Machine Learning: Applicazione dell'architettura Vision Transformer a task con un ridotto set di immagini associato

Alessandro Arezzo
E-mail address
alessandro.arezzo@stud.unifi.it

Abstract

Il progetto descritto nel merito della presente relazione è stato realizzato come elaborato per l'insegnamento Machine Learning previsto dal corso di laurea magistrale in ingegneria informatica dell'Università degli studi di Firenze.

1. Introduzione

Negli ultimi anni l'architettura dei Transformers descritta all'interno del lavoro *Attention is All you Need* [4] viene presa in considerazione per la risoluzione di un sempre maggior numero di problemi di machine learning. Il modello in questione, meglio analizzato nel corso del paragrafo 2, nasce come metodo risolutivo della trasduzione di sequenze e più nello specifico viene originariamente applicato al contesto dell' *NLP (Natural Language Processing)*. In tale ambito, i Transformers si sono rilevati sin da subito in grado di ottenere delle performance migliori rispetto alle reti ricorrenti che ne rappresentavo lo stato dell'arte. Visti i notevoli risultati ottenuti per tale applicazione si sono quindi intensificati gli sforzi per verificarne l'efficacia anche in altri campi. Un esempio di ciò è rappresentato dal lavoro riportato nel merito dell'articolo *An Image Is Worth 16X16 Words: Transformers For Image Recognition At Scale* [3] in cui i Transformers vengono utilizzati in computer vision per il task di classificazione di immagini attraverso la definizione dell'architettura denominata *Vision Transformers (ViT)* descritta al paragrafo 3.
All'interno del paper di riferimento [3], nello specifico, tali modelli sono stati applicati come solito fare in generale per i Transformers attraverso una prima fase di pre-training su grandi dataset ed un successivo fine-tuning per la classificazione di immagini appartenenti a set di ridotte dimensioni. Così facendo i risultati ottenuti hanno consentito alla suddetta architettura di superare il precedente stato dell'arte rappresentato dalle attuali reti convoluzionali residuali (Resnet).

L'obiettivo dell'elaborato descritto all'interno della presente relazione, è stato quello di partire dallo studio dei modelli ViT, e di valutarne le performance quando ne viene effettuato direttamente il training su dataset di piccole dimensioni. Tali prestazioni sono quindi poi state confrontate con quelle ottenute addestrando sugli stessi set di dati una resnet18, la quale rappresenta una rete convoluzionale residuale a 18 layers.
L'implementazione dei modelli, spiegata nel dettaglio al paragrafo 4, è stata realizzata in linguaggio di programmazione Python utilizzando il framework *PyTorch* [1]. I risultati prodotti dagli esperimenti effettuati sono riportati nel corso del paragrafo 5.

2. Transformers

I modelli noti sotto il nome di Transformers sono stati presentati per la prima volta nell'articolo *Attention is All you Need* [4] come una soluzione semplice e scalabile sviluppata al fine di risolvere il problema della trasduzione di sequenze composte da parole: la traduzione di frasi da una lingua sorgente ad una lingua di target. Il loro obiettivo era difatti quello di definire un sistema semplice e scalabile che fosse in grado di risolvere il suddetto task con l'ausilio del solo meccanismo di attenzione, senza cioè ricorrere all'utilizzo delle convoluzioni richieste dai precedenti modelli in uso quali reti ricorrenti (RNN) e reti convoluzionali (CNN). Le principali limitazioni che i Transformers si sono posti lo scopo di rimuovere risiedono difatti da un lato nell'impossibilità di parallelizzare i calcoli propria delle RNN quali LSTM e GRU e dall'altro nella difficoltà di addestrare CNN particolarmente profonde e quindi con un campo ricettivo sufficientemente ampio da poter codificare le cosiddette dipendenze a lungo termine.

Per la realizzazione di tale obiettivo auspicato, i modelli in analisi fanno uso di una semplice architettura encoder-decoder, il cui schema è riportato in Figura 1. Come si nota da una sua interpretazione, questo si divide in tre parti principali: embedding, encoders e decoders.

Il modulo di **embedding** si occupa di fornire una specifi-

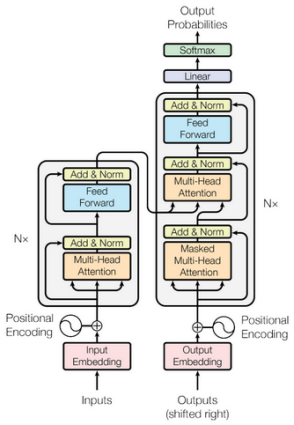

Figura 1. Schema dell'architettura del modello Transformer.

ca rappresentazione della frase di input. Per farlo, viene prima di tutto utilizzato un layer lineare che mappa la codifica one-hot di ciascun termine in un vettore avente una dimensione detta di embedding. A questi vengono poi sommati dei vettori di *positional encoding* allo scopo di consentire al modello di comprendere la posizione di ciascuna parola all'interno della frase.

La rappresentazione viene quindi passata al modulo **encoders**, il quale, utilizzando una serie di sottoblocchi encoder posti in cascata, ha il compito di trasformare l'attuale descrizione della sequenza di input in una codifica che sia rappresentativa di quanto il resto della frase sia rilevante nell'interpretazione di ogni sua parola. Per farlo ogni encoder utilizza in primo luogo l'elemento di *Multi-head attention (MHA)*, il quale si occupa di implementare il meccanismo di self-attention meglio spiegato di seguito e che rappresenta le fondamenta del funzionamento del modello Transformers. Inoltre, in ciascun encoder vi è la presenza di una rete feedforward solitamente composta da pochi layers lineari che computa i dati in uscita dal MHA e di un blocco che esegue un layer di normalizzazione sul risultato di una connessione residuale. Si noti come tale componente sia presente non solo dopo l'esecuzione del modulo MHA ma anche in uscita dalla rete fully connected.

