Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Riconoscimento di varietà di vegetali in immagini digitali

Relatore: Prof. Raimondo Schettini

Co-relatore: Dott. Marco Buzzelli

Relazione della prova finale di:
Belotti Federico
Matricola 808708

Anno Accademico 2017-2018

A mia madre (esistono specifici motivi per ringraziare una madre?).
A mio padre (così come per un padre?).
A mia sorella Mara, per aver costantemente creduto in me.
A Francesca, per essere stata "il sapore della piogga fresca su un campo di grano arido".
Agli amici, quasi fratelli, di sempre, Thomas e Alessia, per sopportarmi e supportarmi da tempi immemorabili.
Al Dott. Marco Buzzelli, per il costante, mai banale e sempre arguto consiglio fornitomi.
Ai colleghi divenuti amici, che sono troppi per essere menzionati tutti, per aver reso quest'esperienza unica, e al contempo più stimolante e leggera.
A tutti coloro che per questioni di spazio e non d'importanza, poiché innumerevoli, non ho potuto citare, che m'hanno accompagnato, sostenuto, consigliato e criticato.

La teoria è quando si sa tutto ma non funziona niente. La pratica è quando tutto funziona ma non si sa il perché. In ogni caso si finisce sempre con il coniugare la teoria con la pratica: non funziona niente e non si sa il perché.

Albert Einstein

Sommario

Il lavoro svolto durante il periodo di stage è stato concentrato nel riconoscere varietà di vegetali mediante l'utilizzo di reti neurali convoluzionali (CNN).
E' stato utilizzato un dataset espressamente pensato per il cosiddetto *Ffine-grained Visual Categorization problem* (FGVC) in cui le immagini raccolte sono state etichettate gerarchicamente secondo le loro caratteristiche alimentari.
Gli esperimenti sono stati così strutturati: in primo luogo sono state addestrate e testate varie architetture di reti neurali convoluzionali nello stato-dell'arte per il suddetto problema; secondariamente si è cercato di sfruttare la suddivisione gerarchica delle immagini per migliorare ulteriormente le performance nel riconoscimento.
I risultati sperimentali dimostrano quantitativamente l'efficacia del metodo sviluppato.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Nella computer vision, con *Fine-grained Visual Categorization* (FGVC) ci si riferisce al problema di classificare oggetti in classi subordinate [26, 33]. Alcuni esempi includono il saper riconoscere la specie d'appartenenza di animali, quali cani o uccelli, il modello di un'automobile o di un aereo.
A differenza della classificazione generica [29], la FGVC deve saper gestire una più sottile differenza tra le diverse classi d'appartenenza, così come una più grande variabilità di oggetti appartenenti alla stessa classe: di fatto immagini rappresentanti lo stesso oggetto possono essere acquisite con differenti condizioni di luce, posizione e sfondo; viene dunque richiesta una più robusta e discriminativa rappresentazione delle immagini da classificare, in termini di features o caratteristiche.
A questo proposito negli ultimi anni le reti neurali convoluzionali, oltre ad essersi imposte come principale strumento d'impiego nella computer vision per il riconoscimento automatico di oggetti [21, 14, 32, 34, 41], sono state largamente utilizzate nell'ambito della FGVC [7, 39, 33, 22].
Ciononostante, i dataset a disposizione per FGVC contengono un numero ridotto o non sufficiente di immagini in ogni classe, ad esempio il dataset CUB-200-2011 [35] contiene non più di 30 immagini per classe, o comunque sono ristretti a specifici ambiti: automobilistico [20], aeronautico [23], naturalistico [35, 18, 19], abbigliamento [17], *etc.*
In particolare, pochi sono i dataset in ambito culinario, aspetto molto importante della quotidianità, e quelli esistenti sono incompleti, come [12] che contiene solo immagini di frutta, o contengono un numero esiguo sia di classi subordinate che di immagini di training [9], oppure non seguono una vera e propria suddivisione tassonomica [2].
A tal riguardo Hou et al. hanno collezionato, in un dataset chiamato VegFru, più di 160000 immagini rappresentanti frutta e verdura allo stato crudo, suddivise in 92 e 200 classi rispettivamente, etichettate con un sistema di annotazione gerarchico a due livelli [15]. Tale dataset, oltre che ad essere d'impiego nella FGVC, può essere utilizzato per mettere a punto sistemi intelligenti nell'ambito delle smart home, che interagiscano con le persone nella loro quotidianità, ad esempio suggerendo ricette, eseguendo un controllo qualità analizzando lo stato di conservazione, *etc* [4].

Il primo passo in questa direzione resta comunque il riconoscimento automatico del materiale culinario, che viene affrontato in questo lavoro mediante l'applicazione di reti neurali convoluzionali su VegFru, con l'intenzione aggiuntiva di sfruttare la struttura gerarchica di tale dataset per migliorare nel riconoscimento degli oggetti ad esso appartenenti.
In particolare, nel secondo capitolo viene data una breve ma esaustiva presentazione di VegFru, dai principi costruttivi alla suddivisione tassonomica delle classi subordinate; nel terzo capitolo, oltre ad essere presentate le architetture neurali utilizzate in questo lavoro, vengono forniti due differenti metodi atti a sfruttare la struttura gerarchica di VegFru; infine nel quarto capitolo verranno elencati i risultati sperimentali.

Capitolo 2

Dataset

2.1 Fine Grained Visual Categorization

Nella computer vision, con *Fine-grained Visual Categorization* (FGVC d'ora in avanti) ci si riferisce al problema di classificare oggetti in classi subordinate, ad esempio data l'immagine di un cane riconoscerne la razza canina d'appartenenza, ovvero "Pastore tedesco", "Lupo cecoslovacco" e così via.
A differenza della classificazione generica [29], in cui, data l'immagine raffigurante un cane, questa viene classificata come appartenente o meno alla classe "Cane", la FGVC deve saper gestire una più sottile differenza tra le diverse classi d'appartenenza, così come una più grande variabilità di oggetti appartenenti alla stessa classe [Figura 2.1]; viene dunque richiesta una rappresentazione delle immagini da classificare, in termini di features o caratteristiche, che sia più robusta e discriminativa.

(a) Cipolla cinese

(b) Porro

(c) Erba cipollina

(d) Cavolo cinese

(e) Cavolo cinese

(f) Cavolo cinese

Figura 2.1: **(a)-(c)**: immagini appartenenti a classi differenti molto simili tra loro. **(d)-(f)**: immagini appartenenti alla stessa classe molto diverse tra loro

2.2 VegFru

VegFru è un dataset creato appositamente per la FGVC contenente immagini rappresentanti frutta e verdura d'uso comune categorizzate secondo le loro caratteristiche alimentari: differenti parti commestibili di un frutto o di una verdura, come ad esempio foglie e radici, sono classificate in classi subordinate differenti. Un'ulteriore peculiarità degli oggetti presenti in ogni immagine è che si tratta di materiali alimentari crudi.
VegFru attualmente contiene più di 160000 immagini di frutta e verdura suddivise in 25 classi di alto livello (*super-classi* d'ora in avanti) e 292 classi subordinate o di basso livello (*sotto-classi* d'ora in avanti) con almeno 200 immagini per ogni *sotto-classe*.
In particolare, a parte l'annotazione a grana fine, ovvero quella che permette la categorizzazione delle immagini come appartenenti ad una delle 292 *sotto-classi*, alle immagini presenti in VegFru sono assegnate delle label gerarchiche seguendo una tassonomia prontata principalmente alla cucina domestica e alla gestione del cibo nel suo senso più generale, per questo motivo ogni immagine contiene almeno una parte commestibile di un frutto o di una verdura con lo stesso metodo di cottura [Figura 2.2].

Figura 2.2: Immagini d'esempio in VegFru. Prima riga: verdure. Seconda riga: frutta

VegFru può dunque essere applicato ai seguenti aspetti, anche se non circoscritto solo ed esclusivamente ad essi:

- *Fine-grained Visual Categorization (FGVC).*
- *Granularità ibrida per FGVC*: E' stato dimostrato che l'annotazione gerarchica è d'aiuto per migliorare il riconoscimento di immagini sul più arduo problema di determinare la classe subordinata, o *sotto-classe* in caso di VegFru, d'appartenenza [15]. Essendo ogni immagine annotata gerarchicamente, VegFru si presta naturalmente ad essere ambito di ricerca sullo sfruttamento di questa tipologia di annotazione atto a migliorare applicazioni nel campo del FGVC.

- *Applicazione pratica per la cucina domestica e per il food managment*: VegFru contiene una grande quantità di immagini raffiguranti frutta e verdura crude categorizzandole secondo le loro caratteristiche alimentari. E' strettamente correlato alla vita quotidiana di tutti, e dunque utile per promuovere applicazioni nell'ambito delle Smart Home [4].

2.2.1 Dettagli e principi costruttivi

Nello specifico, la gerarchia della verdura è stata costruita secondo l'*Agricultural Biological Taxonomy* [15], che suddivide le verdure in *verdura a radice, verdura a foglia, etc.* Per le verdure si hanno dunque 15 *super-classi* e 200 *sotto-classi*.
In maniera del tutto analoga per la gerarchia della frutta è stata adottata l'*Horticultural Taxonomy* [15], in cui viene oraganizzata in 10 *super-classi* e 92 *sotto-classi*.
L'attuale versione di VegFru, che può essere naturalmente diviso in due sottoinsiemi disgiunti, Veg200 e Fru92, contiene 91,117 immagini di verdure e 69,614 immagini di frutta. Il numero di immagini per ogni *sotto-classe* varia dalle 200 alle 2000.
I principi costruttivi seguiti per la costruzione di VegFru sono dunque i seguenti:

- Gli oggetti presenti nelle immagini di ogni *sotto-classe* hanno la stessa applicazione culinaria [Figura 2.1 **(d)-(f)**].
- Ogni immagine contiene almeno una parte commestibile di frutta o di verdura [Figura 2.2].
- Le immagini che contengono differenti parti commestibili di frutta o di verdura, come ad esempio foglie, fiori, radici, sono classificate in *sotto-classi* differenti.
- Se due immagini contengono la stessa parte commestibile di un dato frutto o verdura, ma vengono cucinati in modo differente, allora le due immagini sono classificate in due *sottoclassi* differenti [Figura 2.3].
- Gli oggetti in ogni immagine devono rappresentare meteriali alimentari crudi. Se il materiale crudo del cibo cucinato è indistinguibile, l'immagine viene viene esclusa dagli autori durante la fase di costruzione del dataset [Figura 2.4].

Figura 2.3: Immagini con metodi di cottura differenti

Di seguito è riportata in forma tabellare la struttura gerarchica di VegFru.

