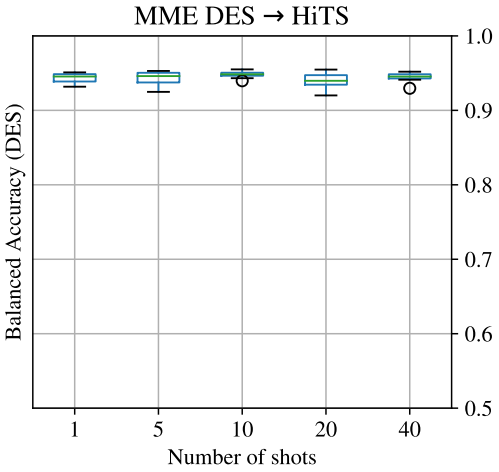
shots	Fine tuning	MME
1	0.873 ± 0.140	0.958 ± 0.008
5	0.936 ± 0.029	0.960 ± 0.017
10	0.946 ± 0.019	0.965 ± 0.005
20	0.956 ± 0.009	0.959 ± 0.010
40	0.964 ± 0.005	0.964 ± 0.010

Cuadro 11: BACC de modelos DES→HiTS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

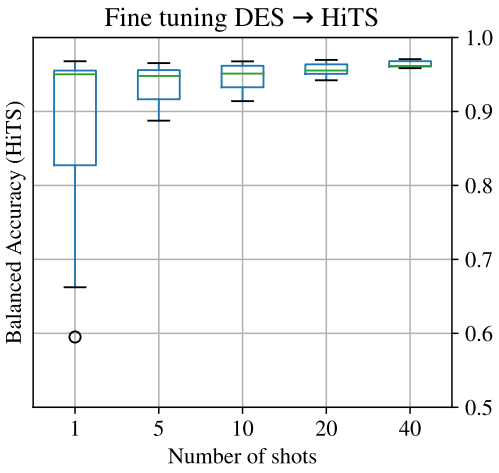
Para estos experimentos, no se observa mucha variación entre fine tuning y MME a partir de 10-shot en el conjunto target. Sin embargo, en 1-shot y 10-shot, MME presenta menos incertidumbre: Los BACC de los 10 particionados están menos separados que fine tuning. En MME para 1-shot, el BACC de la peor instancia está por sobre el 0.9 mientras que en fine tuning el BACC mínimo está por debajo del 0.6. Más aún, MME es capaz de conseguir estos resultados sin empeorar su desempeño en el conjunto source, manteniéndose casi siempre sobre el 0.9 para toda cantidad de shots en todas las instancias, mientras que fine tuning en source presenta outliers que llegan incluso por debajo de 0.6 para 1-shot. Notamos que fine-tuning a partir de 20-shot es capaz de superar el baseline, mientras que MME lo consigue a partir de 1-shot, lo cual es importante considerando que la combinación source DES y target HiTS obtuvo el mayor BACC promedio en el baseline.



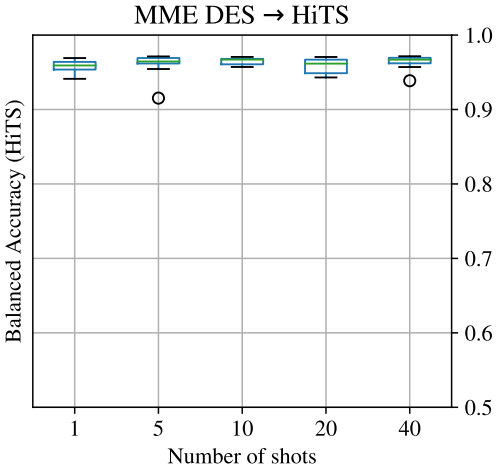
(a) Fine-tuning DES→HiTS evaluado en source.



(b) MME DES→HiTS evaluado en source.



(c) Fine-tuning DES→HiTS evaluado en target.



(d) MME DES→HiTS evaluado en target.

Figura 25: Distribución del BACC para modelos DES→HiTS.

6.2.6. Source DES, Target ZTF

En el cuadro 12 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (DES), mientras que en el cuadro 13 se muestran los mismos modelos evaluados sobre el conjunto target (ZTF).

shots	Fine tuning	MME
1	0.785 ± 0.124	0.922 ± 0.069
5	0.876 ± 0.051	0.942 ± 0.008
10	0.866 ± 0.057	0.925 ± 0.041
20	0.877 ± 0.043	0.940 ± 0.013
40	0.879 ± 0.025	0.943 ± 0.010

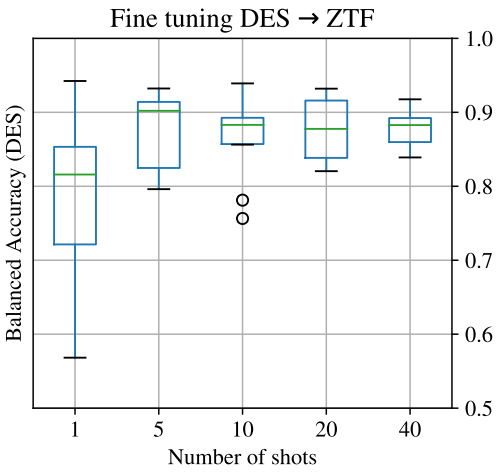
Cuadro 12: BACC de modelos DES→ZTF evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 26 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

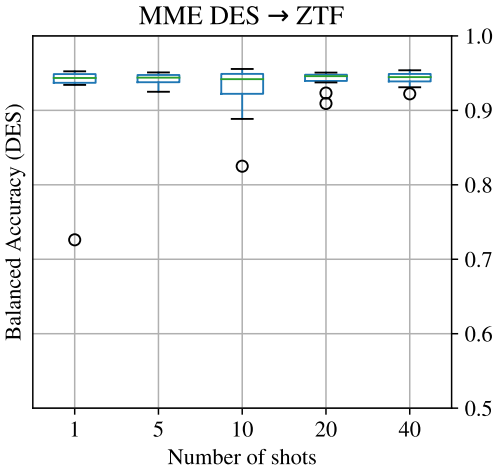
shots	Fine tuning	MME
1	0.695 ± 0.083	0.704 ± 0.072
5	0.795 ± 0.063	0.786 ± 0.070
10	0.846 ± 0.026	0.842 ± 0.036
20	0.879 ± 0.028	0.872 ± 0.032
40	0.898 ± 0.016	0.884 ± 0.017

Cuadro 13: BACC de modelos DES→ZTF evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

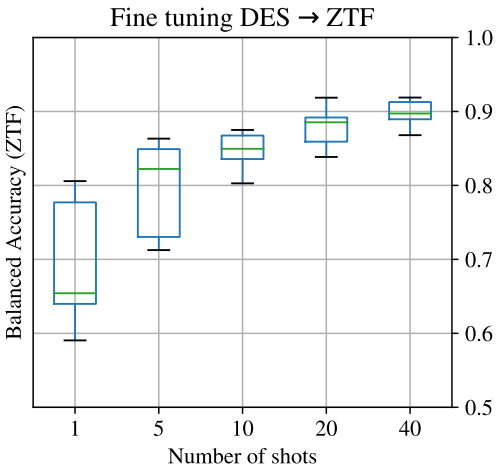
Para estos experimentos, no se observa mucha variación entre fine tuning y MME en el conjunto target. Ambos modelos consiguen superar el baseline tan solo usando 1-shot por clase, diferencia que se hace más notoria al aumentar la cantidad de shots. En cuanto a source, MME consistentemente mantiene su desempeño sobre el 0.9 para cada una de las 10 instancias, para casi toda cantidad de shots, con la excepción de 1-shot y 10-shot que presentan algunos outliers. Por otra parte, fine tuning consigue BACC sobre el 0.9 para por lo menos 1 instancia para cualquier cantidad de shots. Sin embargo, fine tuning en source presenta barras de error mucho más grandes que MME, aunque los rangos se solapan.



