An image is worth 16x16 words: Transformers for image recognition at scale

Cristhian Wiki Sanchez Sauñe
ACECOM IA
Lima, Peru
csanchez@uni.pe

Abstract

Si bien la arquitectura Transformer se ha convertido en el estándar de facto para tareas de procesamiento del lenguaje, sus aplicaciones en visión por computadora siguen siendo limitadas. En visión, la atención se aplica en conjunto con redes convolucionales, o utilizado para reemplazar ciertos componentes de las redes convolucionales manteniendo su estructura general en su lugar. En este trabajo los autores demuestran que esta dependencia de las CNN no es necesaria y un Transformer puro, aplicado directamente a secuencias de parches de una imagen, puede realizar muy bien varias tareas de clasificación de imágenes. Cuando se pre-entrena en grandes cantidades de datos y transferidos a múltiples benchmarks de reconocimiento de imágenes de tamaño mediano o pequeño (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) logra excelentes resultados en comparación con las redes convolucionales de última generación, al tiempo que requieren sustancialmente menos recursos computacionales para entrenarse[1].

1 Introducción

Las arquitecturas basadas en la self-attention, en particular Transformers (Vaswani et al. [1], 2017), se han convertido en el modelo de elección en el Procesamiento del Lenguaje Natural (NLP). El enfoque dominante es el pre-entrenamiento en un corpus de texto grande y luego ajustarlo en un conjunto de datos específico de tarea más pequeño (Devlin et al. [2], 2019). Gracias a la escalabilidad y eficiencia computacional de los Transformers, se ha hecho posible entrenar modelos de tamaño sin precedentes, con más de 100 billones de parámetros. Con el crecimiento de modelos y conjuntos de datos, todavía no hay señales de saturación del rendimiento.

En el campo de la visión por computadora, sin embargo, las arquitecturas convolucionales siguen siendo dominantes (LeCun et al. [3], 1989; Krizhevsky y col. [4], 2012; He et al. [5], 2016). Inspirados por los éxitos en NLP, varios trabajos intentan combinar arquitecturas similares a las de CNN con self-attention (Wang et al. [6], 2018; Carion et al. [7], 2020), algunos reemplazando las convoluciones por completo (Ramachandran et al. [8], 2019; Wang et al. [9], 2020a). Los últimos modelos, si bien son teóricamente eficientes, aún no se han escalado de manera efectiva en aceleradores de hardware modernos (GPU's y TPU's) debido a el uso de patrones de atención especializados. Por lo tanto, en el reconocimiento de imágenes a gran escala, las arquitecturas clásicas como ResNet (en la figura 1 se muestra la arquitectura ResNet-50) siguen siendo de vanguardia (Mahajan et al. [10], 2018; Xie et al. [11], 2020; Kolesnikov et al. [12], 2020).

Los autores de este trabajo se inspiraron en los éxitos de escalamiento de Transformer en NLP, aplicaron un Transformer estándar directamente a las imágenes, con la menor cantidad de modificaciones posibles. Para hacerlo, dividieron una imagen en parches (porciones de la imagen

[1] Sin embargo, aún siguen siendo recursos muy poco accesibles para personas principiantes

original) y proporcionaron la secuencia de incrustaciones (o embeddings) lineales de estos parches como entrada a un Transformer. Los parches de imagen se tratan de la misma manera que los tokens (palabras) en una aplicación de NLP. Ellos entrenaron el modelo de clasificación de imágenes de forma supervisada. Cuando se entrena en conjuntos de datos de tamaño mediano como ImageNet, dichos modelos producen precisiones modestas de unos pocos puntos porcentuales por debajo de ResNets de tamaño comparable.

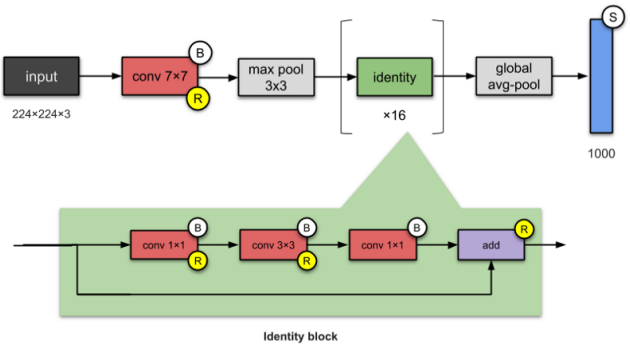


Figure 1: Arquitectura convolucional ResNet-50 [5]

Este resultado aparentemente desalentador puede deberse a que: los Transformers carecen de algunos de los sesgos inductivos inherentes a las CNN, como la invarianza traslacional y de localidad, y por lo tanto no generalizan bien cuando se entrenan con cantidades insuficientes de datos. Sin embargo, el panorama cambia si los modelos se entrenan en conjuntos de datos más grandes (entre 14M-300M imágenes). Se encontró que el entrenamiento a gran escala triunfa sobre el sesgo inductivo. Vision Transformer (ViT) logra excelentes resultados cuando se pre-entrena a una escala suficiente y se transfiere el aprendizaje a tareas con menos datapoints. Con pre-entrenamiento en el conjunto de datos público ImageNet-21k o en el conjunto de datos interno JFT-300M (propiedad de Google), ViT se acerca o supera el estado del arte en múltiples benchmarks de reconocimiento de imágenes. En particular, el mejor modelo alcanza la precisión del 88,55% en ImageNet, el 90,72% en ImageNet-ReaL, el 94,55% en CIFAR-100, y 77,63% en el conjunto de 19 tareas de VTAB.

2 Trabajo Relacionado

Los Transformers fueron propuestos por Vaswani et al. [1] (2017) para la traducción automática, y desde entonces se han convertido en el método más avanzado en muchas tareas de NLP. Los grandes modelos basados en Transformers a menudo se pre-entrenan en un corpus grande y luego se ajustan para la tarea en cuestión: BERT (Devlin et al. [2], 2019) utiliza una tarea de pre-entrenamiento auto-supervisado de eliminación de ruido, mientras que la línea de trabajo GPT utiliza el modelado del lenguaje como tarea de pre-entrenamiento (Radford et al. [13], 2018; 2019; Brown et al. [14], 2020).

La aplicación naive de la self-attention a las imágenes requeriría que cada píxel atienda a los demás píxeles. Con un **costo cuadrático** en el número de píxeles, esto no se escala a tamaños de entrada realistas. Así, para aplicar Transformers en el contexto del procesamiento de imágenes, se han probado varias aproximaciones en el pasado: Parmar et al. [15] (2018) aplicaron la self-attention solo en vecindarios locales para cada píxel de consulta en lugar de globalmente. Dichos bloques de self-attention locales de productos punto de múltiples cabezales (multiheads) pueden reemplazar completamente a las convoluciones (Ramachandran et al. [8], 2019; Cordonnier et al. [16], 2020; Zhao et al. [17], 2020).

