



UNIVERSITÀ DEGLI STUDI DI FIRENZE  
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA  
DELL'INFORMAZIONE

---

Tesi di Laurea Magistrale in Ingegneria Informatica

**RANDOM REBOOT AND FOCAL DISTILLATION  
FOR DATA-INCREMENTAL LEARNING OF VISUAL  
CATEGORIES**

*Candidato*

Marioemanuele Ghianni

*Relatori*

Prof. Andrew David Bagdanov

Prof. Fabio Schoen

*Correlatore*

Dott. Simone Magistri

---

Anno Accademico 2021-2022

# Indice

<b>Elenco delle figure</b>	<b>iv</b>
<b>Elenco delle tabelle</b>	<b>vi</b>
<b>Ringraziamenti</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>1 Introduzione</b>	<b>1</b>
1.1 Apprendimento incrementale e catastrophic forgetting nei modelli di Deep Learning . . . . .	2
1.2 Una possibile soluzione: le metodologie di apprendimento incrementale . . . . .	6
1.3 Contributi di questo lavoro . . . . .	8
<b>2 Lavori Correlati</b>	<b>14</b>
2.1 Image Classification tramite reti neurali . . . . .	14
2.1.1 CNNs e ResNet . . . . .	16
2.2 Knowledge distillation per la Computer Vision . . . . .	19
2.2.1 Focal distillation . . . . .	23
2.3 Class-Incremental Learning . . . . .	26
2.3.1 Approcci . . . . .	26

2.3.2	Learning Without Forgetting . . . . .	31
2.3.3	Elastic Weight Consolidation . . . . .	32
2.3.4	ICaRL . . . . .	36
2.3.5	Metriche per Class-incremental Learning . . . . .	40
<b>3</b>	<b>Data Incremental Learning</b>	<b>44</b>
3.1	Definizione del problema . . . . .	44
3.2	Metriche per Data-Incremental Learning . . . . .	46
3.3	Tecniche di Continual Learning per lo scenario Data-incremental	49
3.3.1	Feature distillation . . . . .	49
3.3.2	LwF . . . . .	50
3.3.3	Focal distillation . . . . .	51
3.3.4	EWC . . . . .	51
3.3.5	iCaRL . . . . .	52
3.4	Data Incremental Learning con metodologia Random Distillation Reboot . . . . .	53
<b>4</b>	<b>Risultati sperimentali</b>	<b>56</b>
4.1	Configurazione degli esperimenti . . . . .	56
4.1.1	Datasets e protocollo di sperimentazione . . . . .	57
4.1.2	Modelli valutati . . . . .	59
4.1.3	Regime di training . . . . .	60
4.2	Esperimenti su CIFAR-100 . . . . .	61
4.2.1	Valutazione della baseline e degli upper bound . . . . .	61
4.2.2	Studi sugli exemplar . . . . .	67
4.2.2.1	Valutazione della miglior policy di selezione .	67
4.2.2.2	Memoria incrementale vs memoria fissa . . . . .	69
4.2.3	Valutazione degli approcci . . . . .	72

4.2.3.1	Feature distillation . . . . .	73
4.2.3.2	EWC . . . . .	77
4.2.3.3	iCaRL . . . . .	82
4.2.3.4	LwF : Knowledge Distillation . . . . .	87
4.2.3.5	Focal distillation . . . . .	92
4.2.4	Riepilogo e confronto approcci . . . . .	98
4.2.5	Random Distillation Reboot . . . . .	101
4.2.6	Risultati su ResNet18 . . . . .	105
4.3	Esperimenti su TinyImageNet . . . . .	109
4.3.1	Valutazione dei vari approcci . . . . .	110
4.3.2	Metodologia proposta: Random Distillation Reboot . .	112
4.3.3	Risultati su ResNet18 . . . . .	115
<b>5</b>	<b>Conclusioni</b>	<b>119</b>
5.0.1	Sintesi del lavoro e conclusioni . . . . .	119
5.0.2	Sviluppi futuri . . . . .	120
<b>Bibliografia</b>		<b>123</b>

# Elenco delle figure

1.1	Confronto static ML vs adaptive ML . . . . .	4
1.2	Esempio di uno scenario di Class-Incremental learning . . . . .	9
1.3	Esempio di uno scenario di Domain-Incremental learning . . . . .	10
2.1	Descrizione visiva di ciò che il computer vede e del processo di classificazione . . . . .	15
2.2	Rete convoluzionale per la classificazione di immagini . . . . .	17
2.3	Esempio di un blocco residuale con skip connection . . . . .	18
2.4	Composizione in cascata di una rete ResNet . . . . .	19
2.5	Training tramite feature distillation . . . . .	21
2.6	Figura d'esempio del problema dei NFs . . . . .	24
2.7	Bordo di decisione a singola linea . . . . .	29
2.8	Struttura del framework FACIL . . . . .	30
2.9	Pseudocodice della procedura di LwF . . . . .	31
2.10	Funzionamento di EWC . . . . .	35
2.11	Pseudocodice della procedura di aggiornamento di iCaRL . . . . .	39
3.1	Struttura dell'apprendimento data-incremental . . . . .	45
3.2	Schema dell'addestramento distillation reboot . . . . .	54
4.1	Esempio di Immagini presenti in CIFAR-100 . . . . .	58
4.2	Valutazione della baseline e degli upper bound di CIFAR-100	63

---

4.3	Drift delle performance su CIFAR-100 . . . . .	64
4.4	Confronto del forgetting su CIFAR-100 . . . . .	65
4.5	Comparazione tra Adam e SGD su CIFAR-100 . . . . .	66
4.6	Confronto tra le accuracy delle policy di selezione degli exemplar	68
4.7	Impatto sulle performance degli exemplar . . . . .	70
4.8	Confronto dell'accuracy tra exemplar a memoria incrementale e exemplar a memoria fissa . . . . .	72
4.9	Forgetting dell'approccio feature distillation . . . . .	73
4.10	Accuracy dell'approccio feature distillation . . . . .	74
4.11	Accuracy dell'approccio feature distillation con exemplar . . .	77
4.12	Accuracy dell'approccio EWC . . . . .	78
4.13	Andamento della loss di EWC . . . . .	79
4.14	Accuracy delle varianti di EWC provate . . . . .	80
4.15	Accuracy di EWC in combinazione con gli exemplar . . . .	81
4.16	Accuracy dell'approccio iCaRL . . . . .	83
4.17	Accuracy delle varianti di iCaRL provate . . . . .	84
4.20	Accuracy (4.18) e forgetting (4.19) dell'approccio LwF . . .	90
4.21	Accuracy dell'approccio LwF con gli exemplar . . . . .	91
4.24	Accuracy (4.22) e forgetting (4.23) dell'approccio focal distil- lation e della sua variante focal feature distillation . . . . .	96
4.25	Accuracy con exemplar di focal distillation e della sua variante focal fd . . . . .	98
4.28	Valutazione della metodologia random reboot . . . . .	102
4.31	Valutazione della metodologia random reboot con gli exemplar	104
4.32	Confronto tra le accuracy dei principali approcci su ResNet18 e CIFAR-100. . . . .	105

4.33 Valutazione della baseline e degli upper bound di TinyImageNet con ResNet32 . . . . .	109
4.34 Confronto tra le accuracy dei principali approcci su TinyImageNet e ResNet32 . . . . .	110
4.37 Valutazione della metodologia random reboot . . . . .	113
4.40 Valutazione della metodologia random reboot con gli exemplar . . . . .	114
4.41 Grafico di confronto tra le accuracy dei principali approcci su ResNet18 e TinyImageNet . . . . .	115

# Elenco delle tabelle

4.1	Risultati della grid search sul learning rate con Adam . . . . .	62
4.2	Grid search sul learning rate con SGD . . . . .	66
4.3	Conteggio dei true positive nell'approccio feature distillation .	74
4.4	Riepilogo numerico di confronto dell'approccio di feature di- stillation. . . . .	75
4.5	Riepilogo numerico di feature distillation con gli exemplar . .	76
4.6	Riepilogo numerico di EWC . . . . .	80
4.7	Riepilogo dei risultati di EWC con gli exemplar . . . . .	82
4.8	Risultati riassuntivi degli esperimenti effettuati su iCaRL . . .	86
4.9	Medie delle distanze in norma 2 tra i prototipi. . . . .	87
4.10	Grid search dell'iperparametro $\lambda$ di LwF . . . . .	88
4.11	Riepilogo numerico dei risultati di LwF . . . . .	89
4.12	Risultati di LwF al variare della temperatura . . . . .	89
4.13	Risultati numerici riassuntivi della valutazione di LwF con gli exempla . . . . .	92
4.14	Grid search sugli iperparametri di focal distillation . . . . .	93
4.15	Grid search di focal feature distillation . . . . .	94
4.16	Risultati numerici riassuntivi di focal distillation e della sua variante focal feature distillation . . . . .	95

4.17 Risultati numerici riassuntivi, con exemplar, di focal distillation e focal fd . . . . .	97
4.18 Riepilogo degli approcci provati su CIFAR-100 e ResNet32 . .	99
4.19 Riepilogo degli approcci provati su CIFAR-100 e ResNet32 con gli exemplar . . . . .	100
4.20 Riepilogo degli approcci provati su CIFAR-100 e ResNet18 . .	106
4.21 Comparazione tra ResNet32 e ResNet18 dell'accuracy degli approcci provati su CIFAR-100 . . . . .	107
4.22 Comparazione tra ResNet32 e ResNet18 del forgetting degli approcci provati su CIFAR-100 . . . . .	108
4.23 Riepilogo degli approcci provati su TinyImageNet e ResNet32	111
4.24 Riepilogo degli approcci provati su TinyImageNet e ResNet18	116
4.25 Riassunto di comparazione tra l'accuracy su ResNet32 e ResNet18 degli approcci provati su TinyImageNet . . . . .	117
4.26 Riassunto di comparazione tra il forgetting su ResNet32 e ResNet18 degli approcci provati su TinyImageNet . . . . .	118

# Ringraziamenti

Ci tengo a ringraziare il mio correlatore Simone Magistri per la pazienza e il supporto offertomi in ogni ambito riguardo la realizzazione di questa tesi. Ringrazio i relatori Prof. Andrew D. Bagdanov e Prof. Fabio Schoen per l'opportunità offertami con questa tesi e per la disponibilità mostrata. Ringrazio i miei genitori Barbara e Giovanni, e mio fratello Leonardo, per avermi sempre supportato e sopportato nel corso del mio percorso di studi contornato da grandi soddisfazioni ma anche da comprensibili momenti di sconforto. Ringrazio la mia ragazza Caterina per il sempre puntuale sostegno nell'ultimo periodo e i miei amici per essere stati sempre al mio fianco, nonostante la pandemia. Ringrazio in particolare Edoardo, Riccardo, Bernardo e Alessandro per avermi accompagnato lungo questa avventura universitaria aggiungendo quel pizzico di leggerezza che mi ha permesso di affrontare al meglio gli esami e di arrivare fin qua.

# Abstract

Questo lavoro di tesi si focalizza sulla proposta e la sperimentazione di uno scenario di apprendimento incrementale inedito in letteratura, nell'ambito della visione artificiale, che è stato denominato "*Data-incremental learning*". Il compito esaminato è il classico riconoscimento di immagini ma il focus e il principale contributo riguardano la sperimentazione e la ricerca di approcci favorevoli per l'apprendimento incrementale nello scenario proposto. Il lavoro svolto è fondamentalmente un lavoro di ricerca e, come tale, segue le linee guida portanti della mentalità del ricercatore; gran parte del contributo di questa tesi è dunque dato dagli esperimenti svolti e dai risultati ottenuti che sono proposti con un approccio critico, interrogandosi spesso sulla coerenza e motivazione alla base di questi ultimi. In questa tesi vengono proposti un approccio, *Focal distillation*, e una metodologia, denominata "*Random Reboot*", che sperimentalmente ottengono risultati significativi affermandosi come soluzioni innovative e performanti per una serie di scenari di rilievo racchiusi dal data-incremental learning.

This thesis work is focused on the purpose and experimentation of an incremental learning scenario in the computer vision field, which is unprecedented in literature, and that has been named “*Data-incremental learning*”. The analysed task is the image classification, but this thesis focus and its main contribute concern testing and searching favourable methods for continuously learn in the established scenario. The accomplished work is basically a research and, as such, it follows the typical guidelines of the researcher’s attitude; the largest contribution of this dissertation is therefore developed on experiments and on the drawn results which are presented through a critical approach, based on questioning on the coherence and on the explanation of the outcomes. This thesis suggests an approach, *Focal distillation*, and a methodology, named “*Random Reboot*”, which experimentally obtain meaningful results, emerging as innovative and efficient solutions for several important contexts held by the data-incremental learning field.

# Capitolo 1

## Introduzione

Gran parte dell'intelligenza degli organismi viventi risiede nella loro capacità di sapersi adattare all' ambiente circostante, fondamentalmente *dinamico*, e di trarre da esso, in maniera continua, nuovi stimoli e input in modo da gestire in maniera sempre più favorevole l'interazione con esso. Spostandosi nello specifico nel contesto umano, ognuno di noi, ogni singolo giorno, aggiunge al proprio bagaglio di esperienze e conoscenze una serie di *informazioni*, più o meno importanti, interagendo con l'ambiente e con le altre persone. Questa collezione di esperienze e conoscenze alimenta continuamente il cervello umano in un processo di apprendimento pressoché infinito.

La capacità più interessante del cervello umano è infatti l'apprendimento continuo con una tendenza piuttosto bassa a dimenticare le vecchie conoscenze acquisite (tasso di **forgetting**). Sarà dunque proprio questo il focus principale di questa tesi che andrà ad esaminare uno scenario di apprendimento incrementale inedito in letteratura e proporrà soluzioni efficaci e generali. Facendo un esempio pratico, una persona che si trova in un paese straniero, ascoltando e relazionandosi con le persone, riuscirà pian piano ad imparare la lingua locale ma, allo stesso tempo, non dimenticherà la sua lingua nativa

se non qualche vocabolo o accento, a distanza di molti anni. Questa dinamica, ovvero dimenticare difficilmente le conoscenze acquisite in precedenza, è fortemente voluta anche nel contesto dell'apprendimento automatico tramite intelligenza artificiale e, in particolare, nel contesto della visione artificiale [37]. Tuttavia, seppur questo procedimento di apprendimento continuo possa sembrare scontato per gli esseri umani, non è altrettanto scontato per un sistema e un algoritmo di *machine learning* (di cui verranno definiti e approfonditi i concetti base correlati a questo lavoro).

## 1.1 Apprendimento incrementale e catastrophic forgetting nei modelli di Deep Learning

Nell'ultimo decennio, il *Deep Learning* e, nello specifico, le reti neurali convoluzionali hanno segnato progressi e risultati sorprendenti, non solo nell'ambito della visione artificiale, affermandosi come lo stato dell'arte e la "componente intelligente" base per molti sistemi e applicazioni che ci circondano nella vita di tutti i giorni [46]. I modelli di Deep Learning infatti, sono risultati capaci di ottenere livelli di accuratezza estremamente alti tali da essere impiegati anche in contesti delicati dove l'affidabilità del sistema è cruciale, come nel caso della sicurezza e della sorveglianza. Gran parte di questi modelli tuttavia, segue una metodologia di apprendimento tradizionale in cui si utilizza un dataset fisso per addestrare la rete; ciò, come vedremo, rappresenta un forte *vincolo*.

Tipicamente, in un approccio tradizionale, l'apprendimento di una rete neurale avviene tramite il *metodo di discesa stocastica del gradiente (SGD)* [18] [32] combinato con l'algoritmo di *backpropagation* [14]. In sostanza, solitamente si ha a disposizione un dataset, rappresentativo del problema, che

viene suddiviso in sottoinsiemi denominati *mini-batch*. I mini-batch vengono dati in input alla rete nella fase "*feed forward*", viene calcolata una *loss* e il suo gradiente rispetto ai pesi della rete secondo la *chain rule* e viene calcolato il gradiente un layer alla volta, iterando in senso "backward" a partire dall'ultimo layer. Il gradiente viene poi usato per aggiornare i pesi della rete e viene ripetuto il procedimento iterativamente per i vari mini-batch.

Il punto fondamentale è che i modelli neurali di apprendimento così descritti, per avere un funzionamento ottimale, richiedono un *dataset* a priori, spesso molto grande, da cui poter apprendere, per poi successivamente classificare i nuovi esempi. Questo forte requisito a "*training time*" non è affatto da sottovalutare in quanto costruire un dataset è spesso dispendioso sia in termini lavorativi che in termini di tempo. In alcuni casi è addirittura impossibile dal momento che abbiamo bisogno di un *deployment* immediato del sistema. Inoltre, il quantitativo di memoria necessaria per l'immagazzinamento dei dati del dataset, è spesso assolutamente non indifferente. Per la natura stessa di questo requisito, l'addestramento tradizionale dei modelli viene definito *offline* o statico, in opposizione a approcci di tipo *online learning* [4] (fig. 1.1).

Negli ultimi anni, molte applicazioni e, in generale, il mondo dell'informatica, si è mosso sempre di più sul concetto di **flusso continuo** di dati dove la crescita del quantitativo di dati disponibili risulta lineare, se non addirittura esponenziale, col passare del tempo. Un approccio tradizionale, in questo caso, porterebbe alla creazione di un dataset che cresce linearmente (nel migliore dei casi) diventando ben presto insostenibile. Il concetto di flusso di dati è inoltre strettamente collegato alla recente crescita esponenziale del mondo dell'*IoT (Internet of things)* [39]. I dispositivi presenti in

questo ecosistema sono solitamente molto limitati dal punto di vista hardware e devono saper gestire e mostrarsi flessibili verso, appunto, un continuo afflusso di dati. I modelli tradizionali invece apprendono da un set di dati tendenzialmente *"statico"* e incapace di adattarsi a cambiamenti futuri.

La staticità è decisamente un problema; una strada possibile per adattarsi ai nuovi dati può essere quella di effettuare il *re-training* completo del modello. Il riaddestramento tuttavia spesso è insostenibile in termini computazionali; molte reti moderne sono piuttosto complesse e con un numero di parametri estremamente grande tale per cui l'addestramento è molto dispendioso in termini di hardware e tempo, senza considerare il conseguente **impatto ambientale** che ne consegue. Riaddestrare da zero il modello, inoltre significa, dover tenere in memoria i dati precedenti con le varie implicazioni sulla **capacità di memoria** già descritte in aggiunta ad altre sulla **privacy**. La legislazione dello stato in questione potrebbe infatti limitare o vietare lo storage di dati di clienti su un server centrale oppure, soprattutto in ambito sanitario, si ha che le norme di privacy potrebbero consentire la conservazione dei dati dei pazienti solo per periodi di tempo limitati.

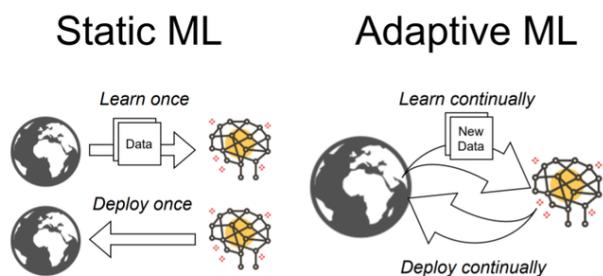


Figura 1.1: Confronto tra una metodologia di apprendimento standard e statica (a sinistra) e un approccio più contemporaneo (a destra). [22].

Gli esempi in cui un approccio di apprendimento tradizionale si mostra inadeguato e ingestibile sono molteplici. Rimanendo nel campo della visione artificiale, nella maggior parte delle applicazioni del mondo reale, si ha a disposizione solo una parte molto parziale di dati all'inizio e si ha che nuovi dati continuano ad arrivare una volta che il sistema è stato installato e messo in funzione. Ad esempio, un sistema di riconoscimento facciale spesso inizialmente funziona su un set ridotto di persone ma quello che si vuole è che sia in grado di gestire e integrare via via nuovi volti, in modo rapido e senza dimenticare quelli vecchi. O ancora, un sistema di riconoscimento di immagini o di *image forensics* [29] che lavora sul settore dei social networks, inevitabilmente si ritroverà con un set iniziale di dati ridotto ma con un flusso continuo di nuovi dati a cui deve sapersi adattare.

La seconda strada alternativa al riaddestramento è quella di addestrare il modello solo sui nuovi dati; questo è fondamentalmente l'approccio incrementale di base denominato **fine tuning**. Un addestramento con approccio fine tuning offre la possibilità di scartare i dati passati e usare solo i nuovi per continuamente addestrare e adattare il modello. La strada del fine tuning ci permette di ovviare a tutte le problematiche descritte in precedenza mantenendo i costi computazionali e i requisiti di memoria molto bassi. L'approccio fine tuning ha tuttavia un grosso problema noto in letteratura come **catastrophic forgetting**. Quando un sistema di apprendimento viene addestrato su un primo task e poi successivamente su un secondo task, per questo effetto "catastrofico" di forgetting, tende a dimenticare il task precedente ottenendo scarse performance per quel task. Tale effetto risulta sempre più marcato per i task precedenti più lontani nel tempo. Il forgetting porta dunque a un *gap* considerevole, e in alcuni casi insostenibile, delle performance tra un addestramento incrementale e un approccio di *joint*

*training*/addestramento cumulato standard.

La causa del forgetting è da ricercare alla base del funzionamento delle reti neurali e del già presentato algoritmo di aggiornamento dei pesi (*back-propagation*) che è fondamentalmente basato sul *gradiente*. In particolare, da vari studi [10] [11], emerge che la causa principale è dovuta al sovrapporsi delle rappresentazioni degli strati nascosti. Nel caso di una rappresentazione distribuita, ogni input alla rete tende a creare cambiamenti nei pesi di vari nodi. In questo contesto, il catastrophic forgetting si verifica perché vengono cambiati i valori di molti dei pesi dove "la conoscenza è immagazzinata" rendendo quindi difficile mantenere le conoscenze acquisite in precedenza.

## 1.2 Una possibile soluzione: le metodologie di apprendimento incrementale

Per far fronte alle problematiche dell'apprendimento tradizionale e tentare di mitigare i problemi anticipati in ambito incrementale, è stato introdotto recentemente un nuovo paradigma, che sta ricevendo attenzioni sempre maggiori dalla comunità scientifica ed è considerato come una soluzione promettente per alcuni compiti e sfide simili a quelli menzionati prima. Il paradigma prende il nome di **incremental learning** e racchiude una serie di scenari, metodologie e architetture mirate a mitigare il catastrophic forgetting.

L'incremental learning è un particolare scenario del *machine learning* che si occupa di gestire applicazioni più coerenti e simili al modo di comportarsi e di pensare dell'essere umano [25]; è inoltre fortemente correlato e usato spesso come sinonimo di vari topic di ricerca, quali il *continual learning* e il *lifelong learning* [7].

I task, nel contesto dell'apprendimento incrementale, sono rappresentati da "blocchetti" o piccoli *chunks* di nuovi dati *supervisionati* (con label) [28] e l'incremental learning ha come obiettivo quello di ridurre il catastrophic forgetting e di aggiornare il modello di apprendimento sui nuovi dati senza utilizzare i dati provenienti da task precedenti. Andando a mitigare il forgetting, si potrebbe tuttavia impedire alla rete di apprendere dai nuovi task; è questo lo **stability-plasticity dilemma**: un vincolo noto che riguarda i sistemi neurali biologici ed artificiali. L'idea di base è che il sistema necessita di *plasticità* per integrare le nuove conoscenze provenienti da un nuovo task ma, allo stesso tempo, *stabilità* per prevenire la perdita delle conoscenze acquisite in precedenza. Troppa plasticità porta il sistema ad apprendere nuove conoscenze ma a dimenticare catastroficamente le conoscenze ottenute in precedenza, mentre troppa stabilità evita il forgetting ma impedisce al sistema di apprendere dai nuovi dati.

Come vedremo, principalmente nel capitolo 2 e nel capitolo 4 dedicato agli esperimenti, verranno presentate varie tecniche e spesso alcune risulteranno non adatte al problema in questione proprio perchè, cercando di mitigare il forgetting, vanno a conferire al sistema troppa stabilità. Un'eccessiva stabilità limita drasticamente l'apprendimento e peggiora il già presente gap di performance rispetto a un addestramento di tipo cumulato( o *joint training*).

Riassumendo, le problematiche per cui l'incremental learning si pone come soluzione sono:

- **Restrizioni di capacità di memoria;** alcuni sistemi hanno limitazioni fisiche per lo storage dei dati e non possono fare affidamento a tecniche di joint training. La motivazione è semplicemente che non possono tenere in memoria tutti i dati che osservano ma solo un sot-

insieme molto limitato di questi. La mancanza di capacità di memoria importanti è molto comune nel campo della robotica e dell'IoT dove i vincoli hardware devono essere tenuti bene in considerazione anche in ottica futura dato che la capacità di questi dispositivi non cresce di pari passo con la quantità di dati interscambiati e rilevati dai sensori.

- **Restrizioni di sicurezza e privacy dei dati;** la legislazione potrebbe infatti limitare o vietare lo storage di dati di clienti su un server centrale. Inoltre, bisogna tener presente che, soprattutto in ambito sanitario, entra in gioco la privacy e la legislazione impedisce la conservazione per lunghi periodi dei dati dei pazienti.
- **Sostenibilità computazione;** come anticipato, il costo di addestrare modelli di deep learning può essere esorbitante e il dover fare il re-training ogni volta che arrivano nuovi dati sarà sempre più insostenibile, con impatti ambientali non trascurabili.

### 1.3 Contributi di questo lavoro

Parlando di incremental learning, lo scenario più studiato e a cui si fa normalmente riferimento è quello del **class-incremental learning** che è una successiva evoluzione del task-incremental learning e si focalizza sull'apprendere incrementalmente nuovi task o classi. Si avrà dunque un modello addestrato inizialmente con i dati di una, o comunque, poche classi e nei training successivi questo dovrà adattarsi all'introduzione di nuove classi senza andare a perdere la capacità di classificare le classi originarie. Un caso tipico

può essere un sistema di visione artificiale che inizialmente deve rilevare e classificare automobili e aerei ma che poi successivamente deve adattarsi e rilevare anche volatili e cani (maggiori dettagli in fig. 1.2).

Altro scenario di particolare interesse è quello del **domain-incremental learning**. In contrapposizione al class-incremental, in questo caso, le classi sono fisse ma sono i dati che variano e che portano a un' "espansione del dominio". Anche qui l'addestramento con dati del nuovo dominio porta a un forgetting nella classificazione dei dati appartenenti al dominio precedente. Per esempio, rimanendo nel contesto della visione artificiale, abbiamo un sistema di *object detection* a bordo di un autoveicolo che inizialmente è addestrato a rilevare altri autoveicoli in un contesto urbano soleggiato ma poi viene addestrato con immagini in presenza di nebbia e infine in caso di pioggia (maggiori dettagli in fig. 1.3).

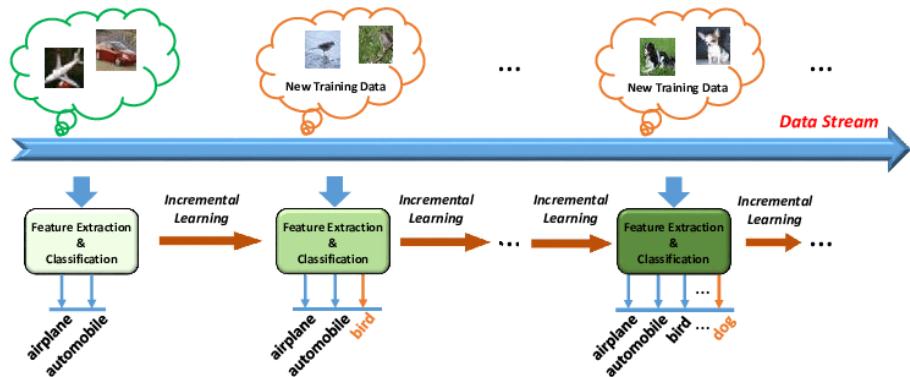


Figura 1.2: Esempio di uno scenario di class-incremental: un classificatore impara inizialmente ad effettuare la classificazione solo su immagini di aerei e automobili e poi, con l'arrivo di nuovi dati dallo stream, estende la sua capacità di classificazione alle classi dei volatili e dei cani. [1]

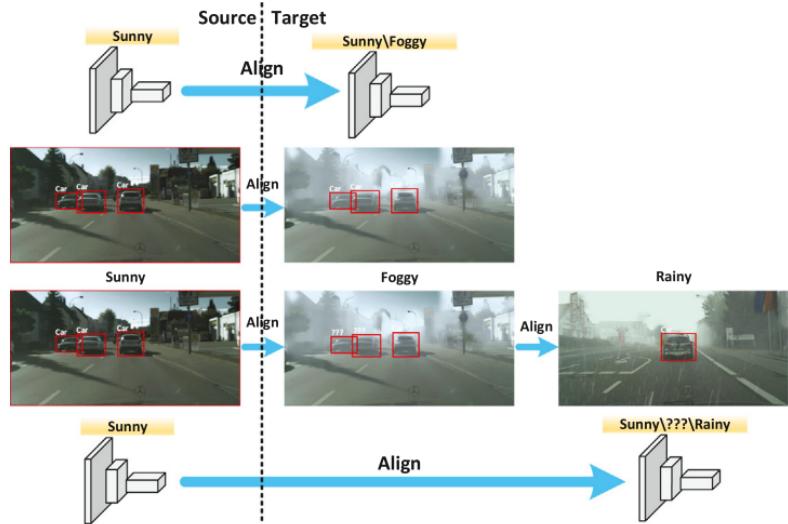


Figura 1.3: Esempio di uno scenario di domain-incremental. In figura, il modello per l'object detection è inizialmente addestrato su dataset urbani soleggiati o in presenza di nebbia e riesce a performare bene in entrambi i domini (in alto). Successivamente, quando il modello è addestrato incrementalmente con dataset di immagini in contesto piovoso, le performance di detection nelle scene in presenza di nebbia diminuisce (in basso) [38].

In questa tesi invece introduciamo un particolare tipo di apprendimento incrementale che abbiamo chiamato **Data-Incremental learning**; totalmente inedito e inesplorato in letteratura. Nel data-incremental learning, le classi sono fissate dall'inizio, a differenza del class-incremental, e i dati stavolta non variano, a differenza del domain-incremental. La particolarità è che avremo dei dati di partenza per ciascuna classe molto ridotti ma un flusso continuo nel tempo di quest'ultimi. Nel contesto data-incremental, il

modello addestrato inizialmente sarà molto inaccurato e specifico dei pochi esempi a disposizione ma diventerà pian piano più generale per ogni classe via via che arrivano nuovi esempi su cui fare il training. Anche in questo scenario di apprendimento incrementale, come vedremo più avanti nel capitolo 4 degli esperimenti, è presente una componente di forgetting e un drift considerevole di performance rispetto al joint training e per questo degno di studio e ricerca.

Lo scenario data-incremental, inesplorato in letterature è comunque molto comune in vari contesti; un’azienda potrebbe avere un sistema di detection tramite videocamere che dunque sfrutta un classificatore binario e voler magari migliorare la sua accuratezza col tempo e con l’arrivo di nuovi esempi. Stessa cosa per quanto riguarda un industria che produce un determinato tipo di prodotti fissato e necessita di un classificatore capace di operare discretamente nell’immediato e migliorare le sue performance con il tempo. O ancora, un personale sanitario potrebbe necessitare di un classificatore di immagini a raggi X per un determinato e fissato tipo di patologie ma non avere a disposizione a priori un dataset per le limitazioni temporali sulla conservazione dei dati dei pazienti (privacy). Questo personale sanitario vorrebbe dunque che il classificatore funzioni inizialmente sui pochi esempi che ha a disposizione per poi aggiornarlo in seguito con nuovi dati senza dover tenere in memoria i precedenti, sempre per implicazioni di privacy. Il modello così generato progressivamente poi può anche essere distribuito e condiviso con altro personale senza violare le norme sulla privacy dei pazienti. L’uso di classificatori su classi fissate e su dati che non variano di dominio ma con la necessità di miglioramenti incrementali è comune in generale in tutta l’industria dove, una componente di forgetting introdotta dagli aggiornamenti incrementali, rischia di "rompere" alcune pipeline di post-processing [41].

Per questi scenari e molti altri, la soluzione risiede proprio nelle metodologie di data incremental learning proposte in questo elaborato.

In questo lavoro andremo ad applicare per la prima volta nel contesto del data incremental, alcune metodologie utilizzate in letteratura per lo scenario del class incremental e valuteremo attentamente i risultati ottenuti. Particolare attenzione verrà posta agli approcci basati o derivati dalla metodologia *knowledge distillation* (approfondita nella sezione 2.2) che, nonostante sia nata in un contesto più generale, rappresenta lo stato dell'arte nel campo dell'incremental learning. Sul concetto di knowledge distillation infatti ruotano gran parte dei metodi celebri di incremental learning quali LwF [23] e iCaRL [30] (che vedremo), ma anche i più recenti LwM [8], LUCIR [16], DMC [45] e M2KD [47].

Nel capitolo 3, sarà poi proposto un approccio del tutto inedito che prenderà spunto dai precedenti e che, sotto alcune condizioni, funzionerà particolarmente bene nel contesto data-incremental. In particolare, dai risultati si otterrà che alcuni approcci di *distillation* saranno particolarmente efficaci in questo contesto con ottimi risultati sia in termini di accuratezza sia in termini di *forgetting* (entrambi i concetti verranno approfonditi nel successivo capitolo 2).