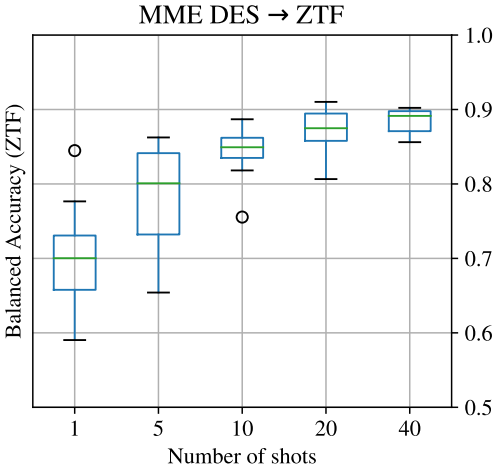
(a) Fine-tuning DES→ZTF evaluado en source.



(b) MME DES→ZTF evaluado en source.



(c) Fine-tuning DES→ZTF evaluado en target.



(d) MME DES→ZTF evaluado en target.

Figura 26: Distribución del BACC para modelos DES→ZTF.

6.2.7. Source HiTS, Target ATLAS

En el cuadro 14 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (HiTS), mientras que en cuadro 15 se muestran los mismos modelos evaluados sobre el conjunto target (ATLAS).

shots	Fine tuning	MME
1	0.876 ± 0.089	0.976 ± 0.006
5	0.892 ± 0.076	0.975 ± 0.007
10	0.911 ± 0.035	0.977 ± 0.004
20	0.927 ± 0.033	0.975 ± 0.006
40	0.920 ± 0.010	0.977 ± 0.006

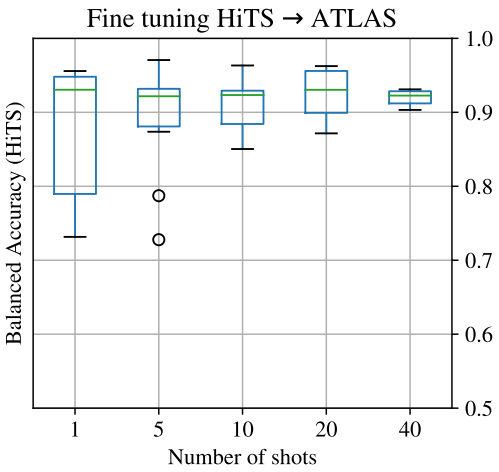
Cuadro 14: BACC de modelos HiTS→ATLAS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 27 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

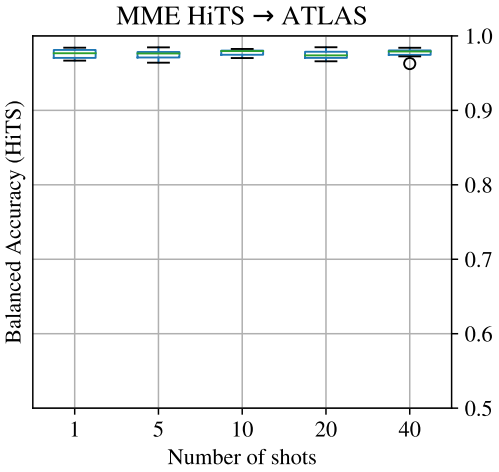
shots	Fine tuning	MME
1	0.740 ± 0.078	0.778 ± 0.086
5	0.827 ± 0.080	0.819 ± 0.084
10	0.850 ± 0.058	0.850 ± 0.031
20	0.892 ± 0.037	0.892 ± 0.020
40	0.921 ± 0.024	0.904 ± 0.015

Cuadro 15: BACC de modelos HiTS→ATLAS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

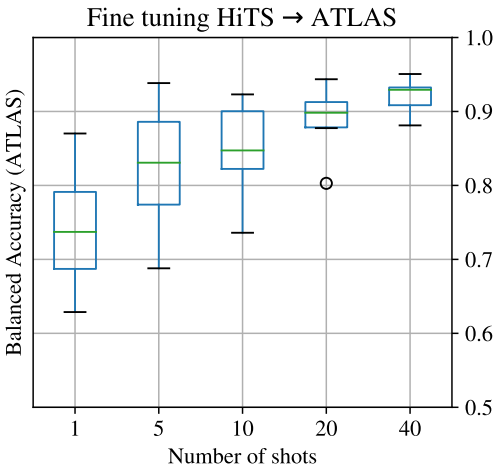
Para estos experimentos, no se observa mucha variación entre fine tuning y MME en el conjunto target. Para 1-shot MME consigue llegar a un BACC promedio ligeramente más alto que fine tuning; para 40-shot fine tuning llega a un BACC promedio ligeramente más alto que MME, mientras que para los otros casos los promedios son muy similares. Los rangos se solapan desde los 1 hasta los 40 shot con barras de error similares, por lo cual no se puede favorecer un modelo sobre otro para source. Ambos modelos son capaces de superar el baseline desde los 1-shot. En source sin embargo, MME es superior a fine tuning: desde el 1-shot, MME presenta un BACC sobre el 0.95 para todas las instancias, mientras que fine tuning, si bien logra pasar el umbral del 0.9 para más de la mitad de sus instancias (todas las instancias en el caso de 40-shot), presenta barras de error mucho más grandes, en especial en 1 y 5-shot. Estas barras de error parecen disminuir a medida que se aumenta la cantidad de shots.



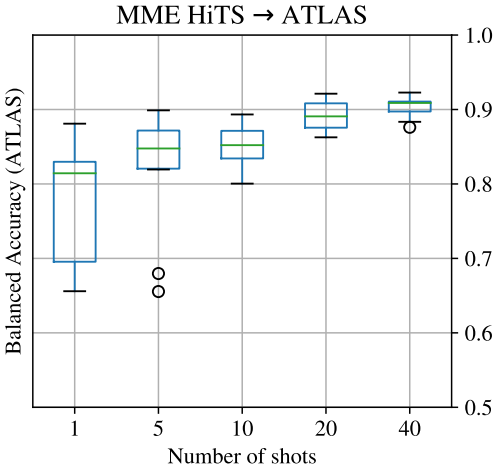
(a) Fine-tuning HiTS→ATLAS evaluado en source.



(b) MME HiTS→ATLAS evaluado en source.



(c) Fine-tuning HiTS→ATLAS evaluado en target.



(d) MME HiTS→ATLAS evaluado en target.

Figura 27: Distribución del BACC para modelos HiTS→ATLAS.