Alternativamente, obras como Sparse Transformers (Child et al. [18], 2019) emplean aproximaciones escalables a self-attention para ser aplicable a las imágenes. Una forma alternativa de escalar la atención es aplicarla en bloques de diferentes tamaños (Weissenborn et al. [19], 2019), en el caso

extremo solo a lo largo de ejes individuales (Ho et al. [20], 2019; Wang y col. [9], 2020a). Muchas de estas arquitecturas de atención especializada demuestran resultados prometedores en tareas de visión por computadora, pero requieren una ingeniería compleja para ser implementada eficientemente en aceleradores de hardware.

También ha habido mucho interés en combinar redes neuronales convolucionales (CNN) con formas de self-attention, por ejemplo aumentando los mapas de características para la clasificación de imágenes (Bello et al. [21], 2019) o procesando aún más la salida de una CNN utilizando self-attention, por ejemplo, para la detección de objetos (Hu et al. [22], 2018; Carion et al. [7], 2020, ver figura 2), procesamiento de video (Wang et al. [6], 2018; Sun et al. [23], 2019), clasificación de imágenes (Wu et al. [24], 2020), descubrimiento de objetos sin supervisión (Locatello et al. [25], 2020) o tareas unificadas de visión-texto (Chen et al. [26], 2020c; Lu et al. [27], 2019; Li et al. [28], 2019).

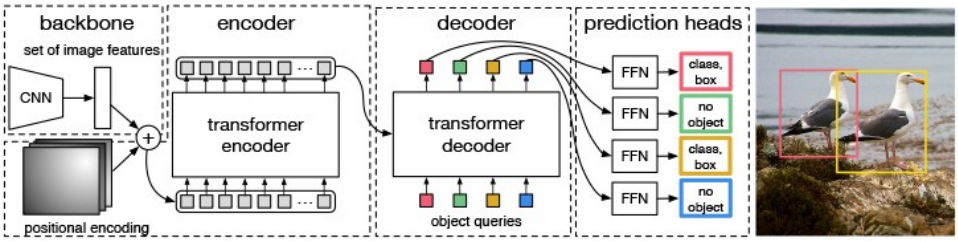


Figure 2: Arquitectura DERT [7] que usa Transformers para la detección de objetos

En el momento de publicación del artículo, los investigadores de ViT, no encontraron alguna aplicación previa de Transformers con self-attention global a imágenes de tamaño completo. El modelo más cercano a ViT, es iGPT (Chen et al. [29], 2020a), que aplica Transformers a los píxeles de la imagen después de reducir la resolución de la imagen y el espacio de color. El modelo se entrena de manera no supervisada como un modelo generativo, y la representación resultante se puede ajustar o probar linealmente para el rendimiento en clasificación, logrando una precisión máxima del 72% en ImageNet. El uso de fuentes de datos adicionales permite lograr resultados de vanguardia en benchmarks estándar (Mahajan et al. [10], 2018; Touvron et al. [30], 2019; Xie et al. [11], 2020).

Además, Sun et al. [31](2017) estudian cómo el desempeño de las CNN's escala con el tamaño del conjunto de datos y Kolesnikov et al. [12] (2020); Djolonga y col. [32] (2020) realizan una exploración empírica del aprendizaje de transferencia de las CNN's a partir de conjuntos de datos a gran escala como ImageNet-21k y JFT-300M. Los autores se centraron en estos dos últimos conjuntos de datos, pero entrenaron Transformers en lugar de los modelos basados en ResNet utilizados en trabajos anteriores.

3 Método

Los autores tratan de seguir el Transformer original[1] lo más cerca posible. Una ventaja de esta configuración intencionalmente simple es que las arquitecturas de Transformers sean escalables y fáciles de implementar.

3.1 Vision Transformer (VIT)

En la figura 3 se muestra una descripción general del modelo. El Transformer estándar recibe como entrada una secuencia 1D de embeddings de tokens. Para manejar imágenes 2D, se hace un reshape a la imagen $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ en una secuencia de parches 2D aplanados $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, donde *(H, W)* es la resolución de la imagen original, *C* es el número de canales, *(P, P)* es la resolución de cada parche de la imagen y $N = \frac{H \cdot W}{P^2}$ es el número resultante de parches, que también sirve como

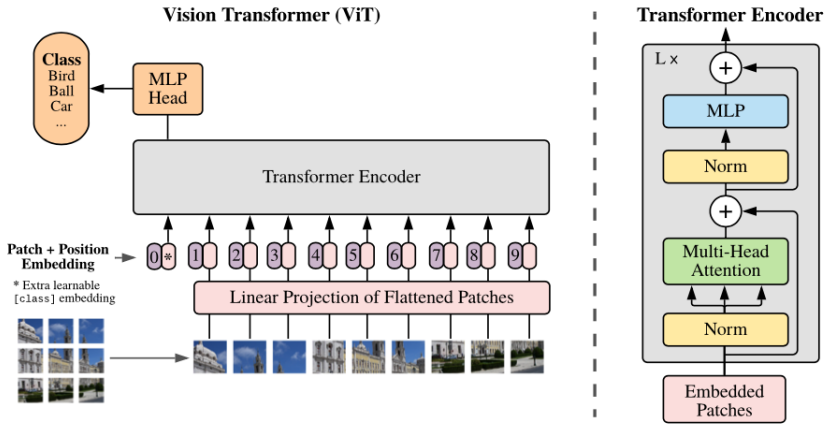


Figure 3: Descripción general del modelo. La imagen fue dividida en parches de tamaño fijo que luego fueron insertadas linealmente. Se agregaron embeddings de posición y posteriormente se introdujo la secuencia resultante al encoder de un Transformer estándar. Para realizar la clasificación, se utilizó el enfoque estándar de agregar un "token de clasificación" extra que se pueda aprender en la secuencia.

longitud de secuencia de *entrada efectiva* para el Transformer.

El Transformer usa un tamaño constante *D* del vector latente a través de todas sus capas, por lo que se aplanan los parches y se mapean a una dimensión *D* con una proyección lineal entrenable (ver ecuación 1). La salida de esta proyección se conoce como *embeddings del parche*. De manera similar al token *[clase]* de BERT, se antepone un embedding que se puede aprender en la secuencia de embeddings del parche ($z_0^0 = x_{class}$), cuyo estado en la salida del encoder del Transformer (z_0^L) sirve como representación de la imagen y de (ecuación 4). Tanto durante el pre-entrenamiento como durante el fine-tuning, se adjunta un cabezal de clasificación a z_0^L. El cabezal de clasificación se implementa mediante un MLP con una capa oculta en el pre-entrenamiento y por una sola capa lineal en el momento del fine-tuning.