Gli approcci e i risultati ottenuti nel capitolo 4 degli esperimenti sono inoltre molto generali poichè, a differenza di molti lavori dedicati all'incremental learning, sono stati validati su due diversi dataset con composizione e numero di classi diverso ma anche su due architetture di rete di diversa complessità. Sono state dunque presentate soluzioni pratiche, applicabili in contesti reali, e importanti da cui partire per eventuali sviluppi futuri. In allegato a questo elaborato sarà infatti rilasciato un semplice codice framework su github<sup>1</sup>, fa-

---

<sup>1</sup>[https://github.com/emanuele-progr/incremental\\_data\\_learning](https://github.com/emanuele-progr/incremental_data_learning)

cile da estendere e sviluppare in futuro, tramite il quale è possibile replicare gli esperimenti descritti nel capitolo 4.

Per quanto riguarda l'organizzazione di questo lavoro:

- nel prossimo capitolo verranno descritti i lavori correlati a questo progetto tra cui: lo stato dell'arte per quanto riguarda l'*Object Recognition* tramite CNNs, gli approcci di class-incremental learning e le relative metriche sviluppate in letteratura.
- nel capitolo 3 verrà invece data una definizione formale e verrà approfondito il contesto proprio di questa tesi del data-incremental learning. Verrà spiegato il problema, definite le metriche *custom* per il data-incremental learning e illustrato l'approccio inedito sviluppato per questo contesto.
- il capitolo 4 è dedicato agli esperimenti e ai risultati; verranno analizzati i vari approcci, i dataset e le reti, valutando le performance con approccio critico e cercando di indagare le motivazioni dietro ai risultati ottenuti.
- il capitolo 5 avrà lo scopo di esporre le conclusioni e idee su possibili sviluppi futuri.

# Capitolo 2

## Lavori Correlati

In questa sezione andremo a vedere le conoscenze di base e tutti i lavori in letteratura inerenti a questo progetto di tesi e che quindi sono spesso stati dei punti di riferimento per lo svolgimento di questo lavoro. In particolare introdurremo parte della letteratura sull'*Image Classification* tramite reti convoluzionali per poi andare ad analizzare i lavori e le metriche proprie del class-incremental learning che verranno adattate al nostro contesto data-incremental nel capitolo 3.

### 2.1 Image Classification tramite reti neurali

Il compito di base preso in esame da questa tesi è il problema dell'*Image Classification*: ovvero assegnare a un'immagine di input, una label da un insieme fissato di categorie. Questo è uno dei problemi fondamentali della *Computer Vision* e ha molte applicazioni pratiche in vari contesti. C'è una grossa differenza su come noi vediamo un'immagine e come la macchina (computer) vede la stessa immagine. Noi, infatti, siamo capaci di visualizzare l'immagine e caratterizzarla basandosi sul colore, i contorni e la grandezza.

D'altra parte, quello che vede la macchina sono fondamentalmente numeri che rappresentano i *pixels* dell'immagine. Nel caso di immagine in scala di grigi, ogni pixel ha un valore di intensità corrispondente tra 0 e 255 e, a partire da questo, l'algoritmo di ML impiegato (tipicamente, reti neurali) deve derivare pattern specifici o features che possano distinguere un'immagine dalle altre, facenti parte di una diversa categoria e dare in uscita valori interpretabili come probabilità di appartenenza a una determinata classe (fig.2.1).

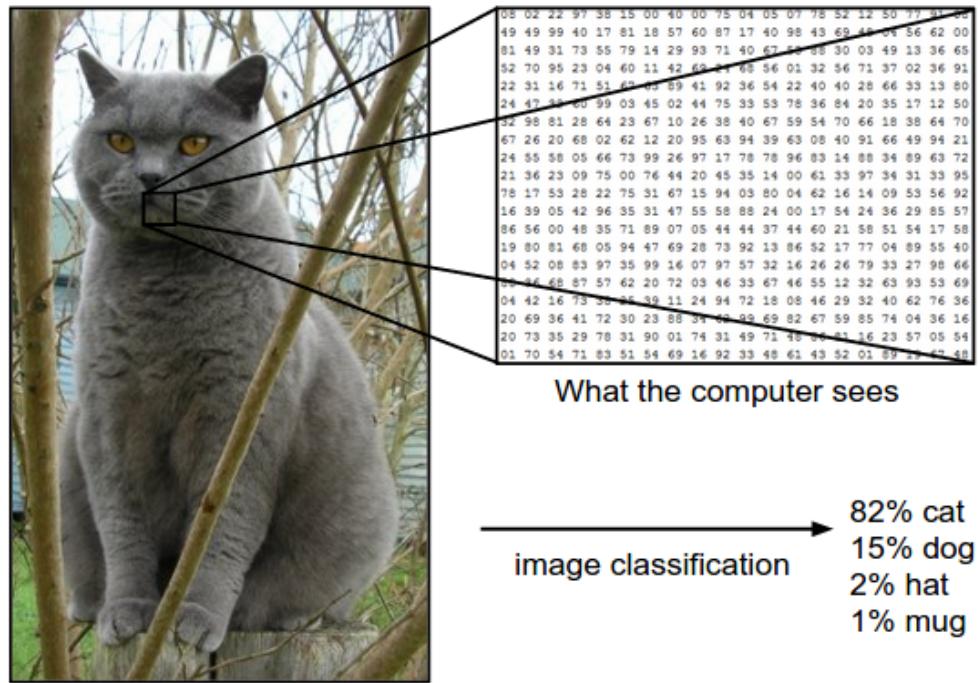


Figura 2.1: Descrizione visiva di ciò che il computer vede e del processo di classificazione [17].

Per introdurre le difficoltà della classificazione di immagini, basti pensare alle trasformazioni e variazioni che può subire un' immagine tra cui: variazione del punto di vista, scala, deformazione, variazione dell'illuminazione, occlusione e molte altre. Un buon modello di image classification deve essere *invariante* rispetto al prodotto vettoriale di tutte queste variazioni e, allo stesso tempo, mantenere *sensibilità* rispetto alle variazioni tra classi diverse di immagini [17]. Per ottenere ciò, dobbiamo fornire alla rete molti esempi appartenenti alle varie classi in modo che possa generalizzare e apprendere l'aspetto visivo di ogni classe. Chiaramente nel nostro contesto data-incremental avremo a disposizione, nei primi task, un numero molto limitato di esempi di training per classe e la corretta classificazione di nuovi esempi non è un compito banale.

### 2.1.1 CNNs e ResNet

Tipicamente, la classificazione di immagini, in tempi recenti, è affidata al Deep Learning attraverso le **reti neurali convoluzionali (CNNs)**. le CNNs sono diventate *mainstream* e si sono affermate come lo stato dell'arte nel campo dell'image recognition, a partire dal 2012 con l'architettura denominata Alex-Net, sviluppata da Krizhevsky et al. [21], che mostrò eccellenti performance in una challenge di visual recognition sul dataset ImageNet. Da quel momento, le reti convoluzionali ricevettero sempre più attenzioni nel campo della ricerca portando allo sviluppo di architetture sempre più performanti.

Fondamentalmente, le reti convoluzionali sono reti che sfruttano la **topologia** dei dati in input per trarne vantaggio e operano la convoluzione al posto di una generale moltiplicazione per matrice. Il loro funzionamento porta naturalmente a un'invarianza agli shift e alle trasformazioni dello

spazio, oltre a vari vantaggi quali la condivisione dei pesi e il numero dei parametri rispetto alle reti neurali *fully-connected*. Una rete convoluzionale è tipicamente una rete multi-strato composta, appunto, da layer di convoluzione, layer ReLU, layer di pooling e layer fully-connected arrangiati in una struttura composta da un input layer, vari hidden layers e un output layer (maggiori dettagli in fig.2.2). Particolarità di queste reti è che funzionano estraendo le feature dall'immagine; queste infatti sono apprese durante l'addestramento attraverso numerosi hidden layers che vanno pian piano ad incrementare la complessità e la capacità discriminativa delle feature apprese.

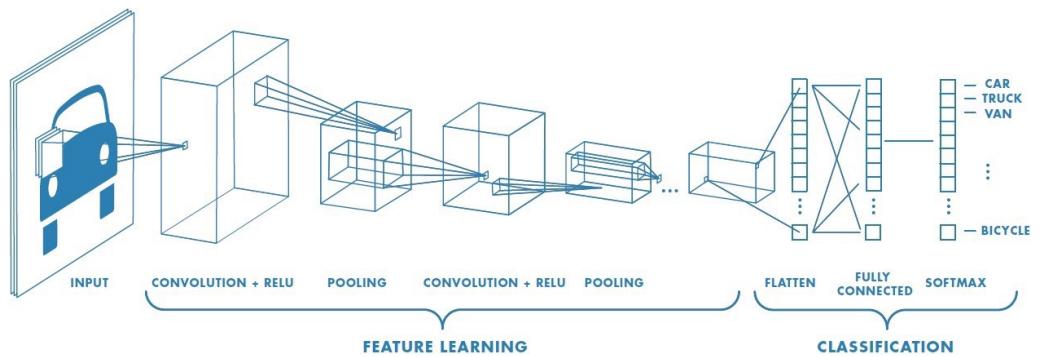


Figura 2.2: Rete convoluzionale nel caso della classificazione di immagini. In figura sono visibili anche le fasi tipiche e i blocchetti costituenti che solitamente vengono combinati per formare la rete. [33]

Nel 2015, una particolare rete convoluzionale denominata **ResNet** vinse la stessa challenge di Alex-Net nel 2015 con risultati sorprendenti (top 5-error 3.6% vs 17% di Alex-Net) grazie a una profondità e quindi un numero di layers molto grande, permessa dall'utilizzo delle **skip connection** (Fig. 2.3). In sintesi, il segnale (i dati elaborati) trasferito al livello successivo è aggiunto anche all'output di quelli precedenti, secondo uno schema definito. Questo meccanismo ha permesso di superare il problema della degradazione nelle reti profonde e ha reso le ResNet uno standard comune nelle applicazioni di Computer Vision odierne, oltre che essere l'architettura primaria presa in esame da questo elaborato.

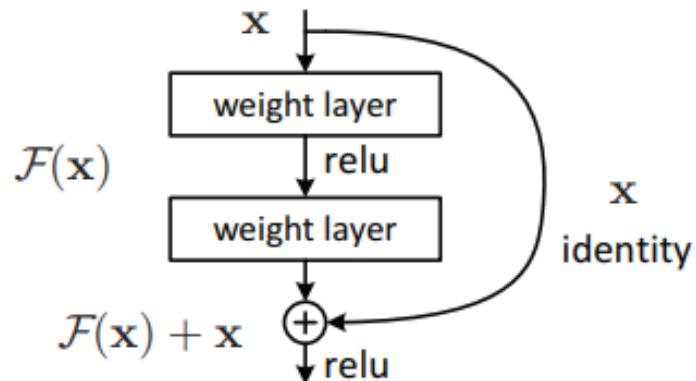


Figura 2.3: Esempio di un blocco residuale con skip connection. [13]

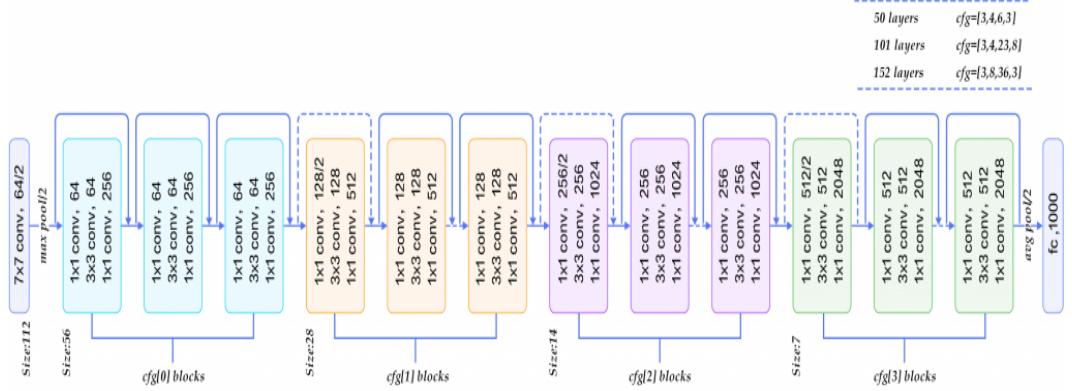


Figura 2.4: Composizione in cascata di una rete ResNet. Si notano i 4 macro-layers costituenti (con colori diversi) e la loro composizione in base a che tipo di resnet si fa riferimento [13].

## 2.2 Knowledge distillation per la Computer Vision

In questa sezione andremo a esplorare il concetto già anticipato di knowledge distillation e i lavori correlati nel contesto del riconoscimento di immagini e del rilevamento di oggetti. A partire dal 2015 infatti, con il lavoro di Hinton et al. [15], iniziarono ad affacciarsi i concetti e le metodologie che vanno sotto il termine di **distillation**. Nello specifico, in questo lavoro non siamo ancora nell'ambito dell'incremental learning ma la distillation viene usata per "comprimere" la conoscenza da un *ensemble* di modelli, spesso an-

che grandi, in un modello piccolo, computazionalmente più gestibile anche per applicazioni real-time e di cui è più facile fare il *deployment*.

L’idea di partenza per trasferire l’abilità di generalizzazione dal modello “ingombrante” a quello piccolo è di usare le probabilità delle classi prodotte dal modello grande come “*soft target*” per l’addestramento del modello piccolo. In sostanza si crea una struttura dove abbiamo un modello grande che ha il ruolo di *teacher* e un modello piccolo a cui viene trasferita conoscenza che svolge il ruolo dello *student* e l’apprendimento di quest’ultimo viene gestito da un iperparametro denominato “*temperatura*”.

In particolare, nel lavoro di Hinton et al. sono stati fatti dei test preliminari di riconoscimento immagini sul dataset dei numeri MNIST utilizzando una rete neurale grande con molte unità e una decisamente più piccola. Valutando il tasso di errori di classificazione sulla rete piccola senza distillation e confrontandolo con quello ottenuto con la distillation usando come teacher il modello grande, si è notata una netta differenza in positivo e verificato l’effettivo “trasferimento di conoscenza”. La distillation proposta da Hinton et al. è quella a cui si fa riferimento col nome di **knowledge distillation** ed è una sorta di “*output distillation*” che agisce sui *logits*. Tuttavia, l’output di una rete teacher molto performante non risulta significativamente diverso dal *ground truth*; dunque, trasferire solo l’output è simile a addestrare lo student con il *ground truth*, limitando di conseguenza le performance dell’output distillation.

Per fare un miglior uso delle informazioni contenute nella rete teacher, dalla distillation proposta da Hinton sono stati derivati e proposti anche vari approcci di **feature distillation** da affiancare o da utilizzare come alternativa alla knowledge distillation . Come anticipato nella sezione 2.1.1 dedicata alle reti convoluzionali, queste ultime, durante l’addestramento, apprendono

una rappresentazione delle features dell'immagine sotto forma di un vettore denominato *feature vector*. E' esattamente sui vettori delle features estratti dalla rete student e dalla rete teacher che andrà ad agire la feature distillation calcolando una distanza da aggiungere alla loss (maggiori dettagli in fig.2.5).

FitNets [31] (A.Romero et al.) è stata la prima architettura a proporre una feature distillation in cui lo student era incoraggiato a mimare i valori delle features nascoste della rete teacher, anche se l'incremento delle performance non è risultato molto significativo.

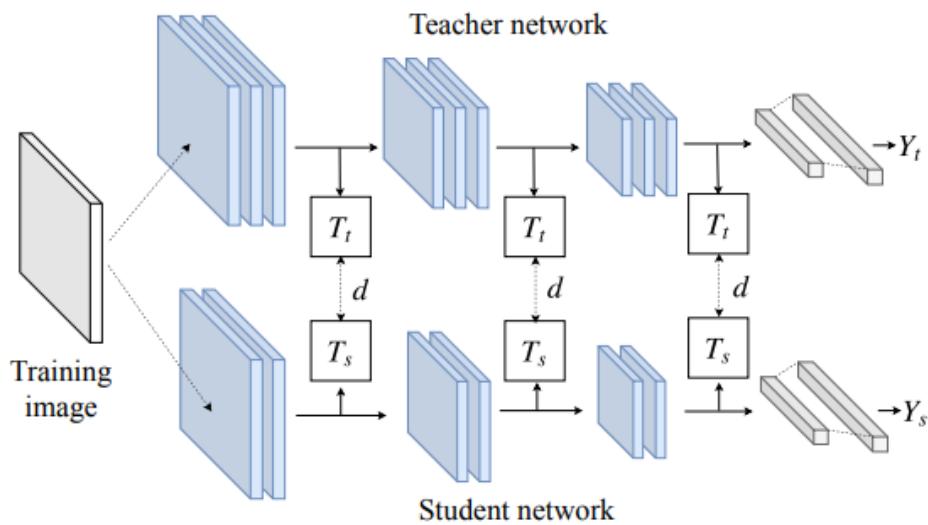


Figura 2.5: Training tramite feature distillation. La forma della trasformazione del teacher  $T_t$ , della trasformazione dello student  $T_s$  e della distanza  $d$  varia a seconda dei vari metodi così come anche i punti della rete dove viene applicata la distillation. [44]

Dopo FitNets, sono state proposte alcune varianti di feature distillation con metodi [44] basati sul trasformare le features in una rappresentazione a dimensione ridotta per poi essere trasferite allo student.

Anche gli approcci di knowledge distillation si sono evoluti col tempo e sono stati proposti vari lavori nel contesto della visione artificiale. Tra questi, nel lavoro di Chen et al. [6] si è indagato l'utilizzo della knowledge distillation (combinata a feature distillation) nel problema dell'object detection e in particolare nell'utilizzo di quest'ultima sui modelli di regressione per la predizione dei *bounding box*. Si rilevano anche lavori, come quello di Liu et al. [24], dove knowledge distillation è applicata nel campo della *semantic segmentation*. In questo lavoro si propone una *structured knowledge distillation* che combina una knowledge distillation *pixel-wise* con una distillation *pair-wise* che va a trasferire le relazioni tra coppie di pixel e una "holistic distillation" che si occupa di allineare reazioni ad alto livello tra le segmentation map generate dal modello teacher e dal modello piccolo student.

Infine, molto recentemente, la distillation è stata applicata anche all'architettura *ViT (Vision Transformer)* [9] nel lavoro di Touvron et al. [35]. Touvron et al. introducono un *distillation token* in input al transformer che andrà a interagire con gli embeddings sfruttando il meccanismo dell'attenzione tipico del Transformer [36]. In questo lavoro si va dunque a trasferire conoscenza da una rete convoluzionale complessa al ViT, ottenendo una rete competitiva e con un ottimo **tradeoff accuratezza-throughput**, che è ciò che si vuole per le applicazioni real-time.

Riassumendo, le tecniche di distillation sono presenti tutt'oggi in vari lavori nel campo di visione artificiale che costituiscono lo stato dell'arte e sono una componente fondamentale per l'efficacia della compressione dei modelli. Successivamente vedremo nel paragrafo 2.3.2 come, la metodologia e il

paradigma di knowledge distillation, con qualche accorgimento, verrà poi trasferito nel contesto dell'incremental learning.

Adesso andremo a vedere invece una recente evoluzione del paradigma di knowledge distillation che costituirà uno dei punti focali di questo lavoro di tesi.

### 2.2.1 Focal distillation

Un lavoro molto recente infatti, ad opera di Yan et al., “*Positive-Congruent Training: Towards Regression-Free Model Updates*” [41], introduce una variante interessante delle tecniche di distillation denominata **focal distillation**. Il lavoro di Yan et al. si focalizza sul contesto di aggiornamento dei modelli (intesi propriamente come cambio di architettura di apprendimento; es. passaggio da AlexNet a ResNet152). Anche in questo caso di studio vi è la presenza di una componente di forgetting nell’aggiornamento; passando al nuovo modello più elaborato infatti, si riduce il tasso di errore globale ma si introducono degli errori di classificazione a cui vecchio modello non era soggetto. La presenza del forgetting a seguito di un aggiornamento del modello rende questo studio molto affine al nostro lavoro in contesto data-incremental.

In questo lavoro vengono infatti definiti **negative flips** gli esempi classificati in maniera corretta dal modello vecchio e classificati erroneamente dal modello nuovo (vedesi Fig.2.6). Il lavoro di Yan et al. si concentra dunque sullo sviluppo di una metodologia per la riduzione del **NFR** (negative flips rate) e, allo stesso tempo, mantenere alta l’accuratezza. Per tali motivi, risulta molto rilevante e perfettamente adattabile al data-incremental learning.

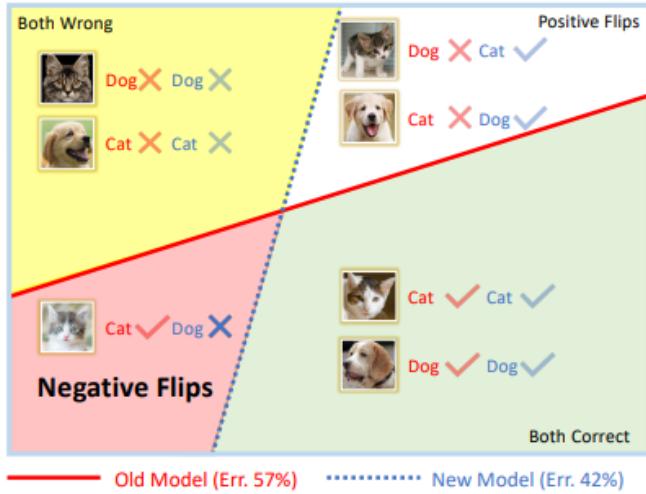


Figura 2.6: Quando si va ad aggiornare un classificatore vecchio (in rosso) in uno nuovo (blu) si vanno a correggere errori (in bianco) ma si introducono anche errori che il vecchio classificatore non faceva (in rosso, in basso a sinistra). Seppur il nuovo modello performa meglio, abbiamo la presenza di un fenomeno di regressione che potrebbe dare problemi [41].

Le tecniche di distillation cercano di spingere l'aggiornamento verso il modello vecchio riducendo allo stesso tempo il tasso di errore ma, ciò che si vuole nel contributo di Yan et al., è di mimare il modello vecchio *solo nei casi in cui è corretto in termini di predizioni*.

In particolare, in questo paper si arriva a proporre la seguente funzione obiettivo generale:

$$\min_w \mathcal{L}_{CE}(\phi^{new}, w) + \lambda \mathcal{L}^{Focal}(\phi^{new}, w; \phi^{old}) \quad (2.1)$$

Dove il primo termine rappresenta la cross-entropy standard per massimizzare le performance del nuovo modello sul training set, il secondo è la *focal distillation loss* che illustreremo a breve e  $\lambda$  è un iperparametro che determina il peso di questa loss.

La focal distillation loss è definita come:

$$\mathcal{L}^{Focal} = - \sum_{i=1}^N [\alpha + \beta * \mathbf{1}(\hat{y}^{old}(x_i) = y_i)] \mathcal{D}(\phi^{new}, \phi^{old}) \quad (2.2)$$

e penalizza una distanza  $\mathcal{D}$  tra gli output dei due modelli, pesata da una funzione filtro  $\mathcal{F} = \alpha + \beta * \mathbf{1}(\hat{y}^{old}(x_i) = y_i)$ . Il filtro applica un peso base  $\alpha$  per tutti gli esempi nel training set e un peso addizionale  $\beta$  agli esempi correttamente predetti dal modello nuovo.

La focal distillation loss porta dunque il nuovo modello verso il precedente solo per quanto riguarda le predizioni corrette e allo stesso tempo va a ridurre il tasso di errore con la cross entropy standard e un termine di distanza calcolata rispetto al modello vecchio. Si nota facilmente che quando  $\alpha = 1$  e  $\beta = 0$ , focal distillation si riduce alla normale distillation mentre quando  $\alpha = 0$  e  $\beta > 0$ , si va ad applicare la distillation con obiettivo solo gli esempi di training predetti correttamente dal modello vecchio.

Per quanto riguarda la distanza  $\mathcal{D}$ , sono proposte varie scelte tra cui la stessa knowledge distillation standard, la divergenza di KL [12] scalata con la temperatura e la distanza tra logits con quest'ultima che risulta in performance leggermente migliori.

La focal distillation proposta, adattata al nostro scenario data-incremental sarà valutata nel dettaglio nella sottosezione 4.2.3.5 del capitolo 4 degli esperimenti.

## 2.3 Class-Incremental Learning

In questa sezione vedremo nel dettaglio i lavori correlati che rientrano nel Class-Incremental Learning in quanto, come anticipato, la ricerca e la letteratura disponibile è principalmente concentrata su questo scenario. Di seguito vederemo i lavori e le metodologie applicate a modelli di Deep Learning (come le CNNs), nel contesto del Class-Incremental Learning, che sono stati il punto di partenza di questo lavoro ispirando gli esperimenti e alcune tecniche che verranno descritte successivamente in questo elaborato. Infine vedremo le metriche proposte in letteratura per il class-incremental learning; questo per comprendere perchè queste non risultino adeguate allo scenario Data-Incremental. Nel capitolo 3 infatti, verranno proposte delle nuove metriche adatte al data-incremental learning.

### 2.3.1 Approcci

Per quanto riguarda la letteratura sugli approcci di Class-Incremental learning, possiamo andare a distinguere tre importanti macrocategorie [27]: gli approcci basati sulla **regolarizzazione**, gli approcci basati su tecniche di **rehearsal** (*exemplar-based*) e gli approcci meno comuni di **bias-correction**. Gli approcci basati sulla regolarizzazione si possono a loro volta distinguere in approcci di *regolarizzazione dei pesi* e approcci di *regolarizzazione dei dati*. La prima classe di approcci si concentra nel prevenire variazioni considerevoli del valore dei pesi ritenuti importanti per le performance sui task precedenti. In sostanza, si assume che i parametri della rete siano indipendenti e si calcola un valore di "importanza" relativo al task precedente. Tra gli approcci di regolarizzazione dei pesi vedremo, nella sottosezione 2.3.3, il metodo più importante in letteratura, denominato EWC.

La categoria che si basa sulla regolarizzazione dei dati invece mira a prevenire variazioni troppo grandi tra i logits prodotti dalla rete ed è basata sulla metodologia knowledge distillation, già introdotta nella sezione 2.2. Capostipite di questi approcci è LwF che approfondiremo nella sottosezione 2.3.2.

La categoria di approcci basata sul rehearsal invece utilizza gli **exemplar** per mitigare il problema del forgetting. Gli exemplar sono degli esempi provenienti dai task precedenti che vengono tenuti in memoria e riutilizzati assieme agli esempi del task attuale per fare il training e "rinfrescare" la memoria del modello sui task precedenti (è una sorta di replay parziale atto a mitigare il forgetting). L'uso degli exemplar in contesto di apprendimento incrementale è stato proposto per la prima volta dall'approccio iCaRL, che vederemo qui di seguito nella sottosezione 2.3.4.

Chiaramente, in un contesto applicativo reale, l'insieme degli exemplar deve essere molto piccolo altrimenti viene meno uno dei vantaggi principali degli approcci di apprendimento incrementale ovvero il non richiedere un dataset a priori. Inoltre, devono essere selezionati, possibilmente, esempi significativi e rappresentativi dai task passati in modo da ottenere un miglior impatto sulle performance. C'è da considerare poi che l'uso degli exemplar non è possibile nel caso già anticipato di vincoli di privacy piuttosto stringenti.

Ipotizzando di non avere forti vincoli di privacy, vediamo adesso le principali politiche di selezione degli exemplar presenti in letteratura [5] e che verranno provate nel contesto data-incremental nel capitolo 4 riguardante gli esperimenti. Le politiche di selezione sono principalmente quattro e si dividono in:

- **Random-based:** vengono scelti, su base randomica, un numero di esempi uniforme per ogni classe.
- **Herding-based:** è la politica di selezione principale di iCaRL in cui viene selezionato un insieme rappresentativo di esempi per una determinata classe in modo che sia una buona approssimazione della media delle features di quella stessa classe.
- **Entropy-based:** vengono selezionati gli esempi di una classe in base all'entropia . Sostanzialmente, dato un esempio di training, al suo output della rete viene applicato il *softmax* e l'entropia di questa distribuzione in uscita ci dice quanta incertezza di classificazione è presente. Più alta (o più bassa, nell'approccio inverso) è l'entropia e maggiore è la probabilità che l'esempio sarà selezionato.
- **Distance-based:** vengono selezionati esempi di una classe in base a quanto sono vicini ai bordi di decisione. I bordi di decisione sono regioni nello spazio del problema dove l'output del classificatore è ambiguo (visualizzazione in figura 2.7). L'assunzione di base è quindi che, più sono vicini, in termini di distanza, (o lontani, nell'approccio inverso) e più sono significativi e rappresentativi. Questi esempi, nel caso in cui lo spazio delle feature e i bordi di decisione non variano molto, rappresentano *esempi di definizione dei bordi*.

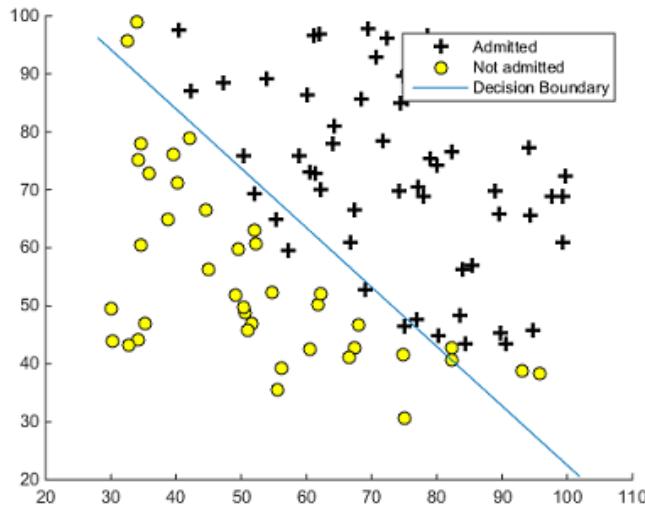


Figura 2.7: Bordo di decisione a singola linea nel caso semplice di regressione logistica.

L’ultima categoria, quella dei metodi di bias-correction, si occupa del problema dell’inclinazione della rete verso le classi apprese nel task più recente. Ciò è principalmente causato dal fatto che, al termine del training, la rete ha visto molti esempi delle classe appartenenti all’ultimo task ma nessun esempio (o pochi, se vengono usati gli exemplar) delle classi dei task precedenti. Di questi metodi non ne vedremo nessuno nel dettaglio in quanto non ne sono stati provati nel nostro contesto data-incremental.

Riassumendo, per visualizzare la divisione tra gli approcci, è utile lo schema descrittivo di *FACIL*(*Framework for Analysis of Class-Incremental Learning*), consultabile in fig. 2.8.

*FACIL* è un framework sviluppato in concomitanza con il paper "**Class-incremental learning: survey and performance evaluation on image classification**"(M. Masana et al.) [27] a cui questo lavoro è fortemente corre-

lato. Il framework<sup>1</sup>, strutturalmente complesso, inoltre racchiude gran parte delle metodologie descritte in precedenza e che andremo ad approfondire, quali Lwf, EWC e iCaRL, ma anche molti altri approcci ed è dunque una sorta di compendio dello stato dell'arte per quel che riguarda il class-incremental learning con 12 metodi allo stato dell'arte e 3 baseline.

Adesso andremo ad approfondire gli approcci che, presi e adattati da FACIL, sono stati valutati nel contesto del Data-Incremental Learning e che ritroveremo nella sottosezione 4.2.3 del capitolo 4 degli esperimenti con i relativi risultati.

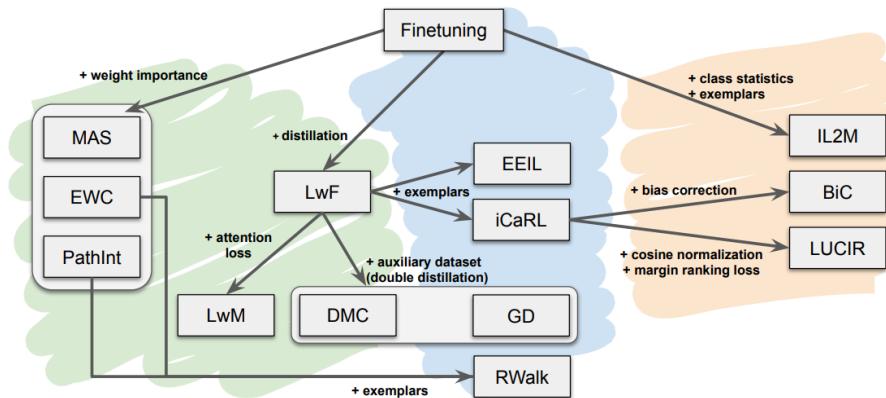


Figura 2.8: Struttura e approcci implementati nel framework FACIL. Sono distinte tre categorie principali: i metodi exemplar-free e basati sulla regolarizzazione (in verde), i metodi basati sul rehearsal (in blue) e i metodi basati sul rehearsal con un meccanismo di correzione dei bias esplicito (in arancione). I metodi che sono in relazione sono racchiusi in un riquadro stondato [27].

<sup>1</sup><https://github.com/mmasana/FACIL>

### 2.3.2 Learning Without Forgetting

Nel 2016, la distillation viene infatti per la prima volta inserita nel contesto dell'Incremental Learning dal paper **“Learning Without Forgetting”** di Z. Li e D.Hoiem [23] che segna un importante contributo, soprattutto nel campo del class-incremental learning. In questo paper viene introdotta una metodologia per **trasferire conoscenza** da una rete all'altra per vincolare quest'ultima a non dimenticare le conoscenze acquisite sui task precedenti ed è uno dei primi grandi passi in avanti per quanto riguarda l'apprendimento incrementale su reti convoluzionali.

