Transfomer : Clasificación aviaria

Álvaro Vázquez Ortiz
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
alvvazort@alum.us.es

Gonzalo Martínez Martínez
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
gonmarmar5@alum.us.es

Resumen—

En este artículo metodológico se examinan dos distintos tipos de redes neuronales con la finalidad de optimizar la clasificación de imágenes aviarias, para ello se ha de obtener, limpiar y preparar los datos, entrenar la red y optimizar los hiperparámetros.

Para la realización del proyecto hemos hecho uso de la arquitectura Transformer, modelo que tiene como principal innovación la sustitución de las capas recurrentes, como las LSTMs usadas en PLN (Procesamiento de Lenguaje Natural), por las denominadas capas de atención. [7] Estas capas de atención codifican paralelamente cada entrada del conjunto de datos de entrada en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática de la imagen. Como alternativa a la arquitectura Transformer hemos hecho uso de un tipo de red convencional lineal uno de los algoritmos más conocidos para entrenar los pesos de una red neuronal artificial feedforward multicapa.

Con la realización de este proyecto hemos podido comprobar la gran eficacia y eficiencia producida por las capas de atención de la red Transformer frente a las capas del algoritmo de la red neuronal lineal, así como un mayor acierto de los resultados en un tiempo notablemente inferior.

Palabras Clave: red neuronal, Transfomer, LSTMs, PLN, red neuroal lineal.

I. INTRODUCCIÓN

En este documento se profundizará en el campo del Deep Learning en específico sobre uno de los modelos más avanzados de la actualidad, como son las arquitecturas Transformer y Visual Transformer (ViT). Por lo cual este documento estará centrado prácticamente en su totalidad en la clasificación de imágenes, imágenes aviarias en este caso. Para cada modelo que se verá durante el desarrollo de este se realizará tanto una explicación teórica profunda. Además, una vez realizada dicha explicación se realizará una implementación en código Python de los conceptos desarrollados en los apartados anteriores y con una base de datos de imágenes aviarias clasificándolas según la tipología, pudiendo así ser capaces de realizar una comparación experimental de una red sencilla lineal con una arquitectura ViT, poseyendo ambas las mismas condiciones de inicialmente. Con la realización de este documento se pretende haber llegado a comprender el funcionamiento característico de esta nueva arquitectura (modelo codificador-decodificador) y ser capaces de mostrar el potencial de esta.

Durante el desarrollo del documento se detallarán los objetivos que se pretenden cubrir con la realización de este y un breve resumen de la estructura del trabajo por puntos. Posteriormente se explicará la metodología seguida durante el proyecto donde se detallará la problemática del modelo y como se ha solucionado, los resultados obtenidos por ambos modelos y, por último, las conclusiones finales obtenidas a lo largo del desarrollo del proyecto.

II. PRELIMINARES

A. Métodos empleados

Este proyecto gira en torno a la arquitectura Transformer de redes neuronales vista en el paper “Attention Is All You Need” [1], en concreto al uso de ella más específico para la clasificación de imágenes, el cual viene definido en “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” [2]. Todas estas parten de redes neuronales, dándole una nueva estructura que permite contextualizar los datos de entrada al modelo.

III. METODOLOGÍA

Esta sección se describirá la metodología implementada en el trabajo. Para la realización del proyecto se han desarrollado como anteriormente se ha descrito 2 redes neuronales distintas. En esta sección se detallará la definición de cada tipo de red y posteriormente se explicará cómo se ha implementado.

Una red neuronal profunda (DNN) es una red neuronal artificial (ANN) con varias capas ocultas entre las capas de entrada y salida. Al igual que en las ANN poco profundas, los DNN pueden modelar relaciones no lineales complejas. Las redes neuronales se utilizan ampliamente en el aprendizaje supervisado y en los problemas de aprendizaje por refuerzo. Estas redes se basan en un conjunto de capas conectadas entre sí. Principalmente utilizamos el método de descenso de gradiente para optimizar la red y minimizar la función de

pérdida. Además, Backpropagation es el algoritmo principal en el entrenamiento de modelos DL para obtener la predicción de salida correcta. Cada nodo en las capas de salida y ocultas tiene sus propios clasificadores. La capa de entrada toma entradas y pasa sus puntuaciones a la siguiente capa oculta para una activación adicional y esto continúa hasta que se alcanza la salida. Este progreso de entrada a salida de izquierda a derecha en la dirección de avance se denomina propagación hacia adelante.