La codifica fornita in output dall'ultimo encoder viene infine utilizzata dal blocco **decoders**, il quale, alla pari del componente appena descritto, fa uso di una pila di decoder per predire ad ogni passo un simbolo della sequenza di output. Per farlo, ogni decoder implementa due moduli di MHA. Il primo, ad ogni step elabora le parole di output prodotte fino a quel momento e per farlo fa uso di un modulo di self-attention di tipo masked, per far fronte al problema che ad ogni passo alcuni termini della sequenza non saranno ancora stati generati. Il secondo modulo di attenzione utilizza invece sia le informazioni prodotte da tale componente, sia quelle relative alla codifica fornita dall'encoder.

L'output dell'ultimo decoder viene posto in input ad un layer fully connected che mappa la rappresentazione in questione in un vettore di logit avente la stessa lunghezza del dizionario della lingua nella quale la frase di uscita è espressa. Le sue componenti vengono quindi passate ad una funzione softmax che le normalizza in un range compreso tra 0 ed 1, così che ad ogni iterazione possa essere generato il simbolo associato alla cella del vettore di logit con il valore di probabilità maggiore.

Ovviamente, risulta evidente di come il problema della traduzione di frasi con la quale i Transformers sono stati introdotti nell'articolo di riferimento Vaswani et al 2017 [4], rappresenta solo un possibile esempio di contesto applicativo e di come dunque questi possano essere utilizzati per la risoluzione di ogni tipo di trasduzione di sequenze, qualunque sia la tipologia dei dati coinvolti nel processo.

2.1. Self-attention

Affinchè i modelli Tranformer siano in grado di comprendere l'attenzione che deve essere attribuita a ciascun componente della frase di input ogni qualvolta ne interpreta un qualunque altro suo elemento, viene utilizzato il meccanismo di self-attention. L'idea di tale processo di primaria rilevanza risiede nella capacità del modulo in questione, ricevuti in input una sequenza di vettori ciascuno rappresentante una determinata parola nella frase, di generare in output un ulteriore insieme di vettori dello stesso numero in cui ognuno di essi si riferisce ad una particolare parola e ne codifica la rilevanza di tutte le altre rispetto ad essa. Per realizzare ciò, per quanto riguarda il modulo di MHA interno al primo encoder del modello, ciascun embedding vector viene moltiplicato per tre diverse matrici dei pesi apprese durante la fase di training. Tale operazione restituisce rispettivamente i vettori di query, key e value relativi alla parola in questione, i quali saranno generalmente di lunghezza inferiore rispetto a quella dell'embedding e nota come dimensione dell'head. L'approccio prevede quindi che per ciascuna parola venga calcolato il prodotto scalare tra il rispettivo vettore di query e quello di key associato ad ogni altro termine della frase. Il risultato prodotto viene interpretato come un indice di similarità tra la parola in oggetto rispetto a tutte le altre, per poi essere normalizzato per mezzo della divisione della radice quadrata della dimensione delle keys. I valori ottenuti vengono quindi passati ad una funzione softmax che li normalizza in un range compreso tra 0 ed 1 così che l'output di tale operazione possa essere interpretato come una misura di quanto le varie parole risultino importanti per l'interpretazione della query corrente. Ciò è riassunto più chiaramente nell'esempio di alto livello riportato in Figura 2, la quale mostra la distribuzione dei livelli di attenzione attribuiti a ciascuna parola di una frase di input d'esempio nella codifica di un suo specifico termine. Per fornire una rappresentazione vettoriale che compatti ta-

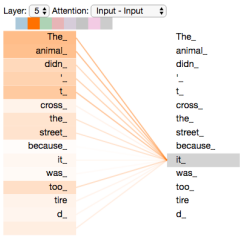

Figura 2. Esempio della distribuzione di attenzione nell'interpretazione di una determinata parola appartenente ad una frase di input d'esempio.

le informazione, ogni punteggio che indica la rilevanza di una certa key con la query viene poi moltiplicato per il rispettivo vettore di value associato alla key in questione ed i risultati ottenuti vengono sommati tra loro. Se ne ottiene quindi un vettore di dimensione pari a quella dell'head, dato dalla somma dei vettori di value di ciascuna parola pesati dall'attenzione che è necessario attribuire al termine al quale questo si riferisce nell'interpretare la query corrente.
Dato che tale procedimento deve essere ripetuto per ogni parola, per velocizzare la computazione i calcoli vengono effettuati in forma matriciale. Per farlo, tutti gli embedding vectors vengono concatenati in un'unica matrice $X \in R^{NXD}$ con N numero di parole della frase e D dimensione dell'embedding. La matrice di attenzione Z risultante viene quindi calcolata come

$$Z = \frac{softmax(QXK^T)V}{\sqrt{d}} \tag{1}$$

dove Q, K e V sono le matrici ottenute moltiplicando X per le matrici dei pesi W^q, W^k, W^v e d è la dimensione dell'head.
Quanto fin qua descritto si riferisce in realtà ad un modulo di self-attention a singola testina. Questo viene quindi generalizzato al caso multi-head in cui vi sono molteplici blocchi di tale tipo, ciascuno con i propri pesi. L'esecuzione di tali moduli in parallelo genera un numero di matrici di attenzione ovviamente pari al numero dei moduli stessi, le quali matrici, una volta concatenate, vengono quindi moltiplicate per un ulteriore matrice dei pesi W^O di dimensione $dNXD$. Il risultato sarà una matrice Z di dimensione NXD, nella quale ciascuna riga codifica la rilevanza dei vari componenti della frase rispetto alla specifica parola alla quale questa si riferisce.