In pratica, data una rete CNN con dei parametri condivisi  $\theta_s$ , e parametri specifici di un task  $\theta_o$ , l'obiettivo di questo algoritmo è aggiungere dei parametri specifici  $\theta_n$  del nuovo task e imparare parametri che funzionano bene sul nuovo e sul vecchio task usando immagini e labels provenienti solo dal nuovo task. L'apprendimento avviene grazie a una *cross-entropy loss* modificata con l'aggiunta di una *knowledge distillation loss* che si applica per tutti i task precedenti.

```

LEARNINGWITHOUTFORGETTING:
Start with:
     $\theta_s$ : shared parameters
     $\theta_o$ : task specific parameters for each old task
     $X_n, Y_n$ : training data and ground truth on the new task
Initialize:
     $Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$  // compute output of old tasks for new data
     $\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$  // randomly initialize new parameters
Train:
    Define  $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$  // old task output
    Define  $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$  // new task output
     $\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \underset{\theta_s, \theta_o, \theta_n}{\operatorname{argmin}} \left( \lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \right)$ 

```

Figura 2.9: Pseudocodice della procedura di Learning Without Forgetting [23].

La procedura è illustrata in figura 2.9 dove:

$$\mathcal{L}_{old}(y_o, \hat{y}_o) = -H(y'_o, \hat{y}'_o) = -\sum_{i=1}^l y'^{(i)}_o \log \hat{y}'^{(i)}_o \quad (2.3)$$

$$y'^{(i)}_o = \frac{(y^{(i)}_o)^{1/T}}{\sum_j (y^{(j)}_o)^{1/T}}, \hat{y}'^{(i)}_o = \frac{(\hat{y}^{(i)}_o)^{1/T}}{\sum_j (\hat{y}^{(j)}_o)^{1/T}}$$

con  $l$  numero delle label,  $y'^{(i)}_o$ ,  $\hat{y}'^{(i)}_o$  versioni modificate delle probabilità passate e correnti  $y^{(i)}_o$  e  $\hat{y}^{(i)}_o$ , e  $\mathcal{R}$  nella procedura è un termine di regolarizzazione. In sostanza, quello che si ottiene è che gli outputs relativi a task vecchi vengono vincolati a mantenere valori simili al modello precedente mentre, allo stesso tempo, si impara il task corrente.

Nel paper di Z. Li e D.Hoiem [23] viene usato il parametro temperatura  $T = 2$  in seguito ad una grid search per incrementare il peso dei valori più piccoli dei logits e incoraggiare la rete a codificare meglio similarità tra classi. Seppur questa metodologia sembrerebbe intuitivamente non funzionare bene in uno scenario data-incremental in quanto va a vincolare in parte i logits al task precedente, nella sottosezione 4.2.3.4 del capitolo 4 degli esperimenti vedremo come in realtà la knowledge distillation loss si va a bilanciare piuttosto bene con la loss data dalla cross-entropy riportando buoni risultati.

### 2.3.3 Elastic Weight Consolidation

L'altra metodologia già anticipata di particolare interesse nell'ambito della regolarizzazione, va sotto il nome di **EWC**, *Elastic Weight Consolidation* (J.Kirkpatrick et al.) [19].

Nel lavoro di J.Kirkpatrick et al., viene definita l' "importanza" dei parametri relativa a un determinato task considerando l'addestramento di una rete neurale da un punto di vista probabilistico. Ottimizzare i parametri della rete significa dunque trovare i loro valori più probabili dato un dataset  $\mathcal{D}$ , ovvero  $p(\theta|\mathcal{D})$ . Dalla regola di Bayes si ottiene:

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D})$$

che, dividendo i dati per due task  $\mathcal{D}_A$  e  $\mathcal{D}_B$ , possiamo riscrivere come:

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B)$$

dove  $\log p(\mathcal{D}_B|\theta)$  corrisponde semplicemente alla loss sul task B. Tutte le informazioni sul task A rimangono dunque condensate nella probabilità a posteriori  $\log p(\theta|\mathcal{D}_A)$  che contiene informazioni sui parametri *importanti* per il task A. Il calcolo esatto della probabilità  $\log p(\theta|\mathcal{D}_A)$  è intrattabile; dunque, seguendo il lavoro sull'approssimazione di Laplace di Mackay [26], questa viene approssimata con una distribuzione gaussiana con media data dai parametri  $\theta_A^*$  e una precisione diagonale data dalla diagonale della matrice di Fisher  $\mathcal{F}$ .

La matrice di Fisher ha tre interessanti proprietà tra cui: (a) è equivalente alla derivata seconda della loss vicino a un minimo, (b) può essere calcolata a partire solo dalla derivata di primo ordine ed è quindi facile da calcolare

anche per modelli grandi e (c) è garantita essere semidefinita positiva. Con questa approssimazione si arriva alla loss da minimizzare nell'approccio di EWC:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} \mathcal{F}_i(\theta_i - \theta_{i,A}^*)^2 \quad (2.4)$$

dove  $\mathcal{L}_B(\theta)$  è la loss calcolata solo sul task B,  $\lambda$  è un iperparametro che determina quanto è importante il task precedente A rispetto al corrente B e  $i$  indica le label. Muovendosi su un terzo task C, si possono accumulare i valori della fisher e quindi cercare di tenere i parametri della rete vicini alle regioni ottimali sia per il task A che per il task B.

Nel cervello umano, le connessioni sinaptiche permettono l'apprendimento continuo riducendo la *plasticity* delle sinapsi che sono vitali per il compito precedentemente imparato. Nel lavoro di J.Kirkpatrick et al. l'approccio proposto esegue un'operazione simile nelle reti neurali vincolando i parametri *importanti* a rimanere vicini ai loro vecchi valori. EWC è fondamentalmente un algoritmo che rallenta il learning su certi pesi in base a quanto questi risultano importanti per il task precedente e che quindi, in sostanza, riduce *selettivamente* la *plasticity* dei pesi della rete (maggiori dettagli visibili in Fig. 2.10).

Importante concetto alla base di EWC è che non esiste una configurazione fissa di parametri  $\theta$  della rete ma svariate configurazioni di  $\theta$  possono risultare nelle stesse performance [34] [14]. Configurazioni "ottimali" multiple dei parametri fanno sì che sia possibile trovare una soluzione per il task B che risulti vicina alla soluzione precedentemente trovata per il task A.

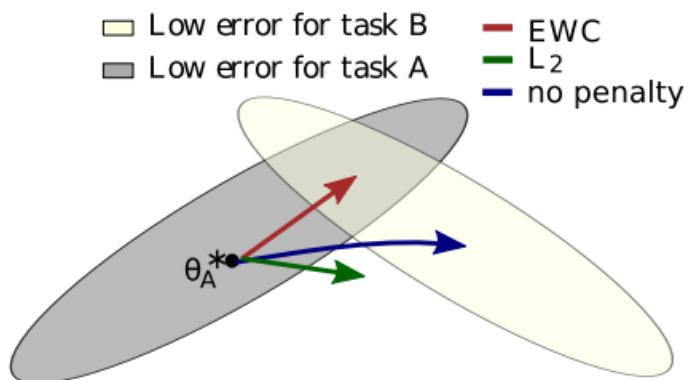


Figura 2.10: Elastic Weight Consolidation si assicura che un task A sia ricordato mentre si addestra la rete per un task B. Le traiettorie di training sono illustrate in modo schematico tramite delle frecce in uno spazio dei parametri che ha regioni di rilevanza per le performance (grigio per il task A, crema per il task B). Dopo aver fatto il training sul primo task i parametri si trovano su  $\theta_A^*$ ; se andiamo a fare passi in base al gradiente usando come training il task B (freccia blu), andremo a minimizzare la loss del task B ma dimenticheremo ciò che abbiamo imparato col task A. Se vincoliamo i pesi con gli stessi coefficienti(freccia verde) arriviamo a ricordare il task A ma senza imparare il task B. EWC (freccia rossa) trova invece il giusto compromesso valutando quanto i pesi sono importanti per il task A [19].

Mentre si sta imparando il task B, la loss di EWC (2.4) salvaguarda le performance sul task A vincolando i parametri a rimanere in una regione di basso errore per A. Il vincolo è rappresentato da una penalità quadratica che risulta più grande per i parametri che influiscono di più sulle performance del task A.

La metodologia EWC è stata applicata nel lavoro di J.Kirkpatrick et al. [19] al riconoscimento di numeri scritti a mano del dataset MNIST ma si è dimostrata efficace anche in contesti di Reinforcement Learning, come i giochi dell'Atari 2600.

In questo lavoro di tesi, EWC è stato preso in esame per valutare se, anche in un contesto data-incremental, riuscisse a vincolare i parametri in modo da giungere a configurazioni ottimali valide per più task. Gli esperimenti relativi a questa metodologia, adattata al problema data-incremental, sono consultabili nella sottosezione 4.2.3.2 del capitolo 4 relativo agli esperimenti.

### 2.3.4 ICaRL

L'architettura **iCaRL** [30], *Incremental Classifier and Representation Learning*, (SA Rebuffi et al.) è un importante contributo allo stato dell'arte ed è il primo approccio che ha introdotto gli exemplar nel contesto del class-incremental learning. I contributi principali dell'architettura iCaRL, come approfondiremo in seguito, sono:

- un algoritmo efficace di costruzione dell'insieme degli exemplar tramite *herding*, approssimando dunque nel miglior modo possibile la media reale delle classi nello spazio delle features.
- un classificatore denominato *Nearest-Mean-of-Exemplars (NME)* che usa la media delle features dell'insieme degli exemplar per la classi-

ficazione, a differenza di un classificatore di tipo *Nearest Class Mean* (*NCM*).

- una variante della knowledge distillation loss per trasferire conoscenza da un rete vecchia a una nuova, incrementata (vedremo nel dettaglio in seguito).

La particolarità di iCaRL è infatti la capacità di apprendere un buon classificatore e allo stesso tempo una rappresentazione delle features. Come anticipato, iCaRL usa una strategia di classificazione di tipo NME “nearest-mean-of-exemplars”. Per predire una label  $y^*$ , per una nuova immagine  $x$ , calcola un vettore prototipo per ogni classe osservata finora  $u_1, \dots, u_t$ , dove  $u_y$  è la media dei vettori di feature di tutti gli exemplar per la classe  $y$ , dove per vettore di feature si intende la rappresentazione appresa dalla rete e introdotta nella sottosezione 2.1.1 di questo capitolo. A questo punto calcola il vettore di feature  $\varphi(x)$  dell’immagine da classificare e assegna la label del prototipo precalcolato più simile secondo la formula:

$$y^* = \underset{y=1, \dots, t}{\operatorname{argmin}} \|\varphi(x) - u_y\| \quad (2.5)$$

La metodologia di classificazione NME, ispirata dalla classificazione *nearest-class-mean* (*NCM*), rende, nel contesto del class-incremental learning, il classificatore robusto ai cambiamenti delle rappresentazioni delle features in quanto i prototipi delle classi cambiano automaticamente al variare delle rappresentazioni. Inoltre vengono usati gli exemplar per calcolare i prototipi in modo tale da dover tenere in memoria solo quelli invece che tutto il dataset precedente, come nel caso di *NCM*.

Per quanto riguarda l'apprendimento, ognqualvolta iCaRL ottiene nuovi dati  $X^s, \dots, X^t$  per delle nuove classi  $s, \dots, t$ , aggiorna la sua routine di estrazione features (come in fig. 2.11) e l'insieme degli exemplar. La rappresentazione viene aggiornata nel modo seguente: inanzitutto iCaRL costruisce un insieme di training "aumentato" che contiene sia gli esempi di training attualmente disponibili sia gli exemplar salvati in memoria. Successivamente, la rete corrente è valutata per ogni esempio e gli outputs delle precedenti classi sono tenuti in memoria. Infine la rete e i parametri sono aggiornati minimizzando una loss che, per ogni nuova immagine, incoraggia la rete a dare in output la label della classe corretta (classification loss), e, per le classi vecchie, viene incoraggiata a riprodurre gli scores salvati nel passo precedente (distillation loss).

Come abbiamo anticipato, gli exemplar vengono selezionati da iCaRL tramite una policy di **herding** dove quindi si va a costruire un insieme rappresentativo di esempi che meglio approssima la distribuzione dei dati e la loro media delle features. Ogni volta che iCaRL incontra nuove classi, aggiusta il suo insieme degli exemplar distribuendo la memoria equamente tra le classi in modo tale che, se  $K$  è la memoria a disposizione per gli exemplar e  $t$  sono le classi osservate finora, iCaRL userà  $m = K/t$  exemplars per ogni classe. L'algoritmo di aggiornamento dell'insieme degli exemplars invece parte al primo task, selezionando e salvando gli exemplars iterativamente fino a raggiungere il target  $m$ . Nei task successivi, ad ogni iterazione un esempio in più del training set viene aggiunto all'insieme degli exemplars; in particolare viene aggiunto l'esempio che causa una media del vettore delle features, calcolato su tutti gli exemplars, che meglio approssima la media del vettore delle features calcolato su tutti gli esempi di training.

**Algorithm 3** iCaRL UPDATEREPRESENTATION

---

```

input  $X^s, \dots, X^t$  // training images of classes  $s, \dots, t$ 
require  $\mathcal{P} = (P_1, \dots, P_{s-1})$  // exemplar sets
require  $\Theta$  // current model parameters
    // form combined training set:

$$\mathcal{D} \leftarrow \bigcup_{y=s, \dots, t} \{(x, y) : x \in X^y\} \cup \bigcup_{y=1, \dots, s-1} \{(x, y) : x \in P^y\}$$

    // store network outputs with pre-update parameters:
    for  $y = 1, \dots, s-1$  do
         $q_i^y \leftarrow g_y(x_i)$  for all  $(x_i, \cdot) \in \mathcal{D}$ 
    end for
    run network training (e.g. BackProp) with loss function

$$\ell(\Theta) = - \sum_{(x_i, y_i) \in \mathcal{D}} \left[ \sum_{y=s}^t \delta_{y=y_i} \log g_y(x_i) + \delta_{y \neq y_i} \log(1 - g_y(x_i)) \right. \\ \left. + \sum_{y=1}^{s-1} q_i^y \log g_y(x_i) + (1 - q_i^y) \log(1 - g_y(x_i)) \right]$$

    that consists of classification and distillation terms.

```

---

Figura 2.11: Pseudocodice per l’aggiornamento incrementale della rappresentazione [30].

Procedendo in questa maniera, l’insieme che si forma è una *lista con priorità* in cui l’ordine è cruciale e gli elementi in cima sono più importanti. Quindi quando dobbiamo andare ad aggiustare le dimensioni dell’insieme e scartare degli exemplars, riducendone il numero da  $m'$  a  $m$ , semplicemente si scartano gli exemplars  $p_{m+1}, \dots, p_{m'}$ , mantenendo  $p_1, \dots, p_m$ .

L’architettura iCaRL, nel lavoro di SA Rebuffi et al., è stata testata sul dataset CIFAR-100 e ImageNet con ResNet32 dando prova del valore degli exemplars e mostrando la capacità di imparare incrementalmente per lunghi

periodi dove altri metodi invece tendono a perdere efficacia facilmente.

La metodologia di selezione e gestione degli exemplar fatta da iCaRL sarà importante in questo lavoro di tesi e verrà effettuato uno studio approfondito sugli exemplar in contesto data-incremental nella sottosezione 4.2.2 del capitolo 4 degli esperimenti. Nello stesso capitolo, nel paragrafo 4.2.3.3 verranno proposti gli esperimenti di data-incremental learning utilizzando l'approccio iCaRL e valutando delle sue varianti.

### 2.3.5 Metriche per Class-incremental Learning

In questa sezione andremo a vedere le metriche sviluppate in letteratura per valutare gli approcci di apprendimento incrementale. Dato che il focus principale della ricerca negli anni è stato sullo scenario di class-incremental learning, la quasi totalità delle metriche proposte in letteratura fanno riferimento a questo contesto.

Poichè l'obiettivo è apprendere in modo continuo nuovi task preservando le conoscenze relative ai precedenti, le performance di un algoritmo di incremental learning devono essere valutate sia sui task precedenti che sul task corrente nella speranza che questo possa riflettere il suo comportamento anche sui task futuri non ancora visti. Chiaramente una metrica banale e immediata ma comunque significativa in questo contesto è l'**accuracy media** che ci dà un'idea iniziale dell'efficacia dell'algoritmo e permette un diretto paragone con l'accuratezza dei modelli di joint training. Oltre a questa prima metrica, in letteratura [5] si individuano due componenti fondamentali che devono essere quantificate: il **forgetting**, ovvero quanto un algoritmo dimentica di ciò che ha imparato nel passato, e l'**intransigence**, cioè la difficoltà di un algoritmo nell'imparare nuovi compiti.

Intuitivamente, se un modello viene fin troppo regolarizzato sui task precedenti per preservare la conoscenza, dimenticherà meno ma avrà un'intransigence piuttosto alta. Se, al contrario, c'è poca regolarizzazione, avremo un'intransigence bassa ma il modello soffrirà del catastrophic forgetting. Idealmente vogliamo un modello che soffra poco di entrambi questi fattori ma, in contesti reali, spesso si osserva una correlazione negativa tra forgetting e intransigence. Questa correlazione negativa rispecchia proprio lo *stability-plasticity dilemma* che è stato anticipato nel capitolo 1.

Solitamente quello che si fa è dunque individuare una configurazione tale per cui si ha un buon *trade-off* tra forgetting e intransigence e un algoritmo che non risulti in una correlazione negativa fin troppo grande. Vediamo adesso nel dettaglio queste metriche e come è stato quantificato il forgetting e l'intransigence.

**Average accuracy:** poniamo  $a_{k,j} \in [0, 1]$  come l'accuratezza (frazione delle immagini correttamente classificate) valutata sul test set del  $j$ -esimo task (con  $j \leq k$ ) dopo aver addestrato la rete incrementalmente dal task 1 al task  $k$ . Per calcolare  $a_{k,j}$ , lo spazio degli output è fatto da  $y^j$  o  $\cup_{j=1}^k y^j$  a seconda se siamo in un architettura di output *single-head* (come nel caso di questo lavoro) o *multi-head* (comune nel contesto del class-incremental).

Definiamo l'accuratezza media al task  $k$  come:

$$A_k = \frac{1}{k} \sum_{j=1}^k a_{k,j} \quad (2.6)$$

Più alta è questa accuratezza, migliore sarà il classificatore ottenuto ma questo non ci fornisce nessuna informazione riguardo al profilo dell'algoritmo di IL per quel che riguarda il forgetting e dell'intransigence.

**Forgetting:** in letteratura si definisce, nel contesto del class incremental, il forgetting per un particolare task (o label) come la differenza tra la massima *knowledge* ottenuta su un determinato task attraverso il processo fatto di apprendimento e la knowledge che il modello ha attualmente a riguardo. Questa metrica ci può dare una stima di quanto il modello, nello stato attuale, ha dimenticato del task in questione. Formalmente, per un problema di classificazione, si quantifica dunque il forgetting per il  $j$ -esimo task dopo che il modello è stato addestrato incrementalmente fino al task  $k > j$ , come:

$$f_j^k = \max_{i \in [1, \dots, k-1]} a_{i,j} - a_{k,j}, \forall j < k \quad (2.7)$$

Da questa definizione si nota che  $f_j^k \in [-1, 1]$  ed è definito per  $j < k$  in quanto siamo interessati a quantificare il forgetting dei task precedenti. In questa formula, al posto di max, si può usare l'*expectation* e sono presenti varie varianti in letteratura. Inoltre, normalizzando rispetto il numero dei task visti precedentemente, il forgetting medio al task  $k$ -esimo si può scrivere come:

$$F_k = \frac{1}{k-1} \sum_{j=1}^{k-1} f_j^k \quad (2.8)$$

Dove, chiaramente, valori più bassi di questo termine  $F_k$  indicano un minor forgetting sui task precedenti.

**Intransigence:** in letteratura si definisce intransigence la difficoltà del modello di apprendere nuovi task. L'effetto dell'intransigence è più pronunciato nel contesti di class-incremental learning a singola head, specialmente in assenza di dati precedenti, poiché ci si aspetta che il modello differenzi il task corrente dai precedenti. Vari esperimenti hanno mostrato in letteratura come l'uso degli *exemplars* riesca efficacemente a migliorare l'intransigence.

Poiché si vuole quantificare la difficoltà a imparare, viene fatto una comparazione con il modello standard di classificazione, che ha accesso a tutti i datasets dei vari task in ogni momento. Viene addestrato dunque un modello di riferimento con il dataset  $\bigcup_{i=1}^k \mathcal{D}$  e valutata la sua accuratezza  $a_k^*$  sul test set del  $k$ -esimo task. Poi, l'intransigence viene definita come:

$$I_k = a_k^* - a_{k,k} \quad (2.9)$$

dove  $a_{k,k}$  denota l'accuratezza sul  $k$ -esimo task quando il modello è addestrato fino al task  $k$  in modo incrementale. E' evidente che  $I_k \in [-1, 1]$  e che minore è questo valore e migliore è il modello che abbiamo ottenuto.

Il modello di riferimento può essere definito in base alla facilità con cui si può ottenere. In situazioni dove risulta particolarmente complesso e costoso, si può usare un'approssimazione al suo posto.

# Capitolo 3

## Data Incremental Learning

In questa sezione entreremo nel vivo del contesto di lavoro di questa tesi con una definizione formale del problema, la definizione di metriche *ad hoc* per questo caso specifico e la proposta di una metodologia di rilievo specifica. Sia le metriche presentate che la metodologia proposta sono il contributo centrale di questa tesi, non essendo presenti attualmente lavori simili in letteratura. Altro contributo importante è ovviamente costituito dagli esperimenti svolti (capitolo 4), in quanto tutte le metodologie approfondite nel capitolo precedente sono state provate e valutate per la prima volta in questo caso di studio.

### 3.1 Definizione del problema

Come anticipato, la particolarità che distingue lo scenario Data-incremental dallo scenario Class-Incremental è che in questo caso specifico le classi sono **fissate** e i dati di partenza per ogni classe sono molto *ridotti*. Si ha dunque un flusso di dati appartenenti a varie classi come in fig.3.1 mentre, nel class-

incremental learning, abbiamo da subito una quantità consistente di dati per una classe e un flusso nel tempo di nuovi dati di nuove classi.

Formalmente, i dati per l'apprendimento incrementale arrivano sotto forma di *chunks* di piccola dimensione. Questo comportamento viene simulato dividendo equamente il dataset  $\mathcal{D}$  in esame in **splits/tasks**  $\mathcal{D}_i$  tali che:  $\mathcal{D} = \cup_{i=1}^n \mathcal{D}_i$ . Gli esempi negli splits inoltre sono equamente distribuiti tra le classi. Al tempo di training  $t = 1$ , avremo a disposizione  $\mathcal{D}_1$  mentre, al tempo  $t = 2$ , avremo a disposizione  $\mathcal{D}_2$  ma  $\mathcal{D}_1$  non sarà più disponibile per il training e così via.

La sfida del data-incremental learning risiede proprio in questo: ottenere un modello di classificazione di partenza con pochi dati per classe e andarlo successivamente a migliorare con nuovi dati di training che diventeranno disponibili, potendo **dimenticare e scartare** i dati precedenti. Naturalmente, non avendo in memoria i dati precedenti, non viene fatto il joint training ma, per  $t = 2$  si continua l'addestramento con i nuovi dati a partire dal modello  $\mathcal{M}_1$  generato a seguito del training al tempo  $t = 1$ .

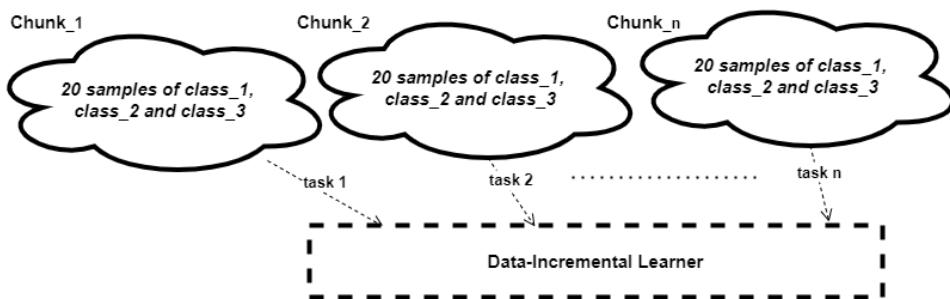


Figura 3.1: Struttura dell'apprendimento data-incremental nel caso specifico di 3 classi, chunks formati da 20 esempi per classe e  $n$  splits.

Anche nel caso data-incremental è rilevabile, come vedremo nel capitolo 4 dedicato agli esperimenti, una componente di *forgetting*. In questo contesto il forgetting è meno “catastrofico” rispetto al class-incremental learning ma comunque presente e da non sottovalutare. Infatti, nel contesto class-incremental, il forgetting è dato propriamente da un sovrapporsi delle rappresentazioni della rete all’arrivo di nuove classi mentre, nel nostro caso di studio, più a una generalizzazione del modello sui nuovi dati. Il modello andrà così a perdere la capacità di classificare correttamente alcuni esempi specifici su cui magari il modello precedente era più “specializzato”, data la scarsità dei dati di training. Inoltre, data la natura del training, l’architettura di rete per le metodologie di data-incremental learning rimane fissa con singola *head* così come sarà fisso il validation e il test set per ogni task.

## 3.2 Metriche per Data-Incremental Learning

Le metriche descritte nella sottosezione 2.3.5 capitolo 2 sono metriche specifiche dello scenario class-incremental e dunque è stato necessario un *rethinking* di quest’ultime e una nuova formulazione per il nostro scenario. Vediamo adesso nel dettaglio le metriche per il data-incremental learning che verranno usate poi nel capitolo 4 degli esperimenti per valutare le performance dei vari approcci provati.

**Average accuracy:** l’adattare l’accuratezza media al contesto Data-incremental è piuttosto intuitivo. Poniamo dunque  $a_{k,s} \in [0, 1]$  come l’accuratezza (frazione delle immagini correttamente classificate) valutata sul test set che si ottiene addestrando la rete incrementalmente fino al task  $k$  con *seed*

$= s$ . Il seed è un valore che determina lo split randomico del dataset e quindi, per diversi valori di  $s$ , gli splits saranno composti da immagini diverse (pur mantenendo lo stesso numero di immagini per classe). Qui chiaramente l'accuracy viene calcolata su tutte le classi, a differenza del Class-incremental learning con la configurazione *multi-head*.

L'accuracy media al task  $k$ , indipendente dalla composizione degli splits, sarà dunque data da:

$$A_k = \frac{1}{N} \sum_{i=1}^N a_{k,s_i} \quad (3.1)$$

dove  $N$  indica il numero di seed valutati (nei nostri esperimenti  $N = 5$ ). Più è alto questo valore, migliore sarà il classificatore addestrato incrementalmente ma questo non ci dice nulla riguardo al fenomeno del forgetting.

**Forgetting:** per quanto riguarda il forgetting introdotto per il caso class-incremental (2.7), la formulazione deve chiaramente essere cambiata del tutto in quanto la formula definita nel capitolo precedente per il class-incremental learning è altamente specifica e inadatta per questo contesto. Andiamo adesso dunque a definire uno dei primi importanti contributi di questo lavoro.

- Denominiamo  $\mathbf{TP}_n$ , i *true positive* ottenuti sul test set addestrando il modello sul dataset  $\mathcal{D}_n$ ; ovvero gli esempi correttamente classificati al task  $n$ .
- Denominiamo inoltre  $\mathbf{TPC}_{n,m}$  i true positive in comune tra  $TP_n$  e  $TP_m$ ; ovvero gli esempi correttamente classificati sia dal modello al task  $n$  che dal modello al task  $m$ .

Definiamo a questo punto il forgetting al task  $n$  come:

$$f_n = 1 - \frac{\sum_{m=1}^{n-1} TPC_{n,m}}{\sum_{m=1}^{n-1} TP_m} \quad (3.2)$$

Questo valore coincide a  $1 - remembering$ , dove il remembering quantifica quanto abbiamo ricordato dai task passati ed è dato dalla sommatoria dei true positive in comune tra  $n$  e tutti i task precedenti diviso la sommatoria dei true positive dei task precedenti. Si nota facilmente che  $f_n \in [0, 1]$  e il valore  $f_n = 0$  indica nessun tipo di forgetting mentre  $f_n = 1$  indica un forgetting totale. Idealmente, quello che vorremmo è un valore di forgetting basso, in quanto, come vedremo nel capitolo 4, è un buon indicatore del corretto funzionamento dell'apprendimento incrementale.

**Performance drift:** altra metrica che, al contrario, è molto immediata e si può valutare facilmente è il drift delle performance. Questa metrica è molto simile all'intransigence (2.9) introdotta nel capitolo precedente e come quest'ultima è una differenza di accuratezza con un modello di riferimento.

Definiamo dunque il drift delle performance al task  $k$  come:

$$drift_k = a_k^* - a_k \quad (3.3)$$

dove  $a_k^*$  è l'accuracy ottenuta dal modello di riferimento ovvero il modello addestrato con  $\mathcal{D}_k = \cup_{i=1}^k D_i$  (joint training su tutti i data chunks fino a

$k$  compreso) e  $a_k$  l'accuracy ottenuta con l'addestramento incrementale sul dataset  $\mathcal{D}_k$ . Se questa differenza è calcolata sull'ultimo task ( $k = n$ ), è fondamentalmente la differenza tra l'accuracy del joint training su tutto il dataset e l'addestramento incrementale completo.

Idealmente, quello che vogliamo è ridurre questo drift di performance che, come vedremo dal capitolo 4 dedicato agli esperimenti, dipende dall'architettura di rete, dal dataset in questione, dal numero degli splits e dalla metodologia di apprendimento incrementale.

### 3.3 Tecniche di Continual Learning per lo scenario Data-incremental

In questa breve sezione andremo a vedere i dettagli di come sono stati implementati e adattati allo scenario Data-Incremental gli approcci di Continual Learning esposti nel capitolo 2. Come vedremo infatti, alcuni approcci sono perfettamente traslabili mentre per altri sono necessarie delle piccole modifiche.

#### 3.3.1 Feature distillation

Per quanto riguarda la feature distillation introdotta nella sezione 2.2, nel contesto data-incremental, il modello student è rappresentato dalla rete addestrata sul task corrente mentre il modello teacher è rappresentato dal modello vecchio ottenuto dall'addestramento sul task precedente e "congelato". A questo punto, vengono estratti i vettori delle feature dei due modelli e usata la loro distanza per l'addestramento del modello student.

E' stata dunque impostata una loss del tipo:

$$\mathcal{L} = \mathcal{L}_{CE}(\phi^{new}, w) + \lambda \mathcal{L}_{FD}(f^{old}, f^{new}) \quad (3.4)$$

dove con  $\phi$  si indicano i logits in output della rete, con  $f$  il vettore di feature e dove:

$$\mathcal{L}_{FD}(f^{old}, f^{new}) = \frac{\sum_{i=1}^M \|f_i^{new} - f_i^{old}\|_2}{M}$$

ovvero la media, calcolata sulla dimensione del minibatch  $M$ , della distanza in norma 2 tra le feature del modello attuale e le feature del modello precedente. Una penalty di questo tipo permette alla rete di elaborare feature che possano mantenere certo grado di coerenza rispetto alle precedenti e allo stesso tempo migliorare grazie alla cross-entropy.

### 3.3.2 LwF

Per quanto riguarda l'approccio Learning Without Forgetting, introdotto nel paragrafo 2.3.2, la struttura student-teacher è la stessa, come visto per feature distillation ma, nel caso data-incremental, non abbiamo task differenti e parametri specifici  $\theta_o$  ma solo parametri condivisi  $\theta_s$ . Questa differenza rispetto al class-incremental è data, appunto, dal fatto che nel data-incremental learning un task è composto da dati di tutte le classi e non di una classe specifica. Dunque, la loss impostata risulterà:

$$\mathcal{L} = \mathcal{L}_{CE}(\phi^{new}, w) + \lambda \mathcal{L}_{KD}(\phi^{old}, \phi^{new}) \quad (3.5)$$

dove con  $\phi^{new}$  si indicano i logits in output dal modello attuale, con  $\phi^{old}$  quelli in output dal modello vecchio “congelato” e dove  $\mathcal{L}_{KD}$  è la cross-entropy scalata col fattore temperatura tra i logits, come descritto dal paper di Hinton et al. [15].

### 3.3.3 Focal distillation

La loss di focal distillation 2.2 definita nella sottosezione 2.2.1 è perfettamente traslabile al contesto data-incremental; semplicemente il modello “vecchio” non sarà una rete differente ma la stessa rete “congelata” dopo il task precedente. In particolare, nel nostro lavoro, la loss impostata sarà dunque:

$$\mathcal{L} = \mathcal{L}_{CE}(\phi^{new}, w) + \lambda \mathcal{L}_{Focal}(\phi^{new}, \phi^{old}) \quad (3.6)$$

dove:

$$\mathcal{L}^{Focal} = - \sum_{i=1}^M [\alpha + \beta * \mathbf{1}(\hat{y}^{old}(x_i) = y_i)] \mathcal{L}_{KD}(\phi^{new}, \phi^{old})$$

con  $x_i$  e  $y_i$  input e rispettiva label,  $y^{old}(x_i)$  label prodotta dal modello teacher con input  $x_i$  e  $M$  numero di elementi del minibatch.

### 3.3.4 EWC

Per l’approccio Elastic Weight Consolidation definito nel paragrafo 2.3.3, non ci sono particolari cambiamenti nel contesto data-incremental e avremo dunque una loss:

$$\mathcal{L} = \mathcal{L}_{CE}(\phi^{new}, w) + \frac{\lambda}{2} \mathcal{F}(\theta^{new} - \theta^{old})^2 \quad (3.7)$$

dove  $\mathcal{F}$  indica la matrice di Fisher,  $\theta^{new}$  i parametri relativi al modello student e  $\theta^{old}$  quelli relativi al modello teacher.