Para la implementación de esta red en primer lugar se ha creado una red neuronal secuencial sencilla, en la que, para trabajar, tendrá como entrada 3*64*64, los cuales se refieren a los 3 canales (RGB) y 64 píxeles de altura y 64 píxeles de anchura que tiene la imagen que le introduciremos. Como capas ocultas, hemos definido 2 de 12288 nodos, y 1 de 6144 intentando aprovechar la potencia del GPU proporcionado por la plataforma kaggle para obtener los mejores resultados posibles. Para la salida, deberá tener 400 outputs, debido a las 400 clases por las que debe clasificar. El mayor de estos outputs será la clase predicha por la red.

Por otra parte, para el desarrollo de la red ViT se ha seguido de manera exhaustiva las pautas descritas en el documento "Attention Is All You Need" [1]. Para poder llegar a comprender el modelo ViT en su plenitud es conveniente primero detallar varios conceptos como la técnica Attention, un concepto esencial para poder entender de manera clara como se focaliza y prioriza algunas partes de la entrada y el modelo Transformer.

La técnica "Attention" fue creada para el ámbito de la traducción mediante Deep Learning y posteriormente fue adaptada al mundo del reconocimiento de imágenes con las CNN. La idea contextualizar la entrada al modelo, como ya se hacía anteriormente con las palabras de una frase, se extrapoló al reconocimiento de imágenes y aquí donde radica la esencia de Transformer en su capacidad tener todas estas características presentes mientras se analiza una fotografía para terminar clasificándola. [4]

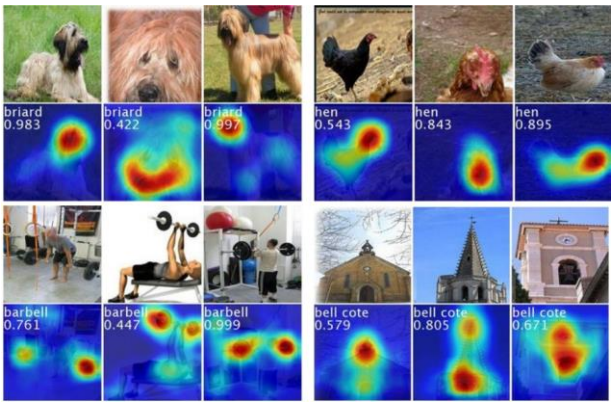

Fig. 1. Representación de mapa de calor de la técnica Attention

En la figura anterior se puede apreciar como esta técnica se centra en diferentes zonas de la imagen conforme va pasando por las diferentes capas.

A continuación, una vez visto cómo funciona la técnica Attention, podemos entrar en detalle en cómo funciona esta nueva arquitectura de Deep Learning. En primer lugar, en la figura 2 se puede apreciar la arquitectura con cada parte que conforma a los Transformers. En la parte izquierda se tiene el codificador y en la derecha se tiene el decodificador, esta división en dos partes de la arquitectura permite hacer que sea más eficiente. A continuación, se van a detallar las características de cada una de ellas por separado.

