UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN
INGEGNERIA INFORMATICA

SVILUPPO MODELLI DI ADVERSARIAL TRAINING

Relatore: Prof. Loris Nanni

Laureando/a: Michele Russo

ANNO ACCADEMICO 2021 –2022

Data di laurea 22/09/2022

Abstract

La generalizzazione ricopre un ruolo importante inerente all'allenamento delle reti neurali. Essa rappresenta la capacità di una rete neurale di astrarre informazioni importanti per la classificazione, dato un training set, e quindi corrisponde pure alla capacità di classificare correttamente gli unseen data, ovvero l'insieme dei pattern con i quali la rete non è stata allenata. Lo sviluppo di queste tecnologie, pertanto, gioca un ruolo fondamentale, perché utile ad ottenere risultati migliori in termini di apprendimento dei machine learning model.

Questo lavoro prende enorme spunto dalla tecnica del MaxUp, la quale offre un miglioramento delle prestazioni di generalizzazione tramite l'utilizzo di Augumented Data ed adversarial training. Sono stati effettuati vari test comprendenti vari dataset, in particolare riconoscimento di immagini e istopatologie.

Lo scopo è quello di dimostrare come la Data Augumentation e l'adversarial training possano migliorare le performance nella generalizzazione e nell'accurancy, offrendo una maggiore robustezza al metodo in caso di Adversarial Examples.

Indice

Introduzione

Le reti neurali sono tra gli strumenti più utilizzati nel campo dell'intelligenza artificiale, in particolare permettono di ottenere ottime prestazioni nella classificazione di immagini e nei problemi di object detection. Un ruolo importante per l'allenamento di reti neurali risulta essere quello di diminuire l'overfitting, mantenendo ottime performance nel riconoscimento dei pattern in ingresso e, a tale fine, sono state sviluppate molte metodologie. La maggior parte di queste, però, per ottenere ottime performance necessitano di training set molto grandi. Nonostante l'enorme mole di pattern usata, la rete allenata può essere comunque facilmente ingannata permutando di poco l'input (pattern) della rete, questi input sono anche conosciuti come "adversarial examples". L'esistenza degli adversarial examples non rileva solo dei limiti nelle capacità di generalizzazione nelle ConvNets, ma pone anche alcuni problemi di sicurezza. L'adversarial Training inserisce questi esempi, dentro al training set al fine di aumentare la robustezza del modello e diminuire i problemi legati alla misclassificazione. Questi approcci mostrano che i modelli allenati "adversarialmente" sono più robusti in datasets piccoli, con allenamento "supervised". Gli argomenti trattati nella tesi, approfondiscono la tematica relativa agli adversarial examples, sul perché essi costituiscono un problema di sicurezza, e che metodologie sono state sviluppate in tal senso. Nel secondo capitolo invece ci sarà un focus sul modello del MaxUp, che prevede un miglioramento sostanziale delle performance con l'utilizzo di varie tecniche di data augumentation, ed una variante "light" dell'adversarial training. Gli ultimi due capitoli invece avranno un focus sull'implementazione della mia variante del MaxUp, tramite vari test.

1. Adversarial Training

Le reti neurali “profonde” sono uno strumento che permette di avere performance eccellenti nei problemi di “visual and speech recognition”.
La loro forza è data soprattutto dal fatto che sono in grado di eseguire un consistente numero di calcoli non lineari parallelamente, il che rende i loro risultati difficili da comprendere e spesso possono avere delle proprietà controintuitive.
Una di queste è strettamente correlata alla stabilità delle reti neurali. Infatti, ci si aspetterebbe che le reti con ottime performance di generalizzazione sui clean data, siano robuste pure a piccole permutazioni dei pattern in ingresso, cosa che sperimentalmente è inesatta.
Questi risultati suggeriscono che i classificatori basati sui moderni modelli di machine learning, anche quelli che ottengono performance eccellenti, non imparano il vero concetto che permette di avere l’output corretto.
La scoperta di queste vulnerabilità ha dato il via alla ricerca di modelli sempre più robusti e che permettano una maggiore generalizzazione.

1.1 Adversarial Examples

Gli adversarial examples consistono in piccole permutazioni dei pattern in ingresso, in grado di confondere anche le reti neurali più robuste e con performance molto alte.
Supponiamo di avere un TS (training set) di immagini, e dato un pattern x appartenente a TS, un suo adversarial example potrebbe essere il seguente:

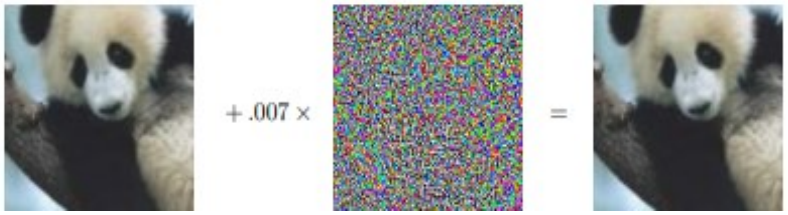
$$\bar{x} = x + \eta \quad (1)$$

Considerando che le immagini sono spesso rappresentate usando 8 bit per pixel, e che la loro rappresentazione scarta valori al di sotto di 1/255; se $||\eta||$ è piccolo abbastanza da essere scartato allora il cambiamento imposto in questo caso, teoricamente, non dovrebbe comportare a misclassificazioni da parte della rete, soprattutto se le classi sono ben separate tra di loro.

Considerando il vettore di pesi w, la precedente diventa:

$$w^T \bar{x} = w^T x + w^T \eta \ (2)$$

La perturbazione inserita causa una crescita della funzione di attivazione $di\ w^T \eta$.
Quindi l'influenza di η all'interno del problema è prettamente legata al crescere non della dimensionalità del problema, ma bensì al crescere di quella di w.
Questa piccola spiegazione da delle delucidazioni su come facilmente possono essere creati degli Adversarial Examples.
Un altro esempio è quello mostrato in figura



[immagine presa da 7]

X	sign($\nabla_x J(\theta, x, y)$)	x+$\varepsilon\ sign(\nabla_x J(\theta, x, y))$
"panda"	"nematode"	"gibbon"
57.7% confidence	8.2% confidence	99.3%

Per l'esempio in questione è stata usata una GoogleNet.
Dove ϑ sono i parametri della rete, x il pattern e y la label, J è la funzione di costo del modello. In questo caso la η della formula precedente assume la seguente forma:

$$\eta = sign(\nabla_x[J(\vartheta,x,y)]) \ (3)$$

Si può notare come progettando un semplice algoritmo matematico, che sfrutta il gradiente della rete stessa, è possibile creare Adversarial examples che ingannino anche ottime reti come la già citata GoogleNet, il che suggerisce una scarsa robustezza per quanto riguarda il modello. Proprio per evitare questo genere di problemi, sia dal punto di vista delle prestazioni della rete che dal punto di vista anche della sicurezza, si rende necessaria l'"immunizzazione" del modello attraverso l'adversarial training.
Infine, un aspetto intrigante degli adversarial examples è che, se generati per un modello, comportano spesso a delle misclassificazioni anche se usati su altri modelli, nonostante presentino architetture diverse.

Motivo di maggior sorpresa, sta nel fatto che quando questi modelli misclassificano un Adversarial Example, spesso lo catalogano
con la stessa classe. Questo fenomeno viene chiamato con il nome di “Adversarial Transferability”, e la sua spiegazione risiede nell’apprendimento stesso dei due modelli; infatti, molto frequentemente due reti tendono ad imparare e fare quindi affidamento sulle stesse features non-robuste, Il che comporta logicamente allo stesso tipo di misclassificazione in caso di permutazione della feature.