Super-classe	#Sotto	Super-classe	#Sotto
Aquatic vegetable	13	Alliaceous	10
Brassia oleracea	9	Beans	15
Bud seeding	4	Cabbage	5
Green-leafy vegetable	31	Eggplant	7
Perennial	13	Melon	14
Tuber vegetable	10	Mushroom	24
Wild vegetable	32	Mustard	2
Root vegetable	11		
Totale	**15 super-classi e 200 sottoclassi per Veg200**		

Tabella 2.1: *Super-classi* in Veg200. Per ogni *super-classe*, #Sotto rappresenta il numero di *sotto-classi* in essa contenute. I nomi di ogni *super-classe* sono stati riportati come in [15]

Super-classe	#Sotto	Super-classe	#Sotto
Berry fruit	22	Drupe	13
Citrus fruit	13	Litchies	2
Persimmons	6	Nut fruit	11
Pome	11	Other fruit	2
Collective fruit	5		
Cucurbites	6		
Totale	**10 super-classi e 92 sottoclassi per Fru92**		

Tabella 2.2: *Super-classi* in Fru92. Per ogni *super-classe*, #Sotto rappresenta il numero di *sotto-classi* in essa contenute. I nomi di ogni *super-classe* sono stati riportati come in [15]

Figura 2.4: Immagini escluse da VegFru

2.2.2 Suddivisione dataset

In VegFru ogni *sotto-classe* contiene almeno 200 immagini, che sono suddivise negli insiemi training, validation e test contenenti 29200, 14600 e 116931 immagini rispettivamente.
Un modo alternativo, suggerito dagli stessi ideatori di VegFru, per la suddivisione delle immagini nei tre insiemi sopra citati è la seguente: riordinare dapprima le immagini in ogni *sotto-classe* randomicamente, selezionare poi per ognuna di esse le prime 100 per l'insieme di training, le successive 50 per l'insieme di validation e le restanti per l'insieme di test.
In questa tesi ci siamo attenuti alla suddivisione fornita di default dagli autori.

Capitolo 3

Metodi proposti

Nel seguente capitolo verrà presentato l'approccio da noi seguito per il riconoscimento automatico di vegetali in immagini digitali. Nello specifico verrà dapprima fornita una breve descrizione delle architetture neurali di base utlizzate a tale scopo: VGG, ResNet e NasNet [32, 14, 41]. Verranno poi proposti due metodi per sfruttare l'annotazione gerarchica di VegFru.

3.1 Reti neurali di base

Le reti neurali convoluzionali (ConvNets o CNN) hanno ottenuto negli ultimi anni un grande successo in quanto ottimi strumenti per il riconoscimento di immagini e video su larga scala [21], che è stato reso possibile dalla presenza di dataset di grandi dimensioni, ad esempio ImageNet [6, 29], e dalle alte prestazioni fornite da sistemi di calcolo quali le GPU o dai sistemi di calcolo distribuito. In particolare, ha avuto un ruolo cruciale nell'evoluzione di tali architetture l'*ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) [29], utilizzato come primo banco di prova per tutti i sistemi di classificazione di immagini su larga scala dal 2010 a oggi.

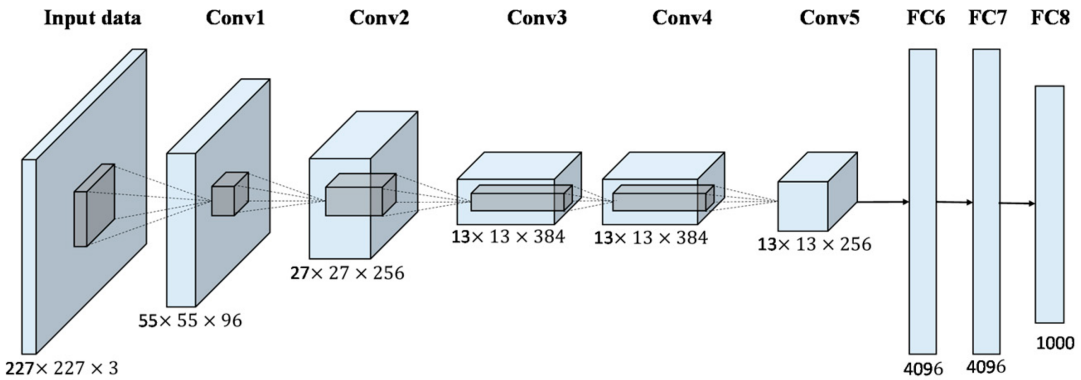


Figura 3.1: Architettura di una AlexNet

3.1.1 VGG

Essendo le reti neurali convoluzionali divenute di vitale importanza nell'ambito della computer vision, molti sono stati gli sforzi improntati a migliorare l'originale architettura sviluppata da [21] [Figura 3.1] per ottenere una più alta precisione di riconoscimento: dall'utilizzare un filtro ricettivo di dimensioni ridotte e uno stride più piccolo nel primo layer di convoluzione [38, 30], all'addestrare e testare la rete su tutta l'immagine e su scale differenti [30, 16].
Simonyan et al., ideatori della VGG [32], hanno invece indirizzato i loro sforzi su un altro aspetto importante dell'architettura di una rete neurale: la profondità.
A questo proposito hanno incrementato notevolmente la profondità della rete aggiungendo altri layer di convoluzione, aggiunta resa possibile dall'utilizzo di filtri di convoluzione molto piccoli (3 x 3, la più piccola dimensione in grado di catturare la nozione di destra/-sinistra, sopra/sotto e centro) in tutti i layer [Figura 3.2].
Gli autori hanno ideato sei diverse configurazioni di architetture neurali, che si differenziano una dall'altra per il numero di layer complessivi (sono omessi dal conteggio i layer di max-pooling e softmax): dalla configurazione A, con 11 layer (8 layer di convoluzione e 3 layer di fully-connected), alla configurazione E, con 19 layer (16 layer di convoluzione e 3 layer di fully-connected), passando per la configurazione D, utilizzata in questo lavoro, con 16 layer (VGG16 d'ora in avanti, con 13 layer di convoluzione e 3 layer di fully-connected) [Tabella 3.1].

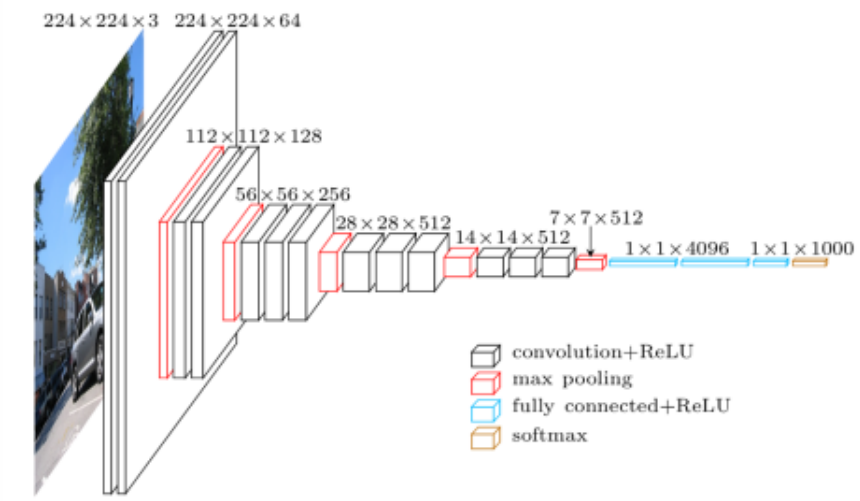


Figura 3.2: Architettura di una VGG16

Piazzatisi secondi all'ILSVRC-2014, hanno dimostrato che a parità di architettura, aumentandone la profondità si ottengono risultati migliori in termini di precisione di riconoscimento.
Pur non avevndo sostanzialmente aumentato il numero di parametri rispetto a [30], un'architettura di questo genere risulta essere computazionalmente molto lenta da addestrare, con una grande occupazione della memoria ($\approx$ 8GB con la dimensione del mini-batch impostata a 4).

Configurazioni VGG					
A	A-LRN	B	C	D	E
11 livelli	11 livelli	13 livelli	16 livelli	16 livelli	19 livelli
input (immagine RGB 224x224)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
fully-connected-4096					
fully-connected-4096					
fully-connected-1000					
softmax					

Tabella 3.1: Configurazioni di VGG. I parametri dei layer di convoluzione sono definiti come "conv[dimensione filtro recettivo]-[numero di canali]"

3.1.2 ResNet

Spinti dall'importanza della profondità in un rete neurale convoluzionale, i ricercatori si sono successivamente posti la seguente domanda: l'apprendimento risulta essere realmente tanto migliore e semplice al crescere del numero di livelli aggiunti?
A parte il noto problema dell'esplosione/azzeramento dei gradienti, che rende difficile rispondere a tale domanda in quanto la convergenza della rete è compromessa in partenza, risolto mediante l'utilizzo di layer di normalizzazione inziale ed intermedi, valutazioni sperimentali sembrano suggerire una risposta negativa [14].
E' stato dimostrato che quando una rete profonda riesce a convergere, il solo aumento della profondità porta la precisione di riconoscimento a saturarsi e a decrescere rapidamente [13, 14].
In [14], He et al. hanno pensato di risolvere il problema della degradazione della precisione introducendo un framework chiamato *deep residual learning* [Figura 3.3].

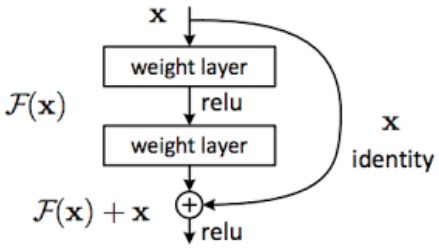


Figura 3.3: Blocco di residual learning

L'idea è cercare di agevolare il processo di backpropagation [28] fornendo ai gradienti un percorso diretto (*shortcut connections*) ai livelli precedenti della rete, diminuendo drasticamente i tempi di addestramento ed evitando al contempo il problema dell'azzeramento dei gradienti.
Di conseguenza, in aggiunta alla massiccia riduzione del numero di parametri ($\approx$ 30 milioni per la ResNet50), ciò ha permesso di aumentare la profondità della rete, ispirata dalla filosofia della VGG [Figura 3.4], ottenendo un netto miglioramento della precisione di riconoscimento nell'ILSVRC-2015.
E' stato però dimostrato in [8] che le cosiddette *residual network* sono molto portate all'overfitting, che può essere comunque ridotto applicando del data augmentation.
Così come gli autori di VGG hanno fornito diverse configurazioni della stessa architettura, differenti una dall'altra per il numero complessivo di layer, anche gli autori di ResNet ne hanno rese a disposizione cinque distinte: a 18, 34, 50, 101 e 152 layer.
Ciò che contraddistingue le diverse configurazioni, oltre ovviamente al numero di livelli complessivi, sono gli elementi costitutivi:

- $\begin{bmatrix} \text{conv3-}d \\ \text{conv3-}d \end{bmatrix}$ per le ResNet18 e 34, dove d è il numero di canali.
- $\begin{bmatrix} \text{conv1-}d_1 \\ \text{conv3-}d_1 \\ \text{conv1-}d_2 \end{bmatrix}$ per le ResNet50, 101 e 152, dove d_1 e d_2 rappresentano il numero di canali.

Concatenenando più elementi costitutivi si ottengono le diverse configurazioni, come specificato nella [Tabella 3.2].