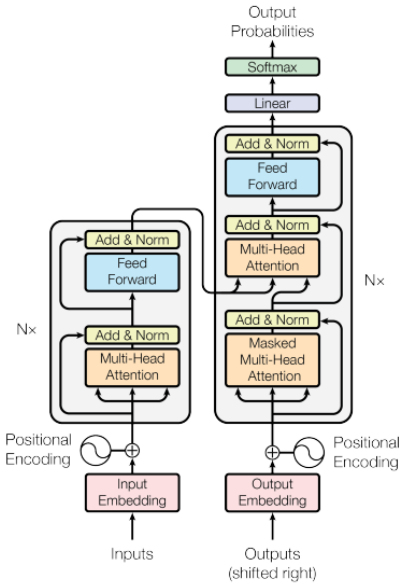

Fig. 2. Transformer. Modelo de arquitectura

· **Codificador**: En cuanto a la entrada se puede encontrar la primera gran diferencia con las redes convencionales, en esta ocasión la entrada son todos los datos a la vez en, al contrario que con las redes recurrentes con las cuales la entrada era una parte de ellos de forma secuencial. Dicha entrada es convertida a "Input Embedding", es decir, se crea un espacio vectorial con los datos aprendidos por la red, los datos que tengan un significado en común estarán más cercanos que las que no, esto permite poder mapear los datos para así poder encontrar el valor de similitud más fácilmente. La segunda diferencia que nos encontramos es la denominada "Positional Encoding", esta herramienta permite tener un control de la posición de las palabras en todo momento. Una vez realizado el Encoding, se introduce el resultado en las capas Multi-Head Attention, múltiples capas que implementan la técnica Attention.

En concatenación con las capas Multi-Head Attention se realiza una concatenación de los valores de entrada y una normalización de los valores de los vectores Query, Key y Value obtenidos para cada uno de los tokens de la entrada.[6]

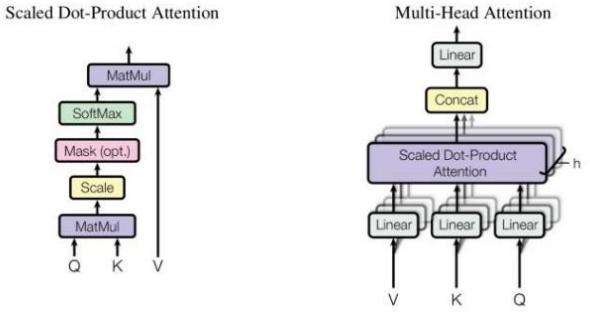

Fig. 3. Transformer. Codificador.

· **Decodificador:** La entrada al decodificador es una secuencia de embeddings con codificación posicional que pasará por el Masked multi-head attention donde múltiples cabezas reciben los datos enmascarados de tal forma que no se utilice la información de las posiciones enmascaradas. Para ello se le otorga valores infinitamente negativos a la atención de esos datos. La función softmax junto con la atención asignará por tanto una probabilidad igual a 0 para esas opciones enmascaradas. Posteriormente se normalizarán los datos para poder pasar por Source-target attention donde múltiples cabezas calcularán la atención entre las características de los datos de entrada y salida. [5,6]

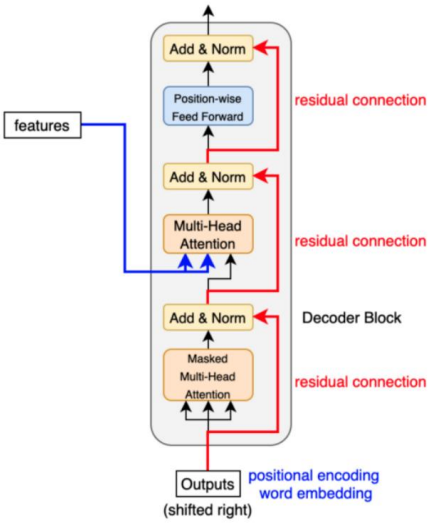

Fig. 4. Transformer. Decodificador.

Una vez detallada la arquitectura Transformer, podemos empezar a detallar con exactitud el modelo posteriormente desarrollado ViT. En el artículo “An image is worth 16x16 words” [2] se mostraban lo que sería una evolución de la técnica Transformer con la que prometían destronar a las CNN, puesto que se necesitaría menos recursos, obteniendo los mismos o mejores resultados. Esto es debido a que en el ViT cada uno de los fragmentos es introducido a la vez, teniendo así tantas capas como fragmentos trabajando en paralelo. Al contrario que en las CNN en las cuales la imagen se tiene que ir recorriendo por el kernel y haciendo más pequeña para ir sacando a su vez, características más importantes. Esto, más la implementación directa de Attention sobre cada uno de los fragmentos hace que sea mucho más eficiente y ver más características conforme se avanza en las capas ViT.

Como se puede intuir por el propio nombre del artículo, para poder implementar el codificador, cada una de las imágenes a se segmentará en N partes de 16x16, por lo tanto la estructura en esta ocasión se subdivide en dos partes principalmente, un primer preprocesado de imagen y posteriormente el codificado de la misma.