1.2 Adversarial Training

Inizialmente gli adversarial examples furono considerati come un problema non correlato alla generalizzazione dei modelli di machine learning; ma bensì, la visione più comune fu quella di considerarli come delle aberrazioni scaturite dall’elevata dimensionalità dei pattern usati dalle reti. Qualche anno dopo la loro scoperta Andrew Ilyas attraverso la sua pubblicazione [21], portò alla luce i possibili benefici derivanti dall’utilizzo della tecnica di Adversarial Training per l’addestramento di reti neurali. Contrariamente agli altri, Andew comprese che il problema dei modelli davanti agli Adversarial Examples risiedesse unicamente nel modo in cui le reti vengono allenate; ovvero cercando di massimizzare esclusivamente la precisione nella classificazione. A tale fine le reti tendono a sfruttare tutti i segnali possibili, anche quelli che sembrano incomprensibili agli esseri umani.
Particolare attenzione, infine, deve essere posta nella distinzione tra adversarial Training e data augumentation. Infatti, nonostante entrambe le tecniche si basano sulla modifica dei pattern in ingresso, gli adversarial pattern permutano l’input rendendolo innaturale ed esponendo la rete a delle criticità nella loro funzione di decisione come già visto.
Gli approcci di Adversarial Training sono molto diversificati, generalmente si tende a permutare i pattern in ingresso, ad esempio tramite gaussiana, cercando di non snaturare eccessivamente l’immagine (vedesi immagine capitolo 1.1), e attraverso una strategia di min max, far convergere il modello.

Per strategia di min max si intende, la massimizzazione del gradiente del pattern permutato rispetto alla loss, come segue:

$$\nabla_\theta(max_{i\in[m]} L(x_i', \theta)) \text{ (4)}$$

Dove per m si intendono le m permutazioni effettuate sul pattern x, mentre per x_i' si intende il pattern x con la permutazione i-esima.
In pratica si effettuano m permutazioni dello stesso pattern, per poi prendere quella che massimizza il gradiente della rete. La parte min consiste nella minimizzazione del gradiente della rete mediante metodi matematici, tra i quali SGD, in modo da migliorare i parametri della rete.
Questa tecnica è utilizzata pure nel MaxUp argomento trattato nel capitolo 3 di questa tesi, la quale per modificare i propri parametri utilizza SGD e la seguente:

$$\theta \leftarrow \theta - \eta E_{x\sim M}[\nabla_\theta(max_{i\in[m]} L(x_i', \theta))] \text{ (5)}$$

Viene utilizzata la discesa stocastica per aggiornare i parametri della rete.
Un'altra metodologia consiste nell'utilizzo di Adversarial objective function basate sul metodo del fast gradient sign method (vedesi (3)) :

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha)J(\theta, x + \varepsilon \text{sign}(\nabla_x J(\theta, x, y)). \text{(6)}$$

Dove α è il learning rate della rete, quindi alla funzione obiettivo tipica del modello, ovvero:

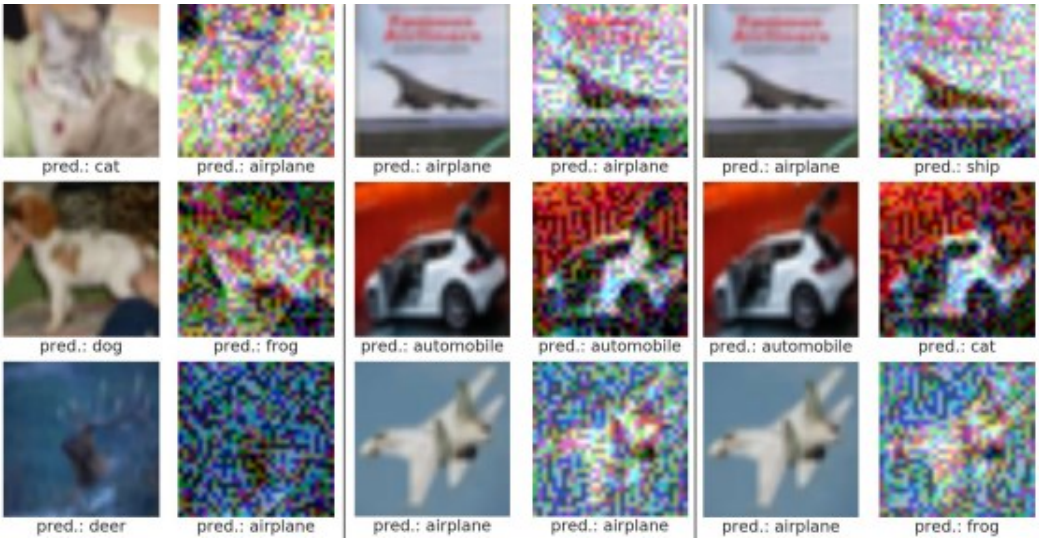
$$\alpha J(\theta, x, y) \text{ (7)}$$

Viene aggiunta pure una sua permutazione.
In questo caso, noi continuiamo a creare nuovi Adversarial Examples che si possano adattare ai miglioramenti della rete tra le varie iterazioni. I risultati sperimentali raggiunti mediante questa tecnica sono molto soddisfacenti, infatti sul dataset MNIST, con modello addestrato senza la tecnica dell'Adversarial Training, senza adversarial examples abbiamo un error rate dello 0.79%, mentre con pattern permutati la soglia si alza all'89.4%. Tramite invece addestramento con pattern permutati, il modello assume robustezza contro quest'ultimi, ottenendo un error rate del 17.9%.

Il metodo più utilizzato per l'Adversarial Training sfrutta la pgd ovvero il projected gradient descent. Il pgd crea ad ogni iterazione una nuova permutazione del patter precedente, basata sulla proiezione del pattern precedente e sul metodo del fast gradient sign:

$$x^{t+1} = \Pi_{x+S}(x^t + \alpha\, sgn(\nabla_x L(\theta, x, y))) \quad (8)$$

La tabella sotto riporta l'applicazione dei metodi sopracitati per mostrare i miglioramenti relativi alla classificazione delle immagini.
Nella prima colonna si fa vedere un modello addestrato con pattern clean, l'output con permutazione creata attraverso il fast sign gradient method comporta ad una misclassificazione totale. Nella seconda colonna invece il metodo viene addestrato tramite utilizzo di PGD e validation set con FSGM per settare al meglio gli iperparametri della rete, come si può notare, abbiamo un miglioramento per quanto riguarda la robustezza nella classificazione di pattern del test set permutati con FSGM. Infine, l'ultima colonna vede un addestramento con clean data, ma un validation set con FSGM, abbiamo un miglioramento rispetto la prima colonna, ma comunque non eguagliabile all'addestramento mediante adversarial training.



[presa da 5]

Infine, vorrei porre l'attenzione sul momento in cui va effettuata la permutazione; ovvero, se effettuarla all'interno della rete, oppure in un hidden layer, oppure prima dell'inserimento del pattern nella rete. Come mostrato da Szegedy [10] le perturbazioni mostrano una migliore regolarizzazione quando applicate ad un hidden layer.

Ovviamente, questo comporta ad alcune problematiche relative alle funzioni di attivazione "unbounded", dove il valore di risposta è
tendenzialmente molto alto, per questo è spesso preferibile effettuare la perturbazione all'esterno, proprio come succede nel caso del MaxUp.