Los embeddings de posición se agregan a los embeddings de parches para retener la información de posición. Se usaron embeddings de posición *1D* estándar aprendibles, ya que no se ha observado un desempeño significativo beneficioso del uso de embeddings de posición *2D* más avanzados. La secuencia resultante de embeddings sirve como entrada al encoder. El encoder del Transformer[1] consiste en capas alternas de self-attention de múltiples cabezales y bloques MLP (Ec. 2, 3). El MLP contiene dos capas con función de activación GELU.

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\, \mathbf{x}_p^1\mathbf{E};\, \mathbf{x}_p^2\mathbf{E};\, \cdots;\, \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D} \tag{1}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L \tag{2}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L \tag{3}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{4}$$

Arquitectura híbrida. Como alternativa a los parches de imagen en bruto, la secuencia de entrada se puede formar a partir de mapas de características de una CNN (LeCun et al. [3], 1989). En este modelo híbrido, la proyección de los embeddings de parches **E** (Ec. 1) se aplica a parches extraídos de un mapa de características de la CNN. Como caso especial, los parches pueden tener un tamaño espacial 1x1, lo que significa que la secuencia de entrada se obtiene simplemente aplanando las dimensiones espaciales del mapa de características y proyectando a la dimensión del Transformer. Los embeddings de entrada de clasificación y de posición se agregan como se describió anteriormente.

4 Experimentos

Se evaluaron las capacidades de aprendizaje de representación de ResNet, Vision Transformer (ViT) y el híbrido. Para comprender los requisitos de datos de cada modelo, se realizó un pre-entrenamiento en conjuntos de datos de tamaño variable y se evaluaron muchas tareas de referencia. Al considerar el costo computacional del pre-entrenamiento del modelo, ViT se desempeña muy favorablemente, alcanzando el estado del arte en la mayoría de los benchmarks de reconocimiento en un menor costo de pre-entrenamiento. Por último, se realizó un pequeño experimento con auto-supervisión y se mostró que ViT auto-supervisado es prometedor para el futuro.

4.1 CONFIGURACIÓN

Conjuntos de datos. Para explorar la escalabilidad del modelo, se usó el conjunto de datos ILSVRC-2012 ImageNet con 1k clases y 1.3M imágenes (ImageNet en lo que sigue),también el superconjunto ImageNet-21k con 21k clases y 14M imágenes (Deng et al. [33], 2009), y JFT (Sun et al. [31], 2017) con 18k clases y 303M imágenes de alta resolución.

También se evaluó en el conjunto de clasificación VTAB de 19 tareas (Zhai et al. [34], 2019b). VTAB evalúa la transferencia de datos bajos a diversas tareas, utilizando 1000 ejemplos de entrenamiento por tarea. Las tareas se dividen en tres grupos: Naturales: tareas como las anteriores, mascotas, CIFAR, etc. Especializadas: imágenes médicas y satelitales, y Estructuradas: tareas que requieren comprensión geométrica como la localización.

Variantes de modelo. Basamos las configuraciones de ViT en las utilizadas para BERT (Devlin et al. [2], 2019), como se resume en la Tabla 1. Los modelos "Base" y "Grande-Large" se adoptaron directamente de BERT y se agregó posteriormente el modelo "Enorme-Huge". En lo que sigue se usará una breve notación para indicar el tamaño del modelo y el tamaño del parche de entrada: por ejemplo, ViT-L/16 significa la variante “Grande-Large” con un tamaño de parche de entrada de 16×16. Tenga en cuenta que la longitud de la secuencia del Transformer es inversamente proporcional al cuadrado del tamaño del parche, por lo que los modelos con un tamaño de parche más pequeño son computacionalmente más caros.

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Detalles de las variantes de Visual Transformer

Para las CNN’s de referencia, se usó ResNet (He et al. [5], 2016), pero se reemplazaron las capas de Batch Normalization (Ioffe & Szegedy [35], 2015) por la normalización de grupos (Wu & He [36], 2018), combinado con convoluciones estandarizadas (Salimans & Kingma [37], 2016). Estas modificaciones mejoran la transferencia (Kolesnikov et al. [12], 2020), el modelo modificado se denota como “ResNet (BiT)”. Para los híbridos, se alimentaron los mapas de características intermedias en ViT con un tamaño de parche de un "píxel". Para experimentar con diferentes longitudes de secuencia, o **(i)** se toma la salida de la etapa 4 de un ResNet50 regular o **(ii)** se elimina la etapa 4, colocando el mismo número de capas en la etapa 3 (manteniendo el número total de capas) y tomando la salida de esta etapa extendida 3. La opción (ii) da como resultado una secuencia 4 veces más larga y un modelo ViT más caro.

Entrenamiento y fine-tuning. En el entrenamiento de todos los modelos, incluidos los ResNets, se usó Adam con $\beta_1 = 0.9$, $\beta_2 = 0.999$, un tamaño de lote de **4096** y se aplicó un alto weight-decay de 0.1, que pareció útil para la transferencia de todos los modelos. Para el fine-tuning se utilizó SGD con momentum y un tamaño de lote de 512, para todos modelos. Para los resultados de ImageNet en la Tabla 2, se ajustó a una resolución más alta: 512 para ViT-L/16 y 518 para ViT-H/14.

Métrica. Se muestran los resultados en conjuntos de datos posteriores, ya sea a través de un accuracy de few-shot o de fine-tuning. Los acccuracy’s de fine-tuning capturan el rendimiento de cada modelo después de ajustarlo en el respectivo conjunto de datos. Se obtienen acccuracy’s de few-shot al

resolver un problema de regresión lineal regularizado que mapea la representación (congelada) de un subconjunto de imágenes de entrenamiento a los vectores objetivo (target) $\{-1, 1\}^K$. Aunque el trabajo se centra principalmente en el rendimiento del fine-tuning, a veces también se utilizan acccuracy's lineales de few-shot para una evaluación rápida sobre la marcha (on-the-fly) donde el fine-tuning sería demasiado costoso.

4.2 Comparación con el SOTA.

Primero se comparan modelos más grandes, ViT-H/14 y ViT-L/16, con CNN's de última generación de la literatura. El primer punto de comparación es Big Transfer (BiT) (Kolesnikov et al. [12], 2020), que realiza aprendizaje de transferencia supervisado con grandes ResNets. El segundo es Noisy Student (Xie et al. [11], 2020), que es una gran EfficientNet formada mediante aprendizaje semi-supervisado en ImageNet y JFT-300M con las etiquetas retiradas. Actualmente, Noisy Student es el estado del arte en ImageNet y BiT-L en los otros conjuntos de datos informados aquí. Todos los modelos fueron entrenados en hardware TPUv3, mostrándose la cantidad de TPUv3-core-days necesarios para el pre-entrenamiento de cada uno de ellos, es decir, la cantidad de núcleos de TPUv3 (2 por chip) utilizados para el entrenamiento multiplicado por el tiempo de entrenamiento en días.