6.2.8. Source HiTS, Target DES

En el cuadro 16 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (HiTS), mientras que en el cuadro 17 se muestran los mismos modelos evaluados sobre el conjunto target (DES).

shots	Fine tuning	MME
1	0.959 ± 0.031	0.983 ± 0.002
5	0.946 ± 0.036	0.981 ± 0.004
10	0.967 ± 0.013	0.980 ± 0.006
20	0.970 ± 0.012	0.981 ± 0.004
40	0.954 ± 0.024	0.981 ± 0.003

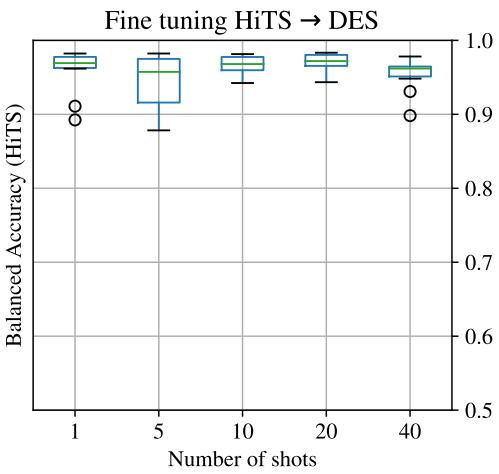
Cuadro 16: BACC de modelos HiTS→DES evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 28 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

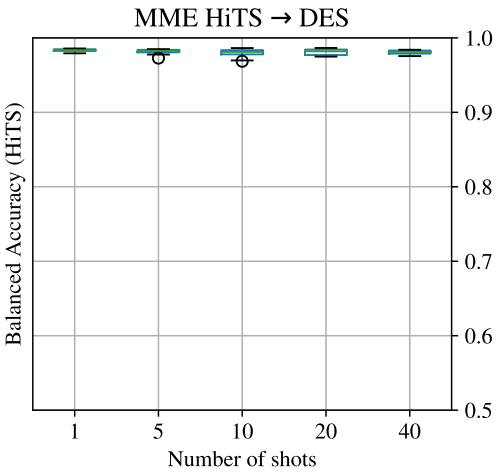
shots	Fine tuning	MME
1	0.770 ± 0.064	0.845 ± 0.017
5	0.831 ± 0.021	0.856 ± 0.016
10	0.854 ± 0.043	0.866 ± 0.016
20	0.872 ± 0.017	0.869 ± 0.019
40	0.888 ± 0.015	0.880 ± 0.011

Cuadro 17: BACC de modelos HiTS→DES evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

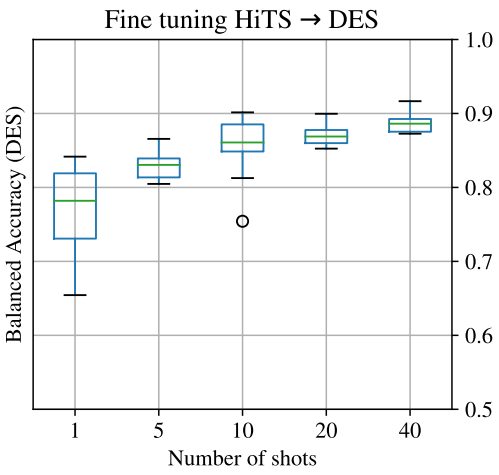
Para estos experimentos, no se observa mucha variación entre fine tuning y MME desde 10-shot hasta 40-shot en el conjunto target. El BACC de MME siempre tiene medias más altas que fine tuning, y si bien MME es superior a fine tuning en 1 y 5-shot, fine tuning consigue alcanzarlo a partir de los 10-shot. Ambos modelos muestran una tendencia a aumentar el BACC al aumentar la cantidad de shots, aunque lentamente. Ambos modelos consiguen superar el baseline, fine tuning a partir de los 5-shot y MME a partir del 1-shot. Por otra parte, en source, MME es consistentemente el ganador, obteniendo siempre un BACC promedio mayor, y con desviaciones muy pequeñas, de a lo más 0.006, Aunque fine tuning lo hace bastante bien de todas formas, presentando solo algunos outliers bajo el 0.9 en 1, 5 y 40-shot.



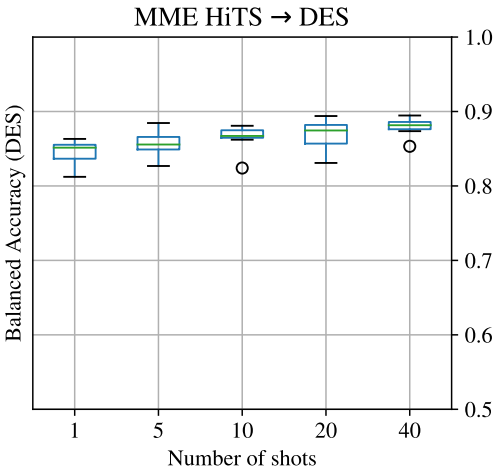
(a) Fine-tuning HiTS→DES evaluado en source.



(b) MME HiTS→DES evaluado en source.



(c) Fine-tuning HiTS→DES evaluado en target.



(d) MME HiTS→DES evaluado en target.

Figura 28: Distribución del BACC para modelos HiTS→DES.

6.2.9. Source HiTS, Target ZTF

En el cuadro 18 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (HiTS), mientras que en el cuadro 19 se muestran los mismos modelos evaluados sobre el conjunto target (ZTF).

shots	Fine tuning	MME
1	0.933 ± 0.047	0.975 ± 0.006
5	0.931 ± 0.067	0.977 ± 0.006
10	0.923 ± 0.047	0.979 ± 0.005
20	0.917 ± 0.039	0.979 ± 0.003
40	0.857 ± 0.132	0.973 ± 0.011

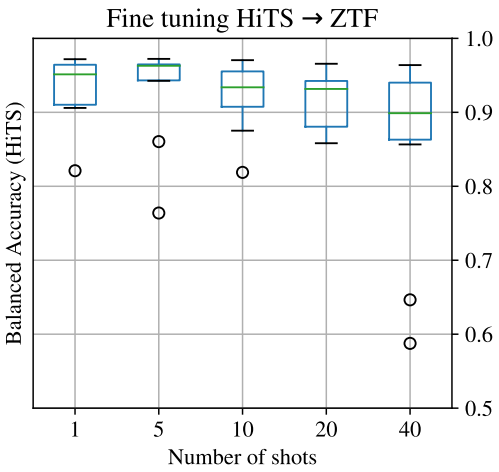
Cuadro 18: BACC de modelos HiTS→ZTF evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 29 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

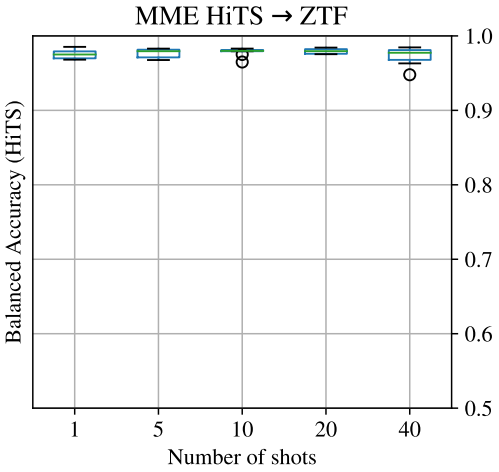
shots	Fine tuning	MME
1	0.693 ± 0.061	0.749 ± 0.066
5	0.773 ± 0.072	0.808 ± 0.089
10	0.839 ± 0.037	0.851 ± 0.035
20	0.871 ± 0.017	0.870 ± 0.031
40	0.882 ± 0.024	0.883 ± 0.013