### 3.3.5 iCaRL

Anche per iCaRL, definito nel paragrafo 2.3.4, non si rilevano troppi cambiamenti necessari per il contesto data-incremental se non l'applicazione della cross-entropy binaria alle probabilità di tutte le classi e non solo a quelle viste finora, come nel class-incremental learning. Avremo dunque una loss:

$$\mathcal{L} = \mathcal{L}_{CE}(\phi^{new}, w) + \lambda \mathcal{L}_{BinaryCE}(p^{new}, p^{old}) \quad (3.8)$$

con  $p^{new}, p^{old}$  probabilità degli output del modello student e del modello teacher.

### 3.4 Data Incremental Learning con metodologia Random Distillation Reboot

Come vedremo nella sottosezione 4.2.3 del successivo capitolo 4 riguardante gli esperimenti, le tecniche di distillation portano benefici considerevoli anche nel contesto data-incremental. Feature distillation, knowledge distillation e, in particolare, focal distillation, sono approcci validi che riducono sostanzialmente il *drift delle performance* (3.2) rispetto a una baseline come fine tuning, con vari benifici anche dal punto di vista del forgetting.

La nostra proposta, che abbiamo denominato **random distillation reboot**, è una modifica di questi metodi con l'aggiunta di un “*soft reset*”. In sostanza, data la struttura model-teacher delle tecniche di distillation, quello che si fa, ad ogni task, è *reinizializzare random solo l'head della rete student, ovvero l'ultimo layer lineare, lasciando la backbone inalterata* (dettagli sulla struttura completa in fig.3.2).

Le motivazioni dietro la modifica proposta sono principalmente due: (i) effettuare un soft reset e un apprendimento da zero dell'head aiuta a “scavalcare” zone di *plateau* e di minimo locale per la funzione obiettivo portando a minimi migliori; (ii) al termine dell’addestramento su un determinato task i pesi dell’ head della rete sono tendenti all’*overfitting* e hanno un valore assoluto prossimo alla “saturazione” per cui diventa meno efficace l’apprendimento sui task successivi.

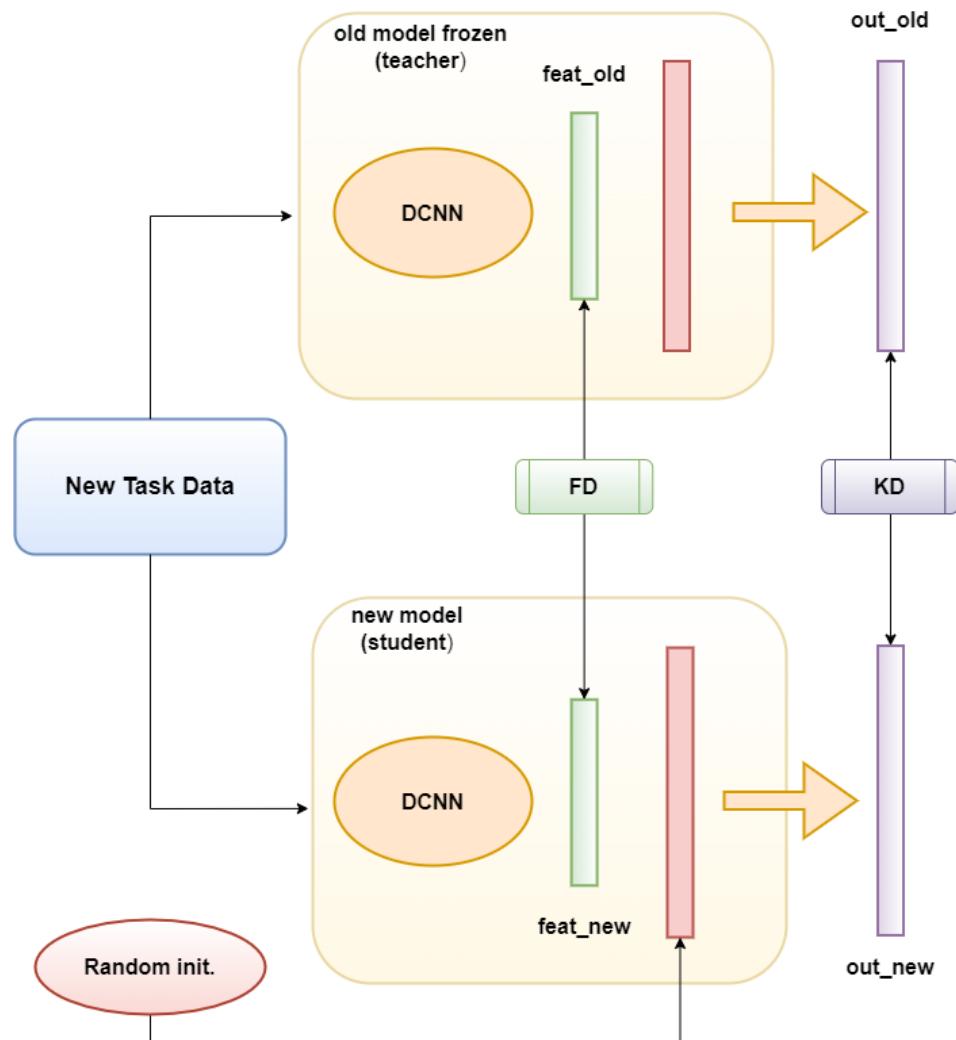


Figura 3.2: Schema dell’addestramento distillation reboot. Nel caso feature distillation reboot, verrà utilizzata la feature distillation (verde), nel caso knowledge distillation/focal distillation reboot, verrà utilizzata la knowledge distillation (viola).

L'head di una rete convoluzionale sostanzialmente applica una trasformazione lineare agli input  $x$  del tipo:

$$y = xA^T + b$$

e, già al termine del primo task,  $|A|$ , matrice dei pesi, tende a valori molto alti di saturazione in quanto la rete diventa sempre più confidente. Al task successivo si avrà di conseguenza una variazione minima di questi pesi che andrà ad apportare cambiamenti minimi o nulli sui primi layer di una rete profonda (= meno apprendimento).

Nella nostra metodologia proposta invece, i pesi vengono reinizializzati random e  $|A|$  parte da valori tendenzialmente molti bassi per poi crescere durante il training del task corrente. Questa modifica porta più *plasticity* alla rete al costo di un pò di *stability* e di un crollo dell'accuratezza nelle epoche iniziali di un task, in quanto l'head deve essere riaddestrata.

Nelle sottosezioni 4.2.5 e 4.3.2 del capitolo 4 degli esperimenti verrà valutata la metodologia random distillation reboot proposta e mostrato come, soprattutto in combinazione con gli exemplar, risulti il miglior approccio tra quelli analizzati nello scenario data-incremental.

# Capitolo 4

## Risultati sperimentali

Questa è la sezione centrale e più importante di questo lavoro; di seguito verrà esposta tutta la sperimentazione svolta e i risultati ottenuti. Avremo dunque una prima sezione dedicata alla configurazione degli esperimenti che è propedeutica per gli esperimenti mostrati in seguito. Un'altra sezione consistente è dedicata agli esperimenti svolti sul dataset CIFAR-100 in quanto gran parte del lavoro di questa tesi si è concentrato su questo dataset. Infine, sarà presente una sezione dedicata agli esperimenti svolti su un sottoinsieme di ImageNet.

### 4.1 Configurazione degli esperimenti

Prima di proseguire con gli esperimenti, vediamo in questa sezione la configurazione base di quest'ultimi. In particolare, verranno introdotti i dataset scelti per gli esperimenti e come questi sono stati suddivisi per simulare lo scenario data-incremental. Vedremo inoltre con quale regime di training sono condotti gli esperimenti e le reti valutate.

### 4.1.1 Datasets e protocollo di sperimentazione

La ricerca in un determinato settore dipende anche dai dataset accessibili e, per quanto riguarda i dataset di classificazione immagini impiegati in contesti incrementali, questi sono un numero piuttosto ridotto. In letteratura, come abbiamo visto nel capitolo 2, gran parte dei lavori sono stati fatti su **ImageNet** [2], **MNIST** [43] e su **CIFAR** [20] (nelle sue varianti *CIFAR-10* e *CIFAR-100*). In questo lavoro, come vedremo, ci siamo concentrati su CIFAR-100 per poi estendere gli esperimenti su TinyImageNet, un sottoinsieme di ImageNet.

CIFAR-100 è un dataset standard molto utilizzato nei riferimenti recenti ed è stato scelto in quanto, essendo suddiviso in 100 classi, risulta non banale dal punto di vista dell'Image Classification (a differenza di MNIST e CIFAR-10 che hanno entrambi solo 10 classi). Nel dettaglio, CIFAR-100 è un dataset che raccoglie 60.000 immagini a colori 32x32 divise in 100 classi (600 immagini per classe). Le immagini fondamentalmente coprono vari settori e raffigurano veicoli, alberi, persone, cibo, animali, insetti ma anche addobbi e apparecchi per la casa (alcuni esempi in fig. 4.1). Per quanto riguarda le immagini, è stata definita la seguente suddivisione :

- 50.000 immagini di training (500 per ogni classe).
- 5.000 immagini di test (50 per ogni classe).
- 5.000 immagini per il validation set (50 per ogni classe) .

Come anticipato nella sezione 3.1, il dataset viene poi suddiviso in splits dove il numero di esempi per classe è distribuito equamente.

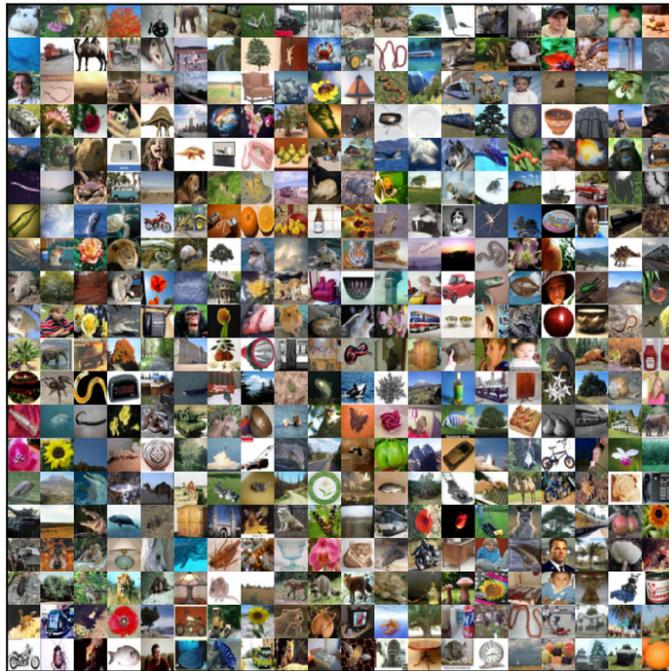


Figura 4.1: Esempio di Immagini presenti in CIFAR-100. [20]

Di questo dataset sono stati valutati diversi splits (come vedremo successivamente in fig.4.3) ma la configurazione principale degli esperimenti, per avere un buon compromesso tra margini di lavoro e numero di esempi per classe presenti nello split, è a **10 splits**. In sostanza, per ogni split/task di CIFAR-100 sono presenti:

- 5.000 immagini di training, ovvero 50 immagini per classe.

Come anticipato, sono stati svolti alcuni esperimenti e validati i risultati su TinyImageNet [3], un sottoinsieme di ImageNet; questo per valutare un problema ancora più complesso rimanendo comunque in tempi computazionali ragionevoli. Il problema di classificazione in questo contesto è molto simile a quello posto da ImageNet solo che sono presenti 200 classi. Vi è quindi una sostanziale differenza in termini di complessità rispetto a CIFAR-100.

Nello specifico, questo dataset contiene 110.000 immagini a colori a cui è stata posta la divisione in:

- 100.000 immagini di training (500 per ogni classe).
- 5.000 immagini di test (25 per ogni classe).
- 5.000 immagini per il validation set (25 per ogni classe) .

Anche qui, mantenendo una certa coerenza con il caso CIFAR-100, si è optato per **10 splits** dove, per ogni split/task di TinyImageNet, avremo:

- 10.000 immagini di training, ovvero 50 immagini per classe.

#### 4.1.2 Modelli valutati

L’architettura di rete di riferimento per gli esperimenti che vedremo è **ResNet32**. Questa rete è uno standard nei lavori di incremental learning essendo una versione alternativa e “*light*” delle reti ResNet tradizionali che presenta una struttura composta da 3 macro-layers (a differenza dei 4 standard). Essendo una rete rapida da addestrare e con performance discrete per molti task di classificazione, si pone come un buon punto di partenza capace di dare un’idea iniziale dell’impatto sul learning dei vari approcci.

Resnet32 tuttavia è una “rete giocattolo” molto basilare che non viene utilizzata in contesti reali; per tale motivo e per avvalorare ancora di più

i risultati ottenuti, sono stati svolti degli esperimenti anche su **ResNet18**. Quest’ultima è una rete con la classica struttura ResNet 2.1.1 e si avvicina di più, per parametri e complessità, alle reti stato dell’arte. In particolare, quantificando la differenza tra le due reti in termini di parametri addestrabili, si ha:

- *ResNet32*  $\approx 479.064$  parametri addestrabili
- *ResNet18*  $\approx 11.218.340$  parametri addestrabili

dove è immediato notare che ResNet18 ha oltre 20 volte il numero di parametri di ResNet32 ed è quindi capace di apprendere feature ben più complesse.

#### 4.1.3 Regime di training

Per quanto riguarda l’addestramento della rete, almeno come configurazione di base, sono stati posti:

- test set e validation set **fissi** per ogni task/split in quanto non ha senso una divisione per task, come nel caso class-incremental.
- **50 epoch** di training per ogni task con meccanismo di *early stopping* basato sulla validation loss e un termine di *patience* = 30. In questo modo si porta la rete alla saturazione su un determinato task evitando l’*overfitting* della rete.
- come ottimizzatore è stato utilizzato principalmente *Adam* anche se su CIFAR-100 vedremo una valutazione di *SGD*. Per quanto riguarda

il learning rate, questo viene ottimizzato tramite una grid search sul validation test.

- batch-size fissata a 64.

## 4.2 Esperimenti su CIFAR-100

Questa sezione raccoglie gran parte degli esperimenti in quanto CIFAR-100 è stato il focus principale di questo lavoro. Partiremo con una valutazione del fine tuning e degli upper bound di ricerca per inquadrare meglio il problema. Successivamente saranno poi esposti gli esperimenti dedicati agli exemplar e saranno valutati una serie di approcci per migliorare le performance. Come anticipato nella sottosezione 4.1.2, gli esperimenti fin qui descritti saranno condotti su ResNet32 ma, al termine di questa sezione, verranno mostrati i risultati su ResNet18 e messe a paragone le performance delle due architetture.

### 4.2.1 Valutazione della baseline e degli upper bound

Saranno riportati adesso i risultati sperimentali valutando la *baseline* (ovvero il fine tuning) e gli *upper bound* di ricerca. Gli upper bound sono il joint training, ovvero l’addestramento cumulativo su tutto il dataset, e il joint incremental, ovvero un addestramento del modello incrementale dove ad ogni task vengono riutilizzati per il training i dati dei task precedenti. Quest’ultimo è un *upper bound incrementale* concettualmente più ragionevole da raggiungere, in termini di performance, rispetto al joint training.

Per valutare questi approcci, è stato utilizzato l'ottimizzatore Adam con un learning rate fisso pari a 0.001 ottenuto tramite una grid search dei valori  $lr = [10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}]$ , valutando l'accuracy media sul validation set ed effettuando 5 prove con 5 diversi seed. In particolare, questo valore è risultato ottimale sia per il joint training che per il fine tuning.

Dalla tabella 4.1 della grid search del fine tuning, si nota inoltre che  $lr = 0.001$  è risultato ottimale sia per quanto riguarda il secondo task che per quanto riguarda le performance al termine dei 10 task. In pratica, il parametro che determina le migliori performance sul secondo task è il miglior parametro, ovvero corrisponde a quello che determina la miglior accuratezza al termine dei 10 task. Questa osservazione sarà un pattern ricorrente che ci tornerà utile più avanti nel seguito degli esperimenti.

In fig.4.2 invece, possiamo osservare il confronto tra gli approcci fine tuning, joint training e joint incremental e si osserva come il *performance drift* (3.2) tra il fine tuning completo al task 10 e il joint incremental sia intorno al 18%. Questo valore sarà dunque il *gap* che, sperimentando approcci differenti, cercheremo di ridurre il più possibile.

	$lr = 0.1$	$lr = 0.01$	$lr = 0.001$	$lr = 0.0001$	$lr = 0.00001$
<b>Accuracy task2</b>	9.58 (+-0.23)	24.50 (+-0.42)	<b>33.02 (+-0.75)</b>	29.81 (+-0.58)	13.55 (+-0.17)
<b>Accuracy task10</b>	13.02 (+-0.11)	33.68 (+-0.26)	<b>45.72 (+-0.35)</b>	42.77 (+-0.22)	27.64 (+-0.12)

Tabella 4.1: Risultati della grid search sul learning rate.

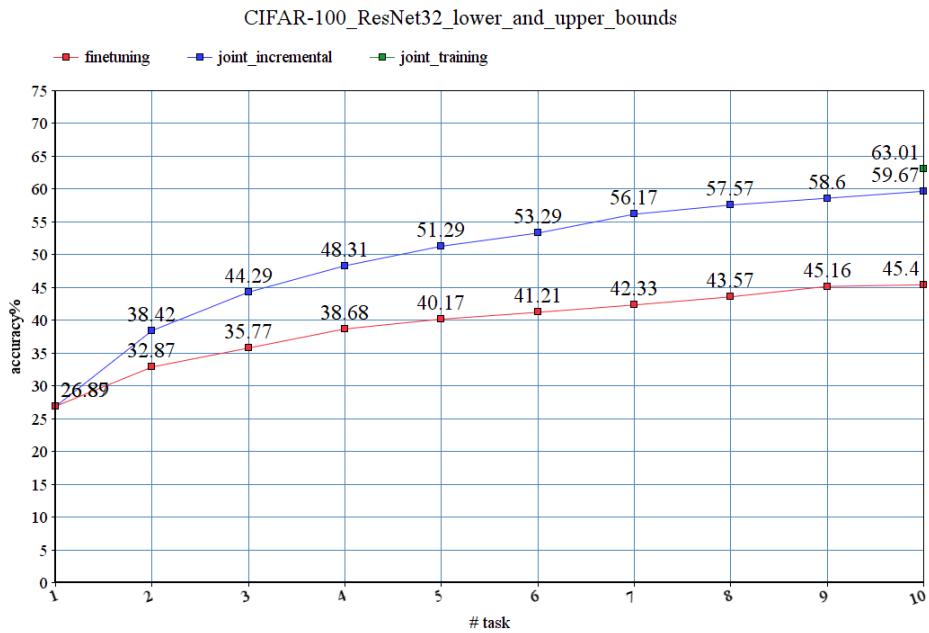


Figura 4.2: Valutazione della baseline (fine tuning) e degli upper bound (joint training, joint incremental) di CIFAR-100 e ResNet32

Particolare da notare in fig. 4.2 è che esiste, seppur molto più leggero, un drift di performance tra il joint incremental training con 10 task e il joint training su tutto il dataset. Questo perchè, nonostante anche il joint incremental usi l'intero dataset per il training al task 10, nel joint training il modello viene addestrato da zero mentre nel joint incremental il modello viene aggiornato incrementalmente lungo i task partendo da configurazioni basate solo su porzioni ridotte del dataset.

In fig. 4.3 invece è stata analizzata la crescita del drift di performance tra fine tuning e joint training al variare del numero degli split effettuati sul dataset.

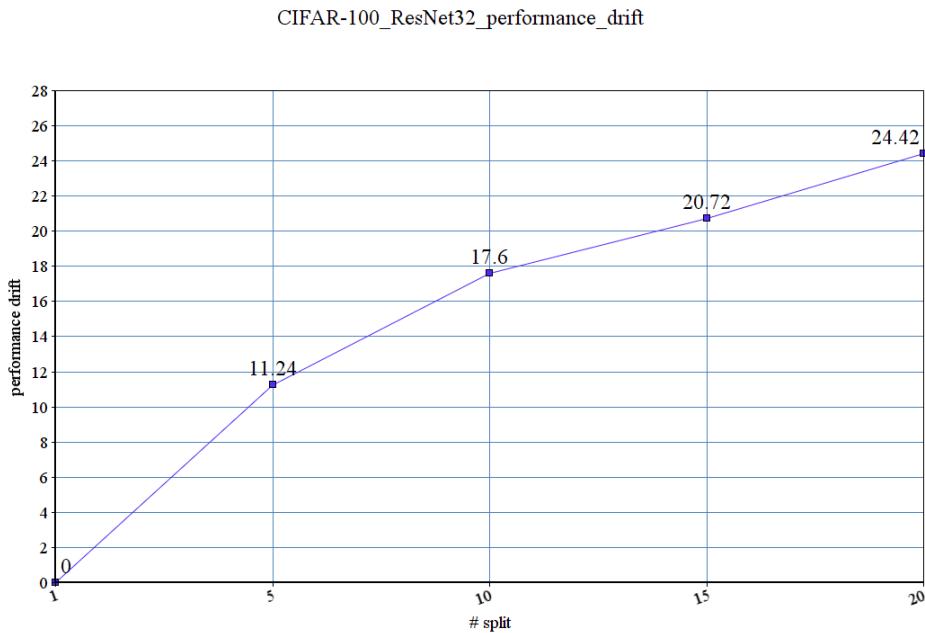


Figura 4.3: Drift delle performance (3.2) su CIFAR-100 e ResNet32 al variare del numero degli split del dataset.

Altro dato di interesse e fondamentale per la nostra ricerca, riguarda il forgetting del joint incremental a confronto con il fine tuning, visibile in figura 4.4. A differenza di ciò che ci si poteva aspettare, ovvero un forgetting basso o quasi nullo nel caso del joint incremental (poichè utilizza i dati dei task passati per il training), è facile notare dal grafico come **esista una componente di forgetting e non sia del tutto trascurabile**. Chiaramente questo valore è notevolmente più basso del fine tuning e la sfida che ci poniamo sarà dunque di sviluppare e provare approcci per migliorare le performance, in termini di accuratezza, rispetto al fine tuning, con un occhio di riguardo anche alla metrica del forgetting.

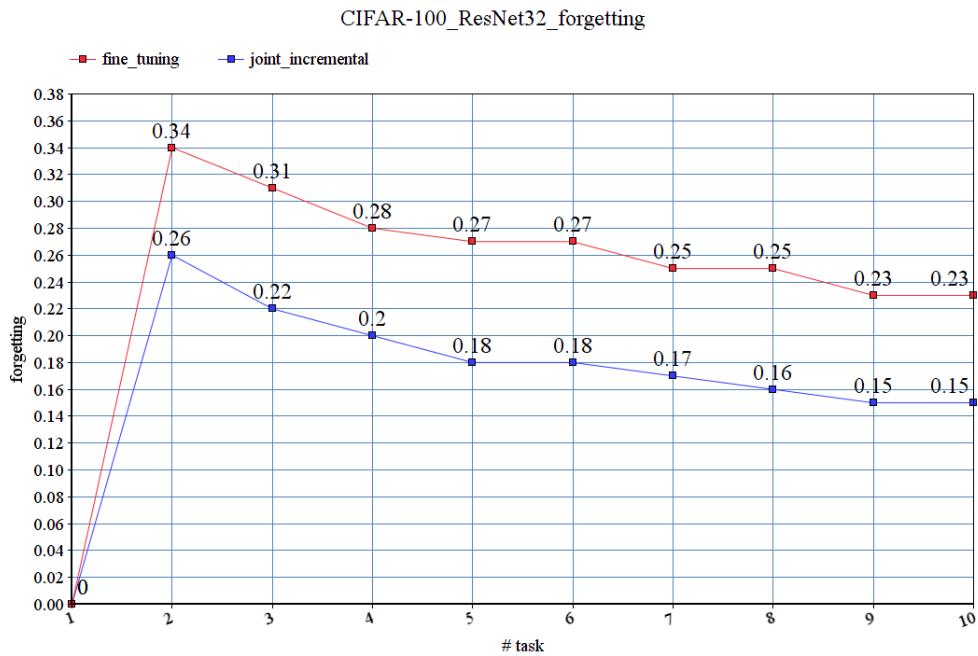


Figura 4.4: Confronto sull’andamento del forgetting(3.2) tra fine tuning e joint incremental su CIFAR-100 e ResNet32.

L’approccio fine tuning è stato valutato anche con *SGD* (*Stochastic gradient descent*), che è solitamente lo standard per quanto riguarda i lavori di class-incremental learning presenti in letteratura. Come per Adam, sono stati provati con la grid search, i learning rate nel range  $lr = [10^{-1}, 10^{-2}, 10^{-3}]$  (vedesi tabella 4.2) ed è stato provato SGD standard e SGD con il *momentum* in range  $\gamma = [0.9, 0.8, 0.7, 0.6]$ .

La configurazione ottimale è risultata  $lr = 10^{-2}$  e  $\gamma = 0.9$ ; i risultati, in configurazione fine tuning, sono visibili in fig. 4.5.

	$lr = 0.1$	$lr = 0.01$	$lr = 0.001$
$Accuracy_{task2}$	13.82 (+-0.28)	<b>31.17 (+-0.43)</b>	13.88 (+-0.34)
$Accuracy_{task10}$	32.58 (+-0.21)	<b>46.18 (+-0.24)</b>	30.44 (+-0.19)

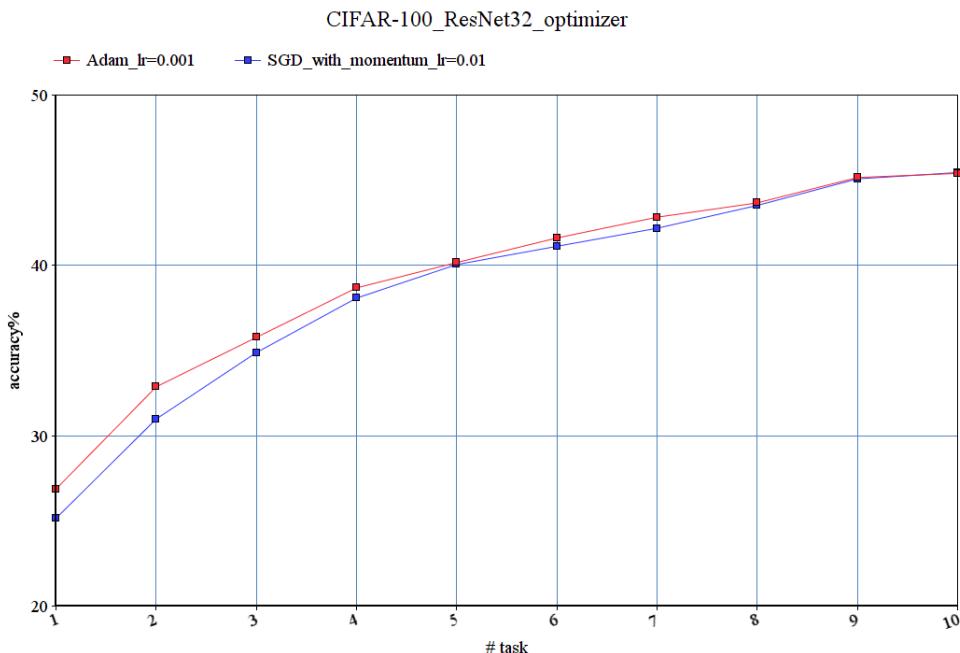
Tabella 4.2: Grid search del  $lr$  effettuata con SGD.

Figura 4.5: Comparazione tra Adam e SGD su CIFAR-100 e ResNet32.

Dal grafico 4.5 si nota come Adam abbia performance migliori sui primi task mentre, per quanto riguarda gli ultimi e il task 10, sostanzialmente entrambi gli ottimizzatori hanno le stesse performance (45.40% Adam vs 45.45% SGD task 10). Per semplicità dunque, nel seguito degli esperimenti

verrà preso come riferimento principale Adam.

Definiti dunque gli upper bound e la baseline e fatte le prime considerazioni, nei prossimi esperimenti ci concentreremo essenzialmente su CIFAR-100 con 10 splits per andare a ridurre in modo sostanziale il drift valutando al contempo l'andamento del forgetting.

### 4.2.2 Studi sugli exemplar

Una delle metodologie atte a migliorare le performance e ridurre il forgetting, è l'introduzione degli exemplar. Gli exemplar risulteranno molto significativi sia perchè il funzionamento dell'approccio iCaRL non può prescindere da questi ultimi, sia perchè, come vedremo, alcuni approcci e metodologie funzionano meglio in combinazione con questi. Nel contesto degli exemplar, devono comunque esser fatti esperimenti per valutare se esista una policy di selezione più ragionevole in grado di selezionare exemplar più rappresentativi e quanta memoria dedicare a quest'ultimi.

#### 4.2.2.1 Valutazione della miglior policy di selezione

Per quanto riguarda le policy di selezione, sono state essenzialmente provate le 4 policy più diffuse introdotte nel paragrafo 2.3.1, ovvero: *random-based*, *herding-based*, *entropy-based* e *distance-based*. le ultime due inoltre sono state provate in entrambi i versi (selezione a maggiore/minore entropia e maggiore/minore distanza). Per testare questi tipi di selezione è stata addestrata la rete con l'approccio fine tuning e le specifiche viste nella sez. 4.2.

Nel nostro esperimento sulle policy di selezione, quello che succede è che, ad ogni task, vengono selezionati incrementalmente 500 exemplar (su 5000 esempi di training) in base alla policy, da utilizzare per il training del task

successivo. Quindi, al task 2 verrà utilizzato per il training un insieme aumentato con 500 esempi provenienti dal task 1, al task 3 l'insieme di training sarà aumentato da 500 esempi provenienti dal task 1 e 500 esempi provenienti dal task 2 e così via. I risultati sono ancora la media di 5 run con diverso seed e sono visibili in fig. 4.6.

Dal grafico e dalla tabella si nota facilmente come le performance siano piuttosto simili tra le varie policy, con una leggera predominanza delle policy *random-based* e *herding-based*. E' da tenere di conto però che la selezione tramite herding, descritta nella sezione 2.3.1 del capitolo 2, è un processo computazionale non banale dove il modello deve ricalcolare le features dei dati di training, fare le medie e eseguire dei confronti.

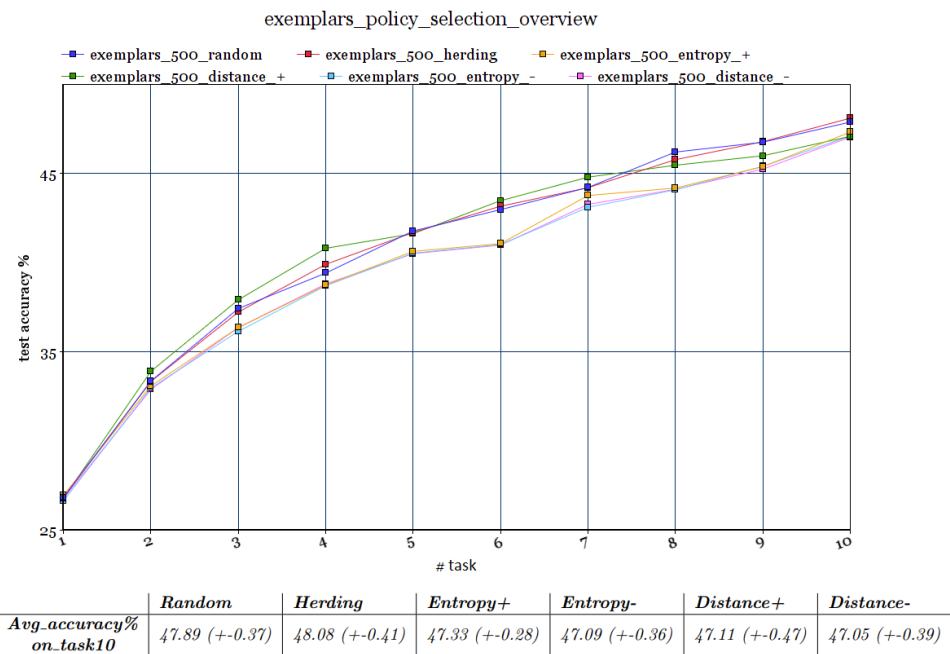


Figura 4.6: Confronto tra le accuracy(3.2 delle policy di selezione degli exemplar su CIFAR-100 e ResNet32.

Possiamo stimare la complessità nel tempo della selezione herding come un  $\mathcal{O}(nfm)$ , dove  $n$  è la size del training set,  $f$  la dimensione delle features e  $m$  il numero di esempi che si vogliono selezionare [5]. D’altro canto, la selezione random è piuttosto banale ed è quindi stata scelta come standard per i prossimi esperimenti in quanto **miglior trade-off performance/complexity**.

I risultati e la scelta della selezione random ha senso alla luce di risultati analoghi in letteratura e se consideriamo il fatto che, nello scenario data-  
Incremental, la distribuzione dei dati non varia tra i task. Questo fatto fa sì che routines complicate di selezione degli exemplar basate su un determinato criterio, non portino necessariamente migliori performance e rende sensata la scelta a random.