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21K (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** $\pm$ 0.04	87.76 $\pm$ 0.03	85.30 $\pm$ 0.02	87.54 $\pm$ 0.02	88.4/88.5*
ImageNet ReaL	**90.72** $\pm$ 0.05	90.54 $\pm$ 0.03	88.62 $\pm$ 0.05	90.54	90.55
CIFAR-10	**99.50** $\pm$ 0.06	99.42 $\pm$ 0.03	99.15 $\pm$ 0.03	99.37 $\pm$ 0.06	–
CIFAR-100	**94.55** $\pm$ 0.04	93.90 $\pm$ 0.05	93.25 $\pm$ 0.05	93.51 $\pm$ 0.08	–
Oxford-IIIT Pets	**97.56** $\pm$ 0.03	97.32 $\pm$ 0.11	94.67 $\pm$ 0.15	96.62 $\pm$ 0.23	–
Oxford Flowers-102	99.68 $\pm$ 0.02	**99.74** $\pm$ 0.00	99.61 $\pm$ 0.02	99.63 $\pm$ 0.03	–
VTAB (19 tasks)	**77.63** $\pm$ 0.23	76.28 $\pm$ 0.46	72.72 $\pm$ 0.21	76.29 $\pm$ 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparación con el estado del arte en los benchmarks más populares de clasificación de imágenes. Se muestra la desviación media y estándar de los acccuracy's, promediadas en tres ejecuciones de fine-tuning. Los modelos de Vision Transformer pre-entrenados en el conjunto de datos JFT-300M superan los baselines basados en ResNet en todos los conjuntos de datos, mientras que requieren sustancialmente menos recursos computacionales para el pre-entrenamiento. ViT pre-entrenado en el conjunto de datos público más pequeño (ImageNet-21k) también funciona bien. *Resultado ligeramente mejorado del 88,5% informado por Touvron et al. [30] (2020).

La tabla 2 muestra los resultados. El modelo ViT-L/16 más pequeño pre-entrenado en JFT-300M supera a BiT-L (que está pre-entrenado en el mismo conjunto de datos) en todas las tareas, mientras que requiere sustancialmente menos recursos computacionales para entrenar. El modelo más grande, ViT-H/14, mejora aún más el rendimiento, especialmente en los conjuntos de datos más desafiantes: ImageNet, CIFAR-100 y la suite VTAB. Curiosamente, este modelo todavía requiere mucho menos cálculo para pre-entrenar que el SOTA anterior. Sin embargo, notamos que la eficiencia del pre-entrenamiento puede verse afectada no solo por la elección de la arquitectura, sino también por otros parámetros, como el programa de entrenamiento (training schedule), el optimizador, la disminución del peso (weight decay), etc. Finalmente, el modelo ViT-L/16 pre-entrenado en el conjunto de datos público ImageNet-21k también funciona bien en la mayoría de los conjuntos de datos, mientras que requiere menos recursos para el pre-entrenamiento: podría entrenarse usando una TPUv3 en la nube estándar con 8 núcleos en aproximadamente 30 días.

La Figura 4 descompone las tareas de VTAB en sus respectivos grupos y se compara con los métodos SOTA anteriores en este benchmark: BiT, VIVI - un ResNet co-entrenado en ImageNet y Youtube (Tschannen et al. [38], 2020) y S4L - aprendizaje supervisado más semi-supervisado en ImageNet (Zhai et al. [39], 2019a). ViT-H/14 supera a BiT-R152x4 y otros métodos en las tareas naturales y estructuradas. En la tarea especializada, el rendimiento de los dos mejores modelos es similar.

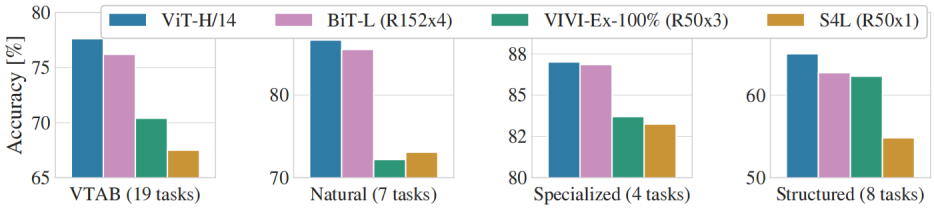


Figure 4: Rendimiento de VTAB en grupos de tareas naturales, especializadas y estructuradas.

4.3 Requisitos de datos para el pre-entrenamiento

Vision Transformer funciona bien cuando se pre-entrena en un gran conjunto de datos tal como JFT-300M. Con menos sesgos inductivos para la visión que ResNets, **¿qué importancia tiene el tamaño del conjunto de datos?** Se realizaron una serie de 2 experimentos. Primero, se pre-entrenaron los modelos ViT en conjuntos de datos de tamaño creciente: ImageNet, ImageNet-21k y JFT-300M. Para obtener el mejor rendimiento posible en los conjuntos de datos más pequeños, se optimizaron tres parámetros de regularización: disminución de peso, dropout y suavizado de etiquetas (label smoothing). La figura 5 (izquierda) muestra los resultados después del fine-tuning en ImageNet. Cuando se pre-entrena en el conjunto de datos más pequeño, ImageNet, los modelos ViT-Large tienen un rendimiento inferior en comparación con los modelos ViT-Base, a pesar de la gran regularización. Sin embargo, con el pre-entrenamiento de ImageNet-21k, sus desempeños son similares. Solo con JFT-300M, se observa el beneficio de modelos más grandes. La Figura 5 también muestra la región de desempeño abarcada por los modelos BiT de diferentes tamaños. Las CNN's de BiT superan a ViT en ImageNet (a pesar de la optimización de la regularización), pero con los conjuntos de datos más grandes, ViT supera.

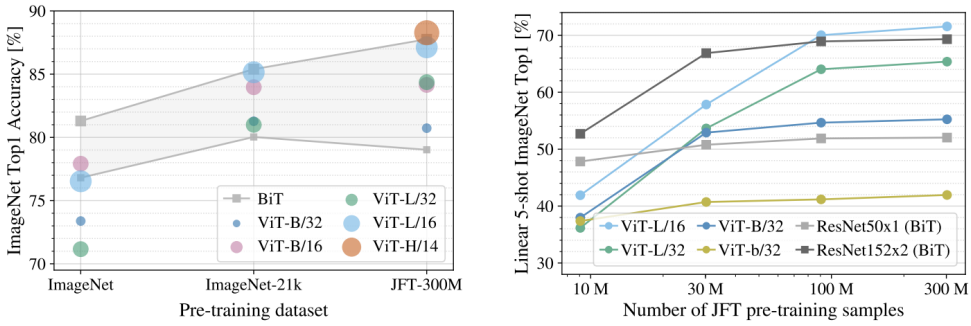


Figure 5: **Transferencia a ImageNet**. (derecha) Si bien los modelos ViT grandes funcionan peor que los BiT ResNets (área sombreada) cuando se pre-entrenan en pequeños conjuntos de datos, brillan cuando se pre-entrenan en conjuntos de datos más grandes. De manera similar, las variantes de ViT más grandes superan a las más pequeñas a medida que crece el conjunto de datos. **Evaluación lineal de few-shot en ImageNet versus tamaño del pre-entrenamiento**. (izquierda) ResNets funciona mejor con conjuntos de datos de pre-entrenamiento más pequeños pero se estabiliza antes que ViT, que funciona mejor con un pre-entrenamiento más grande. ViT-b es ViT-B con todas las dimensiones ocultas reducidas a la mitad.