Cuadro 19: BACC de modelos HiTS→ZTF evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

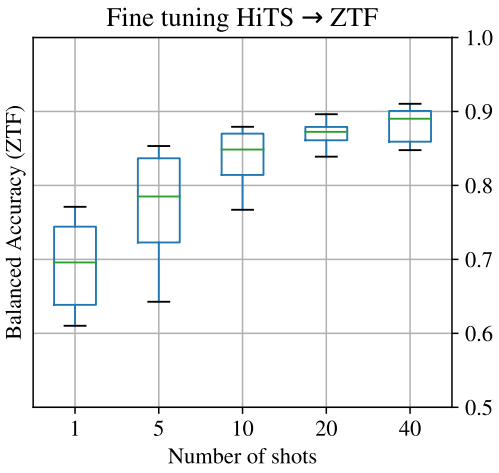
Para estos experimentos no se observan muchas diferencias entre fine tuning y MME en el conjunto target. Ambos modelos muestran comportamientos similares, con barras de error grandes hasta 5-shot y con una tendencia a aumentar el BACC y disminuir las barras de error a medida que se aumenta la cantidad de shots. Ambos modelos consiguen superar el baseline para todas sus instancias y para toda cantidad de shots. En source sin embargo, MME es superior a fine tuning, puesto que mantiene un BACC promedio más alto y con barras de error muy pequeñas, mientras fine tuning presenta barras de error mucho más grandes, además de outliers que parecen empeorar a medida que se aumenta la cantidad de shots. A pesar de esto, fine tuning puede superar el umbral del BACC 0.95 para la mejor de sus instancias desde el 1-shot en adelante.



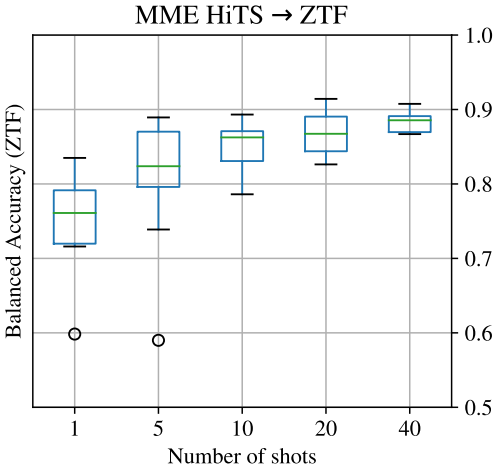
(a) Fine-tuning HiTS→ZTF evaluado en source.



(b) MME HiTS→ZTF evaluado en source.



(c) Fine-tuning HiTS→ZTF evaluado en target.



(d) MME HiTS→ZTF evaluado en target.

Figura 29: Distribución del BACC para modelos HiTS→ZTF.

6.2.10. Source ZTF, Target ATLAS

En el cuadro 20 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ZTF), mientras que en el cuadro 21 se muestran los mismos modelos evaluados sobre el conjunto target (ATLAS).

shots	Fine tuning	MME
1	0.759 ± 0.126	0.948 ± 0.012
5	0.787 ± 0.083	0.950 ± 0.004
10	0.773 ± 0.080	0.946 ± 0.004
20	0.776 ± 0.060	0.946 ± 0.008
40	0.753 ± 0.045	0.937 ± 0.014

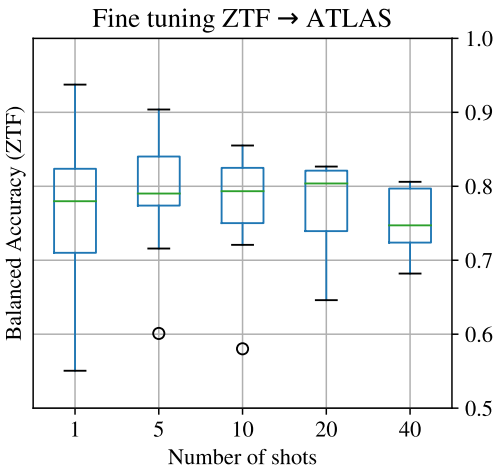
Cuadro 20: BACC de modelos ZTF→ATLAS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 30 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

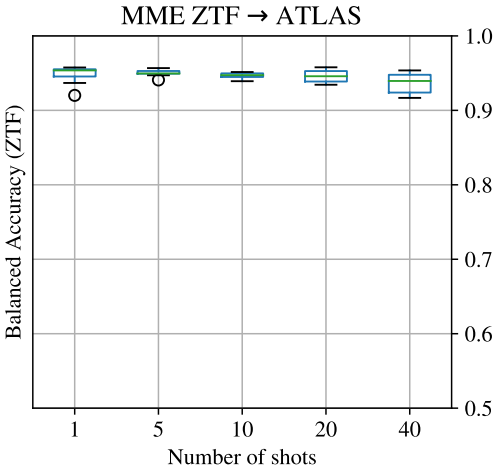
shots	Fine tuning	MME
1	0.749 ± 0.159	0.682 ± 0.018
5	0.852 ± 0.052	0.717 ± 0.041
10	0.870 ± 0.043	0.725 ± 0.057
20	0.898 ± 0.034	0.818 ± 0.042
40	0.920 ± 0.018	0.846 ± 0.036

Cuadro 21: BACC de modelos ZTF→ATLAS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

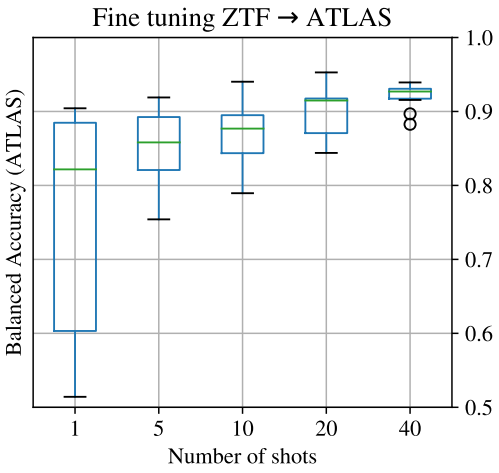
Para estos experimentos, se observan muchas diferencias importantes entre fine tuning y MME en el conjunto target. Fine tuning supera a MME para toda cantidad de shots excepto 1 shot, para el cual fine tuning muestra tanta incertidumbre que no es posible dar un veredicto. Ambos modelos, fine tuning y MME, superan el baseline a partir de los 5-shot, y mejoran el desempeño a medida que se aumenta la cantidad de shots. En el conjunto source, sin embargo, MME consigue llegar a un BACC más alto, y con menos incertidumbre desde los 1-shot hasta los 40-shot, superando el 0.9 para todas las instancias. Fine tuning, por otra parte, muestra barras de error muy grandes y una tendencia a disminuir el accuracy a medida que se aumenta la cantidad de shots.