#### 4.2.2.2 Memoria incrementale vs memoria fissa

Un’altra valutazione da fare attentamente quando si utilizzano gli exemplar riguarda il quantitativo di memoria da dedicare e come questa viene gestita. Chiaramente, dedicando troppa memoria per lo storage degli exemplar, viene meno il vantaggio principale degli approcci di incremental learning; ovvero il poter scartare i dati precedenti senza occupare quantitativi importanti di memoria e il poter evitare un joint training dispendioso in termini computazionali e ambientali. D’altro canto, se l’insieme degli exemplar è veramente piccolo, il rischio è che non si abbia alcun impatto rilevante sulle performance.

In fig. 4.7, possiamo appunto notare l’impatto sulle performance degli exemplar.

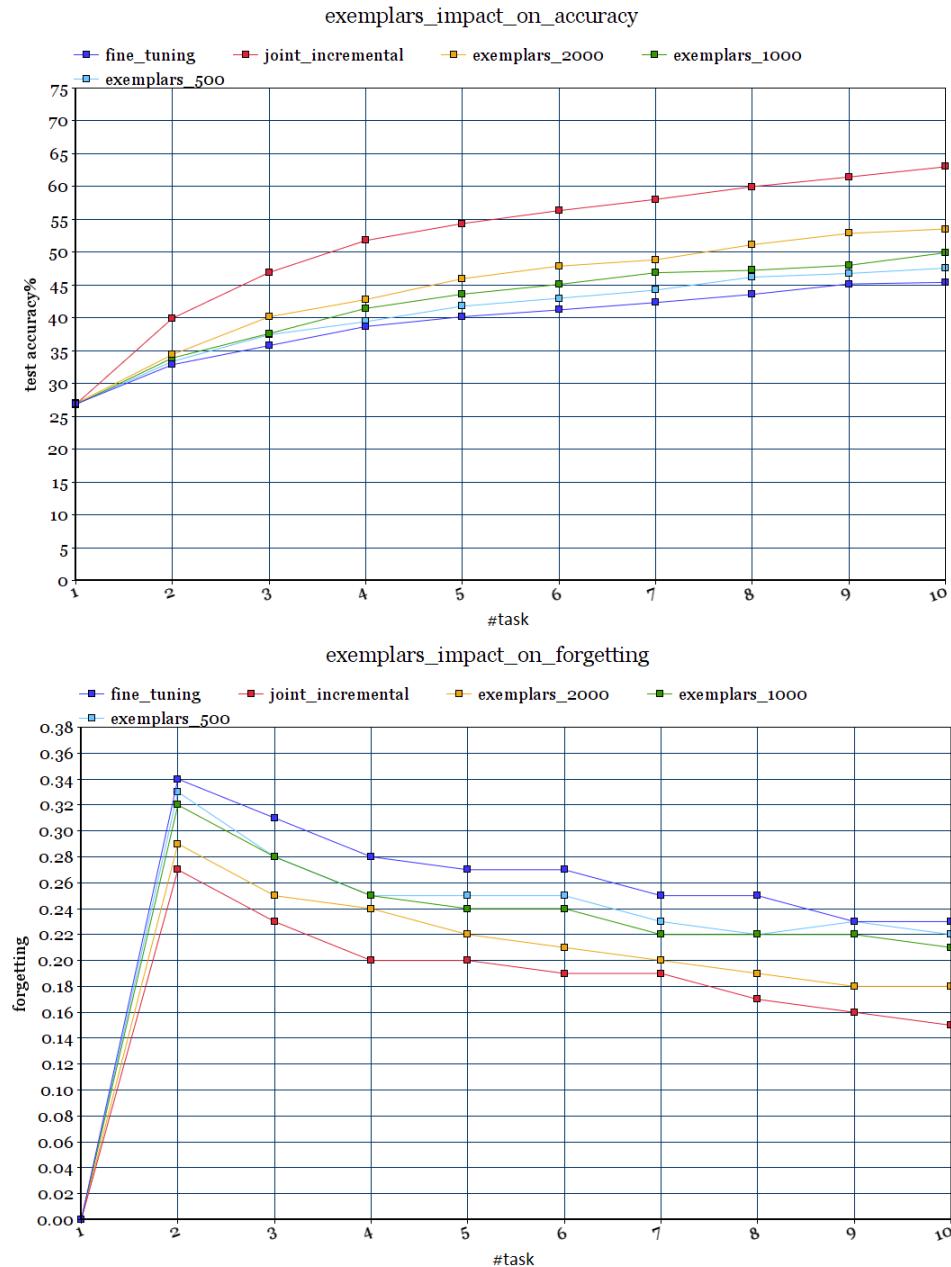


Figura 4.7: Impatto sulle performance degli exemplar. La denominazione "exemplars\_  $N$ " sta ad indicare un addestramento di tipo fine tuning con l'aggiunta incrementale di  $N$  exemplar per task.

Come ci si aspetta, aumentando il numero di exemplar per task, si riduce significativamente il forgetting (3.2) e si aumenta l'accuratezza (3.2). Bisogna ricordarsi però che queste sono valutazioni in un contesto di memoria incrementale per cui, nel caso “*exemplars\_2000*”, al decimo task avremo bisogno di tenere in memoria  $2000 * 9 = 18.000$  exemplar. Questa requisito di memoria non è da poco considerando che CIFAR-100 dispone di un totale di 50.000 esempi di training. L’aspetto più preoccupante tuttavia è che, in un contesto reale, tipicamente, l’apprendimento incrementale prosegue per molti più task di 10 e l’avere un requisito di memoria per gli exemplar che cresce linearmente col numero dei task, è molto sconveniente. Per queste ragioni è stato valutato anche un approccio diverso, a memoria fissa, in linea con l’approccio iCaRL precedentemente descritto.

In pratica, la memoria dedicata agli exemplar viene fissata a un valore costante  $N = k$  e questa viene riempita, al task  $t$ , di  $k/(t - 1)$  exemplar per ogni task passato. Per esempio, ponendo  $N = 1000$ , al task 2 si potranno utilizzare 1000 exemplar provenienti dal task 1 mentre, al task 3, si avranno a disposizione 500 exemplar dal task 1 e 500 exemplar dal task 2 e così via. I risultati, che fanno uso dell’approccio fine tuning e della politica di selezione random, sono visibili in fig. 4.8.

L’aspetto interessante che si nota dai risultati in figura è che *l’approccio a memoria fissa con  $N = 2000$  è comparabile, in termini di accuratezza, all’approccio a memoria incrementale con  $N = 1000$ .* Questa non è una considerazione di poco conto in quanto l’approccio incrementale richiede, al task 10, una memoria di 9000 exemplar (contro i 2000 fissati) e, come già detto, non si adatta molto bene a un’applicazione reale.

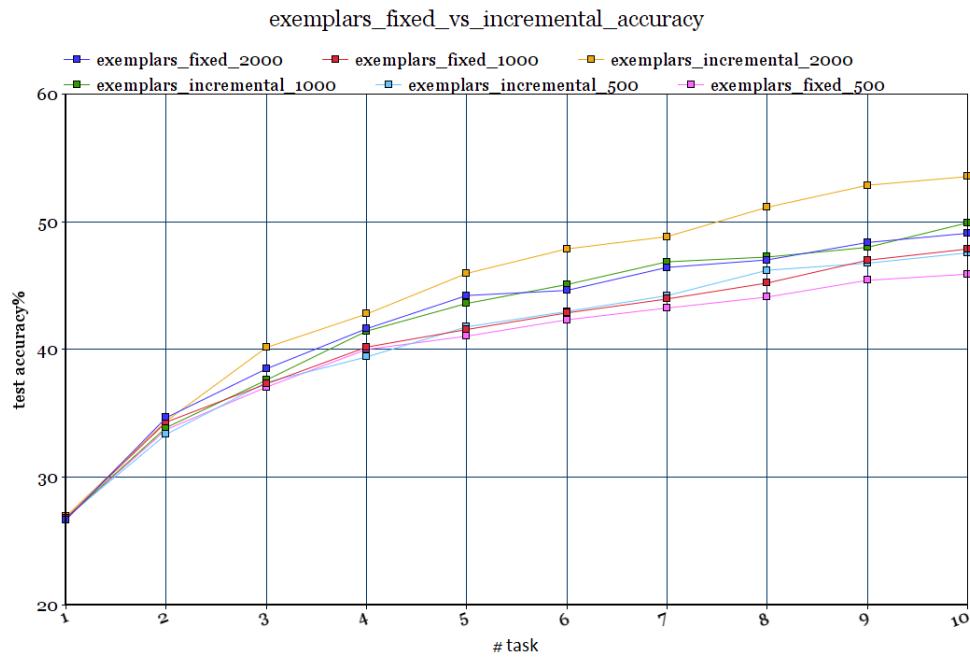


Figura 4.8: Confronto dell’accuracy (3.2) tra exemplar a memoria incrementale e exemplar a memoria fissa.

In conclusione, nei successivi esperimenti in cui verranno integrati gli exemplar si andrà a utilizzare ragionevolmente una selezione **random** e una **memoria fissa di 2000 exemplar**.

### 4.2.3 Valutazione degli approcci

Di seguito verranno proposti gli esperimenti eseguiti con i vari approcci introdotti nel capitolo 2 e adattati allo scenario Data-Incremental Learning. Per ognuno di questi saranno valutate le metriche di accuracy e forgetting paragonate alla baseline del fine tuning e gli upper bound. Gli approc-

ci exemplar-free verranno inoltre valutati anche integrando gli exemplar in modo da fornire un'analisi il più completa possibile.

#### 4.2.3.1 Feature distillation

In questa sezione sono presenti gli esperimenti relativi all'approccio di feature distillation introdotto nella sezione 2.2 e approfondito nel nostro contesto data-incremental nella sottosezione 3.3.1. La loss di feature distillation (3.4) è una distanza tra feature pesata da un parametro  $\lambda$  da ottimizzare. In fig. 4.10 e fig. 4.9 sono rispettivamente visibili dunque l'andamento dell'accuracy e del forgetting a confronto con il fine tuning e il joint incremental.

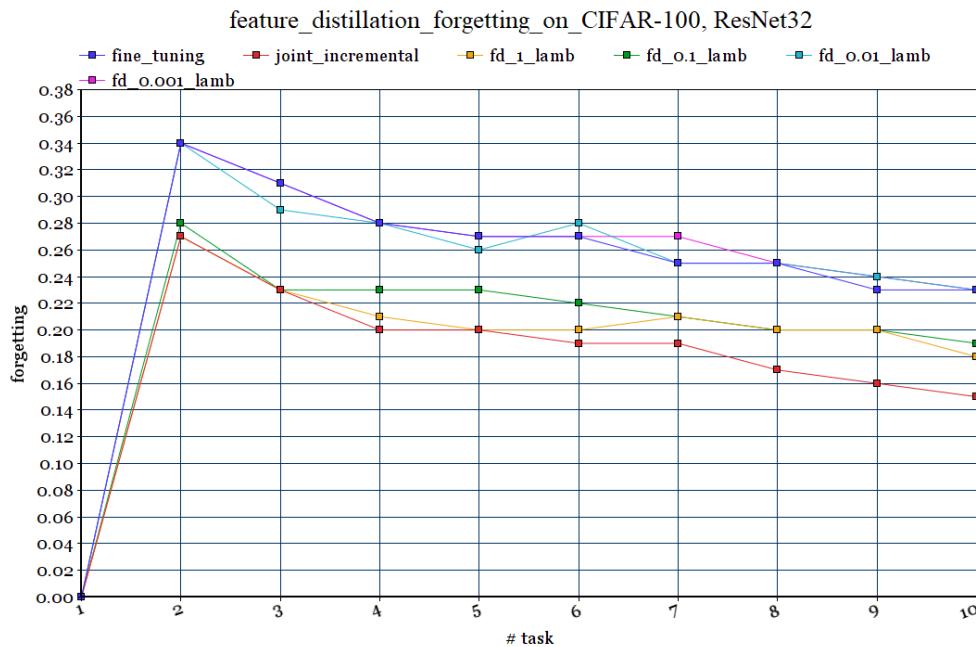


Figura 4.9: Forgetting dell'approccio feature distillation al variare dell'iperparametro  $\lambda$ .

	Joint_incr	Fine_tuning	Fd_lamb_1	Fd_lamb_0.1	Fd_lamb_0.01	Fd_lamb_0.001
TP_task1	1019	1031	1029	1026	1027	1037
TP_task2	1790	1263	1195	1217	1291	1256
CommonTPs	664	569	669	647	606	575

Tabella 4.3: Conteggio dei true positive e dei true positive in comune tra il task 1 e il task 2 con l’approccio feature distillation.

Possiamo notare come, per  $\lambda = 1$ , si ottenga un forgetting prossimo al joint incremental ma un accuracy più bassa mentre, per  $\lambda = 0.1$  si riesca a migliorare leggermente l’accuratezza del fine tuning e abbassare comunque in maniera decisa il forgetting.

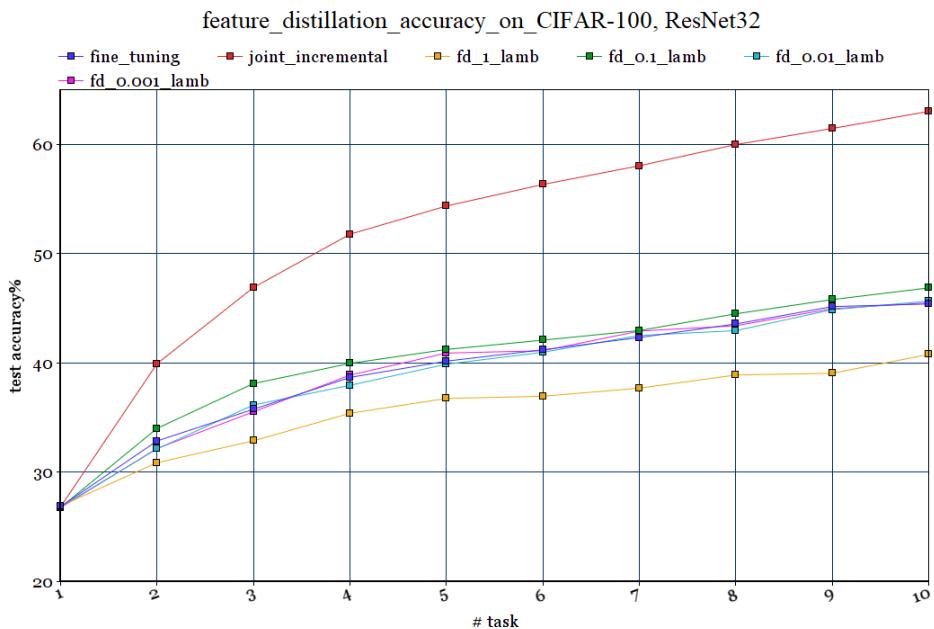


Figura 4.10: Accuracy dell’approccio feature distillation al variare dell’iparametro  $\lambda$ .

Inoltre, in tabella 4.3, per apprezzare meglio gli effetti dell’approccio analizzato, è proposta una comparazione dei true positive ottenuti con feature distillation al variare di lambda a paragone col fine tuning e il joint incremental.

I risultati numerici riassuntivi di feature distillation con  $\lambda = 0.1$  a paragone con la baseline e gli upper bound sono consultabili in tabella 4.4.

Le conclusioni che possiamo trarre da questi esperimenti è che feature distillation, con  $\lambda$  scelto in maniera opportuna, risulta leggermente superiore lungo tutti i task sia in termini di accuracy che di forgetting al fine tuning. L’approccio analizzato riesce dunque ad apprendere piuttosto bene dai nuovi task e a mantenere un certo grado di coerenza con le feature passate, certificato dal sostanziale decremento del forgetting. Questi risultati rendono feature distillation uno degli approcci più promettenti e vedremo, nel riepilogo finale 4.2.4 , le sue performance a paragone con gli altri approcci.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Fine_tuning</i>	<b>2</b>	32.87 (+-0.84)	0.34 (+-0.01)
	<b>5</b>	40.17 (+-0.53)	0.27 (+-0.01)
	<b>10</b>	45.40 (+-0.39)	0.23 (+-0.00)
<i>Joint training</i>		63.01 (+-0.23)	0.00 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	38.42 (+-0.65)	0.26 (+-0.01)
	<b>5</b>	51.29 (+-0.29)	0.18 (+-0.00)
	<b>10</b>	59.67 (+-0.11)	0.15 (+-0.00)
<i>Feature_distillation</i> $l=0.1$	<b>2</b>	34.02 (+-0.84)	0.28 (+-0.01)
	<b>5</b>	41.24 (+-0.50)	0.23 (+-0.01)
	<b>10</b>	46.87 (+-0.41)	0.19 (+-0.00)

Tabella 4.4: Riepilogo numerico di confronto dell’approccio di feature distillation.

**Valutazione con gli exemplar** Per completare l’analisi, sono stati integrati gli exemplar e valutato l’approccio di feature distillation a confronto con fine tuning. I risultati, in termini di accuracy, sono visibili in fig. 4.11 e numericamente, in forma riassuntiva, in tabella 4.5. Le specifiche degli exemplar coincidono con quelle descritte nella sottosezione 4.2.2 (memoria fissa di 2000, selezione random) ma in tabella sono riportati anche i risultati di feature distillation con gli exemplar selezionati tramite policy basata sull’herding. Ciò che si nota nel confronto tra feature distillation e fine tuning (con exemplar) è un comportamento piuttosto simile al caso senza exemplar anche se in questo caso il divario di accuratezza è più marcato. Questi risultati sono sostanzialmente un ulteriore indicazione dell’efficacia di feature distillation e mostrano come l’approccio si integri piuttosto bene con gli exemplar. Inoltre, la differenza tra la policy random e la policy herding visibile in tabella 4.5 è minima e questo conferma i risultati sperimentali ottenuti nella sottosezione 4.2.2.1 dedicata agli exemplar.

<i>Exemplar table</i>			
<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Fine_tuning</i>	<i>2</i>	<i>34.97 (+-1.30)</i>	<i>0.30 (+-0.01)</i>
	<i>5</i>	<i>43.27 (+-0.59)</i>	<i>0.24 (+-0.01)</i>
	<i>10</i>	<i>48.73 (+-0.35)</i>	<i>0.21 (+-0.00)</i>
<i>Joint training</i>		<i>63.01 (+-0.23)</i>	<i>0.00 (+-0.00)</i>
<i>Joint_incremental</i>	<i>2</i>	<i>38.42 (+-0.65)</i>	<i>0.26 (+-0.01)</i>
	<i>5</i>	<i>51.29 (+-0.29)</i>	<i>0.18 (+-0.00)</i>
	<i>10</i>	<i>59.67 (+-0.11)</i>	<i>0.15 (+-0.00)</i>
<i>Feature_distillation</i> <i>l=0.1</i> <i>(rand selection)</i>	<i>2</i>	<i>34.89 (+-0.88)</i>	<i>0.24 (+-0.01)</i>
	<i>5</i>	<i>44.49 (+-0.51)</i>	<i>0.21 (+-0.00)</i>
	<i>10</i>	<i>50.02 (+-0.28)</i>	<i>0.19 (+-0.00)</i>
<i>Feature_distillation</i> <i>l=0.1</i> <i>(herding selection )</i>	<i>2</i>	<i>34.86 (+- 1.01)</i>	<i>0.24 (+-0.01)</i>
	<i>5</i>	<i>44.61 (+-0.61)</i>	<i>0.21 (+-0.01)</i>
	<i>10</i>	<i>50.19 (+-0.30)</i>	<i>0.19 (+-0.00)</i>

Tabella 4.5: Riepilogo numerico di feature distillation con gli exemplar.

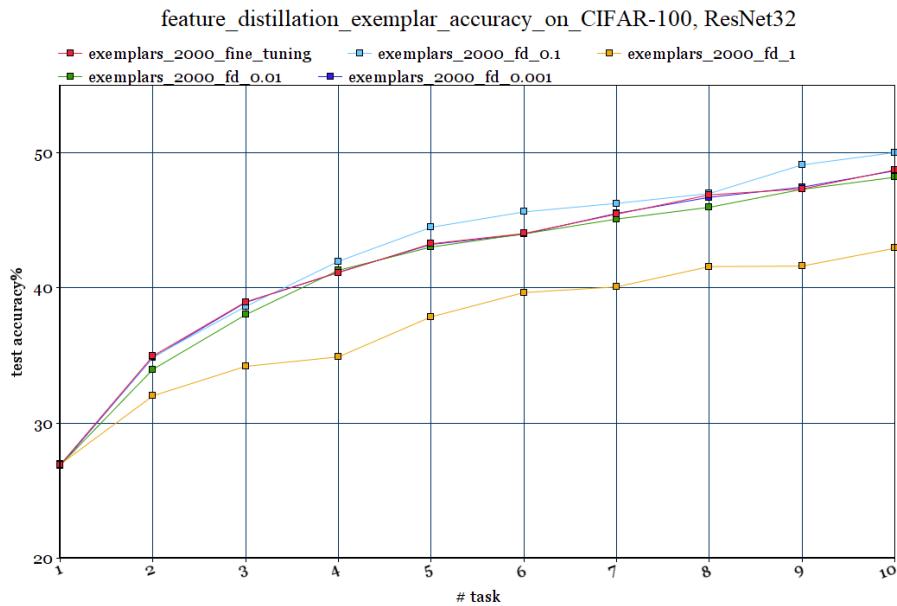


Figura 4.11: Accuracy dell’approccio feature distillation con exemplar al variare dell’iperparametro  $\lambda$ .

#### 4.2.3.2 EWC

Di seguito sono presenti gli esperimenti riguardanti l’approccio EWC (*Elastic Weight Consolidation*), introdotto nel paragrafo 2.3.3 e contestualizzato per lo scenario data-incremental nella sottosezione 3.3.4.

Poichè la *penalty* di EWC (3.7), descritta, include un iperparametro  $\lambda$  per pesare il contributo, sono stati provati vari valori di quest’ultimo e raccolti i valori medi di accuracy su 5 run di ognuno. Sfortunatamente, quello che si evince dal grafico in fig. 4.12 è che, in termini di accuratezza, **il fine tuning risulta un asintoto orizzontale di EWC**. Si nota infatti che, “alleggerendo” la penalty di EWC diminuendo  $\lambda$ , le performance migliorano ma tendono al fine tuning.

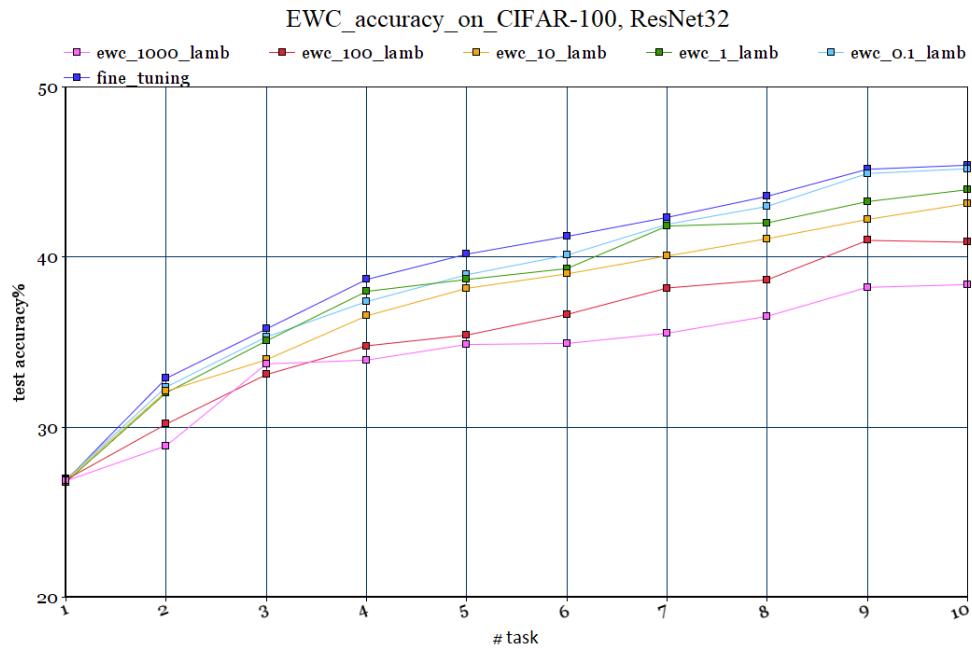


Figura 4.12: Accuracy dell’approccio EWC al variare dell’iperparametro  $\lambda$ .

La causa delle performance inferiori e tendenti al fine tuning non è da attribuire tuttavia a una penalty troppo grande data da EWC che vada a trattenere troppo l’apprendimento. Questo perchè, come è possibile osservare in fig 4.13, è stato verificato che, anche con  $\lambda = 1000$  la penalty di ewc è molto piccola rispetto alla loss totale. Abbassando ulteriormente il valore di  $\lambda$ , questa diventa praticamente irrisiona e si torna al caso fine tuning. Per tali motivi, nella tabella riassuntiva finale 4.6, è stato inserito il valore  $\lambda = 1000$  in quanto, anche se non ottimale, è l’unico valore che porta variazioni significative al fine tuning (vedesi la metrica del forgetting).

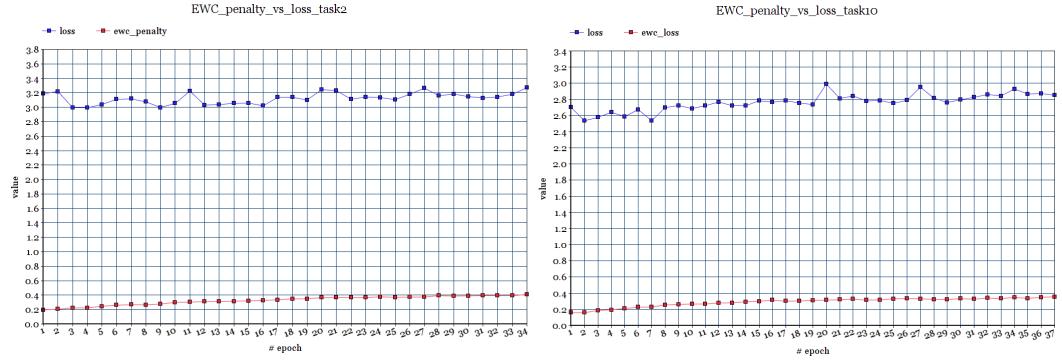


Figura 4.13: Andamento della loss di EWC lungo il task 2 e 10.

In prima valutazione, possiamo concludere che EWC, nel contesto data-incremental, tende a vincolare troppo i pesi della rete, rallentando l'apprendimento con l'arrivo di nuovi dati relativi ai nuovi task. Per cercare di ovviare a questa troppa *stability*, sono state provate varianti dell'approccio EWC in cui non vengono presi in considerazione, per il calcolo della Fisher, i pesi dell'ultimo layer lineare della rete e dei layers di batch normalization. Questa prova è stata fatta con il sospetto che EWC, applicato a questi specifici layer, possa andare a vincolare troppo l'output della rete. In realtà, come si può osservare in fig.4.14, queste modifiche non hanno portato a sostanziali miglioramenti rispetto all'approccio di EWC standard.

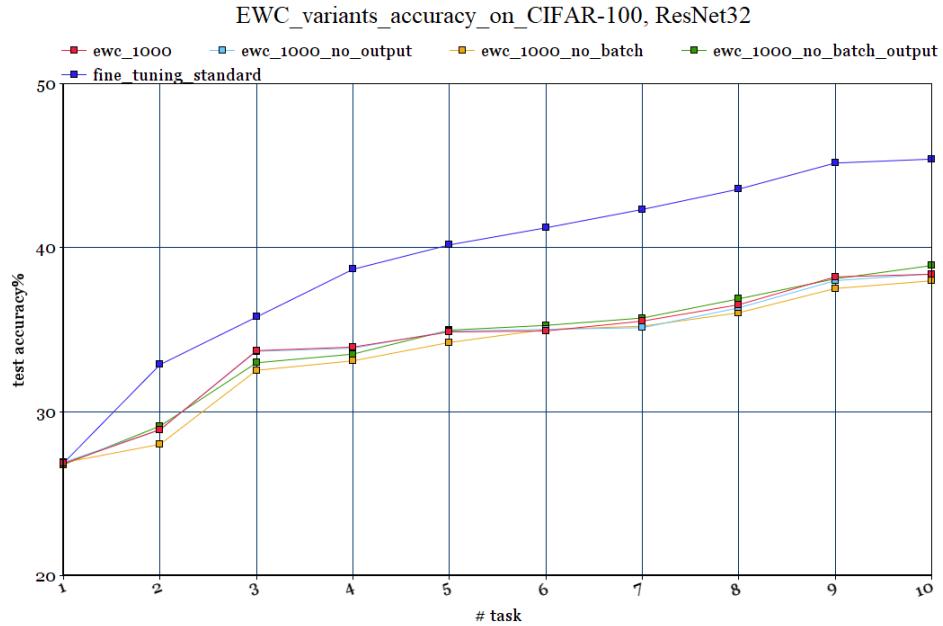


Figura 4.14: Accuracy delle varianti di EWC provate.

Approach	After task	Avg_accuracy	Avg_forgetting
<i>Fine_tuning</i>	2	32.87 (+-0.84)	0.34 (+-0.01)
	5	40.17 (+-0.53)	0.27 (+-0.01)
	10	45.40 (+-0.39)	0.23 (+-0.00)
<i>Joint training</i>		63.01 (+-0.23)	0.00 (+-0.00)
<i>Joint_incremental</i>	2	38.42 (+-0.65)	0.26 (+-0.01)
	5	51.29 (+-0.29)	0.18 (+-0.00)
	10	59.67 (+-0.11)	0.15 (+-0.00)
<i>EWC</i> $l=1000$	2	28.90 (+-1.76)	0.28 (+-0.01)
	5	34.86 (+-0.91)	0.23 (+- 0.00)
	10	38.38 (+-0.68)	0.19 (+-0.00)

Tabella 4.6: Riepilogo numerico di confronto di EWC.

**Valutazione con gli exemplar** Per completare l’analisi, sono stati integrati gli exemplar e valutato EWC a confronto con fine tuning. Le specifiche degli exemplar coincidono con quelle descritte nella sezione 4.2.2 (memoria fissa di 2000, selezione random). Seppur si possa notare globalmente, rispetto al caso senza exemplar, un miglioramento dell’accuracy di circa il 3%, fine tuning con gli exemplar continua a rimanere un asintoto per EWC con gli exemplar, con un andamento caratterizzato dai valori di  $\lambda$  che coincide esattamente con il caso senza exemplar. I risultati, al variare di lambda sono confrontabili in fig. 4.15 e nella tabella riassuntiva 4.7.

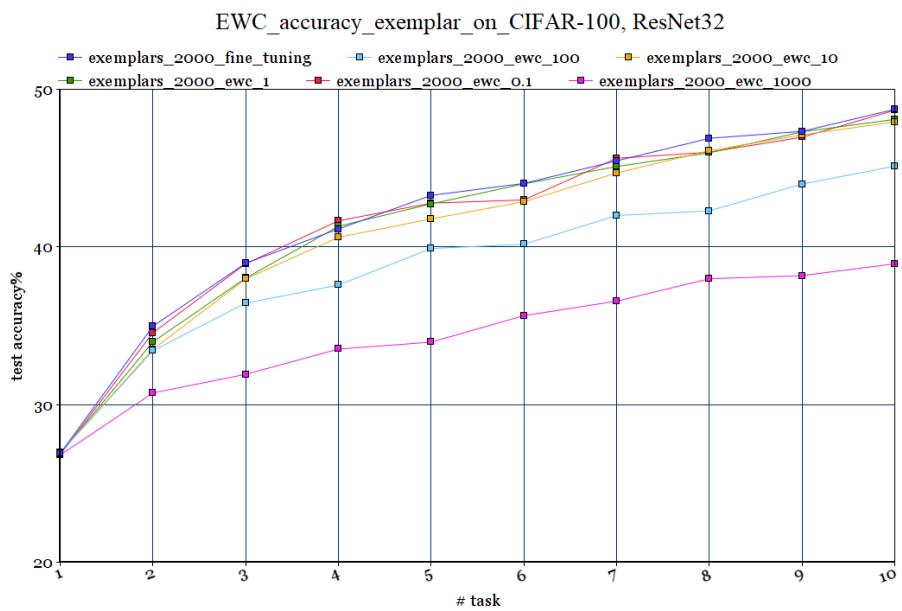


Figura 4.15: Accuracy di EWC in combinazione con gli exemplar al variare dell’iperparametro  $\lambda$ .

<i>Exemplar table</i>			
<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Fine_tuning</i>	<b>2</b>	<i>34.97 (+-1.30)</i>	<i>0.30 (+-0.01)</i>
	<b>5</b>	<i>43.27 (+-0.59)</i>	<i>0.24 (+-0.01)</i>
	<b>10</b>	<i>48.73 (+-0.35)</i>	<i>0.21 (+-0.00)</i>
<i>Joint training</i>		<i>63.01 (+-0.23)</i>	<i>0.00 (+-0.00)</i>
<i>Joint_incremental</i>	<b>2</b>	<i>38.42 (+-0.65)</i>	<i>0.26 (+-0.01)</i>
	<b>5</b>	<i>51.29 (+-0.29)</i>	<i>0.18 (+-0.00)</i>
	<b>10</b>	<i>59.67 (+-0.11)</i>	<i>0.15 (+-0.00)</i>
<i>EWC</i> <i>l=1000</i>	<b>2</b>	<i>30.74 (+-0.98)</i>	<i>0.28 (+-0.01)</i>
	<b>5</b>	<i>33.96 (+-0.42)</i>	<i>0.26 (+-0.00)</i>
	<b>10</b>	<i>38.93 (+-0.29)</i>	<i>0.22 (+-0.00)</i>

Tabella 4.7: Riepilogo dei risultati di EWC con gli exemplar.