2. Max Up

Uno dei modelli di machine learning che sfrutta la data augumentation e l'adversarial training è il MaxUp. L'obiettivo di questa tecnica è quella di migliorare le performance in generalizzazione, e prevenire l'overfitting specialmente nelle reti neurali profonde.

Il metodo si basa sul generare un set di perturbazioni randomiche o trasformazioni per ciascun data point, e poi minimizzare il rischio medio nel caso della peggior data augumentation.

Questa tecnica non solo permette di migliorare la generalizzazione, ma non presenta costi computazionali significativi in più per il modello.

Spesso questa tecnica è stata vista come un metodo valido per l'Adversarial Training, in realtà la progettazione del metodo stesso è più simile alla data augumentation. Ovviamente non possono essere ignorate le caratteristiche relative alla creazione degli adversarial examples, maggiormente sottolineate quando si utilizza come permutazione una gaussiana. Più in generale gli stessi creatori definiscono il MaxUp come "A lightweight Adversarial Training with Data Augmentation method", quindi sottolineando come a differenza di molti altri metodi di Adversarial Training, tra cui il già citato sign fast gradient, presenta una soluzione computazionalmente più leggera ed inoltre è molto più semplice da realizzare a differenza dei metodi basati su PGD. Inoltre, non può essere considerato solo come un metodo di Data Augumentation, in quanto; come più avanti verrà spiegato più nel dettaglio, esso minimizza la loss nel caso della peggiore permutazione del pattern x, invece che la media.

2.1 Main Idea

Dato un data set D=$\{x_i\}$ con $x_i \in R^d$, il MaxUp genera un set di permutazioni, per ogni data point generiamo una serie di permutazioni, minimizzando la massimizzando la loss tra tutte le permutazioni di ciascun data point.

Quindi dato un data point x_i, effettuiamo su di esso una serie di m permutazioni, il ruolo di queste è quello di migliorare la generalizzazione dei dati. Una volta generate il set di perturbazioni di x_i prendiamo tra tutti gli m, la permutazione con la loss più alta:

$$max_{i\in[m]}\mathrm{L}(x_i', \theta) \quad (9)$$

Dove x_i' è il pattern xi permutato.

Una volta aver preso il pattern con la loss più alta cerchiamo di minimizzare:

$$min_{\theta} E_{x \sim D}[max_{i \in [m]} L(x_i', \theta)] \text{ (10)}$$

I metodi per minimizzare questa loss sono molto semplici, può essere utilizzato uno stochastic gradient descent (SGD), in questo caso il miglioramento dei parametri della rete avviene nel seguente modo:

$$\nabla_{\theta}(max_{i \in [m]} L(x_i', \theta)) \text{ (11)}$$

Particolare attenzione deve essere posta alle permutazioni da effettuare a x_i, infatti devono essere i.i.d generate da una distribuzione di probabilità P(|x) . Per rendere sensate queste perturbazioni tipicamente si usano delle piccole perturbazioni attorno al pattern x_i, in modo da non cambiare del tutto l'immagine data, contribuendo a peggiorare le prestazioni della rete.
Ovviamente è sempre possibile utilizzare P(|x) costruite sulla trasformazione del pattern, quali random crops o MixUp.

2.2 Data Augumentation nel MaxUp

Come affermato nel paragrafo precedente, il metodo MaxUp si presta a numerose implementazioni, in base al tipo di data augumentation che si vuole applicare.
Le nuove tecniche di data augumentation che si basano sulle trasformazioni dei pattern, tra cui CutOut, Cutmix e Mixup, possono essere facilmente implementate nel MaxUp. Le tecniche citate si basano sul taglio dell'immagine, ma operano in maniera molto diversa

Image	ResNet50	CutMix	MixUp	CutOut
				
Label	Dog 1.0	Dog 0.6 Cat 0.4	Dog 0.5 Cat 0.5	Dog 1.0
ImageNet Cls (%)	76.3	78.6	77.4	77.1
ImageNet Loc (%)	46.3	47.3	45.8	46.7
Pascal VOC Det (mAP)	75.6	73.9	91	75.1

[8 Table 1]

Come si può notare infatti il CutOut si basa sulla rimozione di una parte di immagine, mentre la tecnica del CutMix va oltre, aggiungendo alla parte rimossa un pezzo di un'altra immagine appartenente al training set, il che comporta naturalmente ad un cambiamento anche della label corrispondente al dato, che ora diventa una composizione probabilistica tra la label di partenza e quella dell'immagine aggiunta nel riquadro tagliato.
Invece per quanto riguarda la tecnica del MixUp costruisce pattern "virtuali" a partire dalla sovrapposizione di immagini del TS stesso, in breve:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \quad (12.1)$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (12.2)$$

Dove λ è un numero randomico tra [0,1].
La differenza principale tra l'utilizzo di queste tecniche in forma pura e più in generale delle tecniche di data augumentation, e la loro implementazione nel MaxUp risiede nella funzione di loss che viene utilizzata, infatti in generale i metodi di data augumentationtendono a minimizzare la media tra tutte le permutazioni:

$$min_\theta E_{x\sim D}[\tfrac{1}{m}\Sigma_{i=1\ldots m} L(x_i', \theta)] \quad (13)$$

Quindi effettuate m permutazioni del pattern x_i, minimizziamo la media della loss function tra tutte. In opposizione, il MaxUp, come citato in precedenza minimizza il caso peggiore tra tutte le m permutazioni (vedi 11)(vedi pure 10).
Inoltre, i metodi di data augumentationsi comportano in maniera molto diversa per quanto riguarda i meccanismi di regolarizzazione, infatti, a differenza del MaxUp che introduce un fattore di

“regolarizzazione della norma del gradiente”, non è possibile avere questo beneficio.
Le ragioni relative a questo sono dovute al fatto che dopo l’espansione di Taylor del primo ordine viene cancellato il gradiente, a causa della presenza della media. Partendo dalla 4, la sua espansione del primo ordine sarebbe:

$$L_{P,m}(\mathrm{x}, \theta) = \mathrm{L}(\mathrm{x}, \theta) + \mathrm{O}(\sigma^2) \quad (14)$$

2.3 SGD vs PGD

Per quanto riguarda la parte relativa all’adversarial training come già accennato, viene preferita invece che la classica PGD, che contraddistingue i modelli che implementano Adversarial Training una SGD, non solo perché risulta più facile da implementare, ma soprattutto perché il projected gradient descent tipicamente risolve il problema di apprendimento in maniera molto aggressiva, e sacrificando spesso l’accurancy.
Per capirne il motivo, è necessario analizzarne i funzionamenti di ciascuno.
La variante stocastica del gradient descent si basa sull’utilizzo di mini-batch.
Per ogni epoca si effettua uno “shuffle” del training set in modo da evitare correlazioni tra i vari dati del TS, e poi vengono suddivisi formando dei minibatch di ugual dimensione, per
ogni iterazione vengono scanditi gli elementi del minibatch, si effettua il forward della rete e si migliorano i parametri. Il vantaggio è che ogni iterazione corrisponde ad un update dei parametri della rete, quindi, il metodo è molto più leggero rispetto al GD.
Il Projected Gradient Descend come già esplicato nel capitolo 1 risolve il problema della massimizzazione della funzione di loss da parte del pattern permutato, effettuando una proiezione e cercando il miglior massimo locale. Sotto il punto di vista della creazione di adversarial pattern sempre più precisi, e che riescano a confondere meglio i classificatori
risulta molto efficiente, ma la ricerca di questo massimo locale, porta ad avere un’immagine
dal punto di vista della rete molto difficile da classificare.

Questo comporta naturalmente ad una perdita di accurancy, come mostrano anche vari esperimenti, di cui riporto i risultati.