Tale ragionamento, può poi essere esteso per l'analisi degli encoder posti più in profondità nel modello, i quali forniranno per ciascuna parola un vettore che ne codifica l'attenzione da attribuire durante la sua interpretazione non rispetto a singoli termini della frase ma piuttosto in relazione ad intere parti della stessa.

3. Vision Transformer

Come preannunciato, l'architettura Vision Transformers (ViT) [3], prevede l'adattamento del modello Transformer fin qua descritto al task di classificazione di immagini ovvero al problema di predire, data un'immagine di input, quale sia la classe alla quale questa appartiene. Vista la diversità di tale problema rispetto a quello della trasduzione di sequenze, lo schema di funzionamento dei ViT, riportato in Figura 3, presenta delle sostanziali differenze rispetto all'architettura encoder-decoder fin qua illustrata.

La pipeline del modello prevede che, data un'immagine, questa venga in primo luogo passata al modulo di embedding, il quale si occupa di fornirne una rappresentazione da porre in input al blocco Transformer Encoder. Dal momento che i componenti di MHA calcolano l'attenzione da attribuire a ciascun componente della sequenza in ingresso rispetto a tutti gli altri, risulta evidente di come non sia computazionalmente accettabile considerare una rappresentazione che codifichi ogni pixel dell'immagine in un vettore così come viene fatto per le parole delle frasi nel task di trasduzione. Per far fronte a ciò, un'immagine $x \in R^{HXWXC}$ dove (H, W) è la risoluzione dell'immagine e C rappresenta il numero dei suoi canali, viene partizionata in **patches** di dimensione PXP. Queste possono essere interpretate come gli elementi corrispondenti a ciò che le parole rappresentano nel problema della trasduzione di frasi. Ciascuna patch 2D viene quindi appiattita in un vettore unidimensionale, con la conseguente ridefinizione dell'immagine di input in una sequenza $X_p \in R^{NX(P^2*C)}$ con N numero di patches. Esattamente come visto per i Transformers, ciascun vettore viene mappato da un layer lineare in un altro vettore avente una certa lunghezza D e noto come patch embedding. Affinchè sia possibile classificare le immagini, viene inoltre aggiunto un ulteriore embedding appreso dal modello durante la fase di training e denominato **cls token**. Tale vettore viene nello specifico aggiunto allo scopo di ottenere in uscita dal successivo blocco Transformer Encoder una rappresentazione della patch che comprenda l'intera immagine, così che sia possibile, come meglio illustrato di seguito, utilizzare tale codifica ai fini della classificazione.

Affinchè il modello sia capace di valutare la posizione delle patch all'interno dell'immagine, ad ogni patch embedding vengono aggiunti dei positional encoding: vettori aventi una composizione appresa dal modello durante l'addestramento.

La rappresentazione così generata, viene quindi posta in input al blocco **Transformer Encoder**, composto da una sequenza di layer a loro volta formati dall'alternanza di un blocco di Multi-Head Attention e di uno di Multi Layer Perceptron (MLP). Quest'ultimo è nello specifico formato da due layers lineari con una GELU come funzione di attivazione. Si noti poi di come layers di normalizzazione e

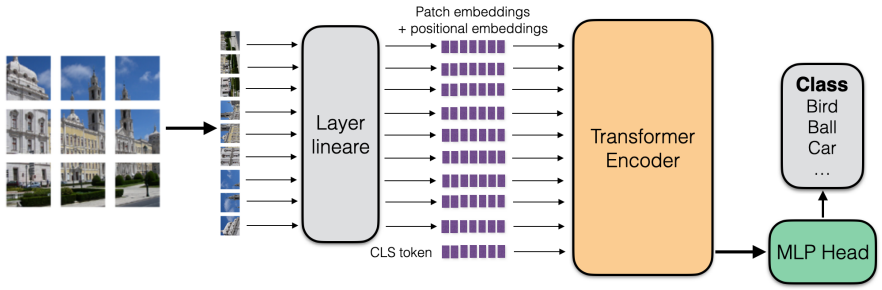

Figura 3. Schema dell'architettura del modello Vision Transformer.

connessioni residuali siano applicati rispettivamente prima e dopo ciascuno di tali blocchi.

Un modulo di MLP head si occupa infine di classificare le immagini. Questo, composto da un semplice layer fully connected, riceve in input il primo vettore in uscita dal Transformer Encoder, e lo mappa in un altro vettore di lunghezza pari al numero delle classi. Applicando quindi una funzione softmax a tale output è possibile ottenere la probabilità che l'immagine elaborata appartenga a ciascuna classe.

Una variante relativa al modello introdotto prevede la definizione d un'architettura ibrida, la quale consiste nel considerare come input da fornire al layer di embedding lineare le features prodotte da una CNN. Nello specifico quindi, una rete convoluzionale viene usata per estrarre le caratteristiche dell'immagine di input da un qualche suo livello e da queste vengono estratte le patch che una volta appiattite definiscono i vari patch embedding. Per tale variante viene in realtà considerato un caso speciale di partizionamento in cui ogni patch ha una dimensione 1x1. Data quindi una feature map 2D questa viene semplicemente appiattita su una dimensione e la sequenza risultante viene proiettata dal layer di embedding nella dimensione del Transformer.