#### 4.2.3.3 iCaRL

Gli studi e l'introduzione degli exemplar in questo lavoro sono stati fatti anche e soprattutto con lo scopo di valutare l'approccio iCaRL in questo contesto. Inizialmente è stata dunque provata la configurazione originaria standard di iCaRL descritta nel paragrafo 2.3.4, ovvero memoria fissa di 2000, selezione degli exemplar tramite herding e classificazione di tipo *NME* (*Nearest-Mean-of-Exemplars*) (2.3.4). Abbiamo visto poi che la loss di iCaRL (3.8) è composta dalla cross-entropy e da una penalty di distillation pesata da un iperparametro  $\lambda$ . I risultati al variare di questo iperparametro, in termini di accuracy, sono visibili in figura 4.17.

Come si può facilmente osservare, iCaRL standard non funziona bene nel contesto data-incremental con performance nettamente inferiori al fine tuning. Inoltre, valori di  $\lambda$  diversi, non migliorano la situazione suggerendo che il problema risieda nella classificazione *NME*.

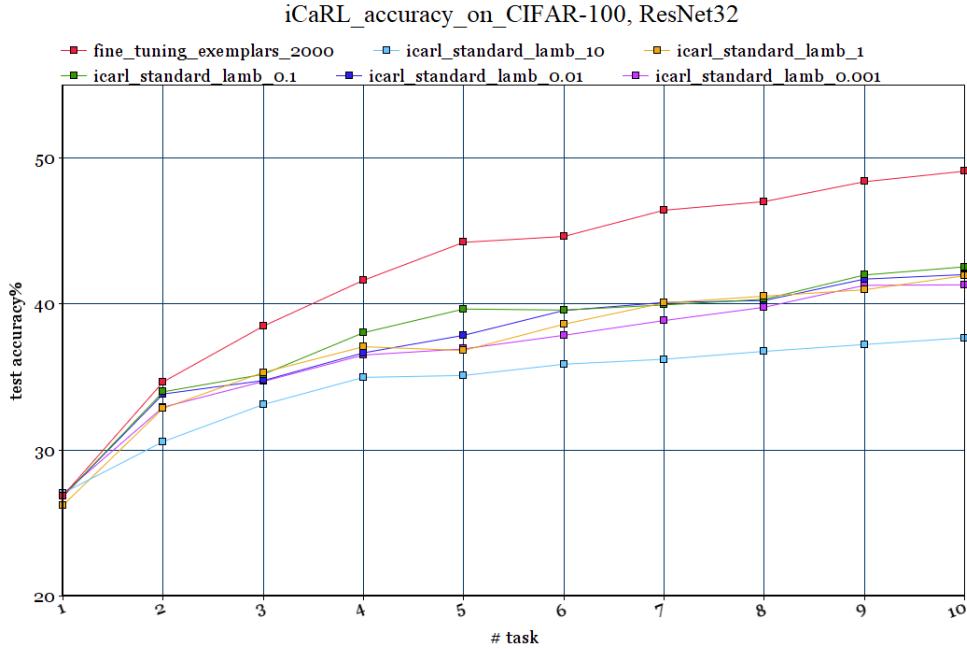


Figura 4.16: Accuracy dell’approccio iCaRL al variare dell’iperparametro  $\lambda$ .

La situazione migliora nettamente se si passa ad una formulazione dell’approccio non basata sugli exemplar e quindi su una classificazione di tipo *NCM* (*Nearest-Class-Mean*). In sostanza, in questo caso, i prototipi usati per la classificazione vengono calcolati su tutto il training set e non solo sugli exemplar. A questo punto, sono stati provati due diversi approcci:

- i prototipi calcolati nei vari task vengono accumulati e usati per l’inferenza, facendo una media tra i prototipi calcolati sul task attuale e quelli calcolati per il task precedente.

- vengono usati solo i prototipi calcolati sul task attuale per l'inferenza e scartati i prototipi precedenti.

Entrambe le varianti sono piuttosto simili con risultati lievemente migliori nel caso di prototipi accumulati, come visibile in figura 4.17. In questo grafico, è stato fissato  $\lambda = 0.1$  e la variante di iCaRL che usa solo i prototipi del task attuale è stata denominata "icarl\_NCM" mentre la variante in cui i task vengono accumulati è stata denominata "icarl\_NCM\_accumulator". Dal grafico si nota facilmente che, utilizzando la classificazione NCM, le performance migliorano rispetto all'approccio standard di iCaRL rimanendo tuttavia ancora una volta inferiori al fine tuning.

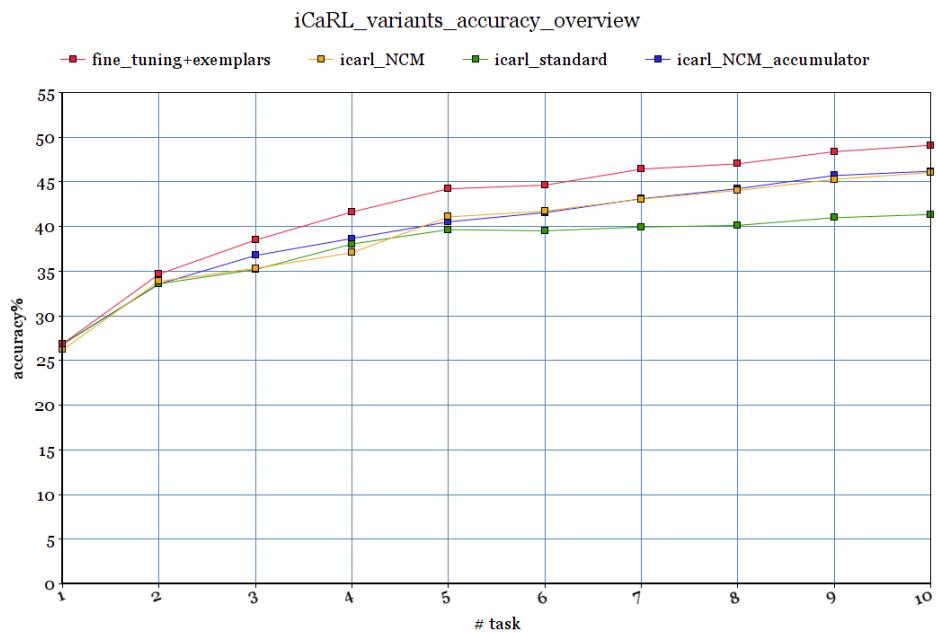


Figura 4.17: Accuracy delle varianti di iCaRL provate.

Un ulteriore esperimento svolto su iCaRL è stato quello di provare a sostituire la distillation loss di iCaRL con la loss di feature distillation (3.4) sia nella sua configurazione standard che nella configurazione NCM. Questa prova è stata fatta alla luce dei risultati promettenti di feature distillation (consultabili nel paragrafo 4.2.3.1).

I risultati sono raggruppati per chiarezza in tabella 4.8 dove è possibile osservare che feature distillation tende addirittura a peggiorare le performance. In particolare, nel caso in cui si vada a combinare feature distillation con “icarl\_NCM” si ha che, per  $\lambda = 0.1$ , la rete ha troppa stability e ottiene un forgetting molto basso e stabile per tutti i task a discapito dell’accuracy; andando ad abbassare  $\lambda$  a 0.01, il forgetting torna presente, le performance migliorano ma risultano comunque inferiori al fine tuning.

Indagando più a fondo sul perchè iCaRL non funziona bene, in un ulteriore esperimento è stato monitorato l’andamento della distanza tra i prototipi del task corrente e quelli del task precedente. I risultati, visibili in tabella 4.9, mostrano come la distanza, col passare dei task, decrementi fino a diventare minima (circa 15%). Questa poca variazione dei prototipi potrebbe spiegare il perchè la classificazione attraverso quest’ultimi non sia efficace nel caso Data-Incremental. Infatti, rimanendo tendenzialmente ferma la media prototipi, non si va ad aumentare la capacità di classificazione al proseguire dei task. A seguito di questa osservazione sperimentale, congiunta con gli esperimenti descritti in precedenza, si può supporre che la poca plasticità nel procedere dei task e la poca discriminazione dei prototipi nel contesto data-incremental, sia la causa principale della degradazione di performance di iCaRL con classificazione NME o NCM.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Fine_tuning</i> + <i>exemplar</i>	<b>2</b>	<i>34.97 (+-1.30)</i>	<i>0.30 (+- 0.01)</i>
	<b>5</b>	<i>43.27 (+-0.59)</i>	<i>0.24 (+-0.01)</i>
	<b>10</b>	<i>48.73 (+-0.35)</i>	<i>0.21 (+- 0.00)</i>
<i>Joint training</i>		<i>63.01 (+-0.23)</i>	<i>0.00 (+-0.00)</i>
<i>Joint_incremental</i>	<b>2</b>	<i>38.42 (+-0.65)</i>	<i>0.26 (+- 0.01)</i>
	<b>5</b>	<i>51.29 (+-0.29)</i>	<i>0.18 (+-0.00)</i>
	<b>10</b>	<i>59.67 (+-0.11)</i>	<i>0.15 (+- 0.00)</i>
<i>iCaRL standard</i> <i>l=0.1</i>	<b>2</b>	<i>33.59 (+-1.38)</i>	<i>0.27 (+-0.02)</i>
	<b>5</b>	<i>39.67 (+-0.98)</i>	<i>0.27 (+-0.01)</i>
	<b>10</b>	<i>41.32 (+-0.62)</i>	<i>0.26 (+-0.01)</i>
<i>iCaRL + fd</i> <i>l=0.1</i>	<b>2</b>	<i>30.56 (+-1.08)</i>	<i>0.26 (+-0.01)</i>
	<b>5</b>	<i>34.68 (+-0.49)</i>	<i>0.27 (+-0.00)</i>
	<b>10</b>	<i>39.92 (+-0.33)</i>	<i>0.26 (+-0.00)</i>
<i>iCaRL_NCM</i> <i>l=0.1</i>	<b>2</b>	<i>33.56 (+-1.11)</i>	<i>0.24 (+-0.01)</i>
	<b>5</b>	<i>40.53 (+-0.95)</i>	<i>0.23 (+-0.01)</i>
	<b>10</b>	<i>46.19 (+-0.42)</i>	<i>0.20 (+-0.00)</i>
<i>iCaRL_NCM+fd</i> <i>l=0.1</i>	<b>2</b>	<i>31.17 (+-0.95)</i>	<i>0.18 (+- 0.01)</i>
	<b>5</b>	<i>38.29 (+-0.97)</i>	<i>0.18 (+-0.01)</i>
	<b>10</b>	<i>42.66 (+-0.29)</i>	<i>0.18 (+-0.00)</i>
<i>iCaRL_NCM + fd</i> <i>l=0.01</i>	<b>2</b>	<i>33.69 (+-0.64)</i>	<i>0.24 (+-0.00)</i>
	<b>5</b>	<i>40.41 (+-0.94)</i>	<i>0.23 (+- 0.01)</i>
	<b>10</b>	<i>45.54 (+-0.25)</i>	<i>0.21 (+-0.00)</i>

Tabella 4.8: Risultati riassuntivi degli esperimenti effettuati su iCaRL.

<i>Approach</i>	<i>task</i>	<i>distance between prototypes</i>
<i>Icarl_</i> <i>NCM</i>	<b>1</b>	0
	<b>2</b>	0.257 (+-0.03)
	<b>3</b>	0.227 (+-0.03)
	<b>4</b>	0.190 (+-0.03)
	<b>5</b>	0.169 (+-0.02)
	<b>6</b>	0.171 (+-0.02)
	<b>7</b>	0.170 (+-0.02)
	<b>8</b>	0.167 (+-0.02)
	<b>9</b>	0.164 (+-0.02)
	<b>10</b>	0.150 (+-0.02)

Tabella 4.9: Medie delle distanze in norma 2 tra i prototipi.

#### 4.2.3.4 LwF : Knowledge Distillation

Vediamo adesso gli esperimenti riguardanti l'approccio LwF introdotto nel paragrafo 2.3.2 e adattato al contesto data-incremental nella sottosezione 3.3.2. La loss di questo approccio (3.5) è composta dalla solita cross-entropy e dalla knowledge distillation loss di cui è necessario fare un tuning dell'iperparametro  $\lambda$ .

Risultato sperimentale che useremo in questo e nei successivi esperimenti riguardanti focal distillation (4.2.3.5), è che, *la configurazione (iperparametri/learning rate) che rende risultati migliori in termini di accuracy media al secondo task coincide con la configurazione che rende risultati migliori al termine di tutti i task*; ciò sostanzialmente significa che, per effettuare il tuning degli iperparametri, possiamo limitarci a valutare l'accuracy media al secondo task sul validation set invece che svolgere l'addestramento incrementale

completo. Questa osservazione è stata introdotta nella sottosezione 4.2.1 ed è osservabile in tutte le tabelle e grafici illustrati negli esperimenti precedenti. Questo risultato sperimentale è molto importante anche a livello pratico in contesti reali in quanto ottimizzare gli iperparametri sul secondo task è fattibile mentre ottimizzare gli iperparametri su un addestramento incrementale completo a priori è impossibile e non ha senso data la natura incrementale del problema affrontato.

In tabella 4.10 è riportata la grid search sul secondo task e evidenziato in grassetto l'iperparametro ottimale ( $\lambda = 1$ ), scelto dando priorità all'accuracy e tenendo conto successivamente del forgetting. I risultati di questo approccio valutato con il suddetto iperparametro sono visibili nei grafici in fig.4.20 e nella tabella riassuntiva 4.11. In particolare, si nota come knowledge distillation funzioni piuttosto bene mostrandosi decisamente meglio di fine tuning sia per quel che riguarda il forgetting che l'accuracy.

<i>Task</i>	<i>avg_accuracy%</i>	<i>avg_forgetting</i>	<i>lambda</i>
2	<i>33.02 (+-0.61)</i>	<i>0.33 (+-0.01)</i>	<i>0.001</i>
2	<i>34.54 (+-0.72)</i>	<i>0.32 (+-0.01)</i>	<i>0.01</i>
2	<i>36.59 (+-0.62)</i>	<i>0.31 (+-0.01)</i>	<i>0.1</i>
2	<b><i>36.63 (+-0.59)</i></b>	<b><i>0.26 (+-0.01)</i></b>	<b><i>1.0</i></b>
2	<i>34.82 (+-0.66)</i>	<i>0.12 (+-0.01)</i>	<i>10</i>
2	<i>28.64 (+-0.27)</i>	<i>0.00 (+-0.00)</i>	<i>100</i>
2	<i>28.64 (+-0.27)</i>	<i>0.00 (+-0.00)</i>	<i>1000</i>

Tabella 4.10: Grid search dell'iperparametro  $\lambda$  di LwF.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Fine_tuning</i>	<b>2</b>	32.87 (+-0.84)	0.34 (+-0.01)
	<b>5</b>	40.17 (+-0.53)	0.27 (+-0.01)
	<b>10</b>	45.40 (+-0.39)	0.23 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	38.42 (+-0.65)	0.26 (+-0.01)
	<b>5</b>	51.29 (+-0.29)	0.18 (+-0.00)
	<b>10</b>	59.67 (+-0.11)	0.15 (+-0.00)
<i>Joint_training</i>		63.01 (+-0.23)	
<i>lwf</i> <i>l=1</i>	<b>2</b>	<b>36.84 (+-0.85)</b>	<b>0.26 (+-0.01)</b>
	<b>5</b>	<b>42.37 (+-0.51)</b>	<b>0.22 (+-0.01)</b>
	<b>10</b>	<b>47.93 (+-0.27)</b>	<b>0.18 (+-0.00)</b>

Tabella 4.11: Riepilogo numerico dei risultati di LwF.

In questi esperimenti è stato usato il parametro di temperatura  $T = 2$ , come suggerito dal paper di Hinton [15] e di LwF [23]. Come prove aggiuntive sono stati provati anche i due casi limite ovvero  $T = 1$  e  $T = 500$  dove quest'ultimo può essere sostanzialmente approssimato con la distanza tra i logits. I risultati sono visibili nella tabella 4.12 qui di seguito e mostrano come effettivamente una temperatura impostata a 2 porti dei benefici all'addestramento.

	$T = 1$	$T = 2$	$T = 500$
<i>Accuracy on_task2</i>	36.09 (+-0.97)	<b>36.84 (+-0.85)</b>	34.70 (+-0.77)
<i>Accuracy on_task10</i>	46.71 (+-0.22)	<b>47.93 (+-0.27)</b>	45.02 (+-0.16)

Tabella 4.12: Risultati di LwF al variare della temperatura.

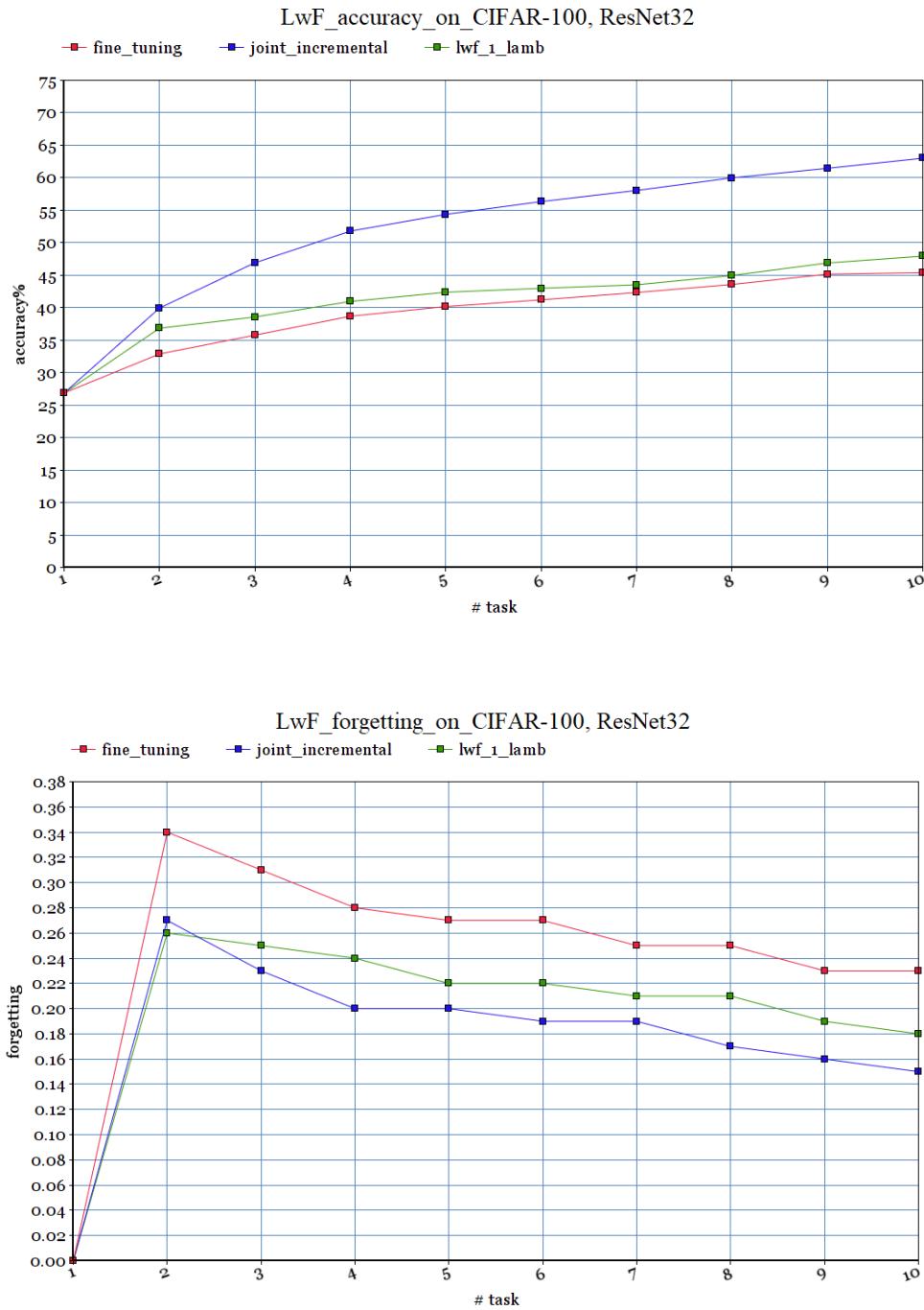


Figura 4.20: Accuracy (4.18) e forgetting (4.19) dell'approccio LwF con  $\lambda = 1$ .

**Valutazione con gli exemplar** A completamento dell’analisi è stato valutato LwF con gli exemplar e l’andamento dell’accuracy è visibile in fig.4.21. In tabella 4.13 sono riassunti invece numericamente i risultati con annessa la metrica del forgetting.

Ciò che si nota sperimentalmente è che l’approccio LwF con gli exemplar continua ad essere migliore rispetto al fine tuning (con exemplar) ma le performance non migliorano moltissimo. Si nota infatti come il divario di accuratezza tra LwF e fine tuning nel caso senza exemplar ( $\approx 2.50\%$ ) tenda ad assottigliarsi ( $\approx 1.20\%$ ) considerando il caso con gli exemplar. Questo risultato sembra suggerirci che l’approccio LwF, seppur vantaggioso, non funzioni molto bene associato agli exemplar.

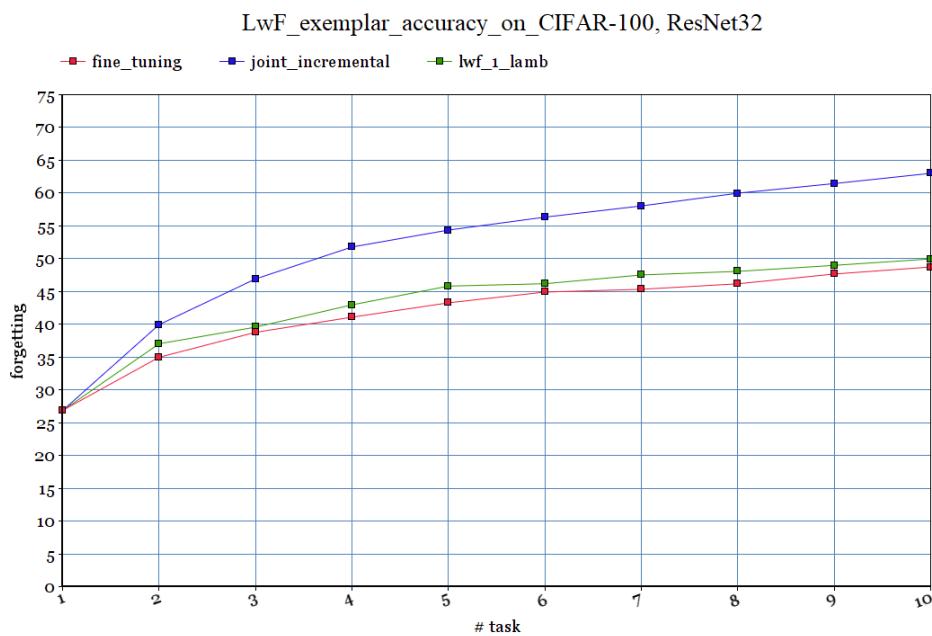


Figura 4.21: Accuracy dell’approccio LwF con gli exemplar.

<i>Exemplar table</i>			
<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		63.01 (+-0.23)	0.00 (+-0.00)
	<b>2</b>	38.42 (+-0.65)	0.26 (+-0.01)
	<b>5</b>	51.29 (+-0.29)	0.18 (+-0.00)
<i>Joint_incremental</i>	<b>10</b>	59.67 (+-0.11)	0.15 (+-0.00)
	<b>2</b>	34.97 (+-1.30)	0.30 (+-0.01)
	<b>5</b>	43.27 (+-0.59)	0.24 (+-0.01)
<i>Fine_tuning</i>	<b>10</b>	48.73 (+-0.29)	0.21 (+-0.00)
	<b>2</b>	<b>37.03 (+-0.82)</b>	<b>0.23 (+-0.01)</b>
	<b>5</b>	<b>45.82 (+-0.78)</b>	<b>0.21 (+-0.01)</b>
<i>lwf</i> <i>l=1</i>	<b>10</b>	<b>49.96 (+-0.36)</b>	<b>0.18 (+-0.00)</b>

Tabella 4.13: Risultati numerici della valutazione di LwF con gli exemplar.

#### 4.2.3.5 Focal distillation

Di seguito sono esposti gli esperimenti con l'approccio Focal distillation introdotto nella sottosezione 2.2.1 e specificato per il contesto data-incremental nella sottosezione 3.3.3. Come è possibile osservare dalla formula della loss (3.6), in questo caso gli iperparametri da ottimizzare sono ben tre:  $\lambda$ ,  $\beta$  e  $\alpha$ .

Come nel paragrafo 4.2.3.4, la grid search viene fatta valutando l'accuracy al secondo task sul validation test utilizzando le varie combinazioni degli iperparametri proposte dal paper [41]. I valori provati per  $\lambda$  sono nel range  $[100, 10, 1, 0.1, 0.001]$  con risultati ottimali per  $\lambda = 1$  visibili in tabella 4.14. Come è possibile notare in grassetto dalla tabella, sono stati selezionati i valori di  $\alpha = 1$  e  $\beta = 5$ , in quanto **miglior tradeoff accuracy-forgetting**, coincidendo inoltre con i valori scelti dal paper di riferimento.

<i>Task</i>	<i>avg_accuracy%</i>	<i>avg_forgetting</i>	<i>lambda</i>	<i>alpha</i>	<i>beta</i>
2	36.62 (+-0.65)	0.27 (+-0.01)	1	0	1
2	36.84 (+-0.83)	0.25 (+-0.01)	1	1	0
2	37.19 (+-0.69)	0.22 (+-0.01)	1	1	1
2	36.23 (+-0.88)	0.19 (+-0.01)	1	1	2
2	<b>36.95 (+-0.71)</b>	<b>0.17 (+-0.01)</b>	<b>1</b>	<b>1</b>	<b>5</b>
2	35.84 (+-0.59)	0.15 (+-0.01)	1	1	10
2	33.34 (+-0.77)	0.14 (+-0.01)	1	1	20

Tabella 4.14: Grid search sugli iperparametri di focal distillation.

Prova ulteriore fatta a scopo di studio, è stata di sostituire la knowledge distillation loss con la feature distillation loss nella formula 2.2 per ottenere quella che è stata denominata **focal feature distillation**. Anche in questo caso è stata effettuata la grid search sul secondo task per il tuning degli iperparametri. Stavolta però il range dei valori  $\lambda$  è stato rimodulato conseguentemente al fatto che il valore ottimale di  $\lambda$  per feature distillation, come è possibile osservare dagli esperimenti del paragrafo 4.2.3.1, è risultato 0.1. Dalla grid search visibile in tabella 4.15, si ottiene la configurazione ottimale sul secondo task con  $\lambda = 0.1$ ,  $\alpha = 1$  e  $\beta = 2$ .

Nei grafici di fig. 4.24 e nella tabella riassuntiva 4.16, possiamo valutare a confronto sia la focal distillation standard che la focal feature distillation.

Task	avg_accuracy%	forgetting	lambda	alpha	beta
2	36.86 (+-0.49)	0.23 (+-0.01)	0.3	0	1
2	36.88 (+-0.55)	0.22 (+-0.00)	0.3	1	0
2	37.18 (+-0.62)	0.24 (+-0.01)	0.3	1	1
2	36.96 (+-0.58)	0.25 (+-0.01)	0.3	1	2
2	35.04 (+-0.77)	0.25 (+-0.01)	0.3	1	5
2	34.80 (+-0.69)	0.25 (+-0.01)	0.3	1	10
2	34.46 (+-0.81)	0.25 (+-0.01)	0.3	1	20
2	36.48 (+-0.65)	0.25 (+-0.01)	0.1	0	1
2	36.82 (+- 0.47)	0.25 (+-0.01)	0.1	1	0
2	36.22 (+- 0.83)	0.24 (+-0.01)	0.1	1	1
2	<b>37.02 (+-0.65)</b>	<b>0.22 (+-0.01)</b>	<b>0.1</b>	<b>1</b>	<b>2</b>
2	35.10 (+- 0.88)	0.26 (+-0.01)	0.1	1	5
2	34.64 (+- 0.64)	0.27 (+-0.02)	0.1	1	10
2	34.04 (+- 0.78)	0.28 (+-0.02)	0.1	1	20
2	36.22 (+- 0.55)	0.25 (+-0.01)	0.08	0	1
2	35.90 (+- 0.75)	0.26 (+-0.02)	0.08	1	0
2	36.84 (+- 0.51)	0.22 (+-0.00)	0.08	1	1
2	36.95 (+- 0.62)	0.22 (+-0.01)	0.08	1	2
2	35.16 (+- 0.70)	0.26 (+-0.01)	0.08	1	5
2	34.14 (+- 0.59)	0.26 (+-0.01)	0.08	1	10
2	34.26 (+- 0.42)	0.26 (+-0.01)	0.08	1	20

Tabella 4.15: Grid search effettuata sugli iperparametri di focal feature distillation.

Dai grafici, si nota bene come entrambi gli approcci riportino risultati molto promettenti in quanto si riesce ad avere un'accuracy migliore del fine tuning e, allo stesso tempo, si riesce addirittura a portare il forgetting a livelli simili al joint incremental per quel che riguarda focal feature distillation, e addirittura a livelli inferiori per quel che riguarda focal distillation. Questo risultato sul forgetting è dovuto al fatto che la loss di focal distillation è mirata proprio a vincolare il modello student sulle predizioni corrette del modello teacher andando, di fatto, a incrementare drasticamente i true positive in comune e quindi abbassando la metrifica del forgetting. Seppur valida, la variante focal feature distillation è tuttavia risultata in questi esperimenti inferiore alla focal distillation standard sia in termini di accuracy che di forgetting.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		<i>63.01 (+-0.23)</i>	<i>0.00 (+-0.00)</i>
	<i>2</i>	<i>38.42 (+-0.65)</i>	<i>0.26 (+-0.01)</i>
	<i>5</i>	<i>51.29 (+-0.29)</i>	<i>0.18 (+-0.00)</i>
<i>Joint_incremental</i>	<i>10</i>	<i>59.67 (+-0.11)</i>	<i>0.15 (+-0.00)</i>
	<i>2</i>	<i>32.87 (+-0.84)</i>	<i>0.34 (+-0.01)</i>
	<i>5</i>	<i>40.17 (+-0.53)</i>	<i>0.27 (+-0.01)</i>
<i>Fine_tuning</i>	<i>10</i>	<i>45.40 (+-0.39)</i>	<i>0.23 (+-0.00)</i>
	<i>2</i>	<i>36.95 (+-0.98)</i>	<i>0.16 (+-0.02)</i>
	<i>5</i>	<i>43.58 (+-0.61)</i>	<i>0.15 (+-0.01)</i>
<i>Focal distillation</i> <i>l=1, alpha=1, beta=5</i>	<i>10</i>	<i>48.87 (+-0.53)</i>	<i>0.15 (+-0.00)</i>
	<i>2</i>	<i>34.25 (+-0.92)</i>	<i>0.24 (+-0.01)</i>
	<i>5</i>	<i>40.68 (+-0.88)</i>	<i>0.19 (+-0.01)</i>
<i>Focal fd</i> <i>l=0.1, alpha =1, beta =2</i>	<i>10</i>	<i>46.56 (+-0.51)</i>	<i>0.17 (+-0.01)</i>

Tabella 4.16: Risultati numerici riassuntivi di focal distillation e della sua variante focal feature distillation.

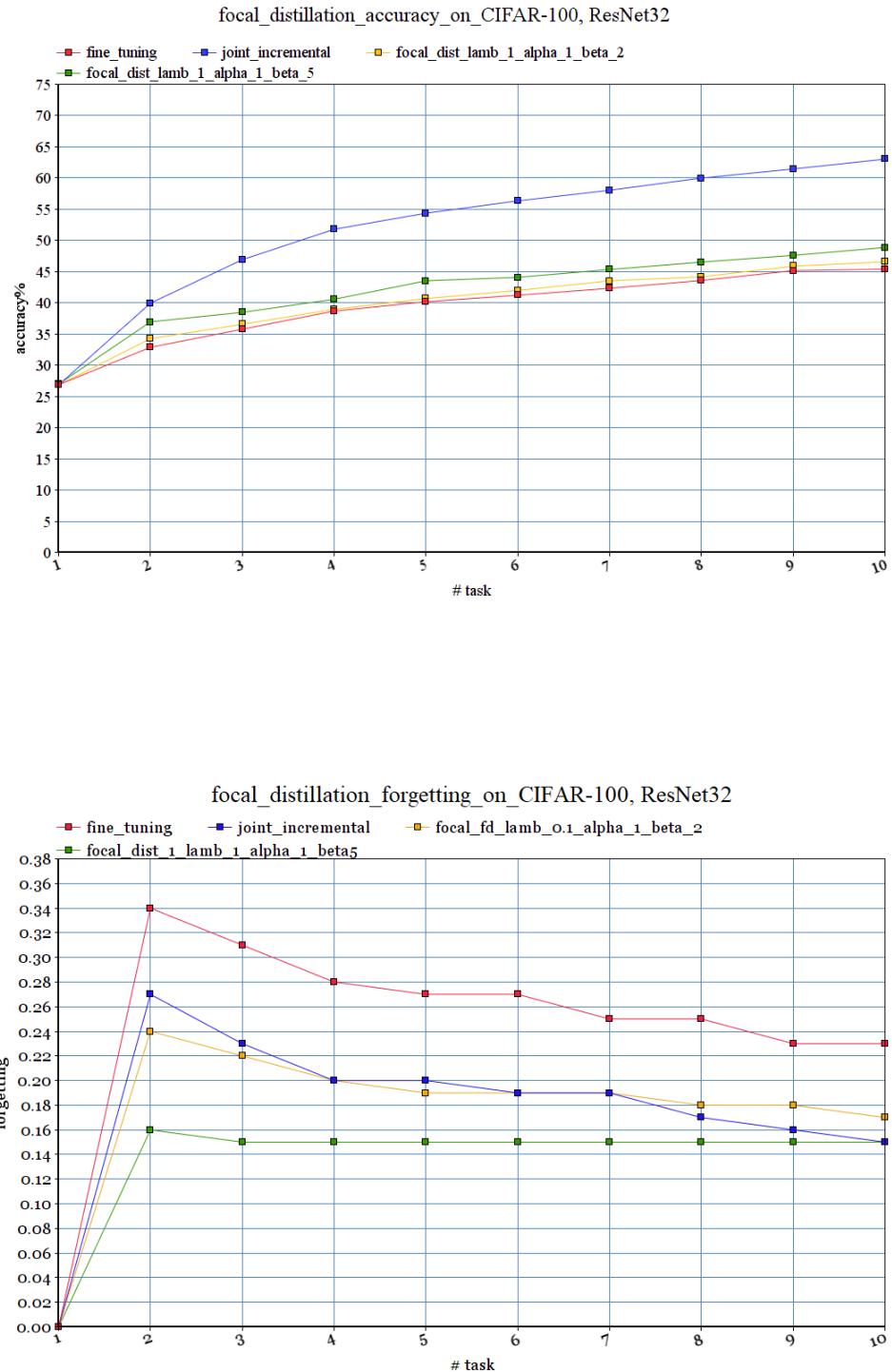


Figura 4.24: Accuracy (4.22) e forgetting (4.23) dell’approccio focal distillation con  $\lambda = 1$ ,  $\alpha = 1$  e  $\beta = 5$  e della sua variante focal feature distillation con  $\lambda = 0.1$ ,  $\alpha = 1$  e  $\beta = 2$ .