Configurazioni ResNet				
18 livelli	34 livelli	50 livelli	101 livelli	152 livelli
input (immagine RGB 224x224)				
conv7-64				
maxpool				
[conv3-64 conv3-64] x 2	[conv3-64 conv3-64] x 3	[conv1-64 conv3-64 conv1-256] x 3	[conv1-64 conv3-64 conv1-256] x 3	[conv1-64 conv3-64 conv1-256] x 3
[conv3-128 conv3-128] x 2	[conv3-128 conv3-128] x 4	[conv1-128 conv3-128 conv1-512] x 4	[conv1-128 conv3-128 conv1-512] x 4	[conv1-128 conv3-128 conv1-512] x 8
[conv3-256 conv3-256] x 2	[conv3-256 conv3-256] x 6	[conv1-256 conv3-256 conv1-1024] x 6	[conv1-256 conv3-256 conv1-1024] x 23	[conv1-256 conv3-256 conv1-1024] x 36
[conv3-512 conv3-512] x 2	[conv3-512 conv3-512] x 3	[conv1-512 conv3-512 conv1-2048] x 3	[conv1-512 conv3-512 conv1-2048] x 3	[conv1-512 conv3-512 conv1-2048] x 3
averagepool				
fully-connected-1000				
softmax				

Tabella 3.2: Configurazioni di ResNet. Gli elementi costitutivi sono racchiusi tra parentesi quadre, con a fianco il numero di blocchi concatenati. I parametri dei layer di convoluzione sono definiti come "conv[dimensione filtro recettivo]-[numero di canali]"

Le versioni utilizzate in questo lavoro sono ResNet34 e ResNet50.

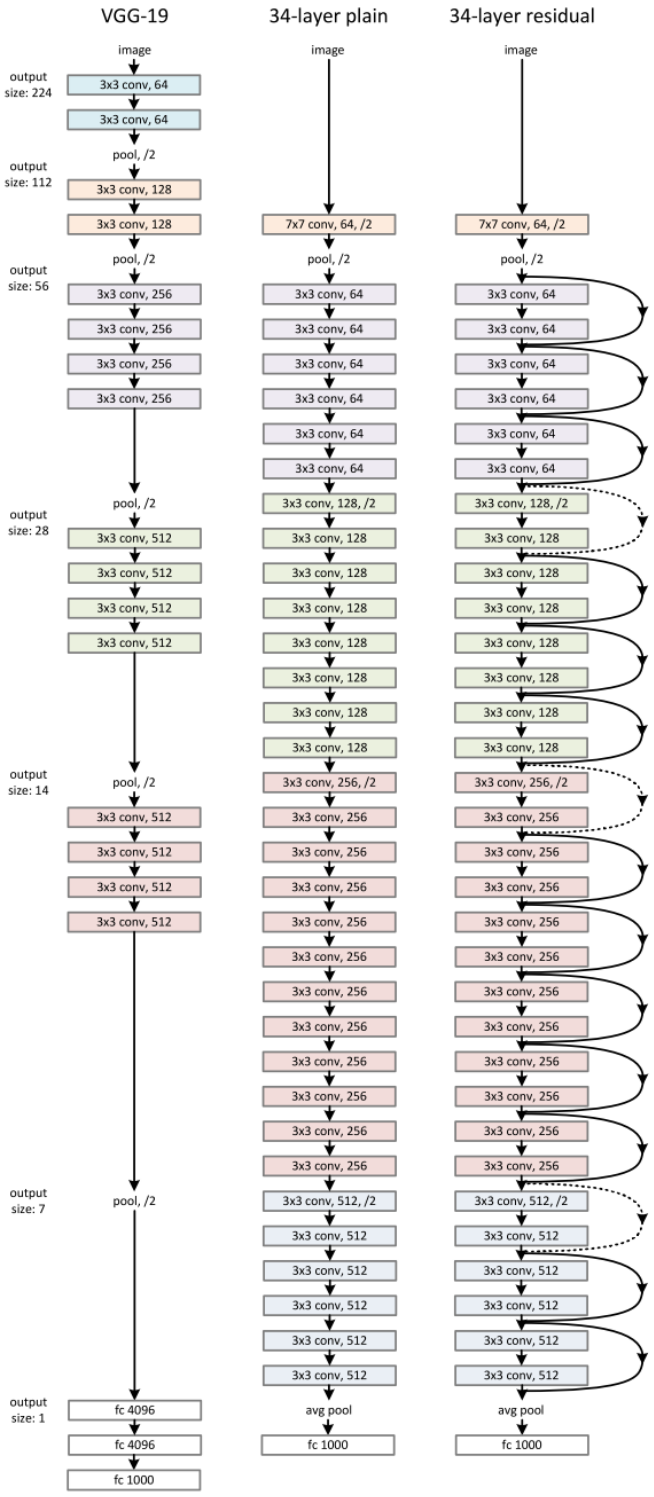


Figura 3.4: Architettura della ResNet34

3.1.3 NasNet

Uno degli aspetti più ostici che da sempre accompagna un'architettura neurale è la sua progettazione. Le scelte che un progettista deve saper compiere sono innumerevoli e non banali: quali e sopratutto quanti layer scegliere, in che ordine disporli, quale dimensione per i filtri adoperare, *etc*; rese ancor più ardue dall'etereogenità dei task in cui le reti vengono adoperate.
Per ovviare a questo problema, nell'Aprile 2018 Zoph et al. [41] hanno messo a punto un innovativo paradigma di progettazione automatica di architetture neurali convoluzionali. Ispirato al framework *Neural Architecture Search* (NAS) [40], questo paradigma mediante reinforcement learning cerca di migliorare nel corso del tempo la configurazione dell'architettura sottoposta su un particolare dataset d'interesse.
In NAS, un controller implementato da una rete neurale ricorsiva (RNN) genera differenti configurazioni della stessa architettura, predeterminata manualmente dagli autori [Figura 3.6]. Queste sono addestrate fino alla convergenza per ottenere una certa percentuale di riconoscimento su un insieme di validation di un certo dataset d'interesse. Le percentuali sono poi successivamente utilizzate dal controller che genererà architetture sempre migliori nel corso del tempo.

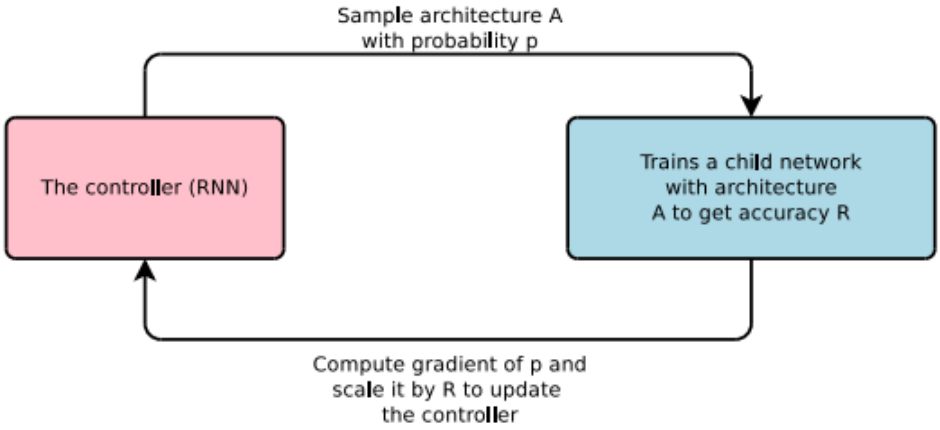


Figura 3.5: Nasnet RNN controller

Dato che l'applicazione di NAS su un dataset di grandi dimensioni, quale ImageNet [6], risulta essere computazionalmente troppo dispendiosa, la ricerca della miglior configurazione è stata eseguita su un dataset di dimensioni ridotte, quale CIFAR-10 [5], trasferendo poi l'architettura addestrata su ImageNet [6].
L'idea alla base della ricerca della configurazione migliore da parte di NAS è guidata dall'intuizione, confermata sperimentalmente da precedenti lavori [32, 14, 34], che molto spesso le architetture neurali sono composte da elementi costitutivi ripetuti, solitamente contenenti diversi layer di convoluzione, seguiti da layer di average/max-pooling [3.1.2]; ciò ha permesso di progettare il controller RNN di NAS in modo che generi una generica *convolutional cell* espressa in termini di tali elementi costitutivi.

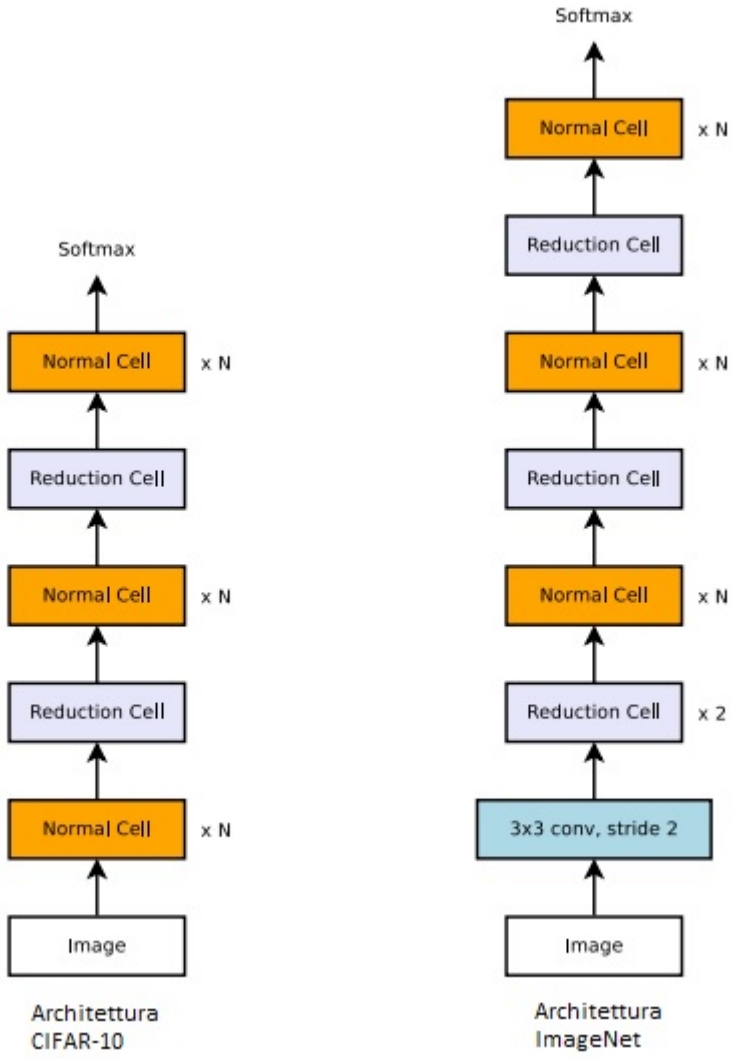


Figura 3.6: Architettura di NasNet per due differenti dataset

Per fare in modo che le configurazioni generate siano in grado di elaborare immagini di dimensioni differenti, gli autori hanno ideato due diversi tipi di convolutional cell:

1. *Normal cell*: lascia inalterata la dimensione della mappa delle features in input.
2. *Reduction cell*: riduce l'altezza e la larghezza della mappa delle features in input di un fattore di 2.

L'algoritmo di ricerca ha prodotto in totale tre differenti configurazioni di convolutional cell: NasNet-A, NasNet-B e NasNet-C [Figura 3.7, 3.8, 3.9].
L'architettura finale, una volta determinate le migliori convolutional cell, è strutturata come in [Figura 3.6].

Per discriminare le diverse architetture finali, ottenibili cambiando il valore di N e il numero di filtri del penultimo layer, gli autori hanno definito la seguente notazione: NasNet-X N @ C, dove:

- $X \in \{\text{A, B, C}\}$
- N rappresenta il numero di ripetizioni delle normal cell [Figura 3.6]
- C rappresenta il numero di filtri del penultimo layer

La versione utilizzata in questo lavoro è la NasNet-A 6 @ 4032.
Con un numero di parametri di 88.9M e una precisione di riconoscimento di 82.7% NasNet si è classificata prima all'ILSVRC-2018.