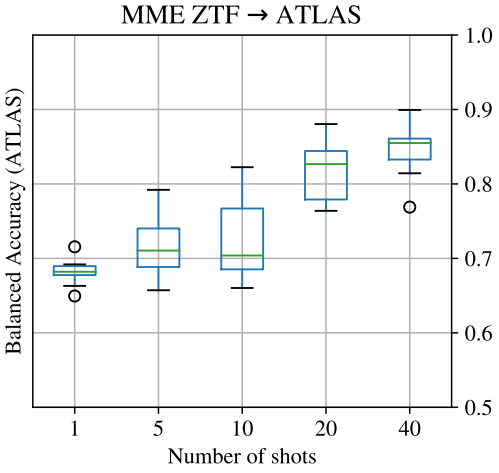
(a) Fine-tuning ZTF→ATLAS evaluado en source.



(b) MME ZTF→ATLAS evaluado en source.



(c) Fine-tuning ZTF→ATLAS evaluado en target.



(d) MME ZTF→ATLAS evaluado en target.

Figura 30: Distribución del BACC para modelos ZTF→ATLAS.

6.2.11. Source ZTF, Target DES

En el cuadro 22 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ZTF), mientras que en el cuadro 23 se muestran los mismos modelos evaluados sobre el conjunto target (DES).

shots	Fine tuning	MME
1	0.813 ± 0.139	0.947 ± 0.006
5	0.918 ± 0.023	0.948 ± 0.007
10	0.917 ± 0.022	0.947 ± 0.005
20	0.898 ± 0.047	0.947 ± 0.004
40	0.889 ± 0.039	0.945 ± 0.007

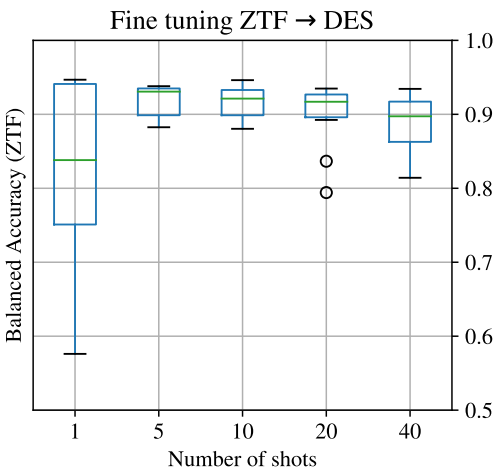
Cuadro 22: BACC de modelos ZTF→DES evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 31 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

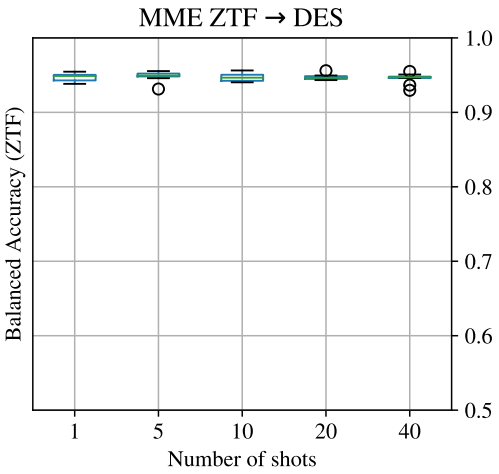
shots	Fine tuning	MME
1	0.686 ± 0.063	0.792 ± 0.022
5	0.787 ± 0.023	0.792 ± 0.022
10	0.812 ± 0.038	0.805 ± 0.028
20	0.848 ± 0.019	0.834 ± 0.031
40	0.868 ± 0.010	0.851 ± 0.023

Cuadro 23: BACC de modelos ZTF→DES evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

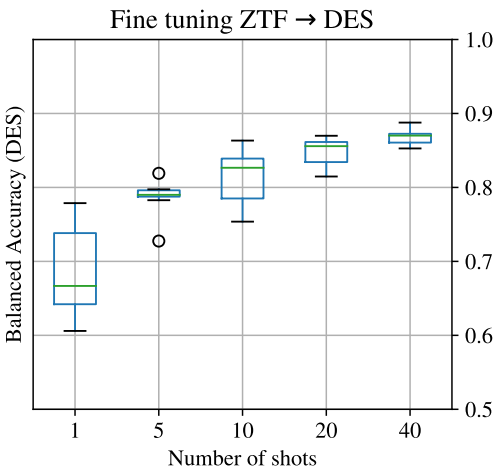
Para estos experimentos, no se observa mucha variación entre fine tuning y MME a partir de 5-shot en el conjunto target. Sin embargo, para 1-shot, MME consigue llegar a un BACC más alto que fine tuning. Ambos modelos, MME y fine tuning logran superar el accuracy promedio del baseline: MME desde el 1-shot y fine tuning desde el 5-shot. MME es capaz de conseguir estos resultados sin empeorar su desempeño en el conjunto source, manteniéndose siempre sobre el 0.9 para toda cantidad de shots. Por otra parte, si bien fine tuning en source es capaz de llegar sobre el 0.9 de BACC para por lo menos 1 instancia con cualquier cantidad de shots, este modelo presenta outliers que llegan incluso por debajo de 0.6 para 1-shot, y en general, barras de error mucho más grandes que MME.



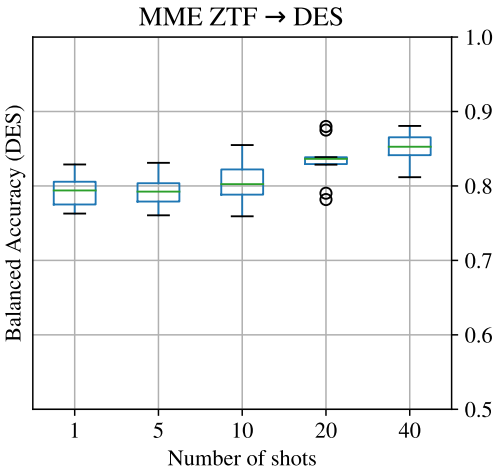
(a) Fine-tuning ZTF→DES evaluado en source.



(b) MME ZTF→DES evaluado en source.



(c) Fine-tuning ZTF→DES evaluado en target.



(d) MME ZTF→DES evaluado en target.

Figura 31: Distribución del BACC para modelos ZTF→DES.

6.2.12. Source ZTF, Target HiTS

En el cuadro 24 se muestran el promedio y las desviaciones estándar del BACC de los modelos de esta combinación source-target, evaluados sobre su conjunto source (ZTF), mientras que en el cuadro 25 se muestran los mismos modelos evaluados sobre el conjunto target (HiTS).

shots	Fine tuning	MME
1	0.904 ± 0.067	0.952 ± 0.005
5	0.905 ± 0.087	0.942 ± 0.027
10	0.917 ± 0.025	0.947 ± 0.005
20	0.905 ± 0.030	0.949 ± 0.006
40	0.892 ± 0.050	0.946 ± 0.006

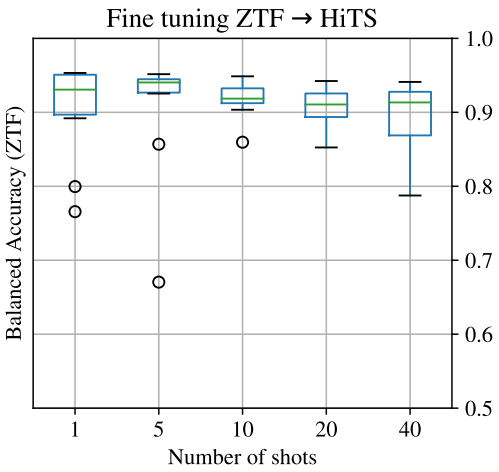
Cuadro 24: BACC de modelos ZTF→HiTS evaluados en source. Media y desviaciones estándar usando 10 particionados distintos.