3.1. Inductive bias

Fino ad oggi, le reti Resnet si sono poste come lo stato dell'arte per la risoluzione del problema di classicazione delle immagini in quanto in grado, con l'aggiunta di connessioni residuali, di rimuovere il problema dell'addestramento di reti convoluzionali particolarmente profonde. I modelli ViT si pongono quindi, al pari delle architetture Transformers con la trasduzione di frasi, come una soluzione semplice e scalabile capace di rimuovere le convoluzioni garantendo delle prestazioni superiori rispetto a quelle fin qua ottenute dalle reti Resnet.
Nell'analisi di come questo sia possibile, occorre innanzitutto evidenziare di quanto i modelli in analisi presentino una quantità inferiore di inductive bias rispetto a quelli delle reti convoluzionali.
Quest'ultime difatti, effettuano in primo luogo un' ipotesi di **equivarianza traslazionale** dovuta al fatto che ogni

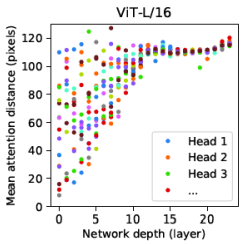

Figura 4. Distanza media di attenzione attribuita alle patches da ciascuna head al variare della profondità del layer in cui queste si trovano. Il grafico è riportato nell'articolo [3] e si riferisce al modello ViT-L/16 addestrato su Imagenet.

layer convoluzionale possiede una singola matrice dei pesi condivisa per il processamento di ogni parte dell'immagine. Se quindi un determinato livello di una CNN è caratterizzato da un filtro che durante il training ha appreso come riconoscere una specifica feature, attraverso l'operazione di convoluzione questo sarà in grado di rilevarla in ogni immagine, indifferentemente dalla posizione globale in cui la caratteristica in questione vi si trova.
In aggiunta a ciò, un'ulteriore ipotesi formulata dalle CNN risiede nella **località**. Per via della natura dell'operatore di convoluzione difatti, le features estratte da un determinato layer tengono conto solo di una ristretta parte dell'immagine con il campo ricettivo che aumenta al crescere della profondità del modello.

Viceversa, i modelli ViT non tengono conto di nessuna di queste ipotesi in quanto solo i layer MLP sono locali ed equivarianti alla traslazione, mentre i moduli di self-attention sono globali. Ciò, può in teoria garantire ai modelli ViT di superare le prestazioni delle reti resnet in quanto a differenza di queste sono in grado di comprendere globalmente un'immagine. Se cioè le reti convoluzionali sono capaci di rilevare le caratteristiche ovunque esse si trovino, sarà necessario definire dei modelli particolarmente profondi affinchè questi riescano ad apprenderne le dipendenze a lungo raggio. Al contrario i ViT, per mezzo del meccanismo di attenzione codificano ad ogni livello il contenuto di

un'immagine valutandone le dipendenze tra patch anche distanti tra loro sin dai primi layer. A sostegno di ciò, in Figura 4, sono riportate le distanze medie di attenzione rilevate per ciascuna head a seguito del training di un modello definito nell'articolo di riferimento [3] in funzione della profondità dei layer in cui queste si trovano. Da una sua analisi si evince di quanto già nei primi livelli i moduli di MHA tendano a concentrarsi sulla relazione tra zone dell'immagine distanti tra loro.

Si noti quindi di quanto tutto ciò possa consentire da un lato di sovraperformare le prestazioni delle attuali reti convoluzionali grazie alla capacità dei modelli ViT di comprendere globalmente le immagini, ma di quanto dall'altro renda necessaria una grande quantità di dati per l'addestramento affinchè i modelli in analisi siano in grado di generalizzare correttamente, come evidenziato dai risultati degli esperimenti effettuati per questo elaborato e riportati al paragrafo 5.

3.2. Risultati noti

Per quanto detto nel merito del paragrafo 3.1, l'assenza di inductive bias tipica dei modelli Transformers rende necessario l'addestramento su dataset di grandi dimensioni affinchè questi siano in grado di apprendere le dipendenze globali. Quando si ha a che fare con tali modelli è quindi pratica comune eseguire una prima fase di pre-training su molti dati ed un successivo adattamento a task con associati set di dati più piccoli. In relazione a ciò, non fa eccezione il lavoro descritto all'interno dell'articolo di riferimento [], per il quale i ViT sono stati prima di tutto addestrati sui dataset ILSVRC-2012 Imagenet (1K classi e 1.3M di immagini), Imagenet-21k (21kclassi e 14M di immagini) e JFT (18kclassi e 300M di immagini). Ai modelli risultanti è stata quindi applicata un'operazione di fine-tuning sui dataset CIFAR-10, CIFAR-100, Oxford-IIIT Pets, Oxford Flowers-102 ed ai 19 tasks della suite di classificazione VTAB. Si noti come l'operazione in questione sia stata implementata sostituendo il layer lineare interno al modulo MLP head dei modelli preaddestrati con un nuovo layer dai pesi inizializzati di dimensione DXK con D dimensione del Transformer Encoder e K numero delle classi del nuovo task.
Al fine di valutare la scalabilità dei modelli e quindi le prestazioni ottenute al variare della loro dimensione, sono state definite tre architetture che in ordine di grandezza prendono il nome di ViT-Base (ViT-B), ViT-Large (ViT-L) e ViT-Huge (ViT-H). Ciascuno di questi tre modelli è stato quindi pre-addestrato sui suddetti tre dataset di grandi dimensioni considerando una risoluzione delle immagini di input di 224x224 e ciascuno di questi è poi stato sottoposto a fine-tuning sugli altri set di dati alla più alta risoluzione di 384x384. Ogni modello è stato inoltre valutato considerando una suddivisione delle immagini di input sia in patch di dimensioni 16x16 che 32x32. In aggiunta a ciò, l'intera procedura è stata ripetuta considerando i modelli ibridi con una resnet50 come rete backbone, e le performance ottenute dalle varie configurazioni sono state confrontate con quelle rilevate addestrando i modelli BiT (Big Transfer).