**Valutazioni con gli exemplar** I risultati ottenuti in combinazione con gli exemplar sono visibili in fig.4.25 e consultabili in forma riassunta nella tabella 4.17. In particolare, si nota come con gli exemplar le performance in termini di accuratezza si assomigliano sempre di più tra la variante focal feature distillation e focal distillation standard. Quest’ultima tuttavia rimane migliore in termini di accuratezza e soprattutto per quanto riguarda la metrica del forgetting.

<i>Exemplar table</i>			
<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		63.01 (+-0.23)	0.00 (+-0.00)
	2	38.42 (+-0.65)	0.26 (+-0.01)
	5	51.29 (+-0.29)	0.18 (+-0.00)
<i>Joint_incremental</i>	10	59.67 (+-0.11)	0.15 (+-0.00)
	2	34.97 (+-1.30)	0.30 (+-0.01)
	5	43.27 (+-0.59)	0.24 (+- 0.01)
<i>Fine_tuning</i>	10	48.73 (+-0.29)	0.21 (+-0.00)
	2	37.08 (+-0.85)	<b>0.15 (+-0.00)</b>
	5	44.78 (+-0.66)	<b>0.15 (+-0.01)</b>
<i>Focal distillation</i> <i>l=1, alpha=1, beta=5</i>	10	<b>50.37 (+-0.41)</b>	<b>0.15 (+-0.01)</b>
<i>Focal fd</i> <i>l =0.1, alpha = 1, beta =2</i>	2	<b>37.88 (+-0.60)</b>	0.19 (+-0.01)
	5	<b>44.92 (+-0.32)</b>	0.17 (+-0.01)
	10	49.77 (+- 0.29)	0.16 (+-0.00)

Tabella 4.17: Risultati numerici riassuntivi, nel caso con exemplar, di focal distillation e focal feature distillation.

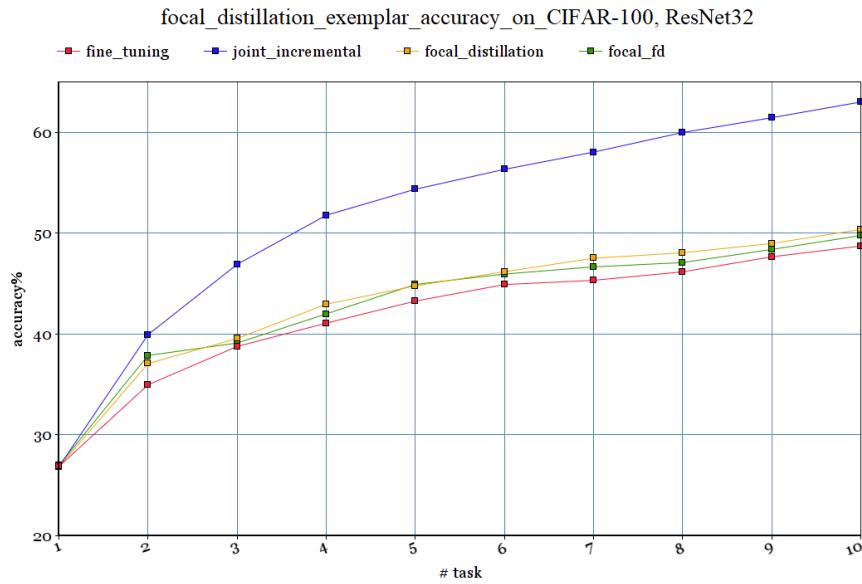


Figura 4.25: Accuracy con exemplar, di focal distillation con  $\lambda = 1$ ,  $\alpha = 1$  e  $\beta = 5$  e focal fd con  $\lambda = 0.1$ ,  $\alpha = 1$  e  $\beta = 2$ .

#### 4.2.4 Riepilogo e confronto approcci

Di seguito è presente la tabella 4.18 che riassume numericamente le performance degli approcci visti finora nei paragrafi precedenti e testati senza exemplar. E' immediato notare dalla tabella che focal distillation (in grassetto) risulta in assoluto, per CIFAR-100 con architettura di rete ResNet32, il miglior approccio tra quelli provati, sia in termini di accuratezza che di forgetting.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		63.01 (+-0.23)	0.00 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	38.42 (+-0.65)	0.26 (+-0.01)
	<b>5</b>	51.29 (+-0.29)	0.18 (+-0.00)
	<b>10</b>	59.67 (+-0.11)	0.15 (+-0.00)
<i>Fine_tuning</i>	<b>2</b>	32.87 (+-0.84)	0.34 (+-0.01)
	<b>5</b>	40.17 (+-0.53)	0.27 (+-0.01)
	<b>10</b>	45.40 (+-0.39)	0.23 (+-0.00)
<i>Feature distillation</i> <i>l=0.1</i>	<b>2</b>	34.02 (+-0.84)	0.28 (+-0.01)
	<b>5</b>	41.24 (+-0.50)	0.23 (+-0.01)
	<b>10</b>	46.87 (+-0.41)	0.19 (+-0.00)
<i>EWC</i> <i>l=1000</i>	<b>2</b>	28.90 (+-1.76)	0.30 (+-0.01)
	<b>5</b>	34.86 (+-0.91)	0.29 (+-0.01)
	<b>10</b>	38.38 (+-0.68)	0.24 (+-0.01)
<i>lwf</i> <i>l=1</i>	<b>2</b>	36.84 (+-0.85)	0.26 (+-0.01)
	<b>5</b>	42.37 (+-0.51)	0.22 (+-0.01)
	<b>10</b>	47.93 (+-0.27)	0.18 (+-0.00)
<i>Focal distillation</i> <i>l=1, alpha=1, beta=5</i>	<b>2</b>	<b>36.95 (+-0.98)</b>	<b>0.16 (+-0.02)</b>
	<b>5</b>	<b>43.58 (+-0.61)</b>	<b>0.15 (+-0.01)</b>
	<b>10</b>	<b>48.87 (+-0.53)</b>	<b>0.15 (+-0.00)</b>
<i>Focal fd</i> <i>l=0.1, alpha =1, beta =2</i>	<b>2</b>	34.25 (+-0.92)	0.24 (+-0.01)
	<b>5</b>	40.68 (+-0.88)	0.19 (+-0.01)
	<b>10</b>	46.56 (+-0.51)	0.17 (+-0.01)

Tabella 4.18: Riepilogo dei risultati degli approcci provati su CIFAR-100 e ResNet32.

### Confronto con gli exemplar

Estendendo il riepilogo e il confronto al caso con gli exemplar, possiamo osservare i risultati in tabella 4.19.

Anche da questa tabella risulta che l'approccio migliore è focal distillation, seguito da LwF, focal feature distillation e feature distillation.

<i>Exemplar table</i>			
<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		63.01 (+-0.23)	0.00 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	38.42 (+-0.65)	0.26 (+-0.01)
	<b>5</b>	51.29 (+-0.29)	0.18 (+-0.00)
	<b>10</b>	59.67 (+-0.11)	0.15 (+-0.00)
<i>Fine_tuning</i>	<b>2</b>	34.97 (+-1.30)	0.30 (+-0.01)
	<b>5</b>	43.27 (+-0.59)	0.24 (+- 0.01)
	<b>10</b>	48.73 (+-0.29)	0.21 (+-0.00)
<i>Feature_distillation</i> $l=0.1$	<b>2</b>	34.89 (+-0.88)	0.24 (+-0.01)
	<b>5</b>	44.49 (+-0.51)	0.21 (+-0.00)
	<b>10</b>	50.02 (+-0.28)	0.19 (+-0.00)
<i>EWC</i> $l=1000$	<b>2</b>	30.74 (+-0.98)	0.28 (+-0.01)
	<b>5</b>	33.96 (+-0.42)	0.26 (+- 0.00)
	<b>10</b>	38.93 (+-0.29)	0.22 (+-0.00)
<i>lwf</i> $l=1$	<b>2</b>	37.03 (+-0.82)	0.23 (+-0.01)
	<b>5</b>	45.82 (+-0.78)	0.21 (+-0.01)
	<b>10</b>	49.96 (+-0.36)	0.18 (+-0.00)
<i>iCaRL standard</i> $l=0.1$	<b>2</b>	33.59 (+-1.38)	0.27 (+-0.02)
	<b>5</b>	39.67 (+-0.98)	0.27 (+-0.01)
	<b>10</b>	41.32 (+-0.62)	0.26 (+-0.01)
<i>iCaRL_NCM</i> $l=0.1$	<b>2</b>	33.56 (+-1.11)	0.24 (+-0.01)
	<b>5</b>	40.53 (+-0.95)	0.23 (+-0.01)
	<b>10</b>	46.19 (+-0.42)	0.20 (+-0.00)
<i>Focal distillation</i> $l=1, \alpha=1, \beta=5$	<b>2</b>	37.08 (+-0.85)	<b>0.15 (+-0.00)</b>
	<b>5</b>	44.78 (+-0.66)	<b>0.15 (+-0.01)</b>
	<b>10</b>	50.37 (+-0.41)	<b>0.15 (+-0.01)</b>
<i>Focal fd</i> $l=0.1, \alpha=1, \beta=2$	<b>2</b>	<b>37.88 (+-0.60)</b>	0.19 (+-0.01)
	<b>5</b>	<b>44.92 (+-0.32)</b>	0.17 (+-0.01)
	<b>10</b>	49.77 (+- 0.29)	0.16 (+-0.00)

Tabella 4.19: Riepilogo dei risultati dei vari approcci provati su CIFAR-100 e ResNet32 con gli exemplar.

Riassumendo, alla luce degli esperimenti svolti e dei risultati su CIFAR-100 e ResNet32, possiamo concludere che le metodologie di distillation sono un ottimo approccio per quel che riguarda lo scenario Data-Incremental. Focal distillation, in particolare, oltre ad essere la metodologia migliore dal punto di vista dell'accuracy, è un approccio che riesce molto bene a limitare il forgetting.

#### 4.2.5 Random Distillation Reboot

Di seguito sono proposti i risultati degli esperimenti sui vari metodi di distillation combinati con la metodologia random reboot proposta e introdotta nella sezione 3.4.

I risultati ottenuti con la metodologia proposta sono consultabili tramite i grafici a barre di fig.4.28 in cui, sia per l'accuracy che il forgetting, sono espressi i valori al task 10 dell'approccio standard e dell'approccio con la modifica random reboot. Ciò che si nota in prima analisi è che il reboot provoca un aumento del forgetting in quanto, andando a reinizializzare la testa, si perde una piccola parte di "conoscenza". La reinizializzazione della testa però comporta un miglioramento in termini di accuracy in alcuni casi (vedesi focal distillation reboot e focal fd reboot).

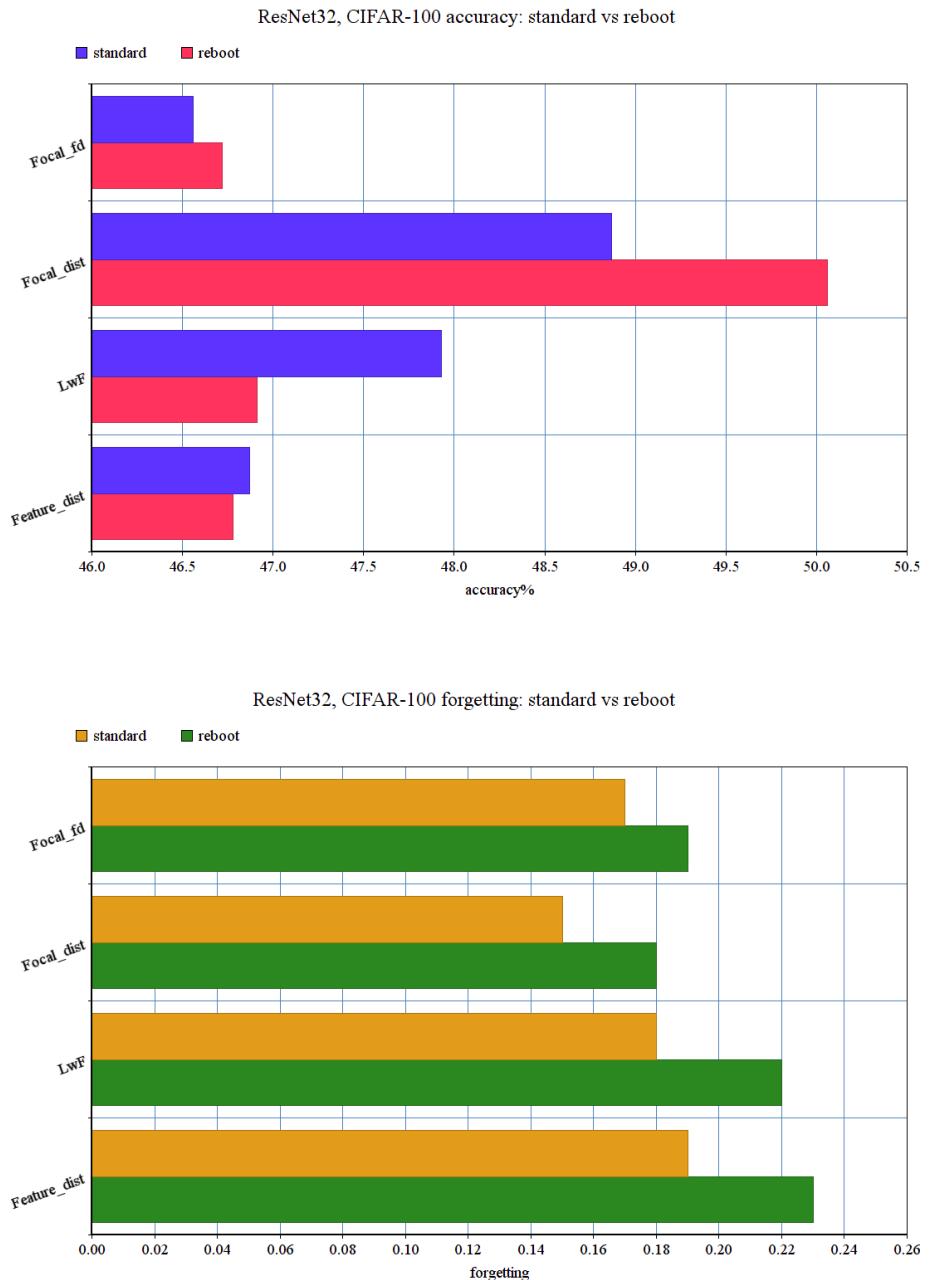


Figura 4.28: Comparazione, in termini di accuracy 4.26 e forgetting 4.27, dei metodi standard e dei metodi reboot.

### Confronto con gli exemplar

I risultati con gli exemplar sono rappresentati dai grafici a barre di fig.4.31 e ci mostrano uno scenario leggermente diverso; infatti, è comunque riscontrabile un aumento di forgetting (anche se più modesto rispetto al caso senza exemplar) ma, in termini di accuratezza, abbiamo un beneficio dato dal reboot per tutte le metodologie analizzate. Questi risultati sono spiegati dal fatto che gli exemplar vanno a mitigare la "perdita" dovuta alla reinizializzazione random mantenendo comunque i benefici di quest'ultima descritti nella sezione 3.4.

In conclusione, si può facilmente osservare, confrontando i risultati del grafico a barre con la tabella riassuntiva 4.19 espressa in precedenza, che *l'approccio focal distillation reboot ottiene le migliori performance in assoluto (50.06% vs 48.87% di focal distillation standard senza exemplar e 52.74% vs 50.37% di focal distillation con gli exemplar) ponendosi come punto di arrivo di tutta la sezione di esperimenti condotta con ResNet32 su CIFAR-100.*

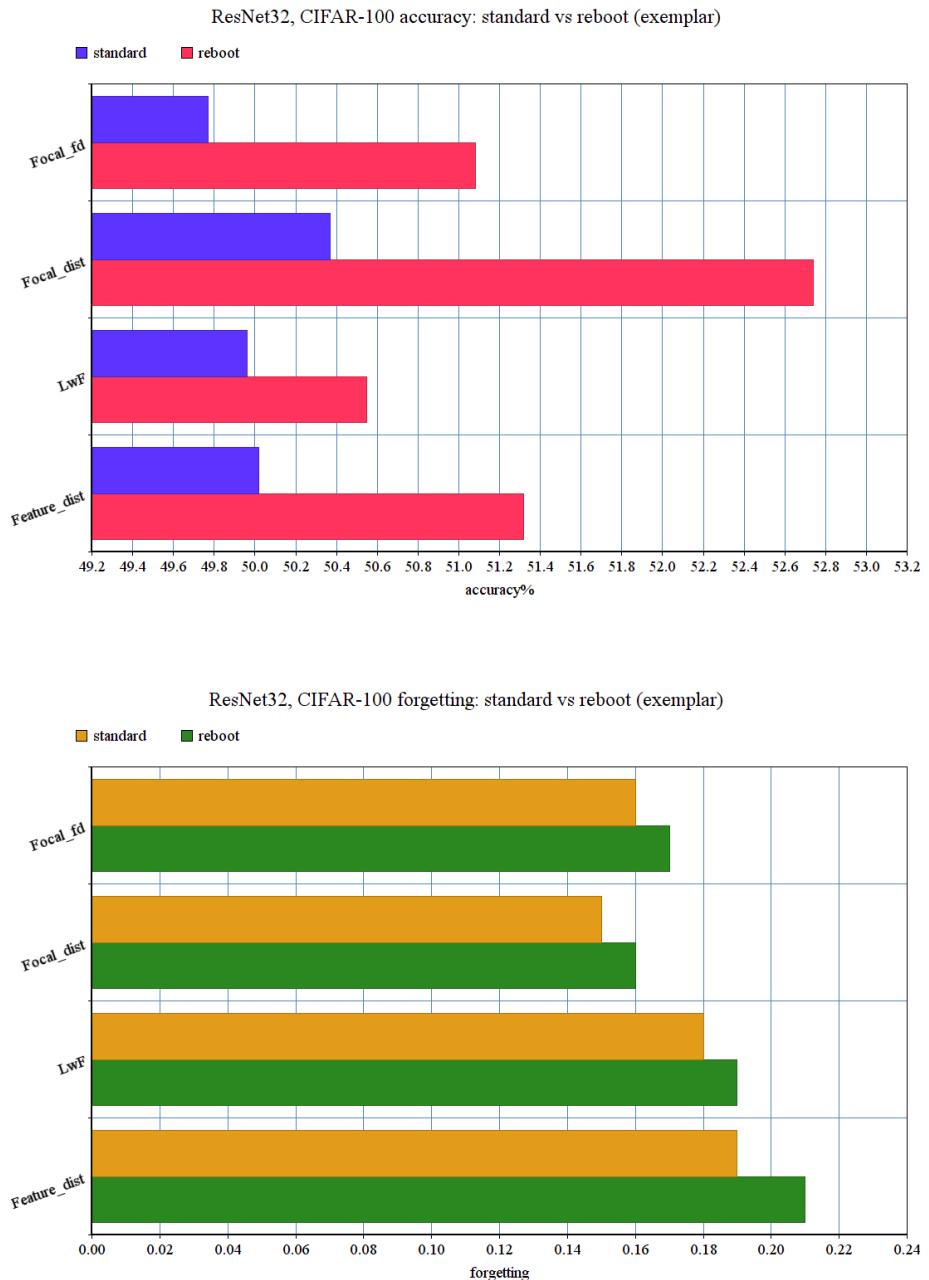


Figura 4.31: Comparazione con exemplar, in termini di accuracy 4.29 e forgetting 4.30, dei metodi standard e dei metodi reboot.

#### 4.2.6 Comparazione dei risultati con un’architettura più complessa: ResNet18

Per quanto riguarda ResNet18, per brevità, è riportato l’andamento dell’accuracy dei principali approcci senza exemplar nel grafico 4.32 e, maggiormente dettagliati nella tabella 4.20. Anche con ResNet18 si osserva l’efficacia degli approcci di distillation che diventa ancora più marcata per quel che riguarda gli approcci LwF e focal distillation. Quest’ultimo in particolare si conferma ancora una volta come il miglior approccio valutato.

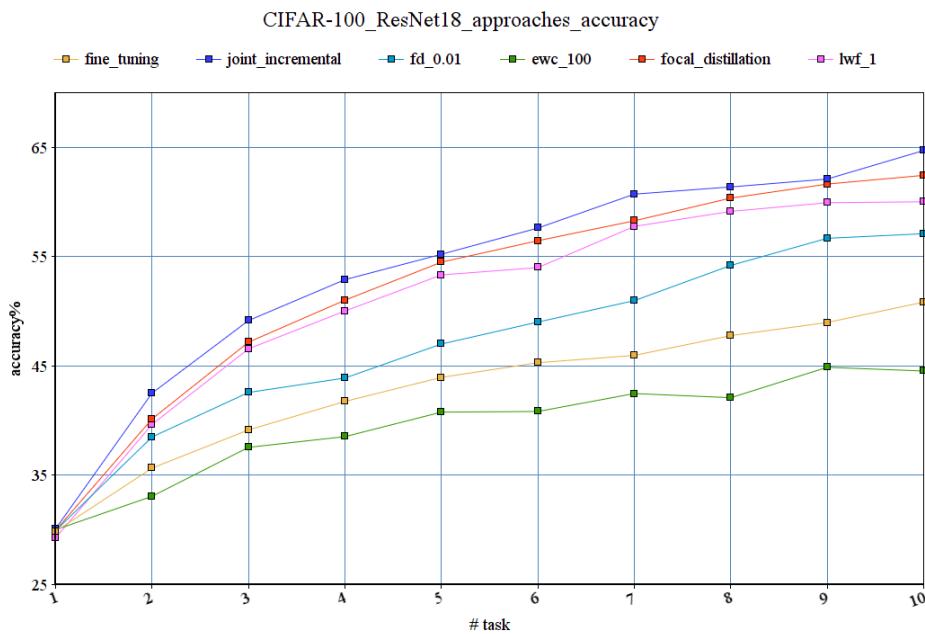


Figura 4.32: Confronto tra le accuracy dei principali approcci su ResNet18 e CIFAR-100.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		67.27 (+-0.03)	0.00 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	42.52 (+-0.65)	0.22 (+-0.00)
	<b>5</b>	55.22 (+-0.44)	0.16 (+-0.00)
	<b>10</b>	64.71 (+-0.39)	0.11 (+-0.00)
<i>Fine_tuning</i>	<b>2</b>	35.66 (+-0.74)	0.27 (+-0.03)
	<b>5</b>	43.96 (+-0.64)	0.26 (+-0.01)
	<b>10</b>	50.84 (+-0.32)	0.20 (+-0.01)
<i>Feature_distillation</i> <i>l=0.1</i>	<b>2</b>	38.02 (+-0.81)	0.24 (+-0.03)
	<b>5</b>	48.37 (+-0.47)	0.16 (+-0.01)
	<b>10</b>	57.10 (+-0.23)	0.14 (+-0.01)
<i>EWC</i> <i>l=1000</i>	<b>2</b>	33.08 (+-0.97)	0.26 (+-0.02)
	<b>5</b>	40.78 (+-0.70)	0.33 (+-0.02)
	<b>10</b>	44.54 (+-0.42)	0.32 (+-0.01)
<i>lwf</i> <i>l = 1</i>	<b>2</b>	39.66 (+-0.66)	0.24 (+-0.02)
	<b>5</b>	52.98 (+-0.42)	0.17 (+-0.01)
	<b>10</b>	60.02 (+-0.21)	0.15 (+-0.01)
<i>Focal distillation</i> <i>l=1, alpha =1, beta =10</i>	<b>2</b>	<b>40.16 (+-0.61)</b>	<b>0.23 (+-0.01)</b>
	<b>5</b>	<b>54.50 (+-0.42)</b>	<b>0.15 (+-0.01)</b>
	<b>10</b>	<b>62.44 (+-0.30)</b>	<b>0.11 (+-0.00)</b>
<i>Focal fd</i> <i>l=0.01, alpha =1, beta =10</i>	<b>2</b>	39.12 (+-0.79)	0.24 (+-0.01)
	<b>5</b>	49.82 (+-0.48)	0.16 (+-0.01)
	<b>10</b>	59.07 (+-0.21)	0.15 (+-0.00)

Tabella 4.20: Riepilogo degli approcci provati su CIFAR-100 e ResNet18.

I risultati estesi agli altri approcci, sono consultabili dalla tabella riassuntiva 4.21 di paragone con ResNet32 per quanto riguarda l'accuratezza, e dalla tabella 4.22 per quanto riguarda il forgetting. Da questa tabella si nota che approcci come *focal distillation*(e la sua modifica reboot) scalano piuttosto bene su architetture più complesse arrivando ad assottigliare molto il performance drift con il joint training. In particolare, utilizzando gli exemplar e l'approccio focal distillation reboot su ResNet18, si riesce addirittura ad arrivare a performance comparabili con il joint incremental che era sostanzialmente il nostro obiettivo prefissato.

<i>Approach</i>	<i>ResNet32</i>	<i>ResNet18</i>
<i>Fine tuning</i>	45.40 (+-0.39)	50.84 (+-0.53)
<i>Joint training</i>	63.01 (+-0.23)	67.27 (+-0.32)
<i>Joint incremental</i>	59.67 (+-0.11)	64.71 (+-0.40)
<i>Feature_distillation</i>	46.87 (+-0.41)	57.10 (+-0.61)
<i>Feature distillation reboot</i>	46.78 (+-0.22)	57.26 (+-0.46)
<i>lwf</i>	47.93 (+-0.27)	60.02 (+-0.33)
<i>lwf reboot</i>	46.91 (+-0.44)	58.36 (+-0.29)
<i>Focal distillation</i>	48.87 (+-0.53)	<b>62.44 (+-0.52)</b>
<i>Focal distillation reboot</i>	<b>50.06 (+-0.55)</b>	61.37 (+-0.44)
<i>Focal fd</i>	46.56 (+-0.51)	57.51 (+-0.40)
<i>Focal fd reboot</i>	46.72 (+-0.48)	56.88 (+-0.64)
<i>exemplars</i>		
<i>Fine_tuning</i>	48.73 (+-0.35)	53.69 (+-0.59)
<i>Feature_distillation</i>	50.02 (+-0.28)	58.64 (+-0.55)
<i>Feature distillation reboot</i>	51.32 (+-0.41)	61.34 (+-0.62)
<i>lwf</i>	49.96 (+-0.36)	61.22 (+-0.51)
<i>lwf reboot</i>	50.55 (+-0.29)	62.73 (+-0.48)
<i>Focal distillation</i>	50.37 (+-0.41)	63.65 (+-0.65)
<i>Focal distillation reboot</i>	<b>52.74 (+-0.44)</b>	<b>64.02 (+-0.59)</b>
<i>Focal fd</i>	49.77 (+-0.29)	59.07 (+-0.56)
<i>Focal fd reboot</i>	51.08 (+-0.20)	59.77 (+-0.49)

Tabella 4.21: Comparazione riassuntiva tra ResNet32 e ResNet18 delle performance di accuracy al task 10 dei vari approcci provati.

<i>Approach</i>	<i>ResNet32</i>	<i>ResNet18</i>
<b><i>Fine tuning</i></b>	0.23 (+-0.00)	0.20 (+-0.01)
<b><i>Joint incremental</i></b>	0,15 (+-0.00)	0.11 (+-0.00)
<b><i>Feature distillation</i></b>	0.19 (+-0.00)	0.14 (+-0.01)
<b><i>Feature distillation reboot</i></b>	0.23 (+-0.01)	0.17 (+-0.00)
<b><i>lwf</i></b>	0.18 (+-0.01)	0.14 (+-0.00)
<b><i>lwf reboot</i></b>	0.22 (+-0.00)	0.16 (+-0.00)
<b><i>Focal distillation</i></b>	<b>0.15 (+-0.00)</b>	<b>0.11 (+-0.00)</b>
<b><i>Focal distillation reboot</i></b>	0.18 (+-0.00)	0.14 (+-0.00)
<b><i>Focal fd</i></b>	0.17 (+-0.00)	0.14 (+-0.01)
<b><i>Focal fd reboot</i></b>	0.19 (+-0.00)	0.15 (+-0.00)
<i>exemplars</i>		
<b><i>Fine tuning</i></b>	0.21 (+-0.00)	0.19 (+-0.00)
<b><i>Feature distillation</i></b>	0.19 (+-0.00)	0.13 (+-0.00)
<b><i>Feature distillation reboot</i></b>	0.23 (+-0.00)	0.16 (+-0.01)
<b><i>lwf</i></b>	0.18 (+-0.00)	0.14 (+-0.00)
<b><i>lwf reboot</i></b>	0.19 (+-0.00)	0.13 (+-0.00)
<b><i>Focal distillation</i></b>	<b>0.15 (+-0.00)</b>	<b>0.11 (+-0.00)</b>
<b><i>Focal distillation reboot</i></b>	0.16 (+-0.00)	<b>0.11 (+-0.01)</b>
<b><i>Focal fd</i></b>	0.16 (+-0.00)	0.12(+0.01)
<b><i>Focal fd reboot</i></b>	0.17 (+-0.01)	0.13 (+-0.01)

Tabella 4.22: Comparazione riassuntiva tra ResNet32 e ResNet18 del forgetting al task 10 dei vari approcci provati.

### 4.3 Esperimenti su TinyImageNet

Per validare maggiormente i risultati ottenuti nella sezione 4.2, sono stati estesi gli esperimenti anche al sottoinsieme di ImageNet denominato *TinyImageNet* [3]. Come per CIFAR-100, sono stati effettuati 10 splits del dataset in modo da avere 10.000 esempi per split con 50 esempi per classe e i test sono stati svolti principalmente su ResNet32. Al termine della sezione, verrà svolta, come per CIFAR-100, una comparazione riassuntiva dei risultati con gli esperimenti aggiuntivi svolti su ResNet18.

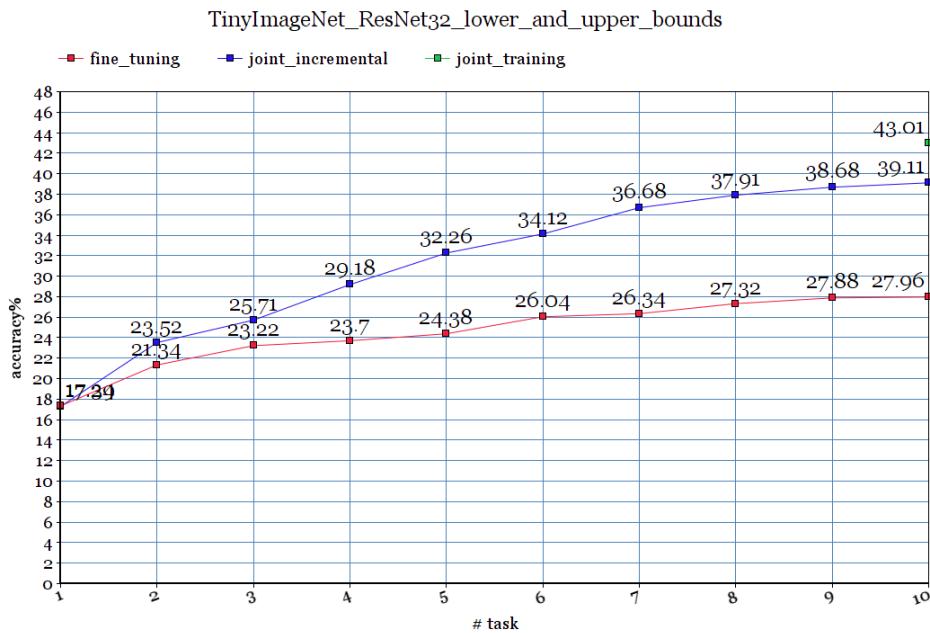


Figura 4.33: Valutazione della baseline (fine tuning) e degli upper bound (joint training, joint incremental) di TinyImageNet con ResNet32.

Dal grafico in fig.4.33, si nota come su questo dataset il drift delle performance tra la baseline del fine tuning e il joint training si aggiri intorno al 16% offrendo, come per CIFAR-100, ampi margini di lavoro per migliorare le performance attraverso i vari approcci incrementali.