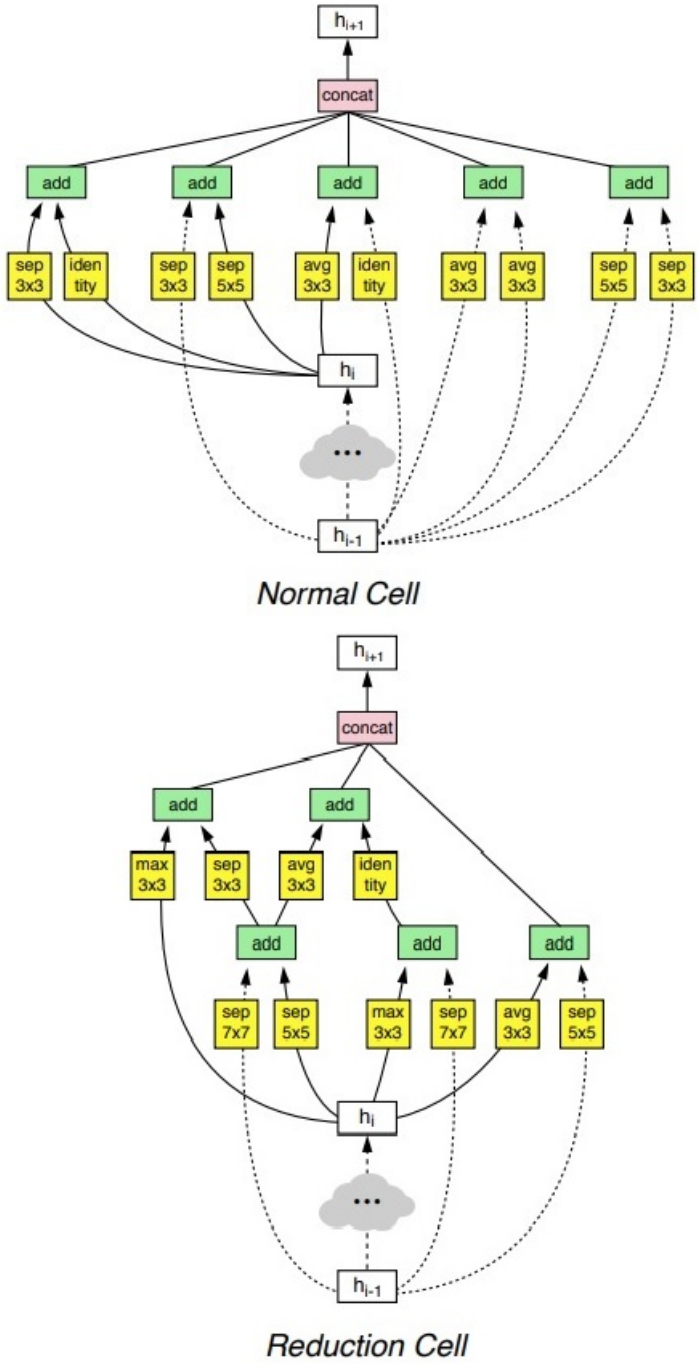


Figura 3.7: Architettura delle migliori convolutional cell (NasNet-A) identificate da NAS su CIFAR-10. L'input (blocco bianco identificato da h_i, h_{i-1}) è l'hidden state proveniente dalle attivazioni precedenti. L'output (rosa) è il risultato dell'operazione di concatenazione di tutti i rami. Ogni convolutional cell è composta da 5 blocchi. Ogni blocco è composto da due operazioni primitive (giallo) e un'operazione di combinazione (verde). Visualizzazione migliore a colori

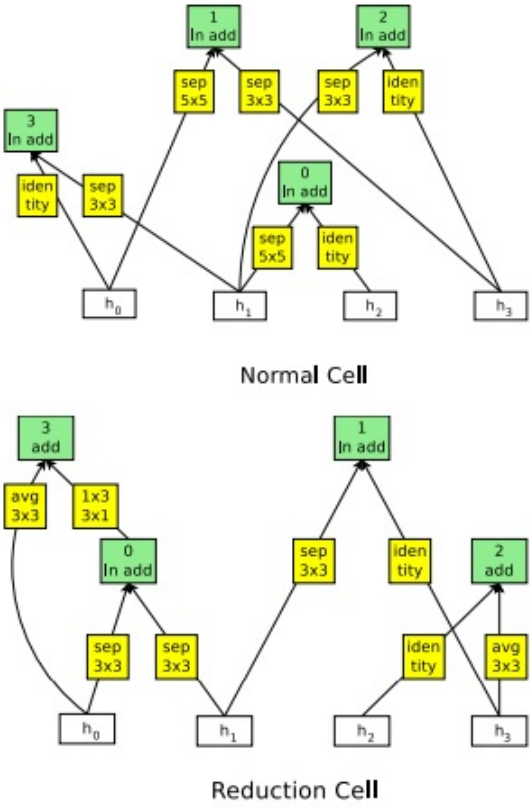


Figura 3.8: Architettura delle convolutional cell di NasNet-B identificate da NAS su CIFAR-10. L'input (blocco bianco identificato da h_i, $i \in \{0,1,2,3\}$) è l'hidden state proveniente dalle attivazioni precedenti. Ogni convolutional cell è composta da 4 blocchi. Ogni blocco è composto da due operazioni primitive (giallo) e un'operazione di combinazione (verde). Dato che gli hidden state non vengono concatenati, ognuno di essi sarà utilizzato come input nei layer successivi. Visualizzazione migliore a colori

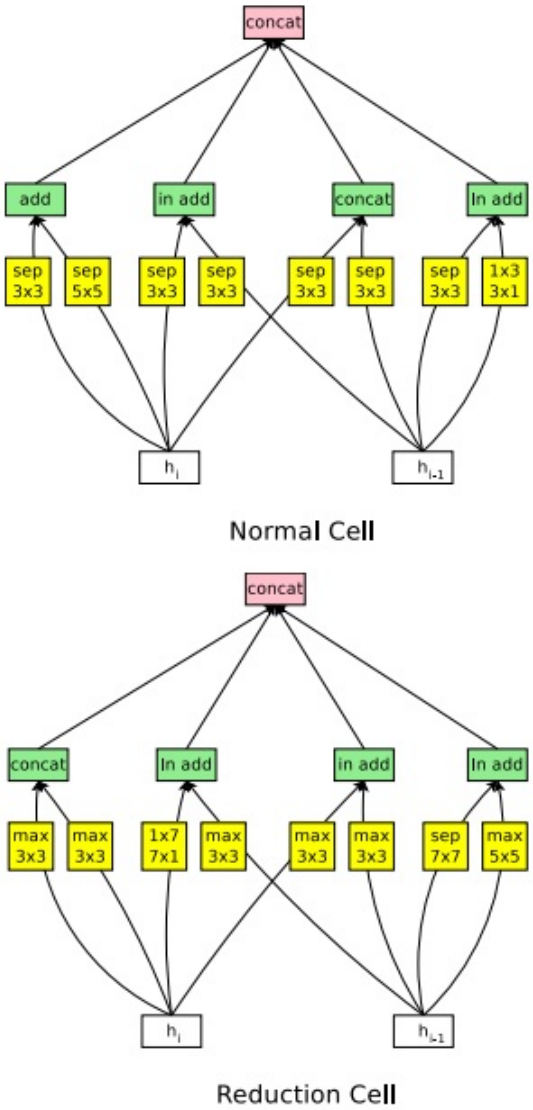


Figura 3.9: Architettura delle convolutional cell di NasNet-C identificate da NAS su CIFAR-10. L'input (blocco bianco identificato da h_i, h_{i-1}) è l'hidden state proveniente dalle attivazioni precedenti. L'output (rosa) è il risultato dell'operazione di concatenazione di tutti i rami. Ogni convolutional cell è composta da 4 blocchi. Ogni blocco è composto da due operazioni primitive (giallo) e un'operazione di combinazione (verde). Visualizzazione migliore a colori

3.2 Annotazione gerarchica

Come descritto in precedenza [2.2.1], VegFru dispone di un doppio livello di annotazione che suddivide le più di 160000 immagini appartenenti in 25 classi di alto livello, o *super-classi*, e 292 classi di basso livello, o *sotto-classi*.
Obiettivo di questo lavoro è quello di cercare di sfruttare l'annotazione gerarchica a due livelli di VegFru per migliorare il riconoscimento delle immagini sul più difficile problema di determinarne la *sotto-classe* d'appartenenza (o problema delle *sotto-classi*). Il metodo da noi sviluppato consiste dapprima nell'addestrare le architetture neurali prese in esame [3.1] sul problema di determinare l'appartenenza di un'immagine ad una delle 25 *super-classi* (o problema delle *super-classi*), per poi specializzarle nel risolvere il problema delle *sotto-classi*.

3.2.1 HybridNet

Gli autori di VegFru, per sfruttare l'annotazione gerarchica a due livelli di cui dispone il dataset da loro ideato, hanno messo a punto un framework chiamato HybridNet [15].
Esso è composto da due architetture neurali parallele che elaborano l'immagine data in input, fornita di entrambe le annotazioni della *super-classe* e *sotto-classe* (denominate anche *annotazione-a-grana-grossa* e *annotazione-a-grana-fine*, rispettivamente) a cui essa appartiene, al fine di risolvere separatamente il problema delle *sotto-classi* e quello delle *super-classi*.
Entrambe le architetture neurali sono logicamente divise in due differenti parti funzionali: un estrattore di features e un classificatore (la suddivisione può teoricamente avvenire ad ogni livello dell'architettura neurale utilizzata, ad esempio dopo l'ultimo livello di max-pool della VGG [Tabella 3.1]). Le features estratte da entrambi gli estrattori, denominate *features-a-grana-grossa* e *features-a-grana-fine*, vengono successivamente combinate al fine di formare una rappresentazione unificata, denominata *features-fuse*, dell'immagine elaborata [Figura 3.10].

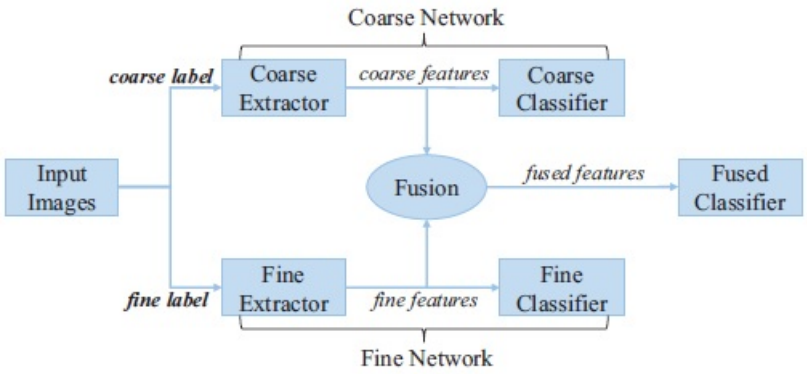


Figura 3.10: Architettura HybridNet framework

L'architettura neurale scelta per HybridNet dagli autori è la VGG16 [3.1.1], la cui suddivisione in estrattore di features e classificatore avviene prima dell'ultimo livello di maxpooling [Tabella 3.1].
Il metodo di fusione delle features che gli autori hanno deciso di adoperare è il Compact Bilinear Pooling [10], il cui utlizzo con le nuove architetture neurali proposte, quali ResNet [3.1.2] e NasNet [3.1.3], non è stato valutato in questo lavoro e si riserva per sviluppi futuri.

