

Deep Learning-based system for document recognition

Authors: Longarini Lorenzo, Ramovini Loris,
Tutor: Rosati Riccardo

Contributing authors: s1110740@studenti.univpm.it;
s1109759@studenti.univpm.it;

Abstract

Il seguente progetto si pone come obiettivo quello di classificare e segmentare documenti di identità per migliorare il sistema di riconoscimento attualmente in utilizzo dall'azienda Ubisive che viene impiegato su dispositivi mobile. Verranno mostrate diverse architetture di modelli di apprendimento, tra cui CNN e ViT per quanto riguarda l'aspetto legato alla classificazione, mentre verranno mostrati modelli quali UNet e SAM per l'aspetto legato alla segmentazione. Il dataset che ci è stato fornito risulta essere molto limitato e per questo motivo è stato incrementato con un dataset pubblico (PRADO) ed è stata effettuato un incremento artificiale di tutti i dati mediante data augmentation.

I risultati mostrano come l'obiettivo legato alla classificazione sia stato ampiamente raggiunto sia con CNN, che con ViT, ottenendo valori superiori al 90%; mentre l'obiettivo legato alla segmentazione resta ancora in parte limitato nonostante i discreti risultati come indicato dalle percentuali di iou quasi all'80%. Infine, un'ulteriore analisi mirata alla Carbon Footprint[1] ci ha permesso di mettere a confronto i consumi legati ai diversi modelli portando l'attenzione anche al fattore ambiente.

Keywords: Classificazione, Segmentazione, CNN, ViT, UNet, SAM

1 Introduzione

Il progetto nasce dalle esigenze dell'azienda Ubisive di modernizzare il loro attuale sistema che si basa sul Feature Matching per poter riconoscere i documenti di identità in un'applicazione mobile. Nel sistema attuale, dopo aver inserito l'immagine del documento, se questo viene classificato correttamente, viene generata la maschera.

L'utilizzo di modelli di Deep Learning dovrebbe consentire una maggiore precisione, efficienza e affidabilità nel processo di classificazione e segmentazione dei documenti rispetto alle metodologie attuali.

Per poter effettuare questi task in maniera corretta ci è stato fornito dall'azienda Ubisive un piccolo dataset contenente immagini di documenti personali suddivisi in fronte-retro a seconda della tipologia.

Lo scopo del progetto era quello di classificare e segmentare le immagini utilizzando reti convoluzionali.

Per poter raggiungere l'obiettivo, dopo aver eseguito alcune ricerche, abbiamo selezionato alcuni modelli di interesse su cui poter effettuare delle prove, in particolar modo:

- VGG16
- MobileNetV2
- MobileNetV3Large
- Resnet50

In base ai risultati ottenuti abbiamo poi selezionato le reti più performanti e le abbiamo allenate per ottenere i risultati finali.

Successivamente, al solo scopo di ricerca abbiamo sperimentato il modello transformer ViT e confrontato i valori con le reti convoluzionali utilizzate in precedenza.

Il task successivo era legato alla segmentazione, ed è stato affrontato mediante l'utilizzo della rete convoluzionale U-Net segmentando a mano le maschere relative al nostro dataset.

Infine, in via sperimentale abbiamo utilizzato il modello SAM sviluppato da Meta che consente di segmentare un'immagine molto rapidamente e con la quale abbiamo poi prodotto alcune delle maschere che abbiamo utilizzato come annotazione per allenare U-Net.

I risultati che andremo a mostrare comprendono anche i valori di carbon footprint.

2 Stato dell'arte

2.1 Classificazione

L'uso del deep learning per la classificazione delle immagini di documenti è un tema molto importante che è esploso negli ultimi anni grazie ai progressi ottenuti con le reti neurali convoluzionali, e alla parallela evoluzione degli hardware, come le GPU, che hanno reso possibili addestramenti e elaborazioni efficienti su grandi moli di dati.

Il tema di interesse è diventato sempre più importante in settori come la sicurezza, il settore finanziario o altri servizi che richiedono l'identificazione degli utenti. Il deep learning applicato alla classificazione delle immagini dei documenti ha consentito di accelerare e semplificare questi processi, portando a una grande riduzione dalla dipendenza di metodi manuali.