En segundo lugar, los modelos fueron entrenados en subconjuntos aleatorios de 9M, 30M y 90M, así como el conjunto de datos completo JFT-300M. No se realizó una regularización adicional en los subconjuntos más pequeños y usaron los mismos hiperparámetros para todos los ajustes. De esta forma, se evaluaron las propiedades intrínsecas del modelo, y no los efectos de la regularización. Sin embargo, se usó early-stopping y se informó el mejor accuracy de validación logrado durante el entrenamiento. Para ahorrar cálculo, se mostró un accuracy lineal de few-shot en lugar de un accuracy total de fine-tuning. La figura 5 (derecha) contiene los resultados. Vision Transformer se adapta más que las ResNets con costo computacional comparable en conjuntos de datos más pequeños. Por ejemplo, ViT-B/32 es ligeramente más rápido que ResNet50; funciona mucho peor en el subconjunto 9M, pero mejor en subconjuntos 90M+. Lo mismo es cierto para ResNet152x2 y

ViT-L/16. Este resultado refuerza la intuición de que el sesgo inductivo convolucional es útil para conjuntos de datos más pequeños, pero para los más grandes, aprender los patrones relevantes es suficiente, incluso beneficioso.

En general, los resultados de few-shot en ImageNet (Figura 5 - derecha), así como los resultados de datos bajos (low-data) en VTAB (Tabla 2) parecen prometedoras para transferencias de datos muy bajas. Análisis adicional de las propiedades de few-shot de ViT es una dirección apasionante del trabajo futuro.

4.4 Estudio de escala

Se realizó un estudio de escalado controlado de diferentes modelos evaluando el desempeño de transferencia desde JFT-300M. En esta configuración, el tamaño de los datos no obstaculiza el rendimiento de los modelos, y se evaluó el rendimiento versus costo de pre-entrenamiento de cada modelo. El conjunto de modelos incluye: 7 ResNets, R50x1, R50x2 R101x1, R152x1, R152x2, pre-entrenados para 7 épocas, más R152x2 y R200x3 pre-entrenados durante 14 épocas; 6 Vision Transformers, ViT-B/32, B/16, L/32, L/16, pre-entrenados para 7 épocas, más L/16 y H/14 pre-entrenados durante 14 épocas; y 5 híbridos, R50 + ViT-B/32, B/16, L/ 32, L/16 pre-entrenados para 7 épocas, más R50 + ViT-L/16 pre-entrenados para 14 épocas (para híbridos, el número al final del nombre del modelo no representa el tamaño del parche, sino la proporción de muestreo descendente total en la red backbone de ResNet).

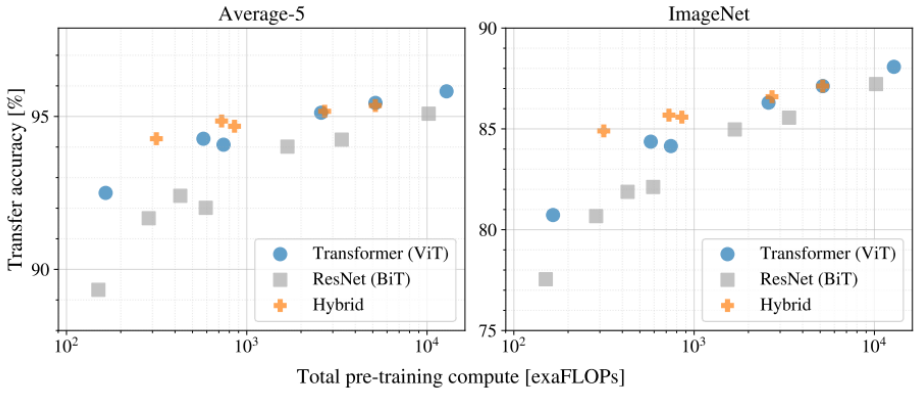


Figure 6: Rendimiento versus costo para diferentes arquitecturas: Vision Transfomers, ResNets e híbridos. Los modelos Vision Transformers generalmente superan a las ResNets con el mismo presupuesto computacional. Los híbridos mejoran los Transformers puros para modelos más pequeños, pero la brecha desaparece para los modelos más grandes.

La Figura 6 contiene el rendimiento de la transferencia frente al cálculo total del pre-entrenamiento. Se pueden observar algunos patrones. En primer lugar, los Vision Transformers dominan las ResNets en la relación rendimiento/cálculo. ViT utiliza aproximadamente 2-4 veces menos procesamiento para lograr el mismo rendimiento (promedio de 5 conjuntos de datos). En segundo lugar, los híbridos superan ligeramente a ViT con presupuestos computacionales pequeños, pero la diferencia desaparece para los modelos más grandes. Este resultado es algo sorprendente, ya que uno podría esperar que el procesamiento de características locales convolucionales ayude a ViT en cualquier tamaño. En tercer lugar, los Vision Transfomers parecen no saturarse dentro del rango probado, lo que motiva futuros esfuerzos de escalado.

4.5 Inspección del Vision Transformer

Para comenzar a comprender cómo Vision Transformer procesa los datos de las imágenes, se analizaron sus representaciones internas. La primera capa del Vision Transformer proyecta linealmente los parches aplanados en un espacio de menor dimensión (Ec. 1). La Figura 8 (izquierda) muestra los principales componentes de los filtros de embedding aprendidos. Los componentes se

asemejan a funciones de base plausibles para una representación de baja dimensión de la estructura fina dentro de cada parche.

Después de la proyección, se agrega un embedding de posición (aprendida) a las representaciones del parche. La figura 8 (centro) muestra que el modelo aprende a codificar la distancia dentro de la imagen en la similitud de embeddings de posición, es decir, los parches más cercanos suelen tener embeddings de posición más parecidas. Además, aparece la estructura fila-columna; los parches en la misma fila/columna tienen embeddings similares. El hecho de que los embeddings de posición aprendan a representar la topología de imagen 2D explica por qué las variantes de embedding con reconocimiento de 2D creadas a mano no producen mejoras.