3.2.2 Specializzazione layer fully-connected

Tipicamente l'ultimo layer di un'architettura neurale è un layer fully-connected, che, connesso ad ogni attivazione del layer precedente, fornisce la rappresentazione finale dell'input, mappando dunque un vettore di features N-dimensionale in un vettore della dimensione dello specifico problema da risolvere.
Le specializzazioni delle architetture neurali atte a sfruttare l'annotazione gerarchica di VegFru proposte in questo lavoro, riguardano appunto l'ultimo layer fully-connected e sono le seguenti:

- *Replace*: L'ultimo layer fully-connected viene sostituito con un nuovo layer fully-connected ((b) in [Figura 3.11]).
- *Append*: Viene concatenato un nuovo layer fully-connected dopo l'ultimo layer fully-connected ((c) in [Figura 3.11]).

Nel dettaglio:

I) Mediante la tecnica del fine-tuning [37] si addestrano le architetture neurali scelte [3.1] nel riconoscere l'appartenenza di un'immagine ad una delle 25 *super-classi*.

II) Si addestrano nuovamente, sempre mediante fine-tuning [37], le architetture neurali precedentemente addestrate al punto I) per risolvere il problema delle *sotto-classi*, applicando una delle due specializzazioni specificate in precedenza.

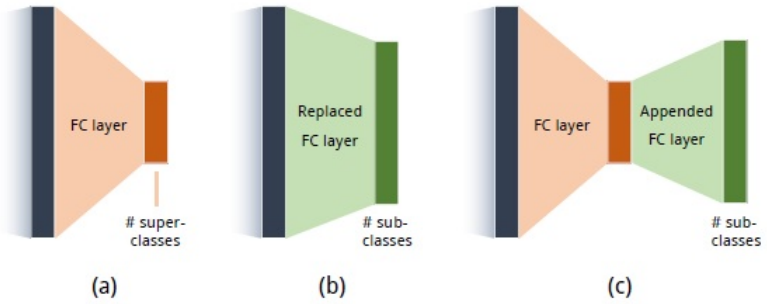


Figura 3.11: Specializzazioni layer fully-connected

Capitolo 4

Esperimenti

In questo capitolo verranno presentati i risultati ottenuti mediante l'utilizzo delle architetture di reti neurali precedentemente presentate [3.1] coerentemente con quanto specificato dagli autori di VegFru [15].
Ogni architettura neurale presa in esame è stata dapprima addestrata per risolvere sia il problema dell'appartenenza di un immagine alle *super-classi* che quello dell'appartenenza alle *sotto-classi*, cercando poi di sfruttare la struttura gerarchica di VegFru [2.2] per migliorare nel secondo, più ostico, problema [3.2].

4.1 Dettagli preliminari

Gli esperimenti da noi svolti sono così strutturati:

Fase I Nella prima fase abbiamo addestrato tre nuove architetture neurali di base, quali ResNet34, ResNet50 e NasNet, a risolvere sia il problema delle *super-classi* sia il problema delle *sotto-classi*.
La scelta della ResNet è stata guidata principalmente dal fatto che la sua particolare struttura porta a ridurre drasticamente il periodo di addestramento mantenendo comunque alta la precisione di riconoscimento.
NasNet è stata invece scelta poichè è l'architettura neurale più recente ad aver ottenuto la miglior precisione di riconoscimento nell'ILSVRC-2018.

Fase II In questa seconda ed ultima fase abbiamo applicato le specializzazioni descritte in [3.2.2] alle architetture neaurali di base addestrate in **Fase I**.

Le architetture neaurali di base, tutte pre-addestrate con l'utilizzo di ImageNet, contenente circa 1.2 milioni di immagini categorizzate in 1000 classi [6], sono state addestrate mediante la tecnica del fine-tuning [37] tramite Stochastic Gradient Descent (SGD) con momento [3] sul sottoinsieme di training di VegFru, composto da 29200 immagini.

Lo stato dell'architettura neurale che ha prodotto la miglior precisione di riconoscimento[1] sull'insieme di validation, composto da 14600 immagini, è stato successivamente selezionato per la valutazione sull'insieme di test, composto da 116931 immagini; valutazione effettuata, come specificato in [15], utilizzando la seguente metrica (*mean top-1 accuracy* d'ora in avanti): la precisione di riconoscimento viene dapprima valutata per ogni classe separatamemente, e successivamente mediata.
Le architetture neurali utilizzate in questo lavoro sono state implementate mediante l'utilizzo di PyTorch [24, 11].

4.1.1 Iperparametri

Un algoritmo d'apprendimento può essere visto come una funzione che prende in input un insieme di dati di training e produce in output una funzione (un classificatore), o un'insieme di funzioni (un modello).
Un iperparametro per un algoritmo d'apprendimento A è definito come una variabile che dev'essere impostata prima dell'applicazione di A sui dati di training, ovvero una che non è direttamente selezionata dall'algoritmo stesso [1].
Gli iperparametri da noi selezionati sono:

- **Numero di iterazioni d'addestramento**: Di base impostato a 100, con la possibilità di fermare manualmente l'addestramento quando si ha raggiunto la convergenza [Figura 4.4].

- **Batch size**: Scelto come il numero massimo che permettesse di non eccedere il limite di memoria a disposizione, tale valore è stato impostato a 100, 64 e 4 per l'addestramento di ResNet34, ResNet50 e NasNet rispettivamente, sia sul problema delle *super-classi* che su quello delle *sotto-classi*.

- **Learning rate**: Per l'addestramento di ResNet34 e ResNet50, sia sul problema delle *super-classi* che su quello delle *sotto-classi*, il learning rate iniziale è stato impostato a 0.01, constatando sperimentalmente la validità di tale scelta.
  Per l'addestramento di NasNet sul problema delle *sotto-classi* il learning rate iniziale è stato impostato a 0.0001, abbassandolo ulteriormente di un fattore di 10 per l'addestramento sul problema delle *super-classi* in quanto, presumibilmente, tale scelta avrebbe potuto ridurre il tempo impiegato da NasNet a convergere [4.4].

- **Decadimento del learning rate**: Il learning rate iniziale viene ridotto di un fattore di 10 ogni 30 iterazioni d'addestramento.

- **Momento**: Parametro che, moltiplicato per i gradienti calcolati all'iterazione precedente, aiuta SGD ad accelerare nella giusta direzione di decrescita rendendo dunque più veloce il raggiungimento della convergenza; è stato impostato a 0.9 [27, 25].

[1] La precisione di riconoscimento è calcolata come: $precisione(y, \hat{y}) = \frac{100}{n} \sum_{i=1}^{n} \mathbb{1}(y_i = \hat{y}_i)$, dove $y, \hat{y} \in \mathbb{R}^n$ rappresentano le vere classi d'appartenenza e le classi predette dall'architettura neurale rispettivamente, n è il numero di immagini presenti nel dataset d'utilizzo e $\mathbb{1}(x)$ è la funzione indicatrice.

4.1.2 Data augmentation

Con data augmentation si fa riferimento alla tecnica che permette di arricchire la quantità di dati disponibile da fornire all'architettura neurale in fase d'addestramento manipolando quelli forniti dal dataset d'utilizzo.
Le motivazioni che portano all'utilizzo di tale tecnica sono molteplici: dalla semplice necessità di ridimensionare le immagini in input, le cui dimensioni devono combaciare con quelle specificate dai progettisti dell'architettura neurale d'utilizzo [Tabella 3.1, 3.2], alla riduzione del fenomeno di overfitting [31, 21], ad un miglioramento della precisione di riconoscimento per problemi in cui il numero di dati in ogni classe non è ben bilanciato [36].
Consci del largo utilizzo di tale tecnica in precedenti lavori [21, 32, 14, 41] e incentivati dalle precedenti motivazioni, le tecniche di data augmentation utilizzate in questo lavoro sono le seguenti:

1. **Rescaling**: le immagini vengono inizialmente ridimensionate ad una dimensione pari a 256x256 quando elaborate con ResNet, a 360x360 quando elaborate con NasNet.

2. **Random resized crop**: viene successivamente eseguito un ritaglio casuale della dimensione di 224x224 o 331x331 se le immagini vengono elaborate con ResNet o NasNet rispettivamente.

3. **Random rotation**: l'immagine ritagliata viene ruotata casualmente di un angolo $\theta \in [0, 360]°$.

4. **Random horizontal flip**: l'immagine ruotata viene infine rovesciata orizzontalmente con una probalità del 50%.

Figura 4.1: Immagini d'esempio a cui è stato applicato del data augmentation. Sopra: immagini originali. Sotto: immagini a cui sono state applicate le tecniche di data augmentation specificate precedentemente

4.2 Risultati

Di seguito verranno presentati i risultati sperimentali degli esperimenti, svolti come specificato in [4.1].
Per ogni architettura neurale addestrata, di base e specializzata, verranno mostrati grafici esplicativi rappresentanti l'andamento della funzione di loss e la precisione di riconoscimento, sia per l'insieme di training che per quello di validation di VegFru, in funzione del numero di iterazioni d'addestramento.
Le figure contenenti i grafici della **Fase I** sono così organizzate:

Loss *super-classi*	Precisione *super-classi*
Loss *sotto-classi*	Precisione *sotto-classi*

Figura 4.x: Architettura neurale

Mentre le figure contenenti i grafici della **Fase II** sono così organizzate:

Loss *sotto-classi*	Precisione *sotto-classi*

Figura 4.x: Architettura neurale

In ogni grafico vengono inoltre mostrate la media mobile con periodo pari a 20 iterazioni d'addestramento e la retta di regressione (**avg 20** e **regression** rispettivamente)

4.2.1 Fase I

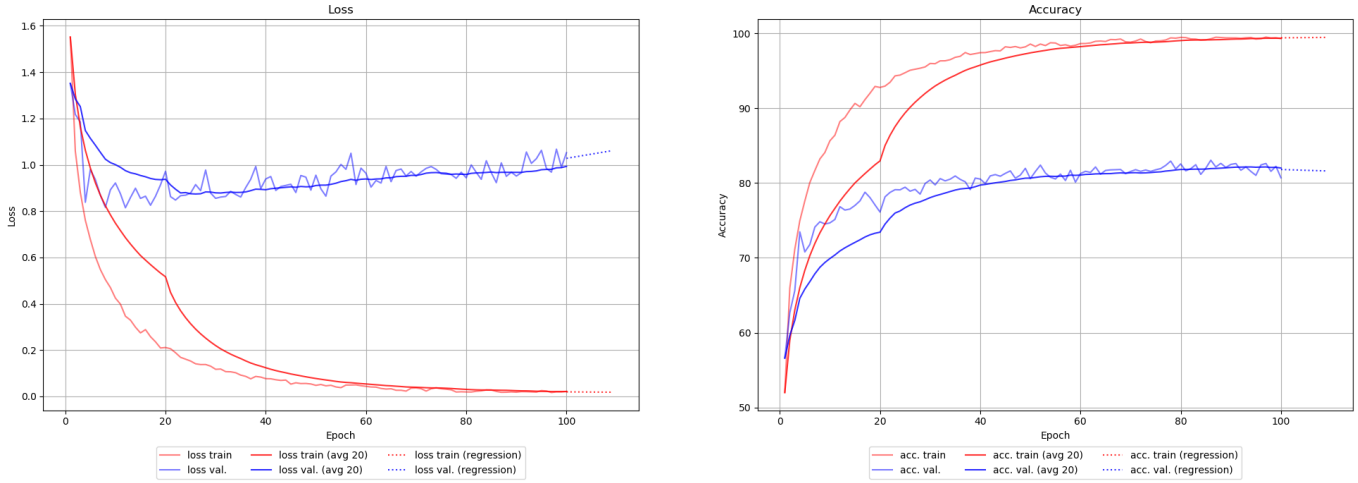
Durante la prima fase degli esperimenti svolti in questo lavoro abbiamo addestrato mediante la tecnica del fine-tuning [37] le architetture neurali di base presentate in [3.1].