Preprocesado de las imágenes: Como se ha comentado con anterioridad, el primer paso es segmentar la imagen en 16x16 píxeles. Esto se hace debido a que, al igual que en la anterior arquitectura, se empleará la técnica Attention, por lo cual, si la imagen es introducida como un registro de bits con una codificación, la técnica calcularía la similitud con cada uno de los bits introducidos, sin tener en cuenta la posición que guarda cada uno de los mismos dentro de la imagen. Una vez segmentada, cada uno de los píxeles de cada una de las partes es pasado a vector y proyectado a un subespacio (embedding).[3]

Esto es parecido a lo que se realizaba con la arquitectura Transformer en el punto anterior, al final, el resultado es tener un subespacio vectorial donde los patches que más se parezcan entre ellos estarán más cercanos en el espacio. Cabe destacar la gran importancia de mantener el orden de cada una de las secciones que se han sacado de la imagen ya que gracias a este orden las capas multi-head Attention son capaces de encontrar la relación entre las diferentes partes.

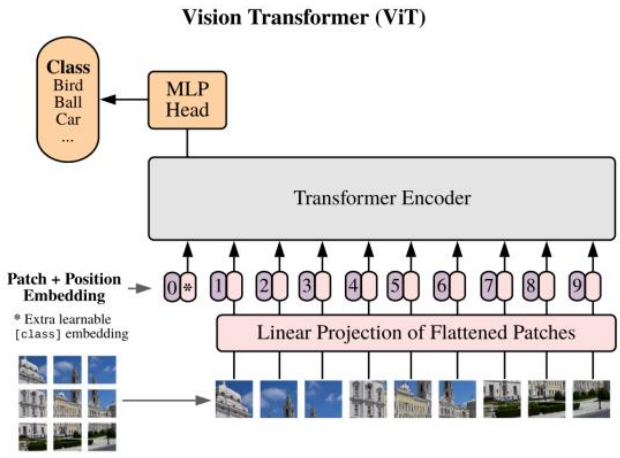

Fig. 5. ViT. Preprocesado de imágenes.

Codificador: Como se puede apreciar en la figura 6, el codificador de la arquitectura ViT guarda una cercana relación con el mostrado en la figura 3 proveniente del modelo de la arquitectura Transformer. Una vez se ha realizado el embedding de cada uno de los fragmentos y de la clase a la cual pertenece la imagen completa, se pasa por una capa de normalización, para posteriormente pasar por las capas de multi-head Attention, al igual que en la arquitectura Transformer original, con estas capas se consigue establecer vectores de similitud entre cada uno de los fragmentos.

Con esto conseguimos que se obtenga la posición a la que pertenecen cada uno de los segmentos que conforman la imagen original y por otro lado, mediante la técnica Attention se están buscando los elementos de interés y estableciendo vectores de similitud entre los mismos, de esta forma, cada uno de los pixeles que conforman la imagen original tienen una función, al contrario que si se hubiera realizado mediante una red neuronal convencional donde la capa de entrada contendría cada uno de los pixeles sin importar el orden espacial. Por último, se realiza una nueva normalización y la salida, es la entrada de una MLP que se encargará de clasificar el resultado, obteniendo así al final una categorización final.

Fig. 6. ViT. Codificador.

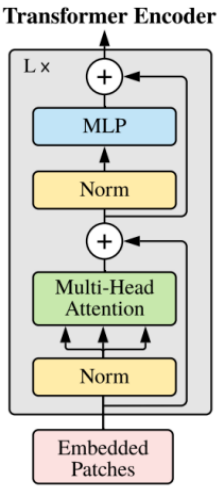

Fig. 7. ViT. Codificador.

Para la implementación del modelo con arquitectura transformer, hemos definido varios módulos:

- **ViT**: Modelo completo del vision transformer. Este módulo se encarga de recibir los datos de entrada y realizar el patch embedding, es decir, dividir la imagen en trozos de 16x16 en nuestro caso. Tras esto, introducirlo en el Encoder. Una vez haya procesado el Encoder estos datos, una capa de red neuronal secuencial se encargará de determinar la clase predicha.
- **Encoder**: Compuesto por el módulo attention y feed forward. Se encargará de procesar estos módulos paralelamente el número que venga indicado en el parámetro depth y tras ello sumar los valores de salida, tanto la salida de attention como de feed forward.
- **PreNorm**: Normaliza el input antes de ser introducido en el módulo de self attention y feed forward.
- **Attention**: Módulo que permite al modelo atender a la información de distintos subespacios.
- **Feed forward**: Red neuronal que procesará tanto los datos introducidos al encoder como la salida del Attention.

