MINISTERO DELLO SVILUPPO ECONOMICO
DIREZIONE GENERALE PER LA LOTTA ALLA CONTRAFFAZIONE
UFFICIO ITALIANO BREVETTI E MARCHI

DOMANDA DI INVENZIONE NUMERO	**102018000007377**
Data Deposito	**20/07/2018**
Data Pubblicazione	**20/01/2020**

Classifiche IPC

Sezione	Classe	Sottoclasse	Gruppo	Sottogruppo
G	06	N	3	08

Sezione	Classe	Sottoclasse	Gruppo	Sottogruppo
G	06	N	3	04

Titolo

Rete neurale avente un numero ridotto di parametri

RETE NEURALE AVENTE UN NUMERO RIDOTTO DI PARAMETRI

Sfondo dell'invenzione

Campo dell'invenzione

La presente invenzione si riferisce al campo delle reti neurali.

Descrizione dell'arte correlata

Le reti neurali artificiali (Artificial Neural Networks, di seguito, brevemente "ANN") hanno visto un'esplosione di interesse negli ultimi anni e vengono applicate con successo in una vasta gamma di campi applicativi, comprendenti sistemi di controllo, robotica, sistemi di riconoscimento di schemi, previsioni, medicina, sistemi di potenza, produzione, ottimizzazione, elaborazione di segnale e scienze sociali / psicologiche .

Tra i campi di applicazione delle ANN, la classificazione di oggetti rappresentati in immagini (di seguito, semplicemente "classificazione di oggetti") è diventata sempre più importante con la crescita dell'uso di dispositivi digitali di acquisizione di immagini – quali smartphone o altri dispositivi portatili comprendenti una fotocamera digitale. La classificazione di oggetti è una procedura che prevede l'assegnazione di etichette ad oggetti raffigurati in una immagine (digitale) in base a classi di immagini predefinite. La classificazione di oggetti prevede la selezione di una classe appropriata per l'oggetto rappresentato nell'immagine tra le classi di immagine predefinite disponibili analizzando modelli visivi dell'immagine. In particolare, la classificazione di oggetti può essere basata sull'approccio conosciuto di apprendimento automatico (Machine Learning) comunemente noto come apprendimento profondo (deep learning) applicato alle ANN.

Come è ben noto agli esperti del settore, l'elemento base di una ANN è il neurone, detto anche nodo. Ogni neurone ha una uscita singola, ma può avere molti ingressi. L'uscita di un neurone è il risultato dell'applicazione di una funzione non lineare ad una combinazione lineare dei suoi ingressi a cui viene sommato un valore costante solitamente noto come bias. I coefficienti di questa combinazione lineare sono solitamente chiamati pesi w e la funzione non lineare è solitamente chiamata funzione di attivazione. Le ANN sono disposte secondo una sequenza di cosiddetti "strati" (layers). Ciascuno strato contiene un set corrispondente di neuroni. L'uscita di un neurone appartenente a uno strato può servire come ingresso per un neurone appartenente allo strato successivo della sequenza.

Come descritto per esempio in *Gradient-based learning applied to document recognition* di LeCun, Yann, et al., Proceeding of IEEE 86.11 (1998), una rete neurale convoluzionale (Convolutional Neural Network, di seguito, brevemente "CNN") è un tipo di ANN che è

particolarmente vantaggiosa da sfruttare nel campo della classificazione di oggetti. Per questo motivo, la maggior parte degli approcci effettivamente impiegati nel campo della classificazione di oggetti sono basati sulla CNN. Una CNN è una ANN comprendente almeno uno strato convoluzionale, cioè uno strato comprendente neuroni che condividono lo stesso insieme di pesi w (nel campo dell'amlisi di immagini, detto insieme di pesi viene generalmente indicato come "kernel" o "filtro") e le cui uscite sono date dalla convoluzione tra i suoi ingressi.