Method	Steps	Source	Accuracy
Natural	-	-	87.3%
FGSM	-	A	56.1%
PGD	7	A	50.0%
PGD	20	A	**45.8%**
CW	30	A	46.8%
FGSM	-	A′	67.0%
PGD	7	A′	**64.2%**
CW	30	A′	78.7%
FGSM	-	A_{nat}	85.6%
PGD	7	A_{nat}	86.0%

[4 Table 2]

Il training set usato è CIFAR10, e le performance di alcuni modelli di adversarial training, sono inserite nella tabella sopra.

Il MaxUp a differenza di Pgd risolve il problema della massimizzazione, effettuando m permutazioni attraverso una funzione di probabilità P(|x), come spiegato nel paragrafo 3.1 e prendendone il caso con loss più alta. In questo modo si ottiene un metodo computazionalmente più veloce e con una buona, ma non ottima resistenza agli adversarial examples.

2.4 Data augumentation MaxUp

Nel paragrafo 3.2 si è osservato come il MaxUp riuscisse a combinarsi molto bene con nuove tecniche di data augumentation. A livello implementativo, la soluzione proposta si basa proprio su una di queste tecniche, ovvero il cutmix. L'idea rimane sempre quella, ma in questo caso, si utilizza come P(|x) le trasformazioni apportate dal cutmix.

A livello operativo, dato un pattern x si applicano m permutazioni, ciascuna delle quali prevede il taglio dell'immagine x e la sostituzione della parte rimossa con un'altra appartenente allo stesso training set, e si modifica la label in modo proporzionale alla
trasformazione effettuata ed infine si seleziona il peggior caso di trasformazione.

L'implementazione mostrata ottiene ottime performance

Method	Top-1 error	Top-5 error
Vanilla [11]	76.3	-
Dropout [21]	76.8	93.4
DropPath [14]	77.1	93.5
Manifold Mixup [26]	77.5	93.8
AutoAugment [5]	77.6	93.8
Mixup [33]	77.9	93.9
DropBlock [9]	78.3	94.1
CutMix [31]	78.6	94.0
MaxUp+CutMix	**78.9**	**94.2**

[14 table 1]

Risultati relativi al contest ImageNet, con ResNet50

Model	Model Size	FLOPs	+CutMix (%)	+*MaxUp*+CutMix (%)
ResNet-101	44.55M	7.85G	79.83	**80.26**
ProxylessNet-CPU	7.12M	481M	75.32	**75.65**
ProxylessNet-GPU	4.36M	470M	75.08	**75.42**
ProxylessNet-Mobile $\times 1.4$	6.86M	603M	76.71	**77.17**
EfficientNet-B7	66.35M	38.20G	85.22*	**85.45***
Fix-EfficientNet-B8	87.42M	101.79G	85.57*	**85.80***

[14 Table 2]

MaxUp applicato a vari modelli nel set di ImageNet contest 2012

m	ResNet-110	WideResNet-28-10
1	73.64 ± 0.15	81.59 ± 0.27
4	75.26 ± 0.21	81.82 ± 0.22
10	75.19 ± 0.13	$\mathbf{82.48 \pm 0.23}$
20	74.37 ± 0.18	82.43 ± 0.24

[14 Table 5]

Test effettuati su CIFAR100, con due tipi di rete, variando il numero di permutazioni effettuate per ogni pattern.

Sono stati effettuati test pure per quanto riguarda dei miglioramenti per l'adversarial training, utilizzando una Gaussiana.

Model	Dataset	Augmentation	Standard		+*MaxUp*		+*MaxUp* (Low R)	
			Acc(%)	Time	Acc(%)	Time	Acc(%)	Time
ResNet-110	CIFAR-10	Cutout	94.8±0.1	36s	**95.4±0.1**	58s	**95.4±0.1**	47s
ResNet-110	CIFAR-100	Cutout	73.6±0.2	36s	**75.3±0.2**	58s	75.0±0.1	48s
WideResNet-28-10	CIFAR-10	RandAugment	97.1±0.1	72s	**97.5±0.1**	106s	**97.5±0.1**	89s
WideResNet-28-10	CIFAR-100	RandAugment	83.1±0.1	72s	**83.8±0.1**	106s	**83.7±0.1**	89s
ResNet-50	ImageNet	CutMix	78.6±0.0	3.6h	**78.9±0.0**	5.2h	**78.9±0.0**	4.4h

[14 Table 7]

Come mostrato nell'ultima tabella anche includendo immagini con una risoluzione bassa si possono raggiungere le stesse prestazioni rispetto al maxup con le immagini ad alta risoluzione, soprattutto quando si usano metodi di data augumentation"aggressivi", che trasformano molto l'immagine. Questi risultati mostrano come possa essere possibile velocizzare il metodo ed ottenere pressappoco le stesse prestazioni.

Infine, gli ultimi test sono stati effettuati sul 3D Point Cloud Classification. In pratica ciascun pattern viene rappresentato su un piano 3D. I test presentano iperparametri diversi, come il numero di "neighbours" (N) e il numero di "particles" (Part), e testati con DataAugumentation differenti.

#Part	#N	Type	Standard (%)	*MaxUp* (%)
1024	20	Gaussian	86.8±0.6	**88.9±0.4**
2048	20	Gaussian	88.4±0.5	**90.4±0.5**
2048	40	Gaussian	92.0±0.5	**92.3±0.4**
1024	20	Jitter	88.1±0.6	**89.2±0.5**
2048	20	Jitter	89.1±0.5	**90.4±0.4**
2048	40	Jitter	92.2±0.5	**92.4±0.5**

[14 table 8]

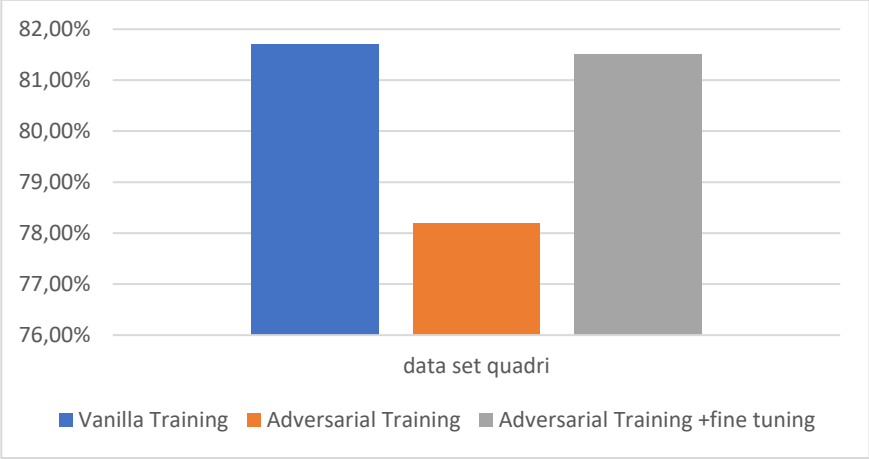
In conclusione, davanti a tutti i test riportati, possiamo dire che empiricamente e praticamente il maxup porta a dei miglioramenti nella classificazione tramite utilizzo di augumented data, offrendo anche una certa “difesa” da attacchi di adversarial Examples.