Self-attention permite a ViT integrar información en todo el imagen incluso en las capas más bajas. Se investigó hasta qué punto la red hace uso de esta capacidad. Específicamente, se calculó la distancia promedio en el espacio de la imagen a través de la cual se integra la información, según los pesos de atención (Figura 8, derecha). Esta "distancia de atención" es análoga al tamaño del campo receptivo en las CNN's. Se encontró que algunos cabezales prestan atención a la mayor parte de la imagen ya en las capas más bajas, lo que demuestra que la capacidad de integrar información globalmente es de hecho utilizada por el modelo. Otros cabezales de atención tienen distancias de atención consistentemente pequeñas en las capas bajas. Esta atención altamente localizada es menos pronunciada en modelos híbridos que aplican una ResNet antes del Transformer (Figura 8, derecha), lo que sugiere que puede tener una función similar a las primeras capas convolucionales en las CNN's. Más lejos, la distancia de atención aumenta con la profundidad de la red. Globalmente, encontramos que el modelo atiende a regiones de imagen que son semánticamente relevantes para la clasificación (Figura 7).

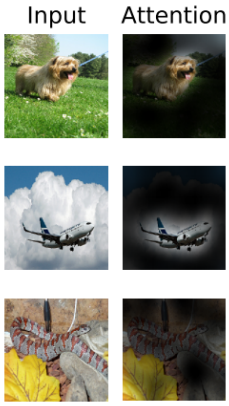


Figure 7: Ejemplos representativos de atención desde el token de salida hasta el espacio de entrada.

4.5.1 Otras técnicas de inspección

Explorando más técnicas para poder visualizar cómo funciona Vision Transfomer, se ha encontrado que la investigación de Jacob Gildenblat [40] "Exploring Explainability for Vision Transformers" hace un buen trabajo al respecto. Dicha investigación intenta conocer cómo fluye la atención desde el principio hasta el final a través de Vision Transformer. Para cuantificar esto se ha utilizado una técnica llamada "Attention Rollout" del paper "Quantifying Attention Flow in Transformers" (Samira Abnar & Willem Zuidema [41], 2020).

En cada bloque Transformer se recibe una matriz de atención A_{ij} que define cuánta atención va a fluir desde el token *j* en la capa anterior al token *i* en la siguiente capa.

Se puede multiplicar las matrices entre cada dos capas, para obtener el flujo de atención total entre ellas. Sin embargo, también hay conexiones residuales (propias de la arquitectura Transformer), que pueden ser modeladas añadiendo la matriz identidad **I** a la matriz de atención de la capa actual: $A_{ij} + I$.
En caso se cuente con múltiples cabezales de atención, [41] sugiere tomar el promedio de las

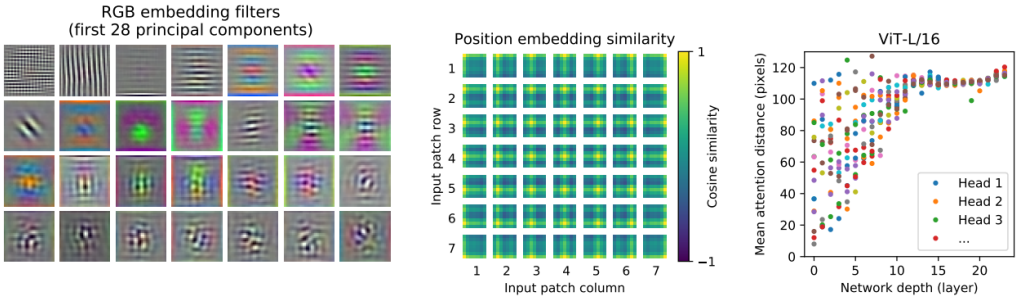


Figure 8: **Izquierda:** Filtros del embedding lineal inicial de valores RGB de ViT-L/32. **Centro:** Similitud de embeddings de posición de ViT-L/32. Los mosaicos muestran la similitud de coseno (cosine similarity) entre el embedding de posición del parche con la fila y columna indicadas y la posición de los embeddings de todos los demás parches. **Derecha:** Tamaño del área atendida por cabezal y profundidad de la red. Cada punto muestra la distancia de atención media en las imágenes para uno de los 16 cabezales en una capa.

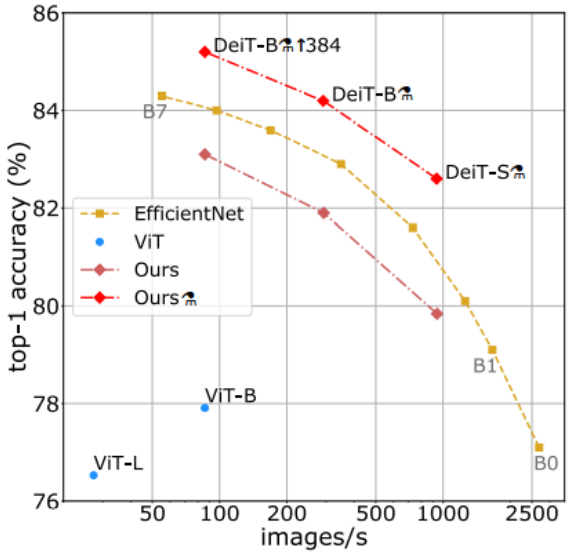


Figure 9: Rendimiento y accuracy en ImageNet de DeiT en comparación con EfficientNets, entrenados solo en ImageNet1k. El rendimiento se mide como el número de imágenes procesadas por segundo en una GPU V100. DeiT-B es idéntico a ViT-B, pero el entrenamiento está más adaptado a un régimen de escasez de datos. Se aprende en unos días en una sola máquina. El símbolo extra se refiere a modelos entrenados con una destilación específica para Transfomers.

cabezales. Sin embargo, también puede tener sentido usar otras opciones (como sugiere Jacob [40]): como el mínimo, el máximo o el uso de diferentes pesos.

Finalmente se obtiene una manera de calcular recursivamente la matriz "Attention Rollout" en la capa L:

$$AttentionRollout_L = (A_L + I)\dot{A}ttentionRollout_{L-1}$$

Se debe normalizar las filas, para mantener el flujo de atención total en 1.

Se probó la técnica mediante el promedio de los cabezales en la arquitectura 'Data Efficient - DeiT' [42] de Facebook (una arquitectura mucho más reciente que ViT y optimizada para ser entrenada en pocos datos mediante técnicas de destilación avanzadas, cuyo rendimiento se observa en la figura 9), pero los resultados no fueron tan agradables como en ViT (ver figura 10 - central).