L'analisi dei risultati, denota che i modelli ViT ottengono delle performance superiori rispetto alle reti Resnet prese in considerazione per datasets sufficientemente grandi. I fine-tuning delle architetture in analisi ottengono difatti un'accuratezza in media inferiore rispetto ai modelli di comparazione quando il pre-training viene effettuato sul più piccolo dataset ILSVRC-Imagenet, per poi migliorare gli approcci convoluzionali se pre addestrati sui set Imagenet-21k e JFT contenenti molte più immagini.
Un ulteriore dato riportato, infine, evidenzia che nella maggior parte degli esperimenti, i ViT addestrati dividendo l'immagine in patch 16x16 ottengono dei risultati migliori rispetto ai corrispettivi modelli allenati considerando partizioni 32x32, a fronte di un costo computazionale richiesto ovviamente maggiore.

4. Implementazione

Al termine dell'analisi dei modelli ViT, descritti al paragrafo 3, affinchè fosse possibile compararne le performance su dataset di piccole dimensioni rispetto a quelle ottenute dalle tradizionali CNN residuali, la loro architettura è stata implementata in linguaggio di programmazione Python utilizzando il framework di deep learning PyTorch alla versione 1.4.0. Per quanto riguarda il modello resnet18 invece, all'interno dell'elaborato, viene fatto ricorso all'implementazione fornita dalla libreria torchivision alla versione 0.5.0. Il progetto, disponibile su Github [2], si articola nello specifico su più moduli le cui caratteristiche sono discusse di seguito nel dettaglio.

4.1. ViT

Ruolo centrale del lavoro è sicuramente rappresentato dall'implementazione del modello Vision Transformer, la quale è incapsulata all'interno del progetto nella classe ViT del file *ViT_model.py*. La classe in questione, nello specifico, estende il modulo nn.Module interno alla libreria Pytorch e nel farlo prevede la definizione di un modello che esegue in sequenza le seguenti tre componenti:

- EmbeddingLayer: ricevuta un'immagine di input ne fornisce la rappresentazione composta dai patch embeddings. Per farlo l'immagine viene in primo luogo partizionata in patch aventi dimensione definita come parametro passato alla classe. Viene quindi usato un layer lineare della libreria Pytorch per proiettare ciascuna patch appiattita in un vettore di dimensione dell'encoder. A tali rappresentazioni, ne viene quindi aggiunta un'altra della stessa dimensione che identifica il

cls token. A ciascuna proiezione vengono quindi sommati i positional encoding, parametro apprendibile dal modello durante il training al pari del token di classe.

- TransformerEncoder: prende in ingresso i patch embeddings ed implementa il meccanismo di codica caratteristico dei modelli ViT. Il modulo è composto da una serie di blocchi encoder elementari, ciascuno dei quali applica come prima operazione una connessione residuale sul risultato della multi-head attention, il quale è a sua volta preceduto dall'esecuzione di un layer di normalizzazione. Dopo di ciò, viene applicata una nuova normalizzazione seguita da una seconda connessione residuale sul risultato dell'esecuzione della rete feedforward implementata così come è stata definita a livello teorico nel paragrafo 3.

- MLPHead: riceve in input la prima uscita del TransformerEncoder e, dopo avervi applicato un layer di normalizzazione, implementa un layer lineare che lo mappa in un vettore di lunghezza pari al numero delle classi del dataset, parametro questo ricevuto in input dalla classe.

4.2. Hybrid ViT

L'architettura ibrida dei modelli ViT è definita dalla classe base astratta HybridViT interna al file *hybrid_ViT_model.py*, la quale utilizza la stessa pipeline descritta per la versione base dei Vision Transformer. La sostanziale differenza rispetto a quest'ultima risiede nella ridefinizione del modulo di embedding, il quale riceve in input la feature map dell'immagine di ingresso generata da una rete CNN di backbone. Per il progetto è stata definita la classe concreta Resnet18HybridViT che utilizza le features estratte dal quarto blocco convoluzionale della rete resnet18, per la quale è stato utilizzato il modello incluso nella libreria torchvision.

4.3. Training

Al fine di addestrare i modelli implementati e di valutarne le performance il progetto include lo script *train_models.py*. Il modulo di codice, che per il suo funzionamento fa ausilio delle funzioni incluse nel file *utils.py*, è eseguibile in tre diverse modalità per mezzo della specifica di un suo parametro denominato *eval_type*. In tal senso è possibile selezionare la tipologia di validation con la quale il modello indicato come parametro allo script viene addestrato sui dati di train e valutato appunto su quelli interni al validation set. In questo caso al termine dell'operazione vengono salvati due grafici che indicano l'andamento della loss function e dell'accuratezza rilevati sia sui dati di training che su quelli di validazione. In alternativa, vi è la modalità di testing durante la quale il modello viene allenato sempre sui dati di training per essere valutato ad ogni epoca su quelli di test. Alla fine della computazione, i risultati rilevati comprendenti la top-1 accuracy rilevata vengono salvati in un apposito file csv. Infine, è possibile selezionare un tipo di esecuzione che unisca le due modalità così da valutare sia il comportamento sul validation che quello sul test set su una singola esecuzione.
In ogni caso il modello del quale eseguire il training viene addestrato per un numero di epoche configurato come parametro allo script utilizzando una cross entropy loss come funzione di perdita ed una softmax per predire la classe dell'immagine a partire dal vettore di logit generato dal modello in questione. Inoltre, è possibile optare per l'utilizzo di Adam oppure di SGD come ottimizzatore, la quale scelta può essere effettuata settando l'apposito parametro dello script, al pari di tutti gli altri fattori caratterizzanti del modello da addestrare e per la quale si rimanda alla più completa descrizione interna al file readme del progetto [2].