3. Metodo creato

Sulla base delle considerazioni fatte nei capitoli precedenti risulta importante che le reti neurali siano in grado di riconoscere il più possibile gli unseen data, minimizzando il più possibile le misclassificazioni. Allo stesso tempo si cerca pure di evitare il problema relativo all'overfitting, quando il training set comincia ad avere delle dimensioni consistenti. Ovviamente, oltre a queste tematiche, altro topic fondamentale è la robustezza dei modelli. Infatti, come spiegato nel capitolo 1 a più riprese, l'esistenza degli adversarial examples lede le prestazioni del modello allenato con estrema facilità. Questo come analizzato è un indicatore di scarsa astrazione del modello stesso, e quindi costituisce uno dei problemi principali per quanto riguarda l'apprendimento delle reti neurali. La soluzione proposta al problema è stata l'adversarial training, il quale permette di avere una robustezza maggiore, ma a discapito di un'accurancy tipicamente più bassa.
In questo contesto abbiamo citato il MaxUp, che rappresenta un buon connubio tra tutte le tecniche analizzate fino ad ora, computazionalmente molto veloce, presenta una discreta resistenza agli adversarial examples, e facilmente adattabile ai metodi più moderni di data augmentation.
Partendo da questa base, ho deciso di apportare dei miglioramenti a questo metodo, per renderlo molto più robusto agli attacchi.

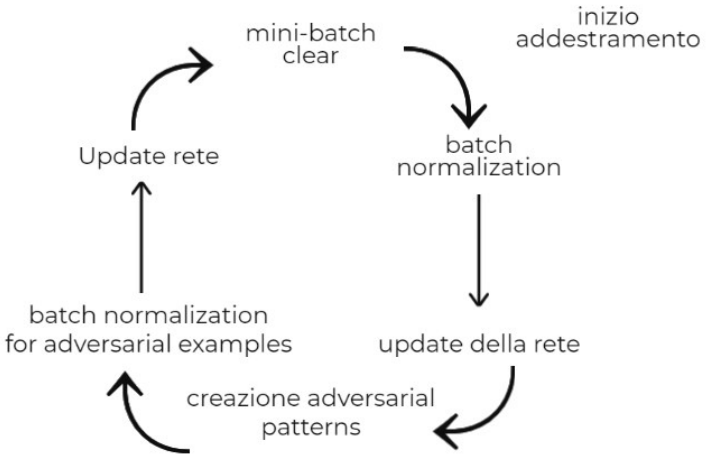
3.1 Descrizione modello

Il problema relativo all'utilizzo della tecnica del MaxUp sta proprio nel fatto che presenta una non eccezionale robustezza agli adversarial examples rispetto ad un modello basato sulla creazione dei medesimi, in base ai miglioramenti della rete, sfruttando le proprietà del gradiente.
Partendo dall'idea del MaxUp, ho cercato di sviluppare un metodo che si incentrasse maggiormente sulla robustezza della rete contro gli adversarial examples, cercando di intaccare il meno possibile l'accurancy in fase di addestramento.
Sono partito proprio con il pensare ad un metodo che possa non intaccare l'accurancy, ma allo stesso tempo che possa offrire una buona robustezza agli input permutati. Per fare ciò ho pensato di creare dei minibatch che contenessero sia pattern permutati che no. Questa tecnica permette di evitare una degradazione delle performance del modello, in particolare quando si utilizzano i clean data nel test set. Infatti, come si mostra in (28) quello che succede quando si allena una rete "adversarialmente", è che si riscontrano delle difficoltà nel riconoscimento dei pattern originali, come mostrato nell'esperimento sottostante



[Table 1]

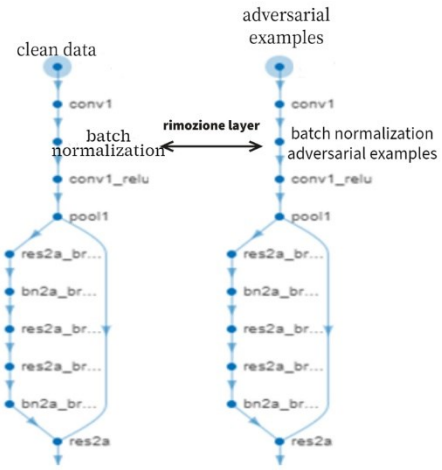
Il problema dell'uso contemporaneo di clean e adversarial pattern consiste nella diversa distribuzione che presentano, e quindi la rete presenta una difficoltà ulteriore nell'apprendimento, che comporta anche a problemi di generalizzazione. Per ovviare a questo problema, in fase di addestramento ho usato due layer di batch normalization differenti in base al minibatch utilizzato. Il layer di batch normalization normalizza i pattern di un dato minibatch sulla base di tutte le osservazioni ricevute, utilizzandone due uno per i clean ed uno per i permutati, elimino i problemi legati alle diverse distribuzioni che ciascun pattern presenta.



[Fig 1]

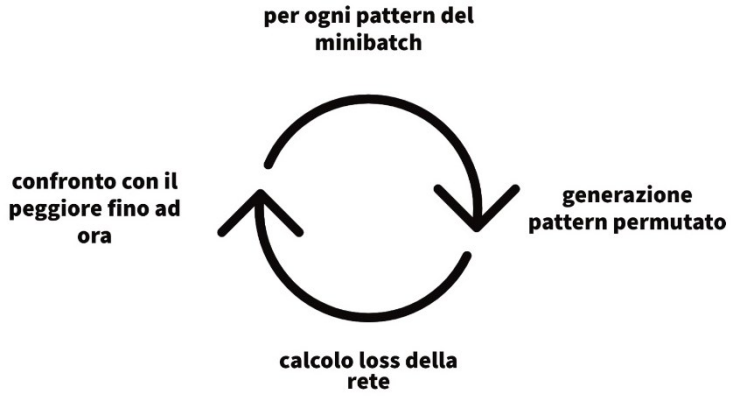
Lo schema sopra mostra come la rete viene allenata

Quando si passa dall'utilizzo di un batch ad un altro viene sostituito il layer di normalizzazione con un altro usato solo per quella tipologia di mini-batch, come è possibile vedere nello schema sotto



[Fig 2]

Per quanto concerne la creazione di adversarial examples, effettuo un attacco della rete permutando l'input ogni volta. In questa fase, ho fatto in modo che il layer di normalizzazione per adversarial examples, non funzionasse. Data la tecnica del MaxUp ho effettuato per ciascun data point una permutazione gaussiana, per m volte, e tra queste ho preso solamente la peggiore tra tutte, ovvero quella con la loss più alta.

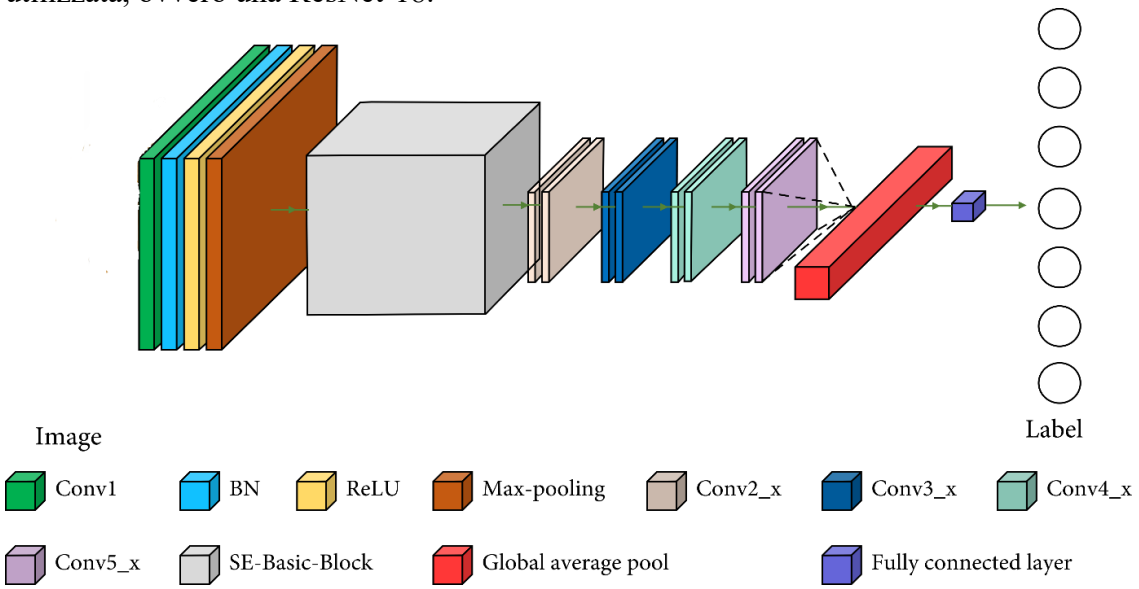


[Fig 3]