En la figura 32 se muestran las distribuciones del BACC de los 10 particionados, en función de la cantidad de shots utilizados para el entrenamiento.

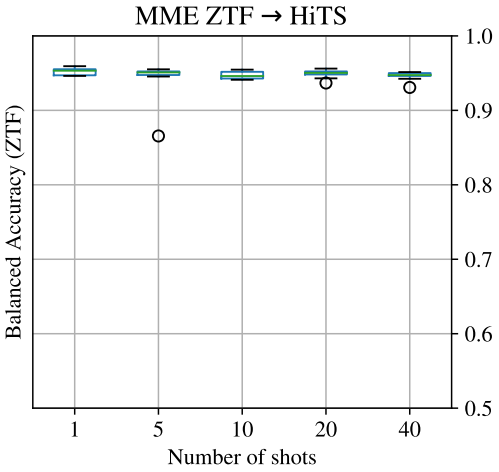
shots	Fine tuning	MME
1	0.815 ± 0.083	0.786 ± 0.043
5	0.865 ± 0.034	0.776 ± 0.033
10	0.897 ± 0.026	0.846 ± 0.036
20	0.913 ± 0.019	0.840 ± 0.042
40	0.920 ± 0.007	0.873 ± 0.023

Cuadro 25: BACC de modelos ZTF→HiTS evaluados en target. Media y desviaciones estándar usando 10 particionados distintos.

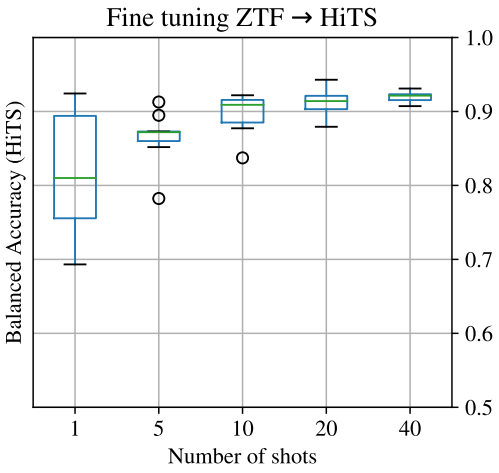
Para estos experimentos, fine tuning muestra un mejor desempeño sobre el conjunto target que MME, con la excepción de 1-shot, donde ambos modelos muestran rangos solapados. Ambos modelos consiguen superar el baseline: fine tuning a partir de los 5-shot, y MME a partir de los 10-shot. Además, ambos modelos tienden a aumentar el BACC a medida que se aumenta la cantidad de shots, aunque MME lo hace con algunas oscilaciones. Para el conjunto source, MME es superior a fine tuning, dado que MME mantiene su accuracy sobre el 0.9 para todas las instancias (excepto un outlier en 10-shot), mientras que fine tuning, si bien logra llegar sobre el 0.9 para algunas instancias en cada cantidad de shots, también presenta outliers que llegan bajo el 0.7 de accuracy.



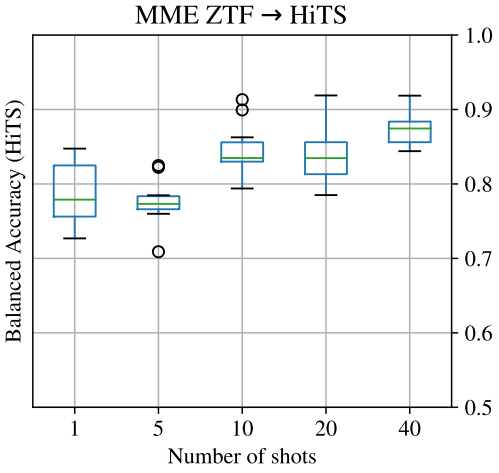
(a) Fine-tuning ZTF→HiTS evaluado en source.



(b) MME ZTF→HiTS evaluado en source.



(c) Fine-tuning ZTF→HiTS evaluado en target.



(d) MME ZTF→HiTS evaluado en target.

Figura 32: Distribución del BACC para modelos ZTF→HiTS.

6.3. Resumen de resultados

6.3.1. Baseline

Cuando se utiliza el modelo de clasificación base entrenado con cualquiera de los conjuntos de datos astronómicos presentados y se evalúa sobre un conjunto de datos distinto al de entrenamiento, el desempeño en términos de Balanced Accuracy se ve disminuido para cada uno de los pares de conjuntos de datos posibles. El caso menos extremo es la combinación DES→HiTS, donde el BACC sobre HiTS medido en el modelo entrenado en DES se reduce en un 3.8 % con respecto al modelo base entrenado en HiTS. El caso más extremo corresponde a la combinación HiTS→ZTF, para la cual el BACC sobre ZTF se reduce en un 46 % con respecto al modelo base entrenado con ZTF.

6.3.2. Fine tuning versus MME

La primera observación respecto a estos modelos, es que todos son capaces de superar el BACC promedio del modelo baseline sobre el conjunto target a partir de alguna cantidad de shots. Esto quiere decir que basta con solo tener una pequeña cantidad de datos etiquetados de target, un modelo preentrenado sobre otro conjunto de datos de imágenes astronómicas, y la técnica de fine tuning o de entrenamiento adversario de MME para lograr un BACC superior al de un modelo pre entrenado sobre otro conjunto datos de imágenes astronómicas. En el cuadro 26 se muestra la cantidad de shots requerida para cada combinación source target posible.

En general con 1 o 5 shots basta, excepto cuando el conjunto de datos es HiTS, ya que este es el conjunto de datos mejor predicho cuando el conjunto de entrenamiento en baseline es otro distinto a HiTS. Notable es el caso DES → HiTS, aquél donde el BACC de baseline es el que menos disminuye. Para esta combinación, fine tuning necesita 20 o más shots para alcanzar el BACC promedio de baseline, mientras que MME requiere solamente 1-shot.