5. Esperimenti

La parte sperimentale del lavoro svolto per il presente elaborato si è basata sulla valutazione delle performance rilevate dai modelli ViT quando addestrati su dataset di piccole dimensioni. Al fine di comprendere quanto l'assenza di inductive bias motivata nel corso del paragrafo 3.1 ne influenzasse le performance nel suddetto caso di riferimento, queste sono state confrontate con quelle ottenute addestrando sugli stessi dataset una rete resnet18, la quale rappresenta una rete convoluzionale residuale a 18 layers.

5.1. Modelli di confronto

Nell'articolo di riferimento [3] sono stati definiti i modelli Vision Transformers di grandi dimensioni denominati ViT-B, ViT-L e ViT-H. Dal momento che, a differenza di quanto riportato nel lavoro in questione, per il progetto in analisi si era interessati all'addestramento dell'architettura su set di dati più ristretti, alle suddette configurazioni se ne sono aggiunte altre due definite come ViT-XS e ViT-S, i cui parametri sono riportati nella Tabella 1. Seguendo la nomencluatura introdotta nel'articolo [3], al nome di ciascun modello di ViT segue la dimensione delle patches nelle quali le immagini vengono partizionate. Per maggior chiarezza, il modello ViT-XS/16 si riferisce ad esempio alla configurazione ViT-XS con immagini divise in regioni 16x16. In aggiunta a questi poi, per ciascun modello è stata considerata la variante ibrida che estrae le features map dal quarto layer di una resnet18.

5.2. Datasets

Come detto, l'obiettivo dell'elaborato è stato quello di analizzare il comportamento dei modelli su set di dati contenenti meno immagini rispetto a quelli considerati per i

Model	Layers	Hidden size D	MLP size	Heads	Parameters
ViT-XS	8	128	384	8	3M
ViT-S	10	256	768	8	9M

Tabella 1. Configurazione dei modelli ViT definiti.

pre-training dell'articolo [3]. Per adempiere a ciò, si sono selezionati i tre datasets CIFAR-10, CIFAR-100 ed Oxford-IIIT Pets, i quali sono associati ad alcuni tra i task scelti per il fine-tuning nel paper stesso. Nello specifico, i datasets CIFAR-10 e CIFAR-100 contengono entrambi 60k immagini 32x32 associate rispettivamente a 10 e 100 classi. Le immagini in questione sono in entrambi i casi suddivise in modo che ve ne siano 50k di train e 10k di test. Il set di dati Oxford-IIIT pets invece ha al suo interno 7350 immagini di dimensione variabile raffiguranti 37 specie di cani e di gatti. Lo split in questo caso prevede un training set di 3680 immagini ed un test set di 3670 elementi. Per l'elaborato, se ne è considerato il più semplice task di classificazione binaria atto cioè a predire, data un'immagine, se questa contenga un cane oppure un gatto.
Seguendo quanto riportato nell'articolo [3], il validation set è stato definito estraendo il 2% dei dati di train per i datasets CIFAR-10 e CIFAR-100 ed il 10% per Oxford-IIIT Pets.

5.3. Parametri di training

Al pari di quanto effettuato nel lavoro di riferimento [3], tutti i modelli sono stati addestrati con immagini alla risoluzione di 224x224. Visto il ristretto numero di campioni contenuti nei datasets presi in considerazione, al fine di ridurre il fenomeno di overfitting, le immagini sono state sottoposte ad un processo di data augmentation atto ad eseguire, a seguito di un resize delle stesse alla dimensione di 250x250, prima un crop alla suddetta risoluzione di training e successivamente un flipping orizzontale casuale. Tutti i modelli sono inoltre stati addestrati considerando un batch size pari a 128 per CIFAR-10 e CIFAR-100, ridotto a 64 nei modelli ibridi per far fronte alla loro maggior memoria richiesta, e di 64 per Oxford-IIIT Pets. Per quanto concerne il training della rete resnet18, questo è stato effettuando per ogni dataset utilizzando SGD come ottimizzatore, e, seguendo quanto indicato nel paper [3], con uno weight decay di $1x10^{-5}$ per CIFAR-10 e CIFAR-100. I ViT invece sono stati addestrati come indicato per il pre-training dell'articolo [3] utilizzando Adam e considerando sia modelli che partizionano le immagini di input in patch 16x16 che 32x32. Inoltre, per mitigare l'overfitting altrimenti evidenziato, il training di questi ultimi è stato effettuato inserendo un dropout di 0.2, a differenza delle reti resnet allenate senza l'introduzione di tale tecnica.
La scelta del learning rate è stata effettuata valutando per ciascun modello il comportamento ottenuto sul validation set dei singoli datasets. Per quanto riguarda i ViT, tale valutazione ha coinvolto i modelli con partizionamento in regioni 16x16, ed il learning rate ottimale rilevato è poi stato utilizzato anche per la relativa versione con patch 32x32 e per la corrispondente variante ibrida. I learning rate effettivamente utilizzati sono risultati essere pari a $1x10^{-4}$ per i modelli ViT e per la resnet18 sul dataset Oxford-IIIT Pets e di $1x10^{-1}$ per la rete convoluzionale addestrata su CIFAR-10 e CIFAR-100. Infine, per tutti i modelli è stato applicato un decadimento del learning rate di tipo coseno, come in uso per il pre-training del più piccolo dataset (ILSVRC-Imagenet) tra quelli considerati nel paper [3].