Architettura neurale		*Super-classi* (25 classi)	*Sotto-classi* (292 classi)
[15]	AlexNet [21]	72.87%	66.40%
	VGG16	82.45%	77.12%
	GoogLeNet [34]	82.52%	79.22%
Questo lavoro	ResNet34	82.06%	81.14%
	ResNet50	85.33%	79.78%
	NasNet-A 6 @ 4032	**87.41%**	**84.50%**

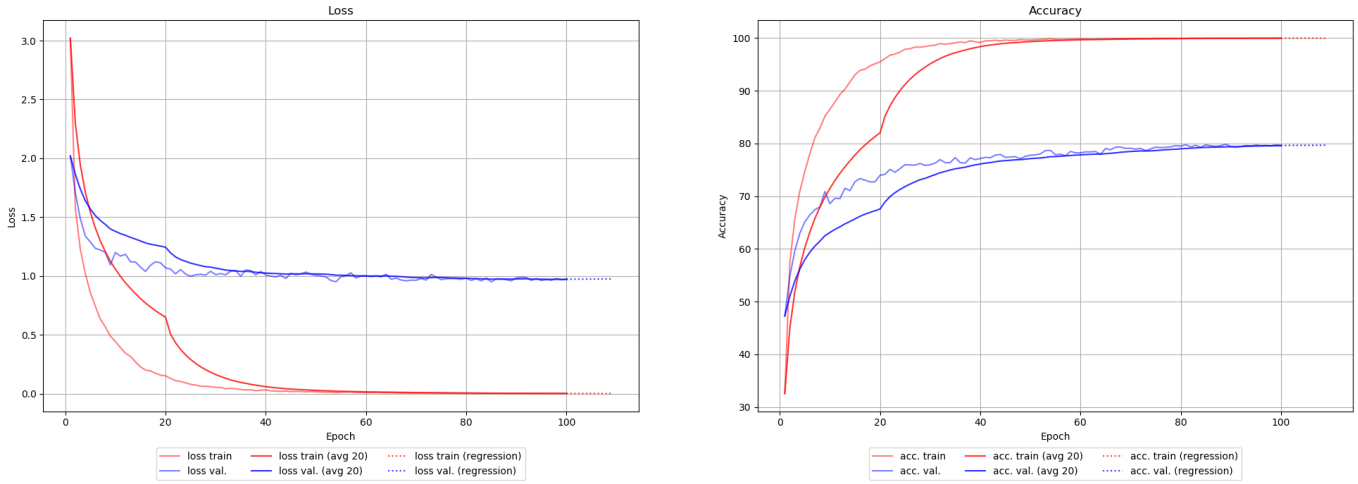
Tabella 4.3: Risultati delle architetture neurali di base sull'insieme di test. Le soluzioni da noi proposte, basate sull'impiego di ResNet e NasNet, producono un notevole miglioramento rispetto a quelle presentate in [15]. La percentuale di riconoscimento rappresentata è la *mean top-1 accuracy* definita in [4.1]

I risultati sperimentali mostrano come sul problema delle *sotto-classi* ResNet34 migliora di circa il 2% le performance di riconoscimento rispetto alla miglior architettura neurale di base utilizzata in [15], mentre ResNet50 ottiene un miglioramento di circa il

3% sul problema delle *super-classi*. I risultati migliori sono ottenuti dall'applicazione di NasNet-A 6 @ 4032, che ragginuge una percentuale di riconoscimento di 87.14% e 84.50% sul problema delle *super-classi* e delle *sotto-classi* rispettivamente.