Allo stato dell'arte non esistono molti studi riguardanti l'utilizzo di CNN per la classificazione dei documenti, questo perchè tendono a risolvere task più complessi (riconoscimento dei documenti falsi[2]), inoltre, sono rari i paper che mettono in evidenza le reti analizzate per poter effettuare classificazione.

Un'analisi approfondita ci ha permesso di estrapolare alcune informazioni:

- è possibile utilizzare reti convoluzionali per effettuare il task di classificazione dei documenti di identità;
- la rete VGG16 presenta valori elevati per poter risolvere il task;
- la maggior parte dei dataset che contengono documenti personali sono proprietari e non consentono di sfruttare le immagini;
- esistono alcuni dataset pubblici come PRADO che contengono documenti suddivisi a seconda della località.

Interessante per la nostra ricerca è il paper del 2018[3], che tratta l'utilizzo della rete VGG16 nella classificazione di documenti di identità. I risultati mostrati sono ottimali per questo task, è stato raggiunto il 98% di accuratezza valutando solo alcune combinazioni degli iperparametri che possono essere dati in pasto al modello. Il paper mette anche in evidenza l'utilizzo di poche immagini in fase di testing e il fatto che si sarebbe raggiunto un risultato migliore con una numerosità maggiore. Sulla base di questo ci siamo concentrati sullo studio e valutazione di diverse combinazioni di iperparametri. Per poter confermare la qualità di questa rete, è stata messa a confronto con altre reti, per dimostrare quale sia la migliore rispetto al task di interesse.

Una svolta significativa in ambito della classificazione delle immagini è l'introduzione recente dei Trasformer, reti neurali che apprendono il contesto e di conseguenza il significato, tracciando le relazioni in dati sequenziali e di come si influenzano, come le parole in una frase.

Nel nostro caso, di particolare importanza, è ViT, una variante dei trasformer che si è dimostrata molto efficiente nella gestione delle immagini. Tramite questo è possibile ottenere informazioni globali e relazioni complesse all'interno della stessa immagine, anche tra porzioni lontane ed ottenere una comprensione migliore del contesto visivo.

A differenza delle CNN, un grande vantaggio di ViT è la possibilità che può essere adattato a immagini di diverse dimensioni senza modifiche all'architettura. Nell'approfondimento di questa tematica di particolare rilievo è il paper del 2021 [4] dove è stato possibile osservare il confronto tra ViT e una CNN come ResNet.

Nello specifico in questo paper sono state valutate le due reti anche in relazione al pre-addestramento su diversi dataset, dimostrando che ViT si comporta in maniera meno efficiente in caso di training set di piccole dimensioni.

2.2 Segmentazione

Il secondo task consiste nella segmentazione delle immagini dei documenti. La segmentazione è una tecnica fondamentale nell'elaborazione delle immagini, che si pone l'obiettivo di identificare e isolare specifiche regioni di interesse all'interno della stessa.

Allo stato dell'arte la maggior parte degli studi si focalizza sulla segmentazione di immagini dei documenti per isolare ad esempio il volto del soggetto per ottenere un riconoscimento facciale o per estrarre dati personali provenienti dai documenti stessi. Questi task sono di maggior complessità rispetto a quello di nostro interesse, dunque è stato difficile trovare paper che analizzino questo compito. Un articolo del 2020 [5] affronta la segmentazione semantica facendo il confronto tra due diversi approcci come U-Net e Fast Fully Octave Convolutional Neural Network. Nostro interesse è stato l'implementazione della U-Net ad hoc con la quale siamo riusciti a generare le maschere dei documenti nella loro interezza.

Un articolo recente[6], tratta di un nuovo modello e set di dati per la segmentazione delle immagini, che prende il nome di "Segment Anything Model (SAM)". Questo è capace mostrare grande efficacia nel generalizzare le maschere a distribuzioni di dati differenti da quelli visti in fase di training. Queste possono essere ottenute a partire da punti o caselle fornite in input oppure, in assenza di quest'ultimi, generando tutte quelle possibili.

3 Materiali e Metodi

3.1 Dataset

Il dataset che abbiamo utilizzato era molto limitato e conteneva alcune tipologie di documento che sono state suddivise da Ubisive come segue:

- dl_I10 contenente patenti in condizioni non ottimali;
- dl_I11 contenente patenti in condizioni ottimali;
- ic_CIE contenente carte di identità in formato tessera del vecchio formato;
- ic_TES contenente carte di identità in formato tessera del formato attuale.