### 4.3.1 Valutazione dei vari approcci

Il grafico in fig.4.34 dell'andamento dei principali appracci e la tabella 4.23 riassuntiva dei risultati conferma fondamentalmente i risultati visti su CIFAR-100, con gli approcci di distillation che portano vari benefici; focal distillation, in particolare, si riconferma il miglior approccio essendo capace di ridurre drasticamente il forgetting e allo stesso tempo ottenere le migliori performance.

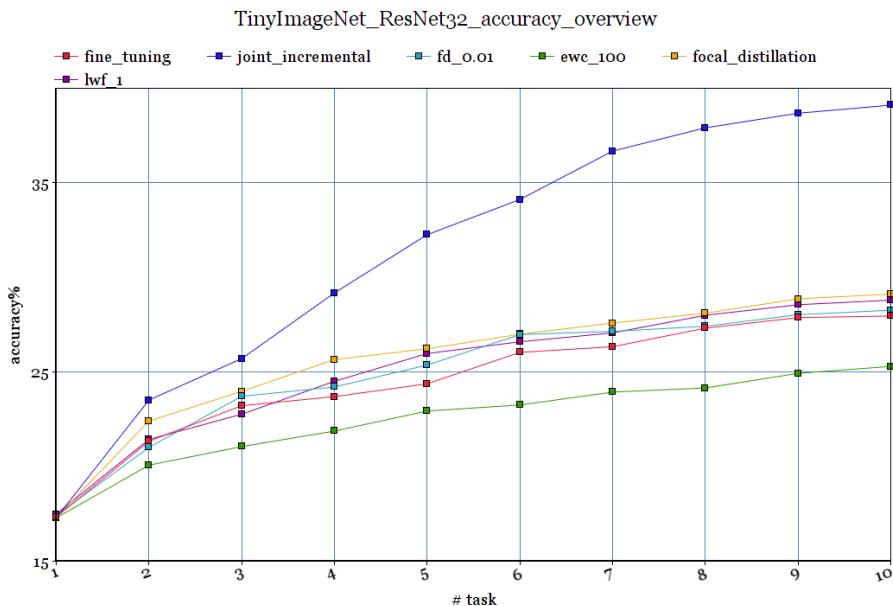


Figura 4.34: Confronto tra le accuracy dei principali approcci su TinyImageNet e ResNet32.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>		43.01 (+-0.03)	0.00 (+-0.00)
<i>Joint_incremental</i>	<b>2</b>	23.52 (+-0.52)	0.32 (+-0.01)
	<b>5</b>	32.26 (+-0.43)	0.27 (+-0.01)
	<b>10</b>	39.11 (+-0.27)	0.25 (+-0.01)
<i>Fine_tuning</i>	<b>2</b>	20.44 (+-0.94)	0.39 (+-0.03)
	<b>5</b>	24.30 (+-0.64)	0.34 (+-0.01)
	<b>10</b>	27.69 (+-0.34)	0.29 (+-0.01)
<i>Feature_distillation</i> <i>l=0.01</i>	<b>2</b>	21.02 (+-0.74)	0.38 (+-0.03)
	<b>5</b>	25.37 (+-0.37)	0.31 (+-0.01)
	<b>10</b>	28.27 (+-0.25)	0.27 (+-0.01)
<i>EWC</i> <i>l=100</i>	<b>2</b>	20.08 (+-0.97)	0.38 (+-0.04)
	<b>5</b>	22.94 (+-0.80)	0.33 (+-0.02)
	<b>10</b>	25.30 (+-0.44)	0.32 (+-0.01)
<i>lwf</i> <i>l = 1</i>	<b>2</b>	21.44 (+-0.66)	0.32 (+-0.02)
	<b>5</b>	25.98 (+-0.44)	0.29 (+-0.01)
	<b>10</b>	29.03 (+-0.31)	0.28 (+-0.01)
<i>Focal distillation</i> <i>l=0.1, alpha =1, beta =10</i>	<b>2</b>	<b>22.41 (+-0.58)</b>	<b>0.27 (+-0.03)</b>
	<b>5</b>	<b>26.23 (+-0.52)</b>	<b>0.26 (+-0.01)</b>
	<b>10</b>	<b>29.12 (+-0.30)</b>	<b>0.25 (+-0.01)</b>
<i>Focal fd</i> <i>l=0.01, alpha =1, beta =10</i>	<b>2</b>	22.12 (+-0.69)	0.34 (+-0.02)
	<b>5</b>	25.82 (+-0.48)	0.28 (+-0.01)
	<b>10</b>	28.62 (+-0.26)	0.25 (+-0.00)

Tabella 4.23: Riepilogo degli approcci provati su TinyImageNet e ResNet32.

### 4.3.2 Metodologia proposta: Random Distillation Reboot

Come per la sottosezione 4.2.5 relativa a CIFAR-100 , di seguito proponiamo l’analisi dei grafici a barre che comparano i metodi di distillation standard e i metodi di distillation con la modifica random reboot. In particolare, in figura 4.37 sono consultabili i grafici a barre relativi al caso senza exemplar e in figura 4.40 quelli relativi al caso con exemplar. I risultati ottenuti sono sostanzialmente allineati con quelli osservati su CIFAR-100 (4.2.5) , validando le nostre affermazioni circa l’efficacia della metodologia random distillation reboot, anche sul dataset TinyImageNet. Si nota facilmente infatti come, nel caso degli exemplar (fig.4.40), tutti i metodi reboot ottengano performance migliori. In particolare, focal distillation reboot risulta ancora una volta l’approccio migliore (29.96% vs 28.80% di focal distillation standard senza exemplar e 32.04% vs 31.37% di focal distillation con gli exemplar).

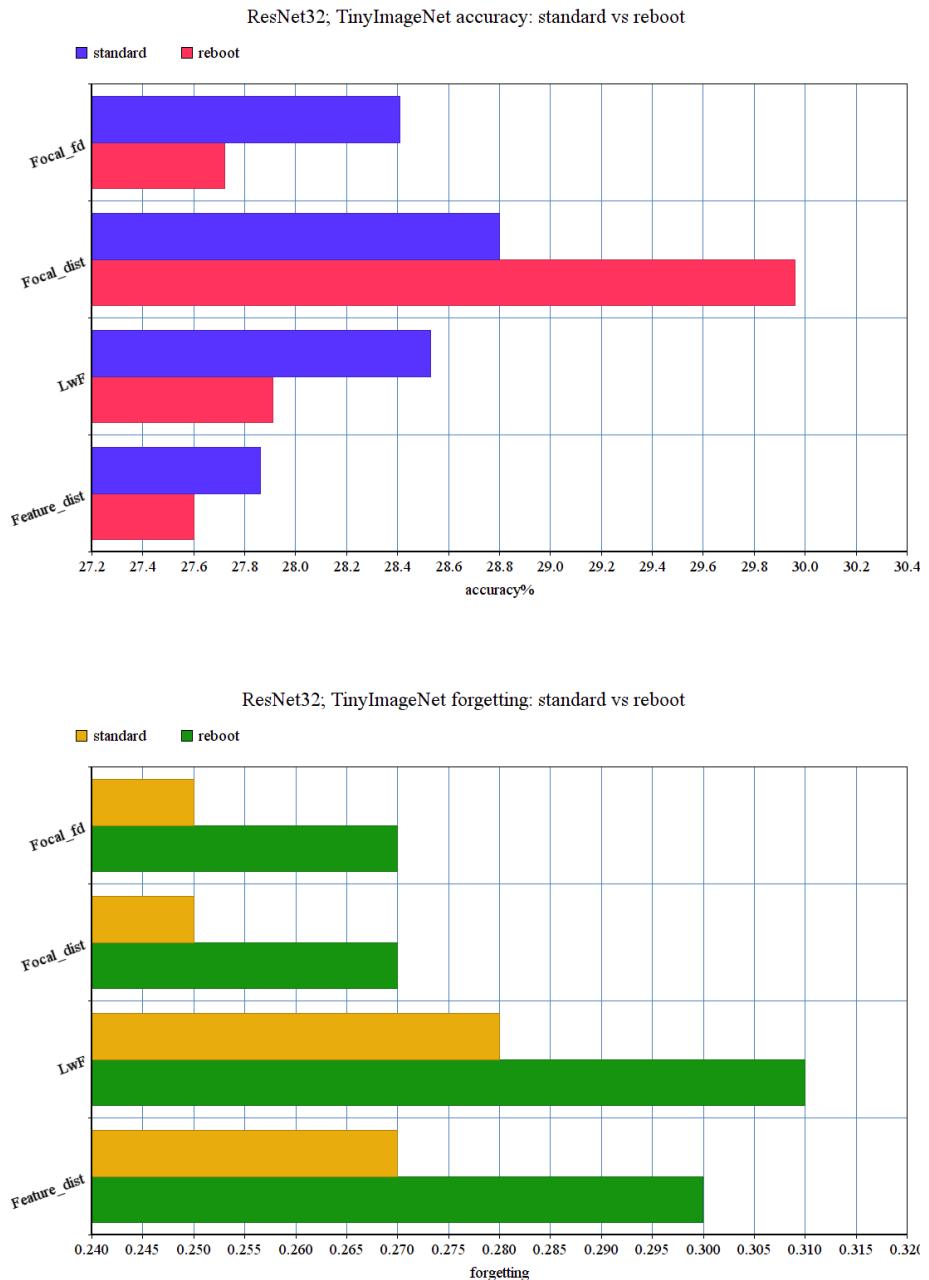


Figura 4.37: Comparazione, in termini di accuracy (4.35) e forgetting (4.36), dei metodi standard e dei metodi reboot.

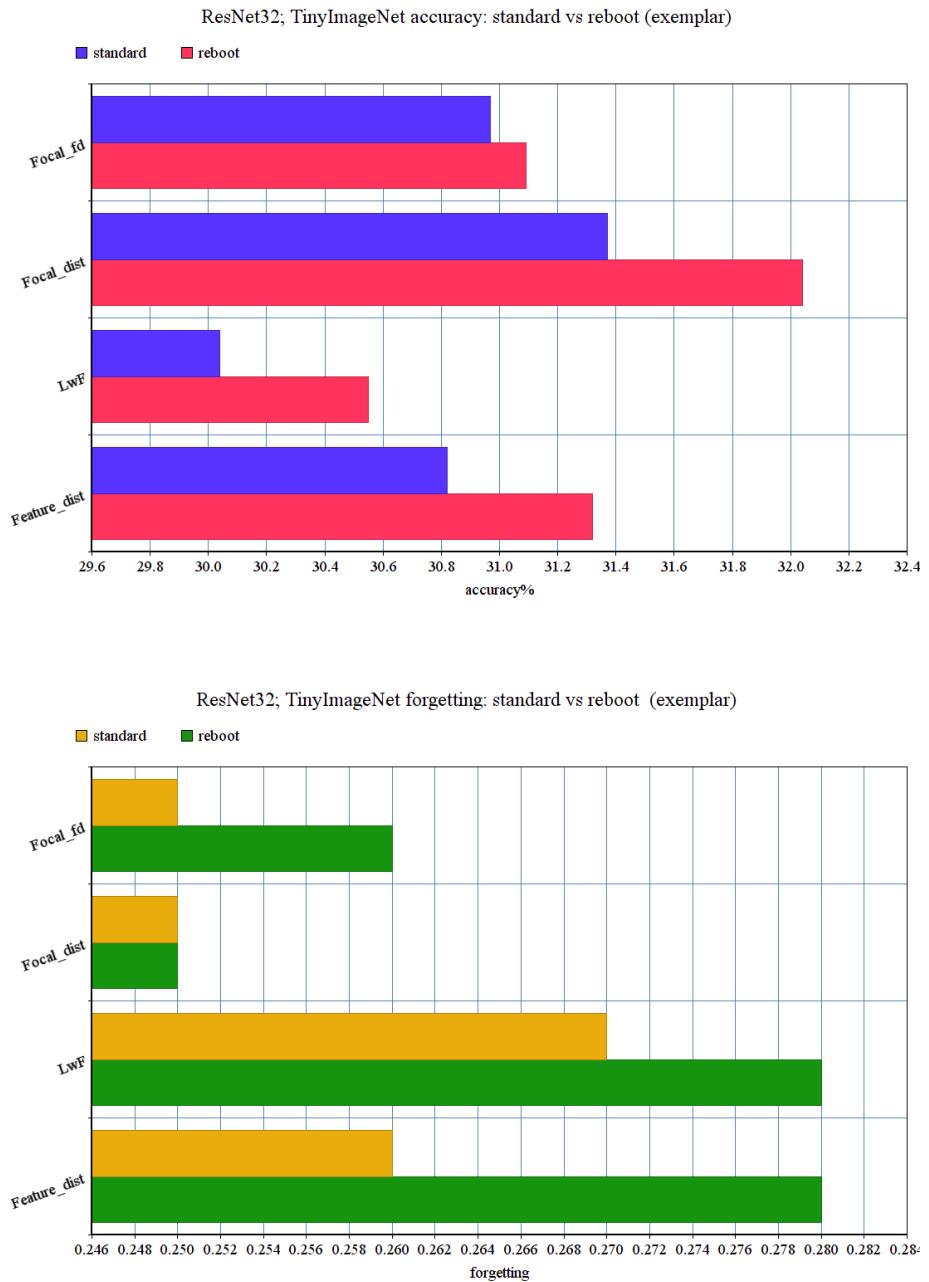


Figura 4.40: Comparazione con exemplar, in termini di accuracy (4.38) e forgetting (4.39), dei metodi standard e dei metodi reboot.

### 4.3.3 Comparazione dei risultati con un'architettura più complessa: ResNet18

Il grafico 4.41 mostra l'andamento delle performance degli approcci principali su ResNet18 mentre nella tabella riassuntiva 4.24 è possibile consultare numericamente i risultati con l'aggiunta del forgetting. Come per CIFAR-100 e TinyImageNet su ResNet32, focal distillation continua a risultare l'approccio più efficace seguito da LwF e feature distillation.

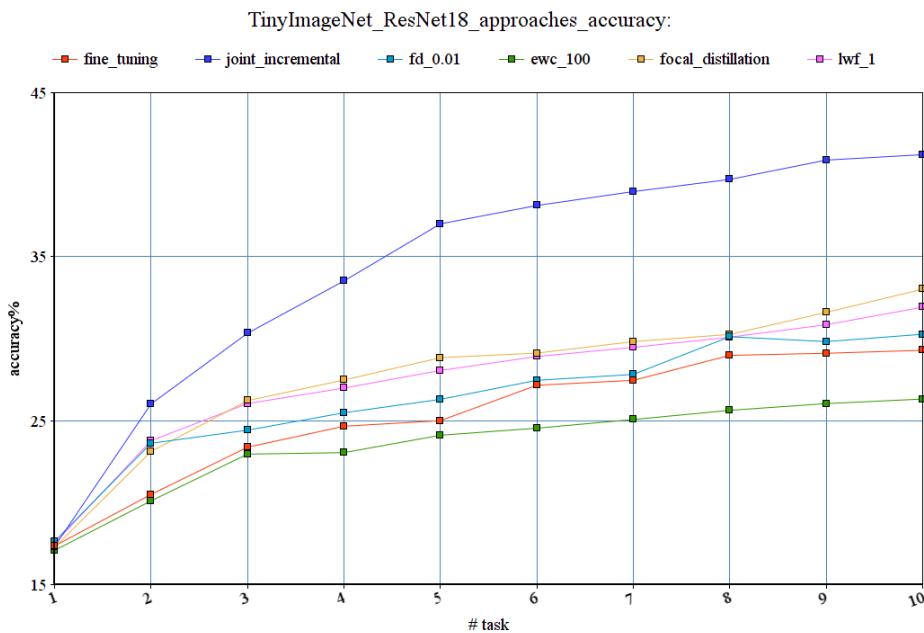


Figura 4.41: Grafico di confronto tra le accuracy dei principali approcci su TinyImageNet e ResNet18.

<i>Approach</i>	<i>After task</i>	<i>Avg_accuracy</i>	<i>Avg_forgetting</i>
<i>Joint_training</i>	<i>2</i>	<i>67.27 (+-0.03)</i>	<i>0.00 (+-0.00)</i>
<i>Joint_incremental</i>	<i>2</i>	<i>26.02 (+-0.55)</i>	<i>0.31 (+-0.01)</i>
	<i>5</i>	<i>37.00 (+-0.34)</i>	<i>0.26 (+-0.00)</i>
	<i>10</i>	<i>41.22 (+-0.29)</i>	<i>0.24 (+-0.00)</i>
<i>Fine_tuning</i>	<i>2</i>	<i>20.50 (+-0.54)</i>	<i>0.40 (+-0.02)</i>
	<i>5</i>	<i>25.00 (+-0.34)</i>	<i>0.34 (+-0.01)</i>
	<i>10</i>	<i>29.29 (+-0.22)</i>	<i>0.29 (+-0.01)</i>
<i>Feature_distillation</i> <i>l=0.01</i>	<i>2</i>	<i>23.62 (+-0.51)</i>	<i>0.36 (+-0.03)</i>
	<i>5</i>	<i>26.30 (+-0.40)</i>	<i>0.30 (+-0.01)</i>
	<i>10</i>	<i>30.27 (+-0.19)</i>	<i>0.27 (+-0.00)</i>
<i>EWC</i> <i>l=100</i>	<i>2</i>	<i>20.11 (+-0.57)</i>	<i>0.37 (+-0.02)</i>
	<i>5</i>	<i>24.11 (+-0.40)</i>	<i>0.33 (+-0.02)</i>
	<i>10</i>	<i>26.32 (+-0.18)</i>	<i>0.31 (+-0.01)</i>
<i>lwf</i> <i>l = 1</i>	<i>2</i>	<i>23.78 (+-0.56)</i>	<i>0.30 (+-0.02)</i>
	<i>5</i>	<i>28.06 (+-0.32)</i>	<i>0.28 (+-0.01)</i>
	<i>10</i>	<i>31.92 (+-0.15)</i>	<i>0.27 (+-0.01)</i>
<i>Focal distillation</i> <i>l=1, alpha =1, beta =10</i>	<i>2</i>	<i>23.52 (+-0.51)</i>	<i>0.26 (+-0.01)</i>
	<i>5</i>	<i>28.84 (+-0.42)</i>	<i>0.26 (+-0.01)</i>
	<i>10</i>	<i>33.02 (+-0.20)</i>	<i>0.24 (+-0.00)</i>
<i>Focal fd</i> <i>l=0.01, alpha =1, beta =10</i>	<i>2</i>	<i>23.12 (+-0.59)</i>	<i>0.31 (+-0.01)</i>
	<i>5</i>	<i>26.91 (+-0.38)</i>	<i>0.26 (+-0.01)</i>
	<i>10</i>	<i>30.57 (+-0.13)</i>	<i>0.24 (+-0.00)</i>

Tabella 4.24: Riepilogo degli approcci provati su TinyImageNet e ResNet18.

Nelle tabelle 4.25 e 4.26 invece, sono riassunti i risultati di tutti gli approcci provati su ResNet18 (incluse le metodologie reboot) a paragone con i risultati ottenuti con ResNet32. Questi risultati sono sostanzialmente un'ulteriore conferma che vede focal distillation reboot, con gli exemplar, come il miglior approccio per distacco, capace di assottigliare il performance drift con il joint incremental sia con ResNet32 che ResNet18 sul dataset TinyImagenet.

<i>Approach</i>	<i>ResNet32</i>	<i>ResNet18</i>
<b><i>Fine tuning</i></b>	27.69 (+-0.34)	29.29 (+-0.28)
<b><i>Joint training</i></b>	42.98 (+-0.18)	45.88 (+-0.24)
<b><i>Joint incremental</i></b>	39.11 (+-0.13)	41.22 (+-0.25)
<b><i>Feature_distillation</i></b>	27.86 (+-0.41)	30.27 (+-0.21)
<b><i>Feature distillation reboot</i></b>	27.60 (+-0.35)	30.33 (+-0.32)
<b><i>lwf</i></b>	28.53 (+-0.37)	31.92 (+-0.36)
<b><i>lwf reboot</i></b>	27.91 (+-0.24)	31.36 (+-0.29)
<b><i>Focal distillation</i></b>	28.80 (+-0.14)	33.02 (+-0.32)
<b><i>Focal distillation reboot</i></b>	<b>29.96 (+-0.31)</b>	31.39 (+-0.20)
<b><i>Focal fd</i></b>	28.41 (+-0.35)	31.51 (+-0.30)
<b><i>Focal fd reboot</i></b>	27.72 (+-0.28)	31.28 (+-0.19)
<i>exemplars</i>		
<b><i>Fine_tuning</i></b>	28.73 (+-0.25)	31.90 (+-0.19)
<b><i>Feature_distillation</i></b>	30.82 (+-0.28)	32.22 (+-0.25)
<b><i>Feature distillation reboot</i></b>	31.32 (+-0.21)	33.34 (+-0.22)
<b><i>lwf</i></b>	30.04 (+-0.36)	32.42 (+-0.31)
<b><i>lwf reboot</i></b>	30.55 (+-0.29)	32.81 (+-0.28)
<b><i>Focal distillation</i></b>	31.37 (+-0.31)	33.68 (+-0.35)
<b><i>Focal distillation reboot</i></b>	<b>32.04 (+-0.24)</b>	<b>35.02 (+-0.19)</b>
<b><i>Focal fd</i></b>	30.97 (+-0.29)	32.57 (+-0.26)
<b><i>Focal fd reboot</i></b>	31.09 (+-0.20)	33.77 (+-0.29)

Tabella 4.25: Riassunto dei risultati di accuracy ottenuti su TinyImageNet sia con ResNet32 che ResNet18.

<i>Approach</i>	<i>ResNet32</i>	<i>ResNet18</i>
<b><i>Fine tuning</i></b>	0.29 (+-0.00)	0.27 (+-0.01)
<b><i>Joint incremental</i></b>	0.24 (+-0.00)	0.23 (+-0.00)
<b><i>Feature distillation</i></b>	0.27 (+-0.00)	0.26 (+-0.01)
<b><i>Feature distillation reboot</i></b>	0.28 (+-0.01)	0.27 (+-0.00)
<b><i>lwf</i></b>	0.27 (+-0.01)	0.25 (+-0.00)
<b><i>lwf reboot</i></b>	0.28 (+-0.00)	0.26 (+-0.00)
<b><i>Focal distillation</i></b>	<b>0.24 (+-0.00)</b>	<b>0.23 (+-0.00)</b>
<b><i>Focal distillation reboot</i></b>	0.25 (+-0.00)	0.24 (+-0.00)
<b><i>Focal fd</i></b>	<b>0.24 (+-0.00)</b>	<b>0.23 (+-0.01)</b>
<b><i>Focal fd reboot</i></b>	0.25 (+-0.00)	0.24 (+-0.00)
<i>exemplars</i>		
<b><i>Fine tuning</i></b>	0.28 (+-0.00)	0.26 (+-0.00)
<b><i>Feature distillation</i></b>	0.26 (+-0.00)	0.25 (+-0.00)
<b><i>Feature distillation reboot</i></b>	0.27 (+-0.00)	0.25 (+-0.01)
<b><i>lwf</i></b>	0.26 (+-0.00)	0.24 (+-0.00)
<b><i>lwf reboot</i></b>	0.27 (+-0.00)	0.25 (+-0.00)
<b><i>Focal distillation</i></b>	<b>0.24 (+-0.00)</b>	<b>0.23 (+-0.00)</b>
<b><i>Focal distillation reboot</i></b>	<b>0.24 (+-0.00)</b>	<b>0.23 (+-0.01)</b>
<b><i>Focal fd</i></b>	<b>0.24 (+-0.00)</b>	<b>0.23 (+-0.00)</b>
<b><i>Focal fd reboot</i></b>	<b>0.24 (+-0.01)</b>	<b>0.23 (+-0.01)</b>

Tabella 4.26: Riassunto dei risultati del forgetting ottenuti su TinyImageNet sia con ResNet32 che ResNet18.

# Capitolo 5

## Conclusioni

In questo capitolo vengono presentate le considerazioni conclusive tratte in seguito alla realizzazione degli esperimenti e all’analisi dei risultati da questi evidenziati. Viene inoltre riportata una breve panoramica riguardante idee e possibili sviluppi futuri, nell’ottica di un’ulteriore prosecuzione del progetto e della ricerca associata.

### 5.0.1 Sintesi del lavoro e conclusioni

Riepilogando, in questo lavoro di tesi abbiamo introdotto per la prima volta in letteratura un particolare scenario di Continual Learning e posto l’attenzione su un determinato tipo di problemi. Sono stati poi presentati i lavori correlati tra cui la classificazione di immagini tramite reti convoluzionali, la metodologia di knowledge distillation nell’ambito della Computer Vision e soprattutto le metriche e gli approcci principali dello scenario Class-Incremental Learning. Si è entrati poi nel dettaglio sullo scenario Data Incremental Learning proposto, definendo delle metriche ad hoc e proponendo una metodologia atta a migliorare le performance in questo contesto.

Infine, è stato proposto un vasto capitolo relativo agli esperimenti in cui sono stati provati tutti gli approcci introdotti in precedenza (con alcune varianti) e la metodologia proposta, interrogandosi sui risultati ottenuti.

Brevemente, per quanto riguarda i risultati sperimentali:

- E' stata provata l'efficacia delle tecniche di distillation nel settore di ricerca del Data-Incremental Learning. In particolare, focal distillation, un approccio utilizzato nel campo dell'aggiornamento strutturale dei modelli, si è mostrato particolarmente adatto anche al contesto data-incremental con ottimi risultati.
- E' stato mostrato che la metodologia random reboot proposta funziona piuttosto bene, in particolar modo se abbinata agli exemplar, in sinergia con gli approcci di distillation. Focal distillation reboot, in particolare, è risultato globalmente il miglior approccio sperimentato, riducendo significativamente il drift di performance tra fine tuning e joint training (obiettivo principale di questo lavoro).
- Sono stati validati su due diversi dataset, CIFAR-100 e TinyImageNet, confrontando per ognuno due diverse architetture di rete, ResNet32 e ResNet18 e valutando l'effettiva generalità dei risultati

### 5.0.2 Sviluppi futuri

Essendo il data-incremental learning un campo di ricerca "nuovo e inesplorato", questo lavoro ha sostanzialmente posto delle basi solide per sviluppi futuri. Lo stesso codice, rilasciato liberamente su github<sup>1</sup> in allegato a questo lavoro di tesi per la replica degli esperimenti, è facilmente espandibile con nuovi approcci, dataset e architetture di rete.

---

<sup>1</sup>[https://github.com/emanuele-progr/incremental\\_data\\_learning](https://github.com/emanuele-progr/incremental_data_learning)

Per quanto riguarda gli approcci, ci sono varie metodologie di particolare interesse, presenti anche in FACIL<sup>2</sup>, che potrebbero essere valutate in questo scenario data-incremental. Tra questi, potrebbe essere interessante valutare la branca degli approcci di *bias-correction* e valutare se, anche nel data-incremental learning, è presente in maniera significativa un bias verso i task più recenti, inteso, in questo caso, non a livello di classi ma a livello di dati. Tra questi, sarebbe interessante valutare una modifica di iCaRL, denominata *BiC* [40], adattandola al data-incremental learning.

Similmente, osservando l'efficacia di LwF, si potrebbe valutare di adattare alcuni approcci derivati quali *LwM* (*Learning Without Memorizing*) [8], che sfrutta il meccanismo dell' *attenzione*, o *DMC* (*Deep Model Consolidation*) [45] che cerca di sopperire ad un'asimmetria di supervisione tra i nuovi e i vecchi dati. Sarebbe interessante anche provare a valutare la focal distillation loss in un'architettura multi-model, come proposto dall'approccio *M2KD* (*Incremental Learning via Multi-model and Multi-level Knowledge Distillation*) [48] che riporta significativi miglioramenti rispetto alla knowledge distillation standard.

Per quanto riguarda valutare altri dataset, l'argomento è piuttosto delicato in quanto ve ne sono veramente pochi adatti e liberamente accessibili. Un'idea, anche alla luce delle potenzialità del data-incremental learning in ambito medico introdotte nel capitolo 1, potrebbe essere quella di valutare qualche sottoinsieme di MedMNISTv2 [42] che è una collezione di dataset di immagini biomediche per valutare alcune patologie tra cui: patologie del colon, patologie polmonari, patologie del sangue e molto altro. Si potrebbe pensare inoltre anche di estendere l'analisi fatta su TinyImageNet a tutto ImageNet aumentando significativamente la complessità del task.

---

<sup>2</sup><https://github.com/mmasana/FACIL>

Per quanto riguarda invece l’architettura di rete, questo lavoro si è focalizzato sulle reti ResNet che costituiscono lo standard nel contesto della classificazione di immagini ma, ultimamente, la già introdotta architettura *Transformer* [36] sta emergendo sempre di più, soppiantando le reti ricorrenti e, in alcuni task, anche le reti convoluzionali. I Transformer sono architetture di rete proposte a partire dal 2017 che sfruttano il meccanismo dell’attenzione e che costituiscono lo stato dell’arte nei compiti di *NLP* (*Natural Language Processing*). Nello specifico, la recente modifica proposta, il *ViT* (*Vision Transformer*) [9], raggiunge performance molto promettenti anche nel campo della classificazione di immagini e sembra molto interessante anche per il nostro lavoro. Questo particolare Transformer divide l’immagine in input in varie *patches* che vengono trattate come tokens (parole) in un’applicazione NLP standard del Transformer.

Andare dunque a valutare performance e approcci incrementali su una rete concepita concettualmente in maniera diversa rispetto alle reti residuali è senza ombra di dubbio un possibile sviluppo futuro interessante. Questo cambio radicale di rete, unito all’implementazione di altri approcci di rilievo del class-incremental learning e ai dataset descritti in precedenza, contribuirebbe a fornire un quadro molto più completo su questo affascinante ambito di ricerca.

# Bibliografia

- [1] A broad neural network structure for class incremental learning. [https://www.researchgate.net/figure/Illustration-of-class-incremental-learning-system\\_fig1\\_325366026](https://www.researchgate.net/figure/Illustration-of-class-incremental-learning-system_fig1_325366026). Accessed: 2022-03-25.
- [2] Imagenet. <https://www.image-net.org/>. Accessed: 2022-03-05.
- [3] Tiny imangenet this is a miniature of imangenet classification challenge. <https://www.kaggle.com/c/tiny-imagenet>. Accessed: 2022-03-16.
- [4] Saul Carliner. An overview of online learning. 2004.
- [5] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.
- [6] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, 30, 2017.

- [7] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.
- [8] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019.
- [9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [10] Robert M French. Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference*, volume 1, pages 173–178, 1991.
- [11] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.
- [12] Jacob Goldberger, Shiri Gordon, Hayit Greenspan, et al. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV*, volume 3, pages 487–493, 2003.
- [13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE*

- conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [14] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.
  - [15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
  - [16] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
  - [17] Justin Johnson Fei-Fei Li Andrej Karpathy. Cs231n convolutional neural networks for visual recognition. 2015.
  - [18] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, pages 462–466, 1952.
  - [19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.
  - [20] Alex Krizhevsky. Cifar-10 and cifar-100 datasets. <https://www.cs.toronto.edu/~kriz/cifar.html>. Accessed: 2022-03-05.

- [21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- [22] Matthias De Lange. Towards adaptive ai with continual learning. *Leuven.AI*, 2021.
- [23] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.
- [24] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019.
- [25] Yong Luo, Liancheng Yin, Wenchao Bai, and Keming Mao. An appraisal of incremental learning methods. *Entropy*, 22(11):1190, 2020.
- [26] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.
- [27] Marc Masana, Xiaolei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification. *arXiv preprint arXiv:2010.15277*, 2020.
- [28] Marc Masana, Xiaolei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. Class-incremental learning: survey and performance evaluation on image classification, 2021.

- [29] Alessandro Piva. An overview on image forensics. *International Scholarly Research Notices*, 2013, 2013.
- [30] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.
- [31] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014.
- [32] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.
- [33] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 ways. *towardsdatascience*, 2018.
- [34] Héctor J Sussmann. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.
- [35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jegou. Training data-efficient image transformers amp; distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.
- [36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention

- is all you need. *Advances in neural information processing systems*, 30, 2017.
- [37] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
  - [38] Xing Wei, Shaofan Liu, Yaoci Xiang, Zhangling Duan, Chong Zhao, and Yang Lu. Incremental learning based multi-domain adaptation for object detection. *Knowledge-Based Systems*, 210:106420, 2020.
  - [39] Felix Wortmann and Kristina Flüchter. Internet of things. *Business & Information Systems Engineering*, 57(3):221–224, 2015.
  - [40] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 374–382, 2019.
  - [41] Sijie Yan, Yuanjun Xiong, Kaustav Kundu, Shuo Yang, Siqi Deng, Meng Wang, Wei Xia, and Stefano Soatto. Positive-congruent training: Towards regression-free model updates, 2021.
  - [42] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.
  - [43] NYU Corinna Cortes Google Labs New York Christopher J.C. Burges Microsoft Research Redmond Yann LeCun, Courant Institute. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>. Accessed: 2022-03-05.

- [44] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017.
- [45] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020.
- [46] Wei Zhang and Zuoxiang Zeng. Research progress of convolutional neural network and its application in object detection, 2020.
- [47] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S. Davis. M2KD: multi-model and multi-level knowledge distillation for incremental learning. *CoRR*, abs/1904.01769, 2019.
- [48] Peng Zhou, Long Mai, Jianming Zhang, Ning Xu, Zuxuan Wu, and Larry S. Davis. M2kd: Multi-model and multi-level knowledge distillation for incremental learning. 2019.