Il numero di permutazioni effettuate per ogni osservazione è stato deciso a priori, così come il tipo di permutazione da effettuare. In generale riprendendo il tratto relativo alla permutazione dei pattern del MaxUp, ho sempre utilizzato permutazioni che derivano da una distribuzione P(|x) in modo da generare pattern x' i.i.d.

4. TEST & RESULTS

Il modello spiegato nel capitolo 3 è stato utilizzato per effettuare vari test per ciascun data set utilizzato.
Cosa che accomuna ciascun addestramento effettuato, è sicuramente il tipo di rete di partenza utilizzata, ovvero una ResNet-18.



[Fig 4] La figura mostra la struttura della rete.

La rete in questione è una rete pre-addestrata su dataset imagenet; quindi, lo sforzo necessario per avere delle buone prestazioni su nuovi dataset è minore rispetto ad una rete partendo da scratch. Al fine di poter riutilizzare la rete per i miei test è stata applicata la tecnica di transfer learning nota con il nome di fine-tuning. Questa tecnica consiste nel cambiare gli ultimi tre layer della rete pre-addestrata, e rimpiazzarli con quelli specifici per il dataset in questione.
Ho separato il dataset in due parti, una utilizzata come training set ed una come test set.
Per utilizzare il modello spiegato nel capitolo 3, per ogni iterazione permuto le immagini del mini-batch attraverso una permutazione Gaussiana, con un numero di permutazioni totali pari a 4, in modo da non appesantire troppo il metodo:

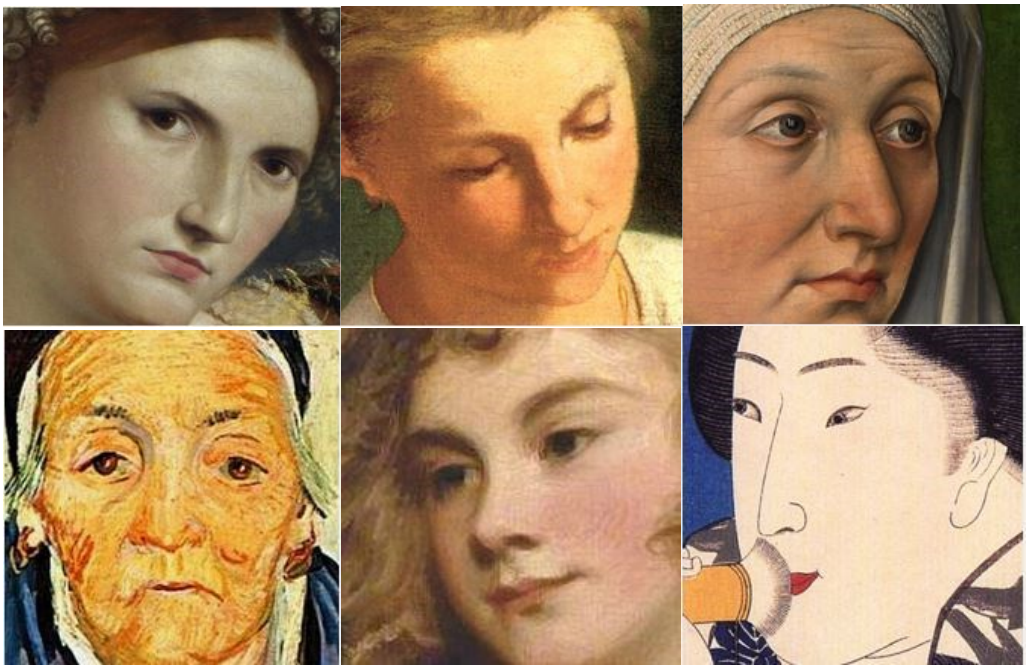
[fig 5] Immagine generata con il metodo

Infine, in fase di test ho utilizzato la rete addestrata con il batch normalization clean (bnc), per il test set non permutato, mentre per il test set permutato ho sostituito il layer di batch normalization (bna) con quello specifico per gli adversarial examples, in pratico ho riproposto lo stesso metodo usato per l'addestramento per la fase di test.

Ovviamente nella vita reale non è possibile avere un test set senza alcun tipo di permutazioni, per questo ho creato dei test set misti, che comprendessero entrambi i pattern citati sopra. Per ottenere dei risultati ottimali ho fatto classificare i pattern due volte, la prima con il bnc e la seconda con bna, e dagli score dati dalla loro classificazione utilizzo la metrica di fusione della somma. Quindi la classe che permette di avere la somma tra le due classificazioni più alta è la classe definitiva a cui viene assegnato il pattern.

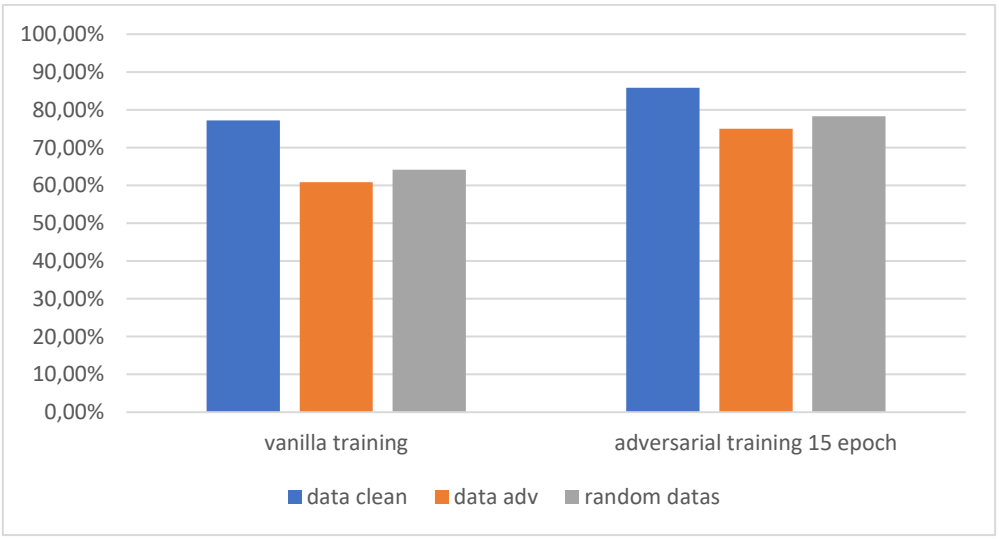
4.1 dataset quadri

Uno dei dataset che sicuramente ho utilizzato di più al fine di testare la rete, presenta quadri di vari artisti:



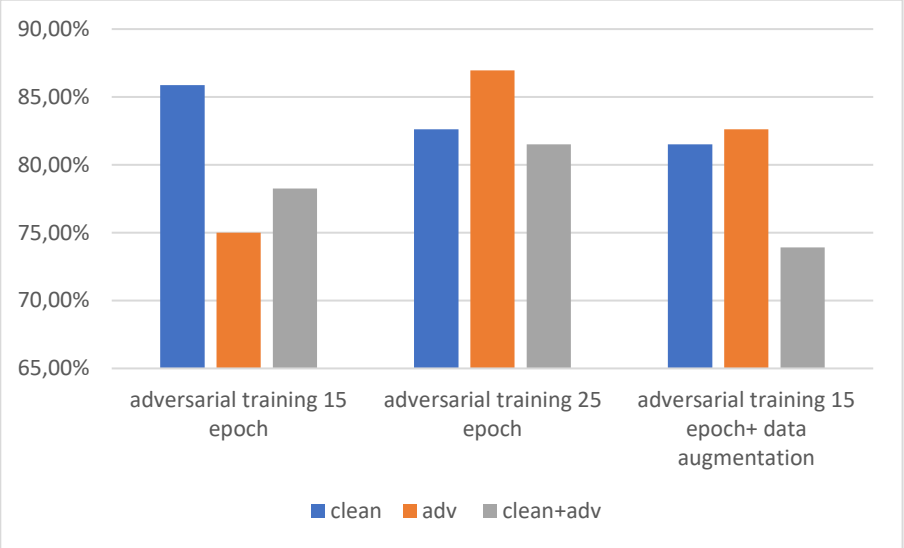
[fig 6.1-6.6]

I risultati relativi a questo dataset con un modello tradizionale



[Table 2]

I campi del confronto sono relativi alla capacità del modello di essere in grado di classificare correttamente i vari pattern. Il dataset usato di prova prevede:
-Nel caso “data clean” pattern appartenenti al test set
-Nel caso “data adv”, tutte le immagini del test set sono state permutate con una gaussiana
- Nel caso “random datas”, casualmente vengono scelte delle immagini e permutate, quindi alcune immagini sono non permutate.
Come si nota dal grafico sopra, il modello creato da me permette di avere un miglioramento considerevole in ambito di apprendimento della rete, ed una considerevole resistenza agli adversarial examples.
La rete è stata addestrata cambiando il numero di epoch e utilizzando varie metodologie di data augmentation

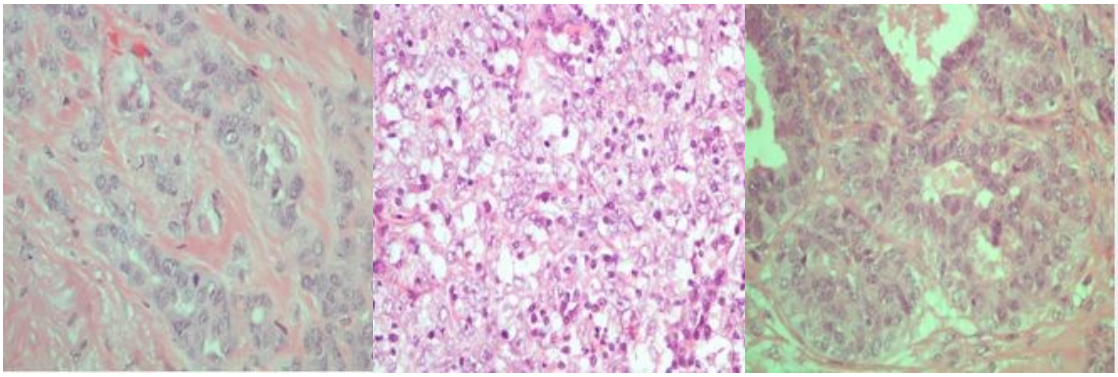


[Table 3]

Come è possibile notare all’aumento del numero di epoch si ha una diminuzione delle prestazioni della rete per quanto riguarda il test set clean, dovuto molto probabilmente ad overfitting della rete; invece, risultano migliorare considerevolmente le prestazioni della rete nella classificazione degli adversarial examples, in quanto come già riportato nel capitolo 1, presentano una maggiore difficoltà per la rete riuscire a generalizzarli.
Nonostante la perdita di accurancy nel caso di utilizzo di 25 epoch, comunque si ha un miglioramento netto nella classificazione di dataset rumorosi.
Infine, analisi importante deve essere posta al caso con 15 epoch e data augmentation, in questo caso, la data augmentation utilizza la tecnica del CutOut. Le prestazioni però tendono a deteriorare, rispetto ai due addestramenti precedente, questo perché il numero di epoch da utilizzare per questo esempio deve essere aumentato.

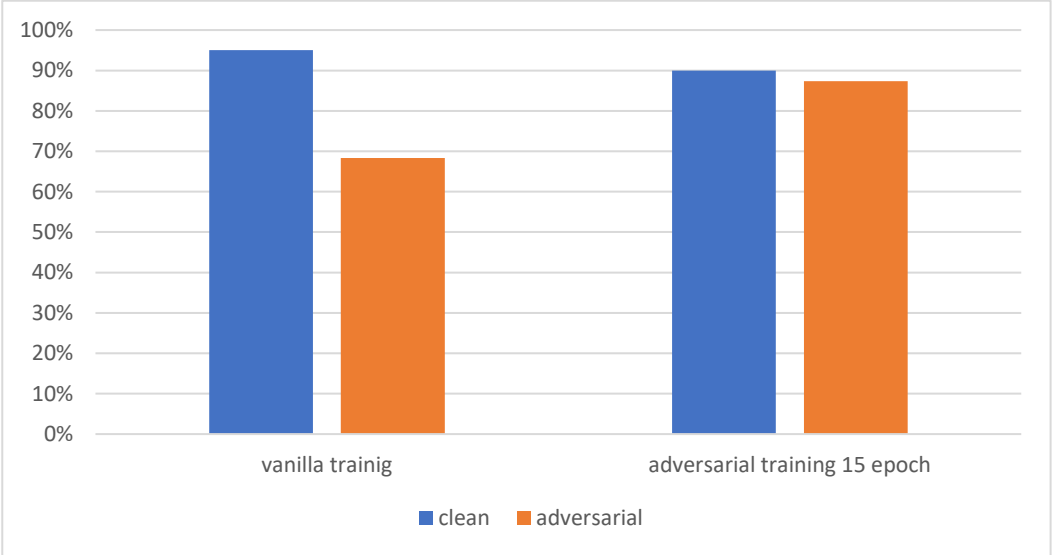
4.2 dataset istopatologie

Il dataset contiene varie immagini di istopatologie, ovvero immagini del tessuto con le loro relative alterazioni.