En el cuadro 27 se muestra una comparación de fine tuning versus MME, basada en sus BACC relativos. En el conjunto de datos source, MME es consistentemente el modelo ganador. En el conjunto de datos target, ambos son capaces de superar el BACC promedio de baseline a partir de alguna cantidad de shots, y ambos modelos muestran tendencias al alza del BACC a medida que se aumenta lla cantidad de shots. Para target, en la mayoría de los escenarios las distribuciones de los BACC de fine tuning y MME para la misma cantidad de shots están lo suficientemente solapados, por lo que no se puede decir que uno es mejor que el otro. En el resto de los experimentos, fine tuning o MME superan al otro, pero solo para un rango limitado de shots. A pesar de esto, MME suele presentar un buen desempeño en las cantidades más bajas de shots (1 y 5) con respecto a fine tuning, y consistentemente aumenta su BACC en target

Experimento	Fine tuning	MME
ATLAS → DES	5	5
ATLAS → HiTS	10	10
ATLAS → ZTF	1	1
DES → ATLAS	1	1
DES → HiTS	20	1
DES → ZTF	1	1
HiTS → ATLAS	1	1
HiTS → DES	5	1
HiTS → ZTF	1	1
ZTF → ATLAS	5	5
ZTF → DES	5	1
ZTF → HiTS	5	10

Cuadro 26: Cantidad de datos etiquetados por clase necesarios para superar al modelo baseline.

Experimento	Source	Target
ATLAS → DES	MME	No se observan diferencias significativas
ATLAS → HiTS	MME	Fine tuning, desde 20-shot
ATLAS → ZTF	MME	Fine tuning, desde 10-shot
DES → ATLAS	MME	No se observan diferencias significativas
DES → HiTS	MME	MME, hasta 10 shot
DES → ZTF	MME	No se observan diferencias significativas
HiTS → ATLAS	MME	No se observan diferencias significativas
HiTS → DES	MME	MME, hasta 5-shot
HiTS → ZTF	MME	No se observan diferencias significativas
ZTF → ATLAS	MME	Fine tuning, desde 5-shot
ZTF → DES	MME	MME, para 1-shot solamente
ZTF → HiTS	MME	Fine-tuning, desde 5-shot

Cuadro 27: Resumen de comparación Fine tuning y MME. En cada fila del cuadro se comparan los resultados de los experimentos para un par de conjuntos-source y target dado. En la columna “Source”, se indica cual de los 2 modelos (fine tuning o MME) fue mejor que el otro en términos de BACC sobre el conjunto de prueba “source”. En la columna “Target”, se indica cual de los 2 modelos (fine tuning o MME) fue mejor que el otro en términos de BACC sobre el conjunto de prueba “target”. Cuando la cantidad de shots está escrita, ésta indica en que rangos de número de shots un modelo fue capaz de superar al otro, siendo los valores restantes comparables. Cuando se omite cantidad de shots, quiere decir que el modelo indicado es superior para toda cantidad de shots (1, 5, 10, 20 y 40).

sin empeorar drásticamente su desempeño en source, lo cual no es el caso para fine tuning.

7. Conclusiones

Se ha observado que en el marco de los catálogos de alertas astronómicas, el fenómeno del domain shift está presente: modelos de clasificación ven sus exactitudes disminuidas, cuando el conjunto de prueba proviene de un dominio diferente que el conjunto de entrenamiento. Esta disminución es asimétrica, y se percibe en mayor medida al utilizar conjuntos de entrenamiento y prueba provenientes de cámaras distintas (ZTF versus HiTS, y ZTF versus DES).

Se ha conseguido mitigar el fenómeno del domain shift en la clasificación de alertas astronómicas mediante el fine tuning y la adaptación de dominio. Ambas técnicas consiguen mejorar el desempeño con respecto a un baseline, a veces utilizando tan poco como 1 objeto etiquetado por clase. MME puede llegar a ser mejor que fine tuning al ser evaluado sobre el conjunto target, aunque no en todos los casos. Sin embargo, MME es siempre y consistentemente mejor sobre el conjunto source, independiente de la cantidad de shots utilizados. Esto es bueno, pues permite reutilizar el modelo entrenado para ambos dominios.

Como trabajo futuro, se espera que se pueda aplicar el modelo presentado en este trabajo para la clasificación de alertas de LSST. Tal vez, con los modelos pre entrenados existentes y solamente 1 objeto etiquetado por clase proveniente del LSST se consiga un buen modelo de clasificación para los terabytes de datos producidos por noche por dicho telescopio.

Glosario

AGN: Active Galactic Nucleus.

ANID: Agencia Nacional de Investigación y Desarrollo.

ATLAS: Asteroid Terrestrial-impact Last Alert System.

BACC: Balanced Accuracy.

CNN: Convolutional Neural Network.

CONICYT: Comisión Nacional de Investigación Científica y Tecnológica.

DA: Domain Adaptation.

DANN: Domain Adversarial Neural Network.

DeCam: Dark Energy Camera.

DES: Dark Energy Survey.

DIICC: Departamento de Ingeniería Informática y Ciencias de la Computación.

FITS: Flexible Image Transport System.

FN: False negatives.

FP: False Positives.

GAN: Generative Adversarial Network.

GIF: Graphics Interchange Format.

GRL: Gradient Reversal Layer.

HiTS: High-cadence Transient Survey.

LSST: Large Synoptic Survey Telescope.

MLP: Multylayered Perceptron.

MME: Minimax Entropy.

MNIST: Modified National Institute of Standards and Technology (database).

NaN: Not a Number.

ReLU: Rectifier Linear Unit.

SGD: Stochastic Gradient Descent.

SN: Supernova.

TL: Transfer Learning.

TN: True negatives.

TP: True Positives.

USPS: United States Postal Service.

VS: Variable Star.

ZTF: Zwicky Transient Facility.

Referencias

[1] S. J. Thomas, J. Barr, S. Callahan, A. W. Clements, F. Daruich, J. Fabrega, P. Ingraham, W. Gressler, F. Munoz, D. Neill, *et al.*, "Vera c. rubin observatory: telescope and site status," in *Ground-based and Airborne Telescopes VIII*, vol. 11445, p. 114450I, International Society for Optics and Photonics, 2020.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[6] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*, pp. 213–226, Springer, 2010.

[7] K. D. Gupta, R. Pampana, R. Vilalta, E. E. Ishida, and R. S. de Souza, "Automated supernova ia classification using adaptive learning techniques," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, IEEE, 2016.

[8] A. Mahabal, U. Rebbapragada, R. Walters, F. J. Masci, N. Blagorodnova, J. van Roestel, Q.-Z. Ye, R. Biswas, K. Burdge, C.-K. Chang, *et al.*, "Machine learning for the zwicky transient facility," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 997, p. 038002, 2019.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon, "Neural network recognizer for hand-written zip code digits," in *Proceedings of the 1st International Conference on Neural Information Processing Systems*, NIPS'88, (Cambridge, MA, USA), p. 323–331, MIT Press, 1988.

[11] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, (Berlin,

Heidelberg), p. 213–226, Springer-Verlag, 2010.

[12] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 8050–8058, 2019.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[14] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[15] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, Dec 1943.

[16] Glosser.ca, "Ann dependency (graph)." `https://en.wikipedia.org/wiki/File:Ann_dependency_(graph).svg`. [Online; accessed 8-June-2020].

[17] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[18] Y.-T. Zhou and R. Chellappa, "Computation of optical flow using a neural network.," in *ICNN*, pp. 71–78, 1988.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[22] R. Barandela, R. M. Valdovinos, J. S. Sánchez, and F. J. Ferri, "The imbalanced training sample problem: Under or over sampling?," in *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)*, pp. 806–814, Springer, 2004.

[23] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, p. 2096–2030, Jan. 2016.