Facendo riferimento ad un esemplare immagine digitale in scala di grigi comprendente h^2 pixel disposti in h righe ed h colonne (immagine di ingresso), in cui a ciascun pixel è associato un corrispondente valore di pixel indicativo di un corrispondente valore di luminanza (ad esempio, maggiore è il valore del pixel, maggiore è la luminanza, ad esempio usando un 1 byte senza segno (unsigned), un valore 0 corrisponde alla luminanza inferiore e un valore 255 corrisponde a quello più alto) e un kernel che comprende k^2 pesi w disposti in una matrice con elementi $k \times k$, la convoluzione tra l' immagine di ingresso e il kernel prevede l'elaborazione dell'immagine per generare una cosiddetta mappa di caratteristiche (d'ora in avanti, "featuremap") comprendente una pluralità di caratteristiche, ciascuna caratteristica essendo associata ad una corrispondente area di k x k pixel (pixel sorgenti) dell'immagine in ingresso, eseguendo le seguenti operazioni:

- il kernel k x k è "sovrapposto" su una porzione k x k corrispondente dell'immagine in ingresso per avere ciascun pixel sorgente di detta porzione dell'immagine in ingresso che è associata ad un corrispondente elemento della matrice del kernel, con l'elemento centrale della matrice di kernel che è associata ad un pixel sorgente centrale di detta porzione;

- il valore in pixel di ciascun pixel incluso in detta porzione dell'immagine in ingresso viene pesato moltiplicandolo con il peso w corrispondente all'elemento associato della matrice di kernel;

- i valori dei pixel pesati sono sommati tra loro, e viene sommato un bias corrispondente;

- viene applicata una funzione di attivazione, ottenendo in questo modo una caratteristica associata alla porzione esaminata dell'immagine di ingresso; tale caratteristica viene salvata in una posizione della featuremap corrispondente al pixel sorgente centrale;

- il filtro viene spostato, orizzontalmente e verticalmente, di un passo ("stride") corrispondente ad un valore intero (ad esempio, 1 pixel);

- i passi precedenti sono ripetuti per coprire tutti i pixel dell'immagine in ingresso, in modo da ottenere una featuremap completa.

Generalizzando, lo strato convoluzionale di una CNN che ha come ingresso un segnale digitale quadrato $h \times h$ (anche generalmente definito come "struttura dati di ingresso" o semplicemente "struttura di ingresso") comprendente h^2 valori ottenuti campionando tale segnale

(come i summenzionati h^2 valori di pixel dell'immagine di ingresso digitale $h \times h$ in scala di grigi esemplificativa) ed avente un kernel comprendente k^2 pesi w disposti in una matrice quadrata con $k \times k$ elementi, produce in uscita un segnale digitale $(h - k + 1) \times (h - k + 1)$ che forma una struttura dati di uscita (featuremap) che comprende $(h - k + 1)^2$ valori (caratteristiche).

Considerazioni simili si applicano se la struttura dei dati di ingresso e / o la matrice di kernel hanno una forma diversa, ad esempio una forma rettangolare.

Con riferimento all'esempio considerato, i pesi w del kernel $k \times k$ possono essere impostati per rappresentare un particolare modello o da ricercare nella struttura di ingresso che rappresenta l'immagine in ingresso. In questo caso, l'uscita dello strato convoluzionale è una struttura dati corrispondente ad una featuremap avente $(h - k + 1) \times (h - k + 1)$ caratteristiche, in cui ciascuna caratteristica di detta featuremap può avere un valore che quantifica quanto tale particolare modello visivo sia presente in una porzione corrispondente dell'immagine di ingresso (ad esempio, maggiore è il valore della caratteristica nella featuremap, più tale particolare modello visivo è presente nella porzione corrispondente dell'immagine di ingresso). Questa operazione è ben nota dall'ingegneria della comunicazione, dove è nota come "rilevamento del segnale mediante filtri abbinati", vedi ad esempio *Modern electrical Communications* di H. Stark e FB Tuteur, Capitolo 11.6, Prentice-Hall, 1979 (pagine 484-488).