Durante los primeros entrenamientos no tuvimos en cuenta una preparación correcta del dataset, lo cual hizo que al rectificar en este aspecto mejorasen notablemente los resultados obtenidos. En primer lugar, normalizamos los tensores de las imágenes para que los valores estuvieran en valores entre -1 y 1. En segundo lugar, se añadió cierta rotación aleatoria en las imágenes, lo que hacía que en cada entrenamiento el dataset de entrenamiento fuese en cierta medida diferente. Por último, aleatorizamos el orden de las imágenes, ya que, al estar en orden, la red iba especie por especie con más de 100 casos en cada una, haciendo que la red se especializase en predecir la última especie en la que habia sido entrenada.

IV. Resultados

En esta sección se detallará tanto los resultados conseguidos para ambas redes neuronales partiendo desde la misma situación inicial y para un mismo número de épocas a entrenar. Nótese que la única diferencia entre las condiciones de ambos modelos es que el modelo ViT se encuentra preentrenado mientras que la red neuronal sencilla no.

Red neuronal secuencial		
Épocas	***Perdida***	***Precisión***
Época 1	5.2378125	0.0433050
Época 2	4.3624863	0.1252909
Época 3	3.9553432	0.1855319
Época 4	3.6313450	0.2350424
Época 5	3.3401269	0.2860966
Época 6	3.0737276	0.3334474
Época 7	2.8354325	0.3755373
Época 8	2.6031801	0.4218270
Época 9	2.3852841	0.4635679
Época 10	2.1730132	0.5075814
Época 11	1.9675176	0.5490416

Tabla 1. Resultados red neuronal sencilla

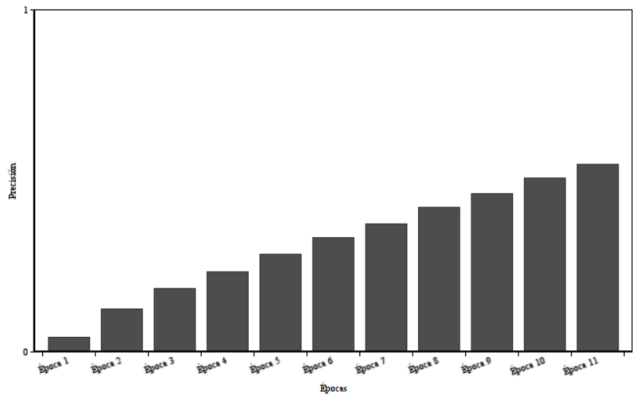

Fig. 8. Grafico resultados red neuronal sencilla

Tiempo de ejecución: 0:29:54.20 segundos

ViT		
Épocas	***Perdida***	***Precisión***
Época 1	5.0189099	0.0763964
Época 2	3.2841980	0.3295591
Época 3	2.3950893	0.4947734
Época 4	1.8322479	0.6107133
Época 5	1.4032191	0.7005545
Época 6	1.0489783	0.7832899
Época 7	0.7452967	0.8569722
Época 8	0.4863475	0.9204443
Época 9	0.2883968	0.9630647
Época 10	0.1581918	0.9873323
Época 11	0.0927655	0.9940615

Tabla 2. Resultados red ViT

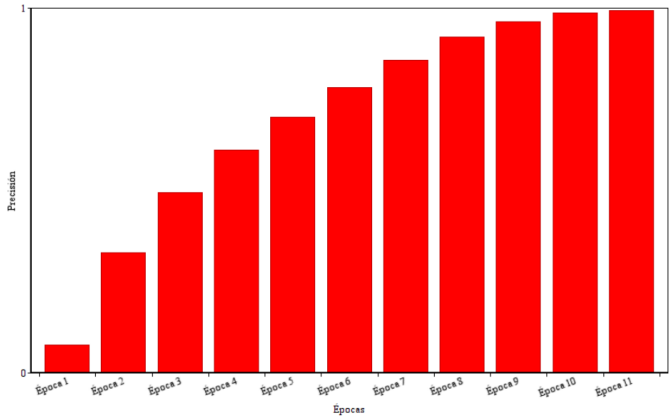

Fig. 9. Grafico resultados red ViT

Tiempo de ejecución: 2:31:33.65 segundos

A continuación, se realizará una comparativa de los resultados creados por ambas redes por cada época para determinar qué modelo es más eficiente teniendo una menor cross entropy loss y una mayor precisión.

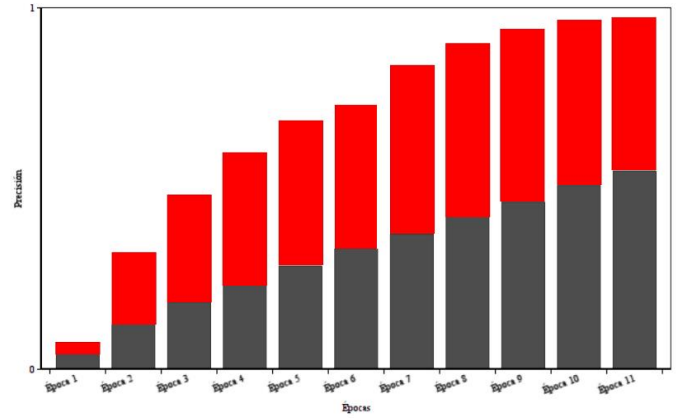

Fig. 10. Grafico comparativo entre los 2 modelos

En la figura 10 podemos observar la comparación de tasa de aciertos de ambos modelos. Podemos observar cómo desde tempranas épocas el modelo ViT obtiene una tasa casi del doble de aciertos con respecto al modelo lineal. A su vez podemos ver una mejora exponencial en el *cross entropy loss*, obteniendo en el modelo ViT valores cercanos a 0 mientras que en el modelo lineal los valores alcanzan en las últimas épocas valores inferiores a 2. Sin embargo, el tiempo empleado para el entrenamiento de la red ViT ha sido 500% mayor al tiempo necesitado por la red lineal, ambas redes consumían la máxima capacidad de RAM y GPU dispuesta por kaggle durante el periodo de entreno, por tanto, el modelo ViT a su vez ha supuesto un 500% más de recursos consumidos con respecto al modelo lineal.

V. Conclusiones

Como hemos podido ver en la comparación entre los modelos, el entrenamiento ha supuesto un consumo de recursos y tiempo excesivamente mayor. Aun así, normalmente este no es un impedimento, ya que las inteligencias artificiales en la práctica si se les da una tarea compleja en las que se requiere precisión, suelen ser entrenadas sin limitarlas en cuanto tiempo y recursos hardware, por lo que se le da mucha más importancia a que se realice la tarea correctamente en vez de a los costes de entrenamiento.

Es por eso que grandes empresas como Google, en servicios como es el motor de búsqueda de Google, Google Translate, Google Lens, etc.

En este campo se puede dar multitud de utilidades a esta arquitectura, identificación de enfermedades mediante imágenes de radiografía o resonancia magnética, enfermedades en plantas, etc.

Referencias

Nótese que todas las referencias indicadas posteriormente han sido utilizadas para la realización de esta memoria o para la realización de los modelos comentados anteriormente. [8,9]

[1] Ashish Vaswani et al. “Attention Is All You Need”. In: NIPS2017 abs/1706.03762(2017). arXiv: 1706.03762. url: http://arxiv.org/abs/1706.03762.

[2] https://arxiv.org/pdf/2010.11929.pdf Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for ImageRecognition at Scale ”. In: CoRR abs/2010.11929 (2020).

[3] https://arxiv.org/pdf/1406.1078.pdf Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014.

[4] P. Rajpurkar, "Visualizing A Convolutional Neural Network's Predictions", mlx, 2017. [Online]. Available: https://rajpurkar.github.io/mlx/visualizing-cnns/.

[5] https://www.youtube.com/watch?v=d_ixlCubqQw

[6] https://kikaben.com/transformers-encoder-decoder/

[7] https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-networkPágina

[8] https://github.com/lucidrains/vit-pytorch

[9] https://github.com/rwightman/pytorch-image-models