[24] B. Flaugher, H. Diehl, K. Honscheid, T. Abbott, O. Alvarez, R. Angstadt, J. Annis, M. Antonik, O. Ballester, L. Beaufore, *et al.*, "The dark energy camera," *The Astronomical Journal*, vol. 150, no. 5, p. 150, 2015.

[25] E. C. Bellm, S. R. Kulkarni, M. J. Graham, R. Dekany, R. M. Smith, R. Riddle, F. J. Masci, G. Helou, T. A. Prince, S. M. Adams, *et al.*, "The zwicky transient facility: system overview, performance, and first results," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 995, p. 018002, 2018.

[26] F. Förster, G. Cabrera-Vives, E. Castillo-Navarrete, P. Estévez, P. Sánchez-Sáez, J. Arredondo, F. Bauer, R. Carrasco-Davis, M. Catelan, F. Elorrieta, *et al.*, "The automatic learning for the rapid classification of events (alerce) alert broker," *arXiv preprint arXiv:2008.03303*, 2020.

[27] G. Csurka, *Domain Adaptation for Visual Applications: A Comprehensive Survey*. 09 2017.

[28] W. Mei and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, 02 2018.

[29] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two-sample-problem," in *Advances in neural information processing systems*, pp. 513–520, 2007.

[30] H. Daumé III, "Frustratingly easy domain adaptation," *arXiv preprint arXiv:0907.1815*, 2009.

[31] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *European conference on computer vision*, pp. 443–450, Springer, 2016.

[32] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[33] M. Long, Z. Cao, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems*, pp. 1640–1650, 2018.

[34] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7167–7176, 2017.

[35] H. Nam, H. Lee, J. Park, W. Yoon, and D. Yoo, "Reducing domain gap via style-agnostic networks," *arXiv preprint arXiv:1910.11645*, 2019.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[37] H. Zou, Y. Zhou, J. Yang, H. Liu, H. P. Das, and C. J. Spanos, "Consensus adversarial domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5997–6004, 2019.

[38] X. Xu, X. Zhou, R. Venkatesan, G. Swaminathan, and O. Majumder, "d-sne: Domain adaptation using stochastic neighborhood embedding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2492–2501, 2019.

[39] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, and K. Saenko, "Visda: The visual domain adaptation challenge," *arXiv preprint arXiv:1710.06924*, 2017.

[40] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[41] Z. Cao, L. Ma, M. Long, and J. Wang, "Partial adversarial domain adaptation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 135–150, 2018.

[42] P. Panareda Busto and J. Gall, "Open set domain adaptation," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 754–763, 2017.

[43] S. Bailey, C. Aragon, R. Romano, R. C. Thomas, B. A. Weaver, and D. Wong, "How to find more supernovae with less work: Object classification techniques for difference imaging," *The Astrophysical Journal*, vol. 665, no. 2, p. 1246, 2007.

[44] D. Wright, S. Smartt, K. Smith, P. Miller, R. Kotak, A. Rest, W. Burgett, K. Chambers, H. Flewelling, K. Hodapp, *et al.*, "Machine learning for transient discovery in pan-starrs1 difference imaging," *Monthly Notices of the Royal Astronomical Society*, vol. 449, no. 1, pp. 451–466, 2015.

[45] D. A. Goldstein, C. Dandrea, J. A. Fischer, R. J. Foley, R. R. Gupta, R. Kessler, A. G. Kim, R. C. Nichol, P. Nugent, A. Papadopoulos, M. Sako, M. Smith, M. Sullivan, R. C. Thomas, W. Wester, R. C. Wolf, F. B. Abdalla, M. Banerji, A. Benoit-levy, E. Bertin, D. Brooks, A. C. Rosell, F. J. Castander, L. N. da Costa, R. Covarrubias, D. L. Depoy, S. Desai, H. T. Diehl, P. Doel, T. F. Eifler, A. F. Neto, D. A. Finley, B. L. Flaugher, P. Fosalba, J. A. Frieman, D. W. Gerdes, D. Gruen, R. A.

Gruendl, D. J. James, K. W. Kuehn, N. Kuropatkin, O. Lahav, T. S. Li, M. A. G. Maia, M. Makler, M. March, J. L. Marshall, P. Martini, K. W. Merritt, R. Miquel, B. D. Nord, R. L. C. Ogando, A. A. Plazas, A. K. Romer, A. Roodman, E. Sánchez, V. E. Scarpine, M. Schubnell, I. Sevilla-Noarbe, R. C. Smith, M. Soares-santos, F. Sobreira, E. Suchyta, M. E. C. Swanson, G. Tarl'e, J. Thaler, and A. R. Walker, "Automated transient identification in the dark energy survey," 2015.

[46] D. A. Duev, A. Mahabal, F. J. Masci, M. J. Graham, B. Rusholme, R. Walters, I. Karmarkar, S. Frederick, M. M. Kasliwal, U. Rebbapragada, *et al.*, "Real-bogus classification for the zwicky transient facility using deep learning," *Monthly Notices of the Royal Astronomical Society*, vol. 489, no. 3, pp. 3582–3590, 2019.

[47] G. Cabrera-Vives, I. Reyes, F. Förster, P. Estevez, and J. Maureira, "Deep-hits: Rotation invariant convolutional neural network for transient detection," *The Astrophysical Journal*, vol. 836, 01 2017.

[48] E. Reyes, P. A. Estévez, I. Reyes, G. Cabrera-Vives, P. Huijse, R. Carrasco, and F. Forster, "Enhanced rotational invariant convolutional neural network for supernovae detection," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[49] R. Carrasco-Davis, G. Cabrera-Vives, F. Förster, P. A. Estévez, P. Huijse, P. Protopapas, I. Reyes, J. Martínez-Palomera, and C. Donoso, "Deep learning for image sequence classification of astronomical events," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 1004, p. 108006, 2019.

[50] R. Vilalta, K. D. Gupta, and L. Macri, "Domain adaptation under data misalignment: An application to cepheid variable star classification," in *2014 22nd International Conference on Pattern Recognition*, pp. 3660–3665, IEEE, 2014.

[51] R. Vilalta, K. D. Gupta, D. Boumber, and M. M. Meskhi, "A general approach to domain adaptation with applications in astronomy," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 1004, p. 108008, 2019.

[52] M. Pérez-Carrasco, G. Cabrera-Vives, M. Martinez-Marín, P. Cerulo, R. Demarco, P. Protopapas, J. Godoy, *et al.*, "Multiband galaxy morphologies for clash: a convolutional neural network transferred from candels," *Publications of the Astronomical Society of the Pacific*, vol. 131, no. 1004, p. 108002, 2019.

[53] M. Pérez-Carrasco, "Matching embeddings for semi-supervised domain adaptation," 2019.

[54] D. C. Wells and E. W. Greisen, “Fits-a flexible image transport system,” in *Image Processing in Astronomy*, p. 445, 1979.

[55] J. Miano, *Compressed image file formats: Jpeg, png, gif, xbm, bmp*. Addison-Wesley Professional, 1999.

[56] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, “The numpy array: a structure for efficient numerical computation,” *Computing in science & engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[57] yu4u, “Convnet drawer.” `https://github.com/yu4u/convnet-drawer`. [Online; accessed 07-June-2020].

[58]