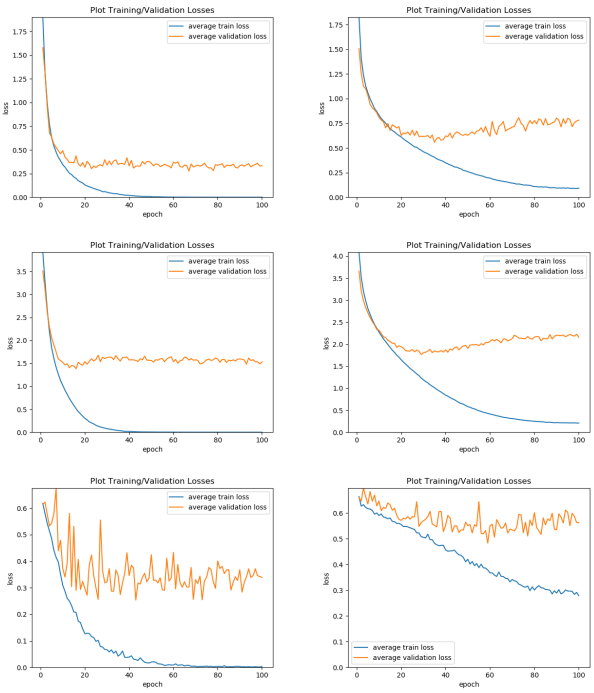

Figura 5. Andamento della loss function rilevato sul training set e sul validation set di ciascun dataset durante il training dei modelli resnet18 (a sinistra) e ViT-S/16 (a destra). Prima riga: CIFAR-10. Seconda riga: CIFAR-100. Terza riga: Oxford-IIIT Pets.

5.4. Risultati

Gli esperimenti effettuati hanno consistito nell'addestrare tutti i modelli sul training set dei tre dataset considerati per un totale di 100 epoche. Le performance sono state quindi comparate per mezzo dei valori di top-1 accuracy ricavati durante il processo di training e riportati nella Tabella

Dataset	ViT-XS/32	ViT-XS/16	ViT-S/32	ViT-S/16	resnet18+ViT-XS	resnet18+ViT-S	resnet18
CIFAR-10	77.70	79.26	80.21	82.14	89.98	91.20	**92.56**
CIFAR-100	51.33	52.30	55.18	55.28	65.29	67.22	**72.54**
Oxford-IIIT Pets	76.07	76.89	76.12	79.10	91.99	**92.06**	91.41

Tabella 2. Top-1 accuracy rilevata sul test set di ciascun dataset addestrando ogni modello per 100 epoche. I ViT sono etichettati dal nome della loro configurazione seguito dalla dimensione delle patch nelle quali vengono divise le immagini di input.

2.
L'analisi dei dati in questione è stata eseguita su due diverse direzioni. In primo luogo ha previsto una comparazione dei risultati ottenuti dai modelli ViT rispetto a quelli della rete convoluzionale. Parallelamente a ciò, sono state inoltre confrontate le performance ricavate con le differenti configurazioni di Vision Transformer definite.

Per quanto concerne la prima parte della suddetta serie di analisi, dall'osservazione dei dati si evince che in nessun caso i modelli ViT si avvicinano alle prestazioni ottenute dalla rete resnet18. Ciò denota quanto effettivamente la mancanza di inductive bias evidenziata nel corso del paragrafo 3.1 non consenta ai modelli di apprendere le dipendenze globali delle immagini necessarie per poter generalizzare efficientemente quando i dati di addestramento non sono sufficientemente numerosi. Ciò si denota già per il dataset CIFAR-10, per il quale la rete resnet18 ottiene circa dieci punti percentuali in più rispetto al miglior modello di ViT rappresentato dalla configurazione ViT-S. Tale divario aumenta poi sia incrementando il numero delle classi come nel caso di CIFAR-100, sia considerando il più semplice task di classificazione binario correlato al dataset Oxford-IIIT Pets che però contiene un numero di campioni estremamente inferiore. A sostegno del fatto che tali prestazioni si devano alla debolezza dei Vision Transformer nel generalizzare su poche immagini, in Figura 5 sono riportati i grafici che mostrano l'andamento della loss function rilevato durante l'addestramento della resnet18 e di ViT-S/16 sui dati di train e su quelli di validation. Dall'osservazione di tali grafici, si denota che, a differenza di quanto avviene per la rete convoluzionale, i modelli in analisi tendono ad overfittare, nonostante l'introduzione di dropout, già dopo circa 40 epoche su CIFAR-10 e CIFAR-100. Sul più piccolo dataset Oxford-IIIT Pet poi, il modello ViT non riesce ad adattarsi perfettamente neppure ai dati di train, con un valore della loss calcolato sugli stessi che rimane notevolmente superiore a quello della resnet18, la quale vi si adatta perfettamente già dopo circa 45 epoche.