Ogni classe presentava i documenti suddivisi in fronte-retro come in Fig. 1:



Fig. 1: Documenti di identità suddivisi per classe di appartenenza

Le immagini utilizzate sono in totale 456 e sono state suddivise come segue:

Table 1: Documenti di identità suddivisi per classe di appartenenza

Etichetta	Tipo	Numero
dl_I10	Fronte	9
dl_I10	Retro	9
dl_I11	Fronte	86
dl_I11	Retro	75
ic_CIE	Fronte	95
ic_CIE	Retro	83
ic_TES	Fronte	22
ic_TES	Retro	22

Dato che il nostro task si concentra solamente su documenti italiani, abbiamo utilizzato i documenti messi a disposizione da PRADO, un registro pubblico online dei documenti di identità e di viaggio autentici con il quale è stato possibile creare la classe dei documenti non riconosciuti.

Abbiamo, quindi, selezionato i documenti di identità e patenti fronte-retro di alcuni paesi appartenenti all'unione europea come mostrato in tabella:

Table 2: Documenti di identità suddivisi per paese di appartenenza

Documento	Carta di identità	Patente
Grecia	2	4
Svizzera	4	4
Gran Bretagna	2	4
Francia	4	4
Germania	2	4
Spagna	4	2
Polonia	6	4
Romania	2	2
Total	26	28

¹I documenti considerati sono fronte-retro.

Seguendo la suddivisione fornita da Ubisive abbiamo creato le classi doc_NR in modalità fronte e retro come mostrato in figura 2:



Fig. 2: Documenti di identità dei paesi europei

3.2 Procedura sperimentale

3.2.1 Classificazione con CNN

La prima problematica si è presentata nel numero di dati su cui svolgere classificazione: essendo in numero limitato abbiamo deciso di effettuare data augmentation per poterli portare ad un numero sufficientemente elevato. L’augmentation è stata effettuata cercando la classe più numerosa e pareggiando tutte le classi allo stesso numero. Le copie che si sono venute a creare sono state modificate scegliendo una tra le possibili trasformazioni di seguito elencate:

- rotazione di 30°;
- aumento della luminosità.

Queste modifiche all’immagine rappresentano condizioni reali in cui un utente si può trovare a scattare le foto. Grazie alla data augmentation siamo riusciti a portare il nostro dataset da 456 a 960 immagini. Abbiamo, successivamente, deciso di effettuare qualche test relativamente alla suddivisione del dataset in:

- 70% training;
- 10% validation;
- 20% testing.

I set di training, validation e testing sono state creati tramite data generator con una batch size di 16 e input size delle immagini di (224,224,3).

Sistemato il dataset abbiamo sviluppato il primo task, il quale era legato alla classificazione delle immagini. Dagli studi che abbiamo svolto abbiamo deciso di testare le quattro reti descritte nell’introduzione.

Se da una parte VGG e Resnet50 rappresentano lo stato dell’arte e mostrano ottime prestazioni, dall’altra abbiamo due reti con architetture leggere per poter essere inserite in un’applicazione mobile.

Per poter testare tutti i modelli è stato necessario implementare una grid search, congelare la rete e inserire diverse tipologie di strati densi con i seguenti valori:

Table 3: Grid search dei modelli selezionati, con ottimizzatore SGD, momentum pari a 0.9, LR pari a 0.001, 10 epochhe e 10 classi finali

Model	1st FC	2nd FC	3rd FC	4rd FC
VGG16 / Resnet50 / MNetV3L / MNetV2	/	/	/	/
Resnet50	2048	1024	512	/
MNetV3L	1000	500	250	/
MNetV2	1280	640	320	/
VGG16 / Resnet50 / MNetV3L / MNetV2	4096	1024	512	/
VGG16 / Resnet50 / MNetV3L / MNetV2	4096	2048	1024	512

In base ai valori ottenuti abbiamo poi deciso di selezionare VGG16 e MobileNetV3Large come reti su cui effettuare un testing più mirato, utilizzando un’ulteriore grid search per selezionare quali iperparametri fossero ideali per entrambi:

Table 4: Grid search dei modelli selezionati con momentum pari a 0.9, 10 epochhe e 4 layer fc con neuroni rispettivamente pari a 4096,2048,1024,512 e 10 classi finali

Model	Type	LR
VGG16 / MNetV3L	SGD	0.001 / 0.0001 / 0.00001
VGG16 / MNetV3L	Adam	0.001 / 0.0001 / 0.00001

Infine abbiamo selezionato gli iperparametri migliori e abbiamo allenato i modelli su trenta epochhe effettuando un’ulteriore tentativo sulle percentuali dei dati:

Table 5: Divisione di training set, validation set e testing set

Training	Validation	Testing
70%	15%	15%
70%	10%	20%

Sulla base di questi, abbiamo fatto inferenza e mostrato a schermo i risultati con le relative metriche di valutazione. I dati di testing sui quali venivano commessi errori sono stati soggetti ad un’ulteriore analisi effettuata tramite GradCam con la quale ci è stato possibile individuare in quali zone dell’immagine la rete si è focalizzata per estrarre le feature.

I risultati relativi alla classificazione e agli errori rilevati con la GradCam verranno approfonditi nell’apposita sezione.

3.2.2 Classificazione con ViT

Successivamente abbiamo provato a risolvere lo stesso task con un transformer ViT pretrainato su un dataset relativamente piccolo e messo a disposizione da Hugging Face. Abbiamo, quindi, suddiviso i set in:

- 80% training;
- 10% del training set per la validation;
- 20% testing.

Anche in questo caso abbiamo congelato la rete e inserito alcuni strati densi con diverse tipologie di iperparametri, utilizzando AdamW come ottimizzatore e variando il learning rate tra 0.001, 0.0001, 0.00001 e la weight decay tra 0.001, 0.0001, 0.00001 e mantenendo gli ultimi strati con neuroni pari a 4096, 2048, 1024, 512 e 10 classi in uscita.

I risultati relativi a ViT sono riportati nell'apposita sezione.

3.2.3 Segmentazione con Unet

Il task di segmentazione è stato affrontato utilizzando U-Net.

Il dataset è stato suddiviso come segue:

- 80% training;
- 10% validation;
- 20% testing.

Abbiamo segmentato manualmente tutte le immagini messe a disposizione da Ubisive grazie alla piattaforma Open Source di labeling chiamata Label Studio, con il quale è stato possibile creare le maschere e successivamente esportarle come jpg.

Abbiamo implementato U-Net con una profondità pari a quattro, in cui vengono applicati $64 \times n$ filtri con n che raddoppia ad ogni strato di profondità fino ad arrivare ai 1024 della bottleneck. Abbiamo seguito due approcci per valutare quale tipologia di U-Net fosse più rapida e consumasse di meno:

- convoluzioni 3×3 ad ogni filtro
- covoluzioni separabili 3×1 e 1×3 così da ridurre il numero di operazioni

Per ogni tipologia abbiamo effettuato test con input size pari a 256×256 e 512×512 con batch size pari ad 8. Infine, abbiamo effettuato un ulteriore test con input size pari a 256×256 e impostando la batch size pari a 16, questo perchè consente una convergenza più rapida.

I risultati per U-Net sono descritti nell'apposita sezione.

3.2.4 Segmentazione con SAM

Infine, abbiamo utilizzato il nuovo modello di segmentazione SAM in cui non è stato necessario allenare un modello, ma nel quale abbiamo testato alcuni approcci:

- Segmentazione tramite punti: abbiamo fornito alcuni punti presenti all'interno dei nostri documenti che siamo riusciti a ricavare dal JSON associato alle

- maschere. In questo modo è stato possibile estrarre correttamente la maschera ad esso associata;
- Segmentazione totale: abbiamo segmentato in modo totale l'immagine da cui vengono ricavate n maschere e di cui abbiamo selezionato quella di interesse.

Il modello SAM ci ha permesso di estrarre alcune delle maschere che poi sono state utilizzate nel modello U-Net. I risultati relativi a SAM sono stati descritti nell'apposita sezione.