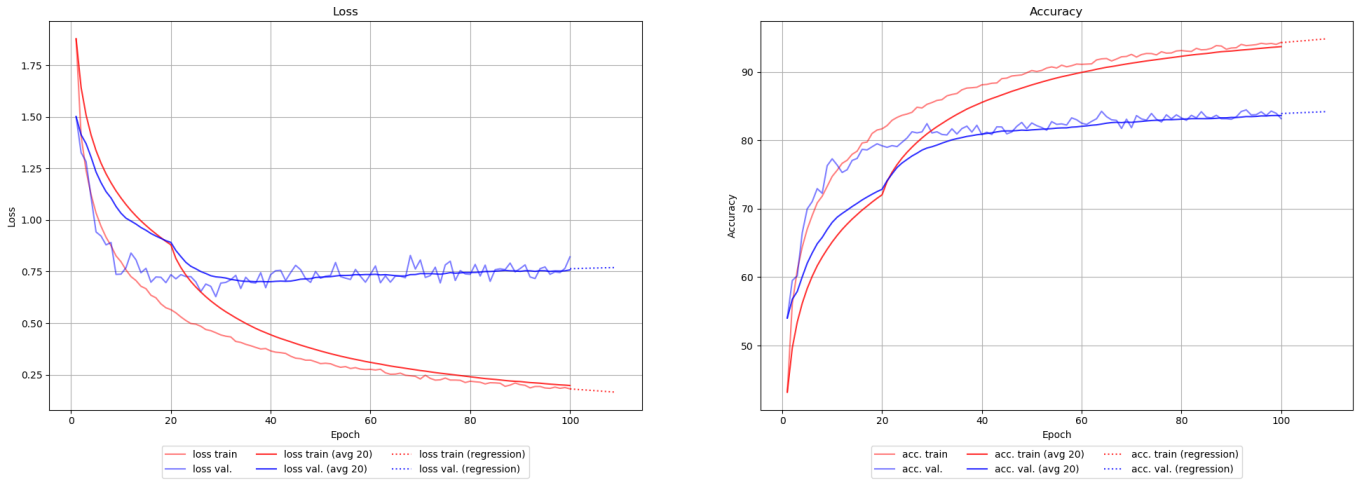


(a) *Super-classi*

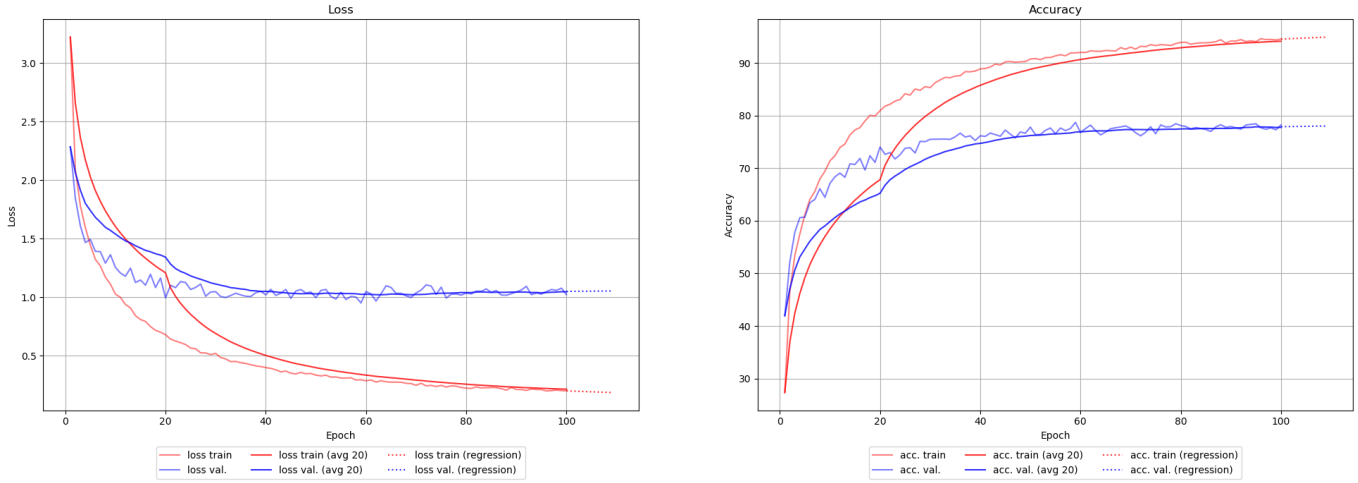


(b) *Sotto-classi*

Figura 4.2: ResNet34

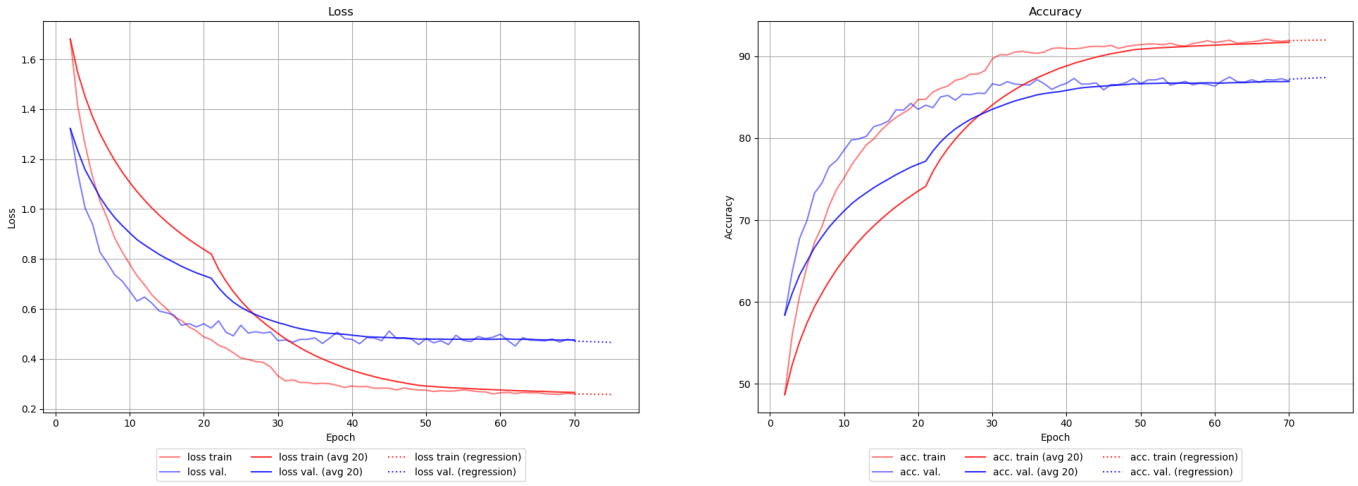


(a) *Super-classi*

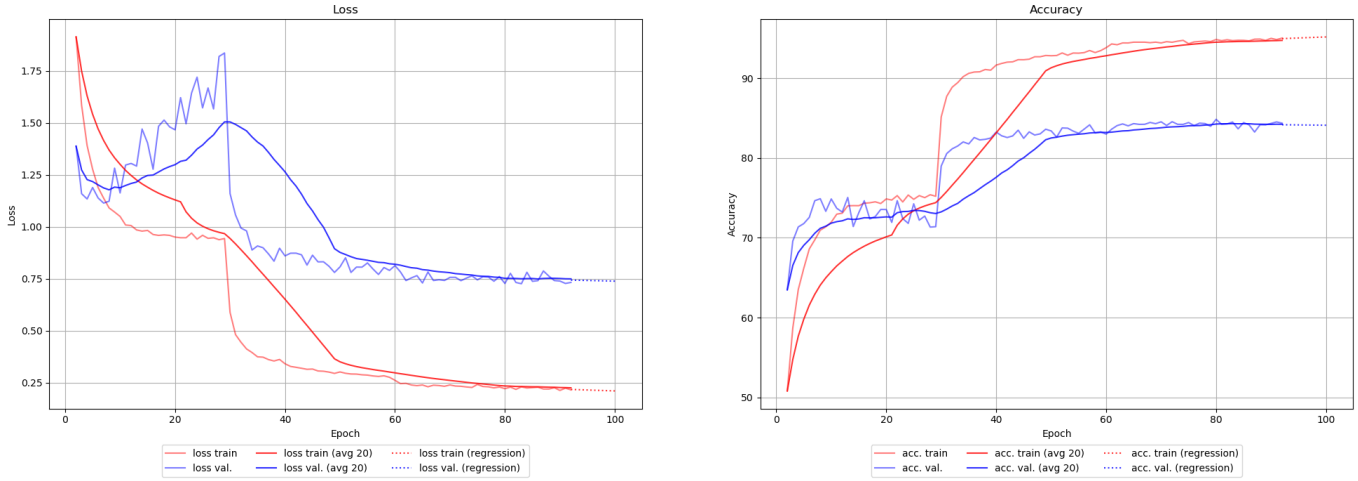


(b) *Sotto-classi*

Figura 4.3: ResNet50



(a) *Super-classi*



(b) *Sotto-classi*

Figura 4.4: NasNet-A 6 @ 4032

4.2.2 Fase II

Nella seconda fase degli esperimenti abbiamo applicato le specializzazioni descritte in [3.2.2] alle architetture neurali di base addestrate in **Fase I** per cercare di migliorare la precisione di riconoscimento nel risolvere il problema delle *sotto-classi*.

Architettura neurale		Tipo spec.	Senza spec.	Con spec.
[15]	VGG16	CBP [10]	77.12%	83.51%
Questo lavoro	ResNet34	Append	81.14%	74.93%
	ResNet34	Replace	81.14%	82.06%
	ResNet50	Replace	79.78%	84.10%
	NasNet-A 6 @ 4032	Replace	**84.50%**	84.08%

Tabella 4.4: Risultati sull'insieme di test delle architetture neurali specializzate. Le soluzioni proposte, basate sulla specializzazione dell'ultimo layer di fully-connected, migliorano ulteriormente le performance rispetto alle loro controparti di base. La percentuale di riconoscimento rappresentata è la *mean top-1 accuracy* definita in [4.1]

I risultati mostrano come l'utilizzo della specializzazione *Append* con ResNet34 degradi le performance nel risolvere il problema delle *sotto-classi* di più del 6%, mentre con la specializzazione *Replace* si ottengano dei notevoli miglioramenti. Ciò potrebbe essere dovuto al fatto che, costringendo l'architettura neurale a mappare un vettore di features di dimensione minore rispetto alla dimensione finale del problema ($25 \rightarrow 292$), di fatto stiamo comprimendo i dati con una possibile perdita d'informazione.
L'architettura neurale che più ha guadagnato in termini di percentuale di riconoscimento mediante l'utilizzo della specializzazione *Replace* è stata ResNet50, che, con un incremento di più del 4% rispetto alla versione di base, migliora le performance di riconoscimento rispetto alla VGG16 specializzata con CBP presentata in [15].
L'utilizzo della specializzazione *Replace* con NasNet non ha invece portato alcun miglioramento. Ciò può essere dovuto al fatto che una così ben progettata architettura neurale è in grado di dedurre le correlazioni gerarchiche direttamente dalle immagini appartenenti alle *sotto-classi*.

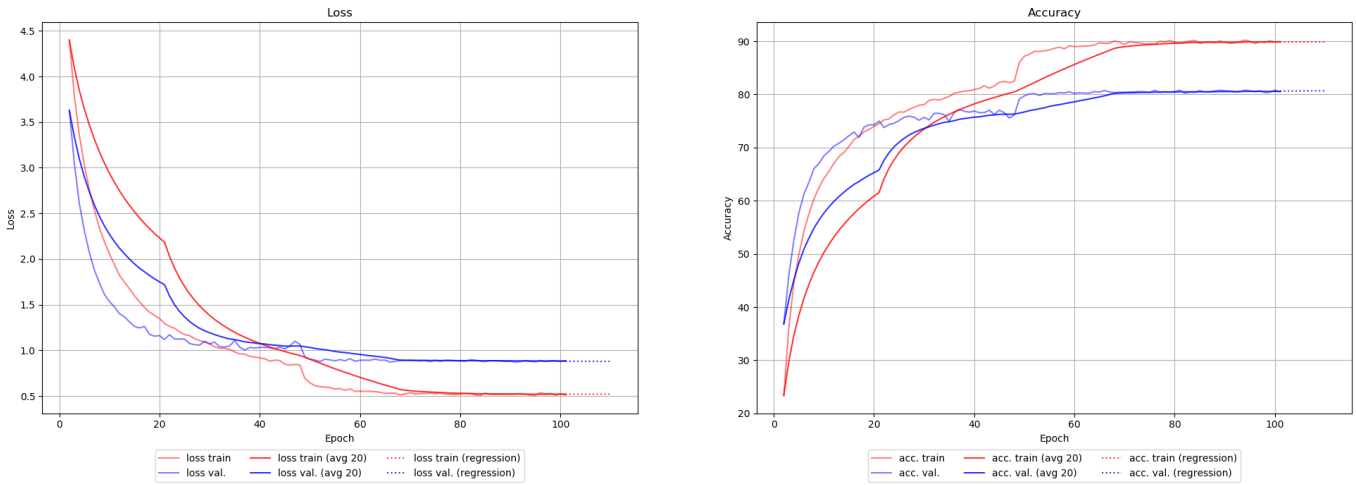


Figura 4.5: ResNet34 con specializzazione *Replace*

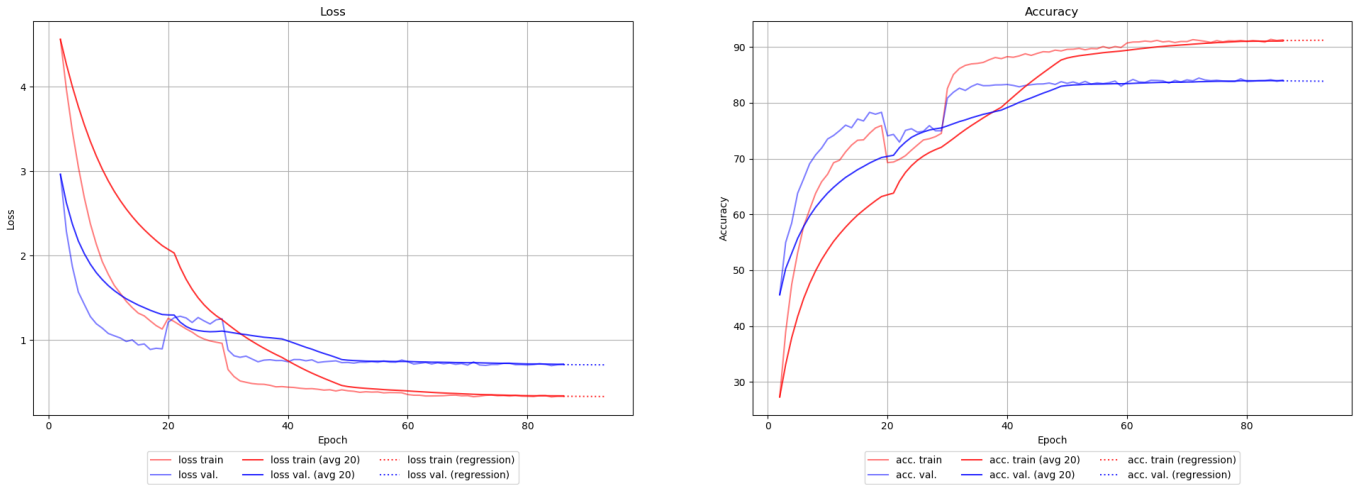


Figura 4.6: ResNet50 con specializzazione *Replace*

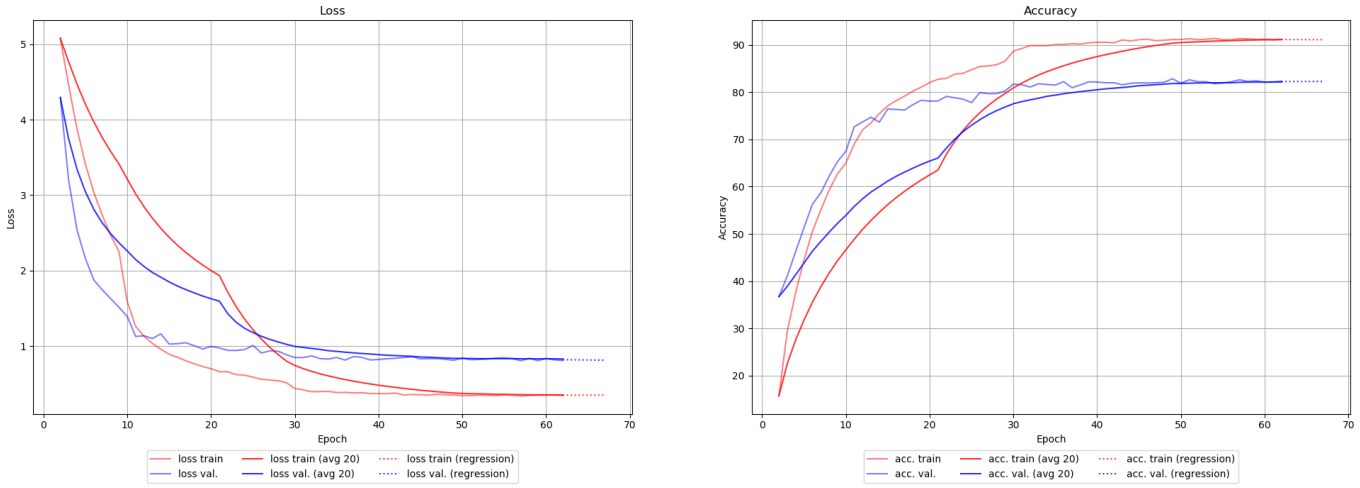


Figura 4.7: NasNet-A 6 @ 4032 con specializzazione *Replace*

Capitolo 5

Conclusioni e sviluppi futuri

Il problema del riconoscimento automatico di oggetti è ben noto nella compuer vision. Negli ultimi anni molti sono stati gli sforzi indirizzati a risolverlo, tutti, o perlomeno la maggior parte, convergenti nell'utilizzo delle reti neurali convoluzionali, che, da Krizhevsky et al. [21], sono divenute il principale strumento d'impiego in quest'ambito.
In questo lavoro, in particolare, è stato affrontato il problema del riconoscimento di varietà di vegetali in immagini digitali mediante, appunto, l'utilizzo di architetture neurali convoluzionali.
Il problema da noi trattato è reso ancor più ostico dalla sua appartenenza ad una più specifica classe di problemi cosiddetti di *Fine-grained Visual Categorization* (FGVC), che consiste nel classificare oggetti in classi subordinate, come ad esempio le specie d'appartenenza di animali, quali uccelli o cani, o il modello di un'automobile o un aereo [35, 19, 20, 23].
A differenza della classificazione generica [6] la FGVC deve far fronte a due sostanziali problemi:

1. Una maggiore differenza intra-classe, in quanto immagini appartenenti ad una stessa classe subordinata possono risultare molto diverse tra loro; diversità dovuta all'acquisizione di tali immagini in condizioni di luce, posizione e sfondo differenti.