[Fig 7.1-7.3]

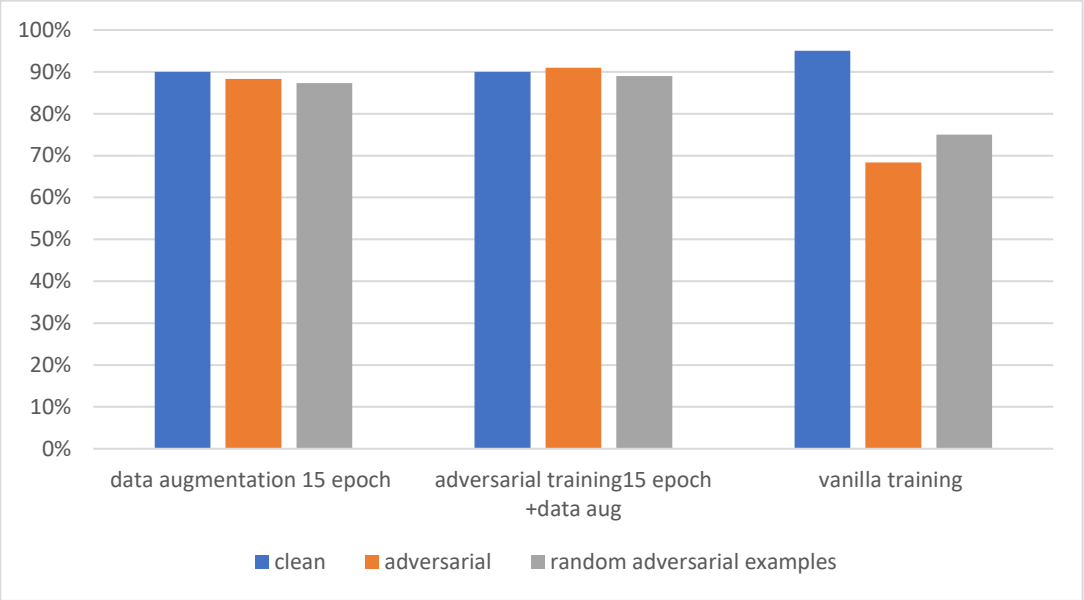
Il colore delle varie immagini è un fattore importantissimo per la distinzione dei vari tipi di tessuto; infatti, la rete è stata addestrata usando le immagini in formato rgb.



[Table 4]

Come dal grafico riportato, qui è possibile vedere in maniera molto evidente il problema relativo agli adversarial examples. Infatti, come si può notare le prestazioni della rete sono addirittura superiori rispetto al metodo creato da me, per quanto riguarda il riconoscimento delle immagini clean. Permutando le immagini, le prestazioni della rete crollano totalmente, venendo di gran lunga surclassate da quelle del metodo me creato.
Il metodo che ho utilizzato per questo primo esperimento presenta 15 epoch, senza nessuna data augmentation, e già le prestazioni si possono considerare come molto soddisfacenti.

Ho provato un ulteriore test con data augmentation, qui sotto riportato.



[Table 4]

Le prestazioni con la data augmentation in questo caso aumentano rispetto alla rete tradizionale, e al modello senza il suo utilizzo. Inoltre, grande vantaggio nell'utilizzo di questa metodologia, sta nel fatto che è possibile allenare la rete ancora per qualche epoch, allontanando l'overfitting La data augmentation che è stata applicata al data set contiene: rotazioni, traslazioni e zoom; casuali.

5. Conclusioni

Le reti neurali sono uno strumento molto potente, soprattutto nell'ambito dell'image recognition. Nonostante la loro efficacia, esse presentano una vulnerabilità, ovvero gli adversarial examples. Questi sono pattern che presentano delle permutazioni, rispetto al dataset che è stato utilizzato per addestrarle. Anche minime permutazioni dei pattern, come è stato dimostrato, possono comportare ad un errore di classificazione da parte della rete. L'adversarial training permette alla rete di acquisire una maggiore robustezza, dovuta specialmente ad un aumento della generalizzazione da parte del modello. Come è stato possibile vedere mediante vari esperimenti, i metodi tradizionali vengono in molti casi surclassati dal metodo da me proposto usando dataset anche leggermente rumorosi, nonostante comunque i metodi tradizionali siano molto più prestanti nel caso di dataset poco rumorosi. Le prestazioni mediante utilizzo di data augmentation, a parità di epoch tendono invece a diminuire leggermente, come riportato nelle tabelle 8 e 6, ma sicuramente presentano una soglia di overfitting più lontana, il che sicuramente aiuta la generalizzazione e pertanto le prestazioni. Risultati incoraggianti sono ottenuti e riportati nella tabella 8, dove l'utilizzo di data augmentation permette di avere prestazioni simili ed in certi casi superiori, rispetto all'addestramento senza.

Bibliografia

[1] Intriguing properties of neural networks. Christian Szegedy, Ian Goodfellow, Rob Fergus https://arxiv.org/abs/1312.6199

[2] Domain-Adversarial Training of Neural Networks. Mario Marchand, Hugo Larochelle https://arxiv.org/abs/1505.07818

[3] Adversarial Diversity and hard Positive Generation. Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. https://arxiv.org/abs/1605.01775

[4] Towards Deep Learning Models Resistant to Adversarial Attacks. Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras https://arxiv.org/abs/1706.06083

[5] INTRIGUING PROPERTIES OF ADVERSARIAL EXAMPLES. Ekin D. Cubuk, Barret Zoph, Samuel S. Schoenholz, Quoc V. Le. https://arxiv.org/abs/1711.02846

[6] Adversarial Image Generation and Training for Deep Neural Networks. Hai Shu, Ronghua Shi, Hongtu Zhu, Ziqi Chen. https://www.researchgate.net/publication/341997931_Adversarial_Image_Generation_and_Training_for_Deep_Convolutional_Neural_Networks

[7] The Vulnerability of the Neural Networks Against Adversarial Examples in Deep Learning Algorithms. Rui Zhao. https://arxiv.org/abs/2011.05976#:~:text=The%20Vulnerability%20of%20the%20Neural%20Networks%20Against%20Adversarial%20Examples%20in%20Deep%20Learning%20Algorithms,-Rui%20Zhao&text=With%20further%20development%20in%20the,gradually%20exposed%20certain%20security%20risks.

[8] CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. Sangdoo Yun Dongyoon Han Seong Joon Oh Sanghyuk Chunk. https://arxiv.org/abs/1905.04899

[9] Improved Regularization of Convolutional Neural Networks with Cutout. Terrance DeVries and Graham W. Taylor. https://arxiv.org/abs/1708.04552

[10] EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES. Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy. https://arxiv.org/abs/1412.6572

[11] Exploring the Space of Adversarial Images. Pedro Tabacof and Eduardo Valle https://arxiv.org/abs/1510.05328

[12] mixup: BEYOND EMPIRICAL RISK MINIMIZATION. Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz. https://arxiv.org/abs/1710.09412

[13] Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization. Uri Shaham, Yutaro Yamada. https://arxiv.org/abs/1511.05432

[14] MaxUp: Lightweight Adversarial Training with Data Augmentation Improves Neural Network Training. Chengyue Gong, Tongzheng Ren, Mao Ye, Qiang Liu. https://arxiv.org/pdf/2002.09024.pdf

[15] AutoAugment: Learning Augmentation Strategies from Data. Ekin D. Cubuk ∗ , Barret Zoph∗ , Dandelion Mane, Vijay Vasudevan, Quoc V. Le. https://www.researchgate.net/publication/338511824_AutoAugment_Learning_Augmentation_Strategies_From_Data

[16] Deep Residual Learning for Image Recognition. Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. https://arxiv.org/abs/1512.03385https://arxiv.org/abs/1512.03385

[17] Adversarially Robust Generalization Requires More Data. Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras. https://arxiv.org/abs/1804.11285

[18] Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky , Ilya Sutskever, Ruslan Salakhutdinov http://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf

[19] ENSEMBLE ADVERSARIAL TRAINING: ATTACKS AND DEFENSES. Florian Tramer, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Ian Goodfellow, Patrick McDaniel. https://arxiv.org/abs/1705.07204

[20] Adversarial Examples Improve Image Recognition. Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan Yuille, Quoc V. Le. https://arxiv.org/abs/1911.09665

[21] Adversarial Examples Are Not Bugs, They Are Features. Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, Aleksander Madry. https://arxiv.org/abs/1905.02175