Studiando invece le differenze nelle performance ottenute dai vari modelli di Vision Transformer, si rileva in primo luogo che l'architettura ViT-S migliora i risultati della meno complessa ViT-XS. Ciò si deve, come evidenziato nell'articolo [3], non tanto ad una maggior larghezza del primo modello rispetto al secondo, quanto piuttosto nella sua più elevata profondità. ViT-S infatti utilizza due layers di encoder in più rispetto a ViT-XS e ciò si ripercuote in una maggior capacità delle head poste negli ultimi blocchi di concentrare l'attenzione tra patch maggiormente distanti tra loro nello spazio immagine, il che consente a sua volta al modello di apprendere meglio le dipendenze globali tra i soggetti raffigurati nei campioni.

Esattamente come rilevato dai risultati riportati nel paper [3], anche in questo caso i ViT traggono poi maggiori benefici da un partizionamento delle immagini in patch più piccole, in quanto dividendole in regioni 16x16 i risultati tendono ad essere migliori rispetto alle versioni che partizionano in aree di 32x32. Si nota comunque che questo divario non risulta essere particolarmente marcato, soprattuto per il caso del modello ViT-XS. Data quindi la maggior complessità di modelli che elaborano immagini divise in patch di dimensione inferiore, i risultati inducono a pensare che, in caso di pre-training su poche immagini, il rapporto tra costo computazionale richiesto e benefici apportati alle performance verta dalla parte di modelli che partizionano in regioni più grandi, come meglio illustrato al paragrafo 5.5.

Infine, in ultima analisi, si sono analizzate le prestazioni ricavate con i modelli ibridi. In tal senso, si nota di come questi facciano uso delle features prodotte dalla rete convoluzionale di backbone così da consentire ai Vision Transformer di avvicinare le performance ottenute dalla resnet18. Tuttavia, solo nel caso del dataset Oxford-IIIT Pets questi sono realmente in grado di migliorare, seppur di poco ed a fronte di un costo maggiore, gli scores rivelati con la suddetta rete.

5.5. Costo computazionale

In ultima analisi, al fine di valutare la scalibilità ed il costo computazionale dei modelli, in Figura 6 è riportato un grafico che ne mostra le performance in funzione del costo espresso usando la misura di MACs. Dall'osservazione di tale rappresentazione, si nota in primo luogo che effettivamente una tipologia di ViT tra quelle definite (ViT-S/16) ha un costo comparabile a quello richiesto dalla resnet18, il che consente di comprendere quanto quest'ultima riesca su piccoli dataset non solo ad ottenere delle performance nettamente superiori, ma anche a farlo utilizzando le stesse risorse dei Vision Transformer.

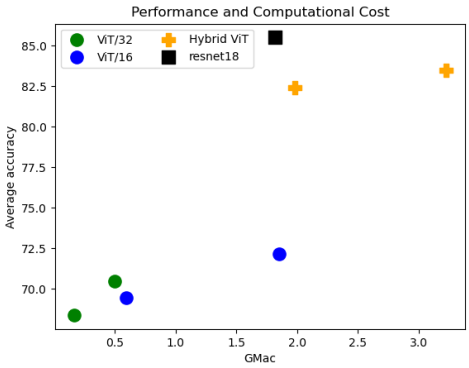

Figura 6. Grafico che mostra il rapporto tra costo computazionale espresso in Macs e accuratezze medie rilevate sui tre dataset dai differenti modelli.

In aggiunta a ciò, si nota anche di come in effetti i ViT che dividono le immagini in patch 32x32 ottengano dei risultati leggermente peggiori rispetto ai modelli che partizionano le stesse in regioni più piccole, ma riescono a farlo solo ad un costo effettivamente di molto superiore a parità di configurazione in analisi.

Infine, discorso analogo vale anche in riferimento ai modelli ibridi messi a confronto con la rete resnet18. Questi ultimi difatti, come già osservato durante l'interpretazione dei risultati, ottengono performance che si avvicinano a quelle di tale rete convoluzionale a fronte però di un costo aggiuntivo. Quest'ultimo risulta di molto superiore soprattutto nel caso del modello ibrido che utilizza ViT-S, i cui risultati si rivelano essere comunque estremamente simili rispetto al modello facente uso dell'architettura più piccola e molto meno onerosa.

6. Conclusioni

In conclusione, l'elaborato discusso nel merito della presente relazione consiste in un'analisi delle performance ottenute dall'architettura Vision Transformer presentata nel paper *An Image Is Worth 16X16 Words*[3] quando questa viene addestrata su dataset di piccole dimensioni.
Dai risultati prodotti si evince quanto tali modelli, che in processi di transfer learning sono capaci di sovraperformare lo stato dell'arte identificato dalle reti convoluzionali residuali, non riescano ad eguagliare le prestazioni ottenute da quest'ultime quando il pre-training avviene direttamente su task con associate poche immagini. Ciò si deve alla mancanza di inductive bias dei Vision Transformer rispetto alle tradizionali CNN, fattore questo che da un lato rappresenta un punto di forza dei modelli in analisi quando si dispone di molti dati, ma che dall'altro non gli consente di apprendere correttamente le dipendenze globali delle quali questi necessiterebbero per una corretta generalizzazione sui dati di test quando le immagini di train non sono sufficientemente numerose.

Riferimenti bibliografici

[1] Pytorch framework. `https://pytorch.org`.

[2] Vision transformer project on github. `https://github.com/AlessandroArezzo/vision_transformer`.

[3] Anonymous. An image is worth 16x16words: Transformers for image recognition at scale, 2020.

[4] N. P. J. U. L. J. A. N. G. K. I. P. Ashish Vaswani, Noam Shazeer. Attention is all you need, 2017.