3.3 Metriche e Hardware utilizzati

Mostriamo qui le metriche utilizzate per valutare le prestazioni dei modelli analizzati:

- Per la classificazione abbiamo valutato l'accuratezza, metrica definita come il rapporto tra il numero di campioni classificati correttamente e il numero totale di campioni.
- Per la segmentazione abbiamo valutato la IoU, metrica che viene definita come il rapporto tra l'area intersezione e l'unione tra la maschera di segmentazione prevista e la maschera di segmentazione di riferimento.

Abbiamo utilizzato Google Colab per eseguire il codice su Cloud e sfruttare la potenza di calcolo fornita da Google. Sono disponibili diverse opzioni hardware:

- Le CPU sono versatili e adatte per la maggior parte dei compiti.
- Le GPU sono più utilizzate nel caso di esecuzione di molti calcoli.
- Le TPU sono le più performanti in quanto sono in grado di eseguire più operazioni in parallelo rispetto alle GPU.

Queste risorse messe a disposizione degli utenti sono limitate e variano a seconda della fluttuazione della domanda. Nel nostro caso abbiamo sfruttato le GPU e TPU solamente per carichi di lavoro elevati. Da notare che Google Colab assegna in modo casuale la potenza dell'hardware tra 42.5W e 120W.

4 Risultati e Discussioni

In seguito ai molteplici esperimenti effettuati sui diversi modelli, possiamo affermare di aver raggiunto ottimi risultati, anche e grazie al fatto che il task non fosse eccessivamente laborioso.

4.1 Classificazione CNN

4.1.1 GridSearch

In base all'analisi effettuata in precedenza, possiamo affermare che il miglior modello per il nostro task di classificazione è risultato essere VGG16, mentre MobileNetV3Large risulta essere il miglior modello tra le architetture più leggere. In tabella vengono mostrati i valori in fase di training delle quattro reti analizzate:

Table 6: GridSearch dei modelli allenati su 10 epoches

Modello	Loss	Accuracy	Val_Loss	Val_Accuracy
VGG16	0.004	0.99	0.20	0.96
Resnet50	0.54	0.91	0.76	0.73
MobileNetV2	1.23	0.77	1.27	0.72
MobileNetV3Large	0.59	0.90	0.62	0.88

Dopo aver ottenuto i risultati abbiamo deciso di fare inferenza su VGG16 e MobileNetV3Large, ottenendo valori di accuratezza rispettivamente di 98% e 76%.

In seguito, abbiamo testato su quest'ultimi i vari iperparametri ed abbiamo ottenuto i seguenti risultati:

Table 7: Grid search dei modelli selezionati con momentum pari a 0.9, 10 epochhe e 4 layer fc con neuroni rispettivamente pari a 4096, 2048, 1024, 512 e 10 classi finali

Model	Type	Loss	Val_Loss	Accuracy %	Val_Accuracy %
VGG16	SGD	0.013/0.002/0.0018	0.02/0.026/0.027	99/100/100	99/99/99
VGG16	Adam	0.15/2.06/1.06	0.14/0.02/0.04	97/100/100	95/99/97
MNetV3L	SGD	0.13/0.09/0.094	0.14/0.12/0.11	98/97/98	96/94/94
MNetV3L	Adam	0.24/0.08/0.06	0.21/0.31/0.35	93/97/98	92/95/94

4.1.2 VGG16

Abbiamo allenato su 30 epochhe VGG16 utilizzando una callback di early stopping fissando il valore di patience a 5 ed abbiamo ottenuto i seguenti risultati:

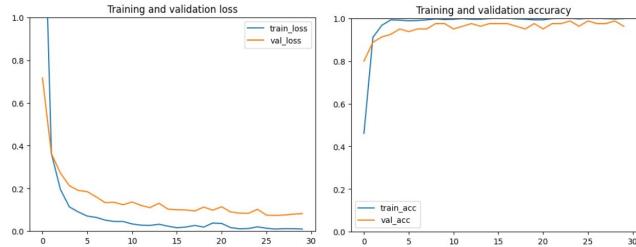


Fig. 3: Valori di Loss, Val_Loss, Accuracy e Val_Accuracy di VGG16

Successivamente abbiamo fatto inferenza ottenendo una testing accuracy pari al 98%:

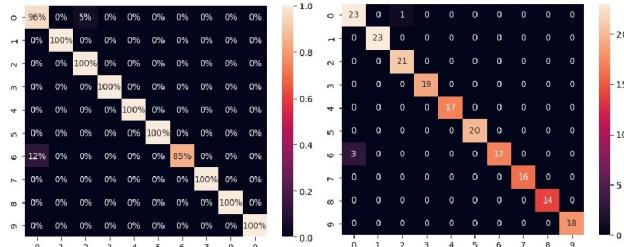


Fig. 4: Matrici di confusione del modello VGG16

Infine abbiamo valutato i casi di errore con la GradCam come mostrato in Fig.4:

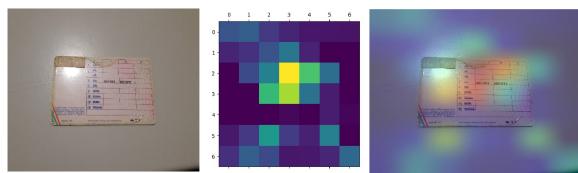


Fig. 5: GradCam relativa agli errori del modello VGG16

Da questa analisi è emerso come la rete tenda a volte ad estrarre le feature nella parte alta della patente e a confondere conseguentemente le classi dl_I10 e dl_I11.

4.1.3 MobileNetV3Large

Abbiamo allenato su 30 epochi anche MobileNetV3Large utilizzando una callback di early stopping fissando il valore di patience a 5 ed abbiamo ottenuto i seguenti risultati:

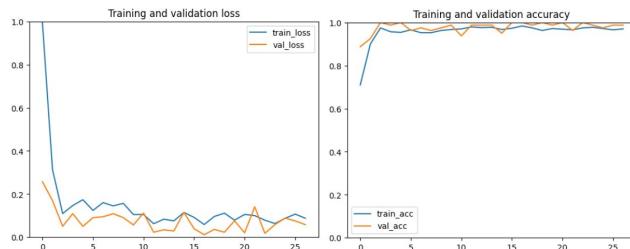


Fig. 6: Valori di Loss, Val Loss, Accuracy e Val Accuracy di MobileNetV3Large

Successivamente abbiamo fatto inferenza sulla rete ottenendo una testing accuracy pari al 96%:

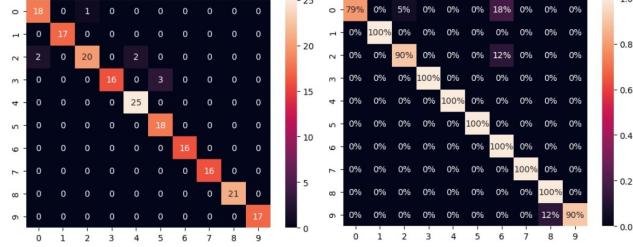


Fig. 7: Matrici di confusione del modello MobileNetV3Large

Infine abbiamo valutato i casi di errore con la GradCam come mostrato in Fig. 3:

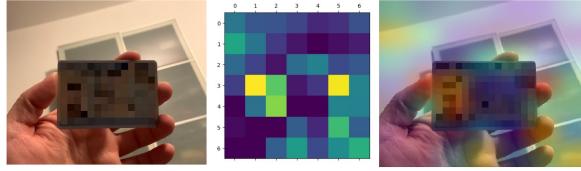


Fig. 8: GradCam relativa agli errori del modello MobileNetV3Large

Da questa analisi è scaturito il fatto che le immagini delle carte di identità della classe ic_CIE vengono confuse con quelle della classe ic_TES a causa del focus nella zona situata vicino alla foto del documento.

4.1.4 ViT

Per quanto riguarda il modello ViT abbiamo allenato su 10 epoch le diverse metriche su cui è stato valutato e da cui è scaturito che le migliori metriche riguardano AdamW con Learning rate pari a 0.0001 e Weight decay pari a 0.0001. Il vero e proprio allenamento è stato effettuato su 30 epoch totali utilizzando una callback di early stopping fissando il valore di patience a 5, dal quale sono derivati i seguenti risultati:

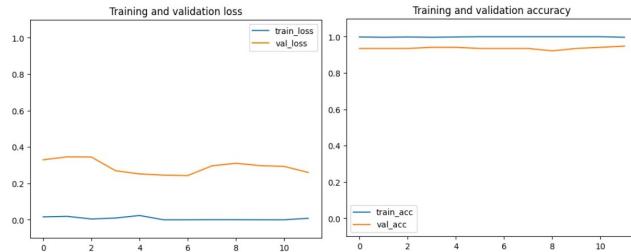


Fig. 9: Valori di Loss, Val_Loss, Accuracy e Val_Accuracy del modello ViT

I valori di loss e val_loss ci indicano che il nostro modello potrebbe essere soggetto ad overfitting, e questo risulta più evidente quando alleniamo il modello su un numero maggiore di epoche. Questo è dovuto alla piccola mole di dati impiegata rispetto alla quantità di cui necessita normalmente ViT e può essere risolto incrementando il dataset.

Infine, abbiamo fatto inferenza sulla rete ottenendo una testing accuracy pari al 95%:

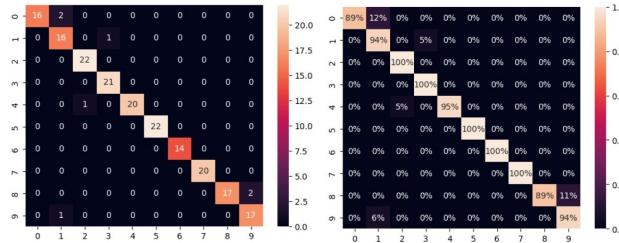


Fig. 10: Matrici di confusione del modello ViT

4.1.5 Carbon Footprint

Una valutazione molto interessante è stata effettuata in relazione ai consumi di CO₂ di ogni modello e che ci consente di poter trovare un compromesso tra efficienza ed emissioni. Nella seguente tabella riportiamo i consumi delle reti effettuati su 10 epoche:

Table 8: Consumi in termini di Kg di CO₂ per ogni modello utilizzato, su 10 epoche

Model	CO ₂ Kg
VGG16	$4,7 \times 10^{-3}$
Resnet50	$4,5 \times 10^{-3}$
MNetV2	$2,45 \times 10^{-5}$
MNetV3L	$4,5 \times 10^{-3}$
ViT	4×10^{-4}

I valori mostrati potrebbero essere condizionati dalla tipologia di processore fornita da Colab, problematica che affronteremo nelle conclusioni.

4.2 Segmentazione

4.2.1 U-Net

Nell'allenamento della rete U-Net abbiamo seguito i due diversi approcci elencati nella sezione 3.2.3.

In entrambi abbiamo condotto l'allenamento su 50 epoche con due input size differenti e pari a 256×256 e a 512×512 , ottenendo i risultati come seguono:

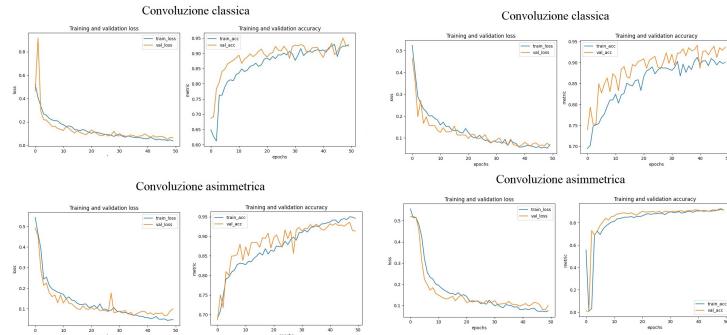


Fig. 11: Valori di Loss, Val_Loss, Iou e Val_Iou dei due diversi approcci a U-Net con input size pari a 256×256 a sx e pari a 512×512 a dx

I risultati delle inferenze dei due approcci sono pari al 72.8% di iou nel primo caso contro il 73,3% iou del secondo.

Le successive immagini presentano il risultato effettivo:

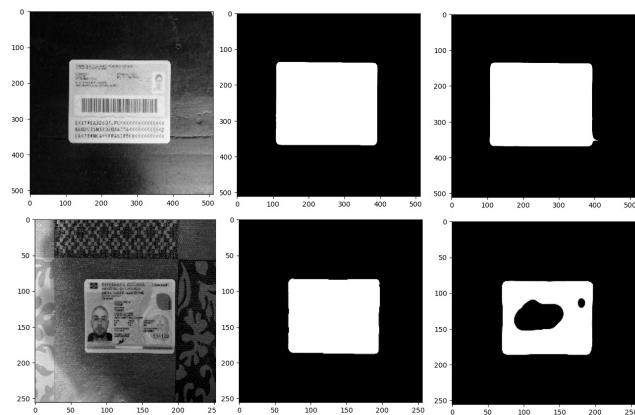


Fig. 12: Le immagini mostrano da sx a dx: il documento, la maschera effettiva e la maschera predetta. In alto troviamo un esempio con input size (256×256), in basso uno con in input size (512×512)