2. Una più sottile differenza inter-classe, dovuta all'enorme somiglianza di immagini appartenenti a classi subordinate distinte.

In quest'ottica, dataset per la FGVC che siano strettamente correlati con la quotidianità culinaria delle persone ne esistono pochi, che risultano però essere incompleti o mal strutturati [12, 9, 2].
A tal proposito Hou et al. [15] hanno creato VegFru, un dataset costruito appositamente per la FGVC contenente più di 160000 immagini di frutta e verdura allo stato crudo, suddivise in 92 e 200 classi subordinate rispettivamente ed organizzate secondo un doppio livello di annotazione gerarchica, composto di 25 *super-classi* e 292 *sotto-classi*.
Utilizzato in questo lavoro, per risolvere i problemi intrinsechi dovuti alla sua appartenenza ai dataset di FGVC, sono state impiegate le reti neurali convoluzionali, che negli

ultimi anni hanno dimostrato di essere gli strumenti migliori nel riconoscimento automatico di oggetti [21, 14, 32, 34, 41]. In particolare sono state adoperate Resnet34, ResNet50 e NasNet-A 6 @ 4032 [14, 41], a cui sono state applicate due differenti tipi di specializzazioni, che consistono nella modifica dell'ultimo (*Replace*), o nell'aggiunta di un nuovo (*Append*) layer fully-connected, per sfruttare la suddivisione gerarchica di VegFru cercando di migliorare le performance nel cosiddetto, più ostico, problema delle *sotto-classi*.
La specializzazione di *Append* di fatto non ha portato alcun miglioramento, degradando notevolmente le performance nel riconoscimento. Ciò può essere dovuto alla notevole compressione dei dati, con conseguente perdita d'informazione, del penultimo layer di fully-connected che fornisce un vettore 25-dimensionale, rimappato poi dall'ultimo layer di fully-connected in un vettore 292-dimensionale. La specializzazione di *Replace* ha portato invece notevoli miglioramenti, come dimostrato sperimentalmente in precedenza da [37]. NasNet ha comunque ottenuto i risultati migliori senza utilizzo di specializzazioni, dimostrando come una ben progettata architettura neurale possa dedurre correlazioni gerarchiche di fatto senza aver alcuna nozione di gerarchia nota a priori.

5.1 Sviluppi futuri

Oltre ad essere oggetto di ricerca per la FGVC, VegFru può essere utilizzato come primo tassello verso la costruzione di sistemi intelligenti nell'ambito delle smart home, che interagiscano con le persone nello loro quotidianità, ad esempio suggerendo ricette o eseguendo un controllo qualità del materiale culinario [4].
In tal caso è necessario disporre di strumenti sempre più specializzati nell'ambito del riconoscimento automatico di oggetti per FGVC. A tal proposito può essere utile utilizzare differenti metodi di sfruttamento della suddivisione gerarchica di VegFru: una specializzazione promettente sembra essere il Compact Bilinear Pooling (CBP) [10], utilizzato dagli stessi autori di VegFru.
Un differente approccio consiste invece nell'applicare il Neural Architecture Search (NAS) [40, 41] a VegFru, ricercando dunque la miglior architettura neurale specificatamente per esso.

Bibliografia

[1] Yoshua Bengio. «Practical recommendations for gradient-based training of deep architectures». In: (24 giu. 2012). arXiv: http://arxiv.org/abs/1206.5533v2 [cs.LG].

[2] Lukas Bossard, Matthieu Guillaumin e Luc Van Gool. «Food-101 – Mining Discriminative Components with Random Forests». In: *European Conference on Computer Vision*. 2014.

[3] Léon Bottou. *Large-Scale Machine Learning with Stochastic Gradient Descent*. 2010. DOI: 10.1007/978-3-7908-2604-3_16.

[4] Marco Buzzelli, Federico Belotti e Raimondo Schettini. «Recognition of Edible Vegetables and Fruits for Smart Home Appliances». In: *2018 IEEE 8th International Conference on Consumer Electronics - Berlin (ICCE-Berlin) (ICCE-Berlin 2018)*. Berlin, Germany, set. 2018.

[5] *CIFAR-10 and CIFAR-100 datasets*. https://www.cs.toronto.edu/~kriz/cifar.html.

[6] J. Deng et al. «ImageNet: A large-scale hierarchical image database». In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition*. Giu. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[7] Jeff Donahue et al. «DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition». In: (6 ott. 2013). arXiv: http://arxiv.org/abs/1310.1531v1 [cs.CV].

[8] Mohammad Sadegh Ebrahimi e Hossein Karkeh Abadi. «Study of Residual Networks for Image Recognition». In: (21 apr. 2018). arXiv: http://arxiv.org/abs/1805.00325v1 [cs.CV].

[9] *Food Image Dataset | MMSPG*. https://mmspg.epfl.ch/food-image-datasets.

[10] Yang Gao et al. «Compact Bilinear Pooling». In: (19 nov. 2015). arXiv: http://arxiv.org/abs/1511.06062v2 [cs.CV].

[11] *GitHub - Cadene/pretrained-models.pytorch: Pretrained ConvNets for pytorch: NASNet, ResNeXt, ResNet, InceptionV4, InceptionResnetV2, Xception, DPN, etc.* https://github.com/Cadene/pretrained-models.pytorch.

[12] *GitHub - Horea94/Fruit-Images-Dataset: Fruits-360: A dataset of images containing fruits*. https://github.com/Horea94/Fruit-Images-Dataset.

[13] Kaiming He e Jian Sun. «Convolutional Neural Networks at Constrained Time Cost». In: (4 dic. 2014). arXiv: http://arxiv.org/abs/1412.1710v1 [cs.CV].

[14] Kaiming He et al. «Deep Residual Learning for Image Recognition». In: (10 dic. 2015). arXiv: http://arxiv.org/abs/1512.03385v1 [cs.CV].

[15] S. Hou, Y. Feng e Z. Wang. «VegFru: A Domain-Specific Dataset for Fine-Grained Visual Categorization». In: *Proc. IEEE Int. Conf. Computer Vision (ICCV)*. Ott. 2017, pp. 541–549. DOI: 10.1109/ICCV.2017.66.

[16] Andrew G. Howard. «Some Improvements on Deep Convolutional Neural Network Based Image Classification». In: (19 dic. 2013). arXiv: http://arxiv.org/abs/1312.5402v1 [cs.CV].

[17] *iMaterialist Challenge at FGVC 2017 | Kaggle*. https://www.kaggle.com/c/imaterialist-challenge-FGVC2017.

[18] *iNaturalist Challenge at FGVC 2017 | Kaggle*. https://www.kaggle.com/c/inaturalist-challenge-at-fgvc-2017.

[19] Aditya Khosla et al. «Novel Dataset for Fine-Grained Image Categorization». In: *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*. Colorado Springs, CO, 2011.

[20] Jonathan Krause et al. «3D Object Representations for Fine-Grained Categorization». In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[21] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton. «ImageNet classification with deep convolutional neural networks». In: 60 (2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386.

[22] Tsung-Yu Lin, Aruni RoyChowdhury e Subhransu Maji. «Bilinear CNNs for Fine-grained Visual Recognition». In: (29 apr. 2015). arXiv: http://arxiv.org/abs/1504.07889v6 [cs.CV].

[23] S. Maji et al. *Fine-Grained Visual Classification of Aircraft*. Rapp. tecn. 2013. arXiv: 1306.5151 [cs-cv].

[24] *PyTorch*. https://pytorch.org/.

[25] Ning Qian. «On the momentum term in gradient descent learning algorithms». In: 12 (1999), pp. 145–151. ISSN: 0893-6080. DOI: 10.1016/s0893-6080(98)00116-6.

[26] Qi Qian et al. «Fine-Grained Visual Categorization via Multi-stage Metric Learning». In: (3 feb. 2014). arXiv: http://arxiv.org/abs/1402.0453v2 [cs.CV].

[27] Sebastian Ruder. «An overview of gradient descent optimization algorithms». In: (15 set. 2016). arXiv: http://arxiv.org/abs/1609.04747v2 [cs.LG].

[28] David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams. «Learning representations by back-propagating errors». In: 323 (1986), pp. 533–536. ISSN: 0028-0836. DOI: 10.1038/323533a0.

[29] Olga Russakovsky et al. «ImageNet Large Scale Visual Recognition Challenge». In: (1 set. 2014). arXiv: http://arxiv.org/abs/1409.0575v3 [cs.CV].

[30] Pierre Sermanet et al. «OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks». In: (21 dic. 2013). arXiv: http://arxiv.org/abs/1312.6229v4 [cs.CV].

[31] P. Y. Simard, D. Steinkraus e J. C. Platt. «Best practices for convolutional neural networks applied to visual document analysis». In: *Proc. Seventh Int. Conf. Document Analysis and Recognition*. Ago. 2003, pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801.

[32] Karen Simonyan e Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: (4 set. 2014). arXiv: http://arxiv.org/abs/1409.1556v6 [cs.CV].

[33] Ting Sun, Lin Sun e Dit-Yan Yeung. «Fine-Grained Categorization via CNN-Based Automatic Extraction and Integration of Object-Level and Part-Level Features». In: (22 giu. 2017). arXiv: http://arxiv.org/abs/1706.07397v1 [cs.CV].

[34] C. Szegedy et al. «Going deeper with convolutions». In: *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*. Giu. 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

[35] C. Wah et al. *The Caltech-UCSD Birds-200-2011 Dataset*. Rapp. tecn. CNS-TR-2011-001. California Institute of Technology, 2011.

[36] Sebastien C. Wong et al. «Understanding data augmentation for classification: when to warp?» In: (28 set. 2016). arXiv: http://arxiv.org/abs/1609.08764v2 [cs.CV].

[37] Jason Yosinski et al. «How transferable are features in deep neural networks?» In: *Advances in Neural Information Processing Systems 27, pages 3320-3328. Dec. 2014* (6 nov. 2014). arXiv: http://arxiv.org/abs/1411.1792v1 [cs.LG].

[38] Matthew D. Zeiler e Rob Fergus. *Visualizing and Understanding Convolutional Networks*. 2014. DOI: 10.1007/978-3-319-10590-1_53.

[39] Ning Zhang et al. «Part-based R-CNNs for Fine-grained Category Detection». In: (15 lug. 2014). arXiv: http://arxiv.org/abs/1407.3867v1 [cs.CV].

[40] Barret Zoph e Quoc V. Le. «Neural Architecture Search with Reinforcement Learning». In: (5 nov. 2016). arXiv: http://arxiv.org/abs/1611.01578v2 [cs.LG].

[41] Barret Zoph et al. «Learning Transferable Architectures for Scalable Image Recognition». In: (21 lug. 2017). arXiv: http://arxiv.org/abs/1707.07012v4 [cs.CV].