Figure 10

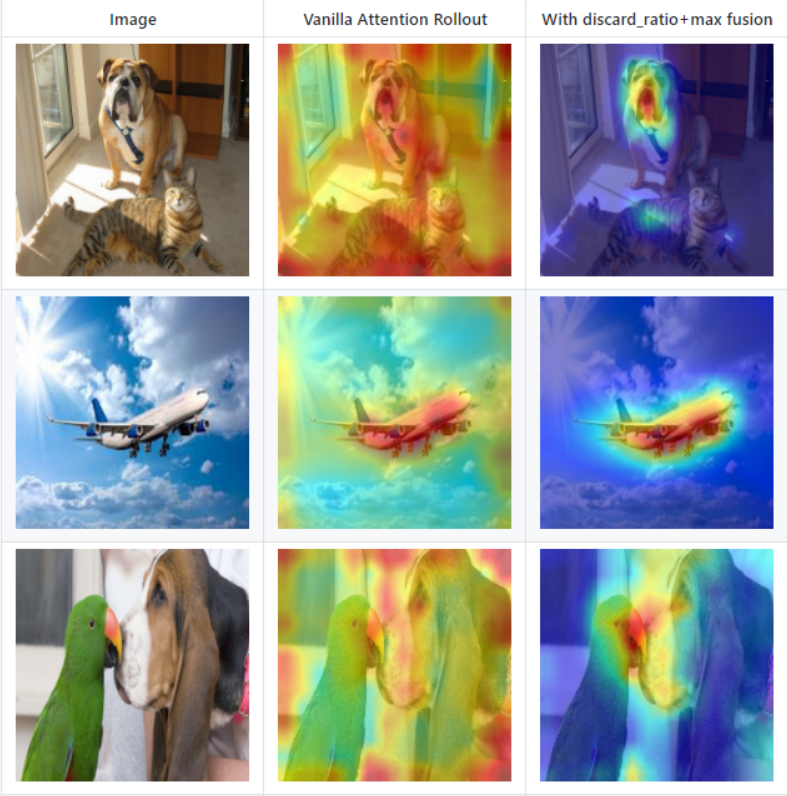


Figure 11: Se descartan píxeles con baja atención

Los resultados fueron muy ruidosos, y la atención no parece centrarse sólo en la parte interesante de la imagen. Por lo que se probaron los otros enfoques como el mínimo y máximo de los cabezales, de tal manera que funcione de forma similar en ViT y DeiT.

Mínimo. En la figura 10 (izquierda) se toma el valor mínimo entre los cabezales de atención, en lugar del valor promedio como sugiere [41]. Diferentes cabezales de atención miran diferentes cosas. Tomar el mínimo elimina el ruido al encontrar un denominador común.

Máximo. Combinado con descartar píxeles de baja atención, fusionar los cabezales de atención con el operador máximo parece funcionar mejor. Podemos centrarnos sólo en las atenciones principales, y descartar el resto. Como se puede ver en la figura 11, cuantos más píxeles eliminemos, podemos aislar mejor el objeto destacado en la imagen.

Otra pregunta que podemos hacer es: *"¿Qué en la imagen contribuye a una puntuación de salida más alta en la categoría 42?"* O en otras palabras, la explicabilidad específica de cada clase. Para ello se usaron las gradientes como se muestra a continuación.

Figure 12: ¿**Dónde** ve el Transformer un perro (categoría 243) y un gato (categoría 282)? En esta figura se muestran el resultado de aplicar esta técnica de gradiente. Consulte [40] para más detalles.

Gradient Attention Rollout. Al fusionar los cabezales de atención en cada capa, se pondera todas las atenciones por el gradiente de la clase objetivo, y luego se toma el promedio entre los cabezales de atención.

$$A_{ij} * grad_{ij}$$

4.6 Auto-supervisión

Los Transformers muestran un rendimiento impresionante en tareas de NLP. Sin embargo, gran parte de su éxito se debe a no solo por su excelente escalabilidad sino también por el pre-entrenamiento auto-supervisado a gran escala (Devlin et al. [2], 2019; Radford et al. [13], 2018). También se realizó una exploración preliminar sobre la predicción de parches enmascarados para la auto-supervisión, imitando la tarea de modelado de lenguaje enmascarado utilizada en BERT. Con pre-entrenamiento auto-supervisado, el modelo más pequeño ViT-B/16 logró alcanzar una precisión del 79,9% en ImageNet, un mejora significativa del 2% al entrenamiento desde cero, pero todavía un 4% por detrás del pre-entrenamiento supervisado.

5 Despliegue

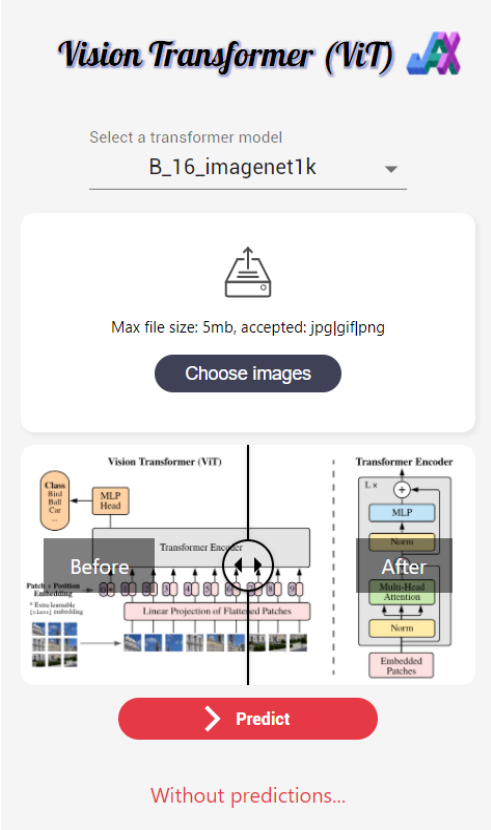
Para el despliegue del modelo se ha usado herramientas de desarrollo web, tales como React para la interfaz de usuario y Flask para levantar el servidor que manejará los modelos de inferencia. Respecto a los modelos de inferencia, sólo se dispuso de los modelos B16 y B32 (el número indica el tamaño de parche) debido a que la carga de dichos modelos ocupa menos memoria[2] comparado con el resto de modelos disponibles[3]. Todos los modelos usados han sido escritos originalmente en el framework JAX, pero se tuvo que convertir a un modelo Pytorch para hacer los evaluaciones.

La aplicación completa se desplegó en un Raspberry Pi 4, usando la compilación ARM de 64 bits de Pytorch [43][4]. De momento no se ha usado Torchscript para poder optimizar el modelo; tampoco se han usado técnicas de destilación o cuantización debido a la complejidad de algunas capas de la red.