In termini di consumi di CO_2 possiamo notare come nonostante le minori operazioni delle convoluzioni separabili, nel secondo caso abbiamo un leggero aumento delle emissioni: $2.1 \times 10^{-2}\text{kg}$ rispetto $2.9 \times 10^{-2}\text{kg}$ con input size fissato a 256×256 e $6.2 \times 10^{-2}\text{kg}$ rispetto a $6.5 \times 10^{-2}\text{kg}$ con input size fissato a 512×512 .

Possiamo, infine, osservare un notevole miglioramento con l'utilizzo della batch size a 16, con iou pari al 79% con convoluzioni separabili e un notevole diminuzione

dei consumi di CO₂ pari a 1.6×10^{-4} kg. Con le classiche convoluzioni questo miglioramento non risulta, invece, essere troppo rilevante dato che abbiamo iou pari al 70% ed emissioni di CO₂ pari a 2×10^{-2} kg.

4.2.2 SAM

Mostriamo qui i due diversi approcci utilizzati per la segmentazione con SAM. Nel primo caso abbiamo fornito in input due punti all'interno della regione di interesse che si vuole segmentare ed abbiamo ottenuto la maschera desiderata:



Fig. 13: Utilizzo di SAM con due valori di input in ingresso

Nel secondo caso SAM riesce ad segmentare tutte le possibili maschere dall'immagine di input. Qui mostriamo la totalità di esse contrassegnate con diversi colori e la singola maschera riferita al documento. Questo modello ci ha permesso di estrarre alcune delle maschere di nostro interesse, come quella riferita al documento stesso, che poi sono state utilizzate nel modello U-Net per imparare a segmentare correttamente l'oggetto specifico.



Fig. 14: Utilizzo di SAM per ricavare tutte le maschere di un'immagine

5 Conclusioni e Sviluppi futuri

In conclusione possiamo affermare di aver raggiunto l'obiettivo posto da Ubisive nonostante ci siano diversi margini di miglioramento:

- un incremento del dataset potrebbe portare migliore accuratezza, poichè diminuendo l'aumento di dati artificiali riusciamo ad avvicinarci quanto più possibile ad una casistica reale;
- la decisione di utilizzare un'architettura rispetto ad un'altra potrebbe essere influenzata in maniera significativa dai consumi e dall'architettura hardware su cui deve essere utilizzato, per questo motivo potrebbero essere presi in considerazione ulteriori modelli;
- il modello SAM risulta limitato rispetto al nostro task in quanto richiede necessariamente un punto di ingresso nel primo caso e presenta un limite nella scelta della maschera nel secondo caso. Una possibile soluzione potrebbe essere quella di sfruttare la detection messa a disposizione da Meta per facilitare la localizzazione del documento;
- U-Net potrebbe essere resa più profonda per raggiungere valori più accurati, modificando la batch size e i valori di input.
- gli addestramenti delle nostre reti sono stati effettuati completamente sulla piattaforma Google Colab. Qui ci sono state rilasciate in modo randomico CPU e GPU di diversa potenza, ottenendo a volte risultati di emissioni non confrontabili.

References

- [1] CodeCarbon:: <https://codecarbon.io/>
- [2] Bui, T.: Applications of machine learning in ekyc's identity document recognition (2021)
- [3] Vil'as, P.: Classification of identity documents using a deep convolutional neural network (2018)
- [4] Bhojanapalli, S., Chakrabarti, A., Glasner, D., Li, D., Unterthiner, T., Veit, A.: Understanding robustness of transformers for image classification, 1–3 (2021)
- [5] Neves Junior, R.B., Vercosa, L.F., Macedo, D., Bezerra, B.L.D., Zanchettin, C.: A fast fully octave cnn for document image segmentation (2020)
- [6] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S.: Segment anything (2023)