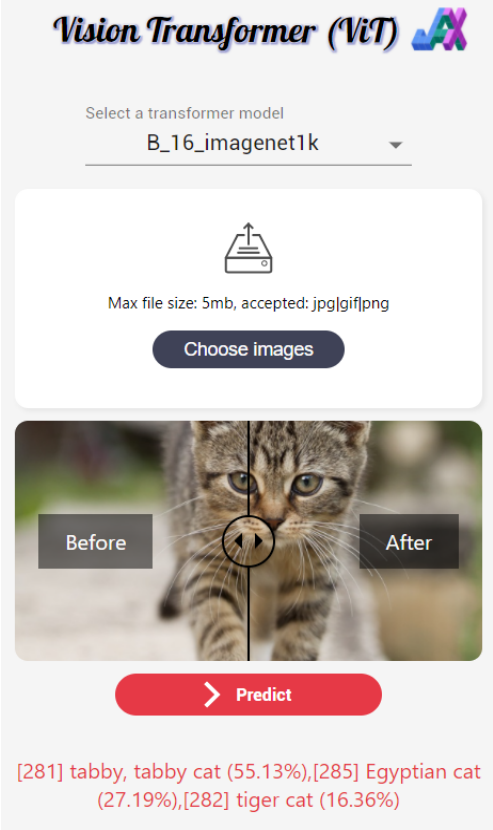
[2]El tamaño de dichos modelos es de aprox. 300MB, que se considera todavía muy grande para ser usado en producción

[3]Cuyos tamaños superan 1GB

[4]Es una compilación no oficial pero tiene soporte de uno de los tres autores originales de Pytorch, Thomas Viehmann



(a) Pantalla de inicio



(b) Después de la inferencia

6 Conclusión

Se ha explorado la aplicación directa de Transformers al reconocimiento de imágenes. A diferencia de trabajos anteriores utilizando self-attention en visión por computadora, no se introdujo ningún sesgo inductivo específico de la imagen en la arquitectura. En su lugar, se interpretó una imagen como una secuencia de parches que luego fue procesada mediante el encoder de un Transformer estándar como se usa en NLP. Esta estrategia simple, pero escalable, funciona sorprendentemente bien cuando se combina con el pre-entrenamiento en grandes conjuntos de datos. Por lo tanto, Vision Transformer iguala o excede el estado del arte en muchos conjuntos de datos de clasificación de imágenes, aunque es relativamente barato de pre-entrenar.

Si bien estos resultados iniciales son alentadores, aún quedan muchos desafíos. Una es aplicar ViT a otras tareas de visión artificial, como la detección y la segmentación. Estos resultados, junto con los de Carion et al. [7] (2020), indican la promesa de este enfoque. Otro desafío es seguir explorando métodos de pre-entrenamiento auto-supervisados. Los experimentos iniciales presentados aquí muestran una mejora del pre-entrenamiento auto-supervisado, pero todavía existe una gran brecha entre el pre-entrenamiento auto-supervisado y supervisado a gran escala. Finalmente, un mayor escalado de ViT probablemente conduciría a un mejor rendimiento.

Agradecimientos

Agradecimientos especiales a los autores originales del trabajo principal sobre el que me basé: Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit y Neil Houlsby. Gracias especiales a Jacob Gildenblat por proporcionar un ilustrativo post que fue de mucha ayuda para poder comprender la explicabilidad de ViT.

References

[1] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[2] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[3] Y. LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1 (1989), pp. 541–551.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012, pp. 1097–1105. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[5] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] Xiaolong Wang et al. *Non-local Neural Networks*. 2018. arXiv: 1711.07971 [cs.CV].

[7] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV].

[8] Prajit Ramachandran et al. *Stand-Alone Self-Attention in Vision Models*. 2019. arXiv: 1906.05909 [cs.CV].

[9] Huiyu Wang et al. *Axial-DeepLab: Stand-Alone Axial-Attention for Panoptic Segmentation*. 2020. arXiv: 2003.07853 [cs.CV].

[10] Dhruv Mahajan et al. *Exploring the Limits of Weakly Supervised Pretraining*. 2018. arXiv: 1805.00932 [cs.CV].

[11] Qizhe Xie et al. *Self-training with Noisy Student improves ImageNet classification*. 2020. arXiv: 1911.04252 [cs.LG].

[12] Alexander Kolesnikov et al. *Big Transfer (BiT): General Visual Representation Learning*. 2020. arXiv: 1912.11370 [cs.CV].

[13] Alec Radford et al. *Improving language understanding by generative pre-training*. 2018.

[14] Tom B Brown et al. "Language models are few-shot learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[15] Niki Parmar et al. "Image transformer". In: *arXiv preprint arXiv:1802.05751* (2018).

[16] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. "On the relationship between self-attention and convolutional layers". In: *arXiv preprint arXiv:1911.03584* (2019).

[17] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. "Exploring self-attention for image recognition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10076–10085.

[18] Rewon Child et al. "Generating long sequences with sparse transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

[19] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. "Scaling autoregressive video models". In: *arXiv preprint arXiv:1906.02634* (2019).

[20] Jonathan Ho et al. "Axial Attention in Multidimensional Transformers". In: *arXiv preprint arXiv:1912.12180* (2019).

[21] Irwan Bello et al. "Attention augmented convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 3286–3295.

[22] Han Hu et al. "Relation networks for object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3588–3597.

[23] Chen Sun et al. "Videobert: A joint model for video and language representation learning". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 7464–7473.

[24] Bichen Wu et al. "Visual Transformers: Token-based Image Representation and Processing for Computer Vision". In: *arXiv preprint arXiv:2006.03677* (2020).

[25] Francesco Locatello et al. "Object-centric learning with slot attention". In: *arXiv preprint arXiv:2006.15055* (2020).

[26] Yen-Chun Chen et al. "Uniter: Universal image-text representation learning". In: *European Conference on Computer Vision*. Springer. 2020, pp. 104–120.

[27] Jiasen Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13–23.

[28] Liunian Harold Li et al. "Visualbert: A simple and performant baseline for vision and language". In: *arXiv preprint arXiv:1908.03557* (2019).

[29] Mark Chen et al. "Generative pretraining from pixels". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 1691–1703.

[30] Hugo Touvron et al. "Fixing the train-test resolution discrepancy". In: *Advances in neural information processing systems*. 2019, pp. 8252–8262.

[31] Chen Sun et al. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[32] Josip Djolonga et al. "On robustness and transferability of convolutional neural networks". In: *arXiv preprint arXiv:2007.08558* (2020).

[33] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[34] Xiaohua Zhai et al. "A large-scale study of representation learning with the visual task adaptation benchmark". In: *arXiv preprint arXiv:1910.04867* (2019).

[35] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[36] Yuxin Wu and Kaiming He. "Group normalization". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.

[37] Tim Salimans and Diederik P Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *arXiv preprint arXiv:1602.07868* (2016).

[38] Michael Tschannen et al. "Self-supervised learning of video-induced visual invariances". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 13806–13815.

[39] Xiaohua Zhai et al. "S4l: Self-supervised semi-supervised learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1476–1485.

[40] Jacob Gildenblat. *Exploring Explainability for Vision Transformers*. URL: `https://jacobgil.github.io/deeplearning/vision-transformer-explainability`.

[41] Samira Abnar and Willem Zuidema. *Quantifying Attention Flow in Transformers*. 2020. arXiv: `2005.00928 [cs.LG]`.

[42] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *arXiv preprint arXiv:2012.12877* (2020).

[43] Thomas Viehmann. *PyTorch packages for ARM64*. URL: `https://mathinf.eu/pytorch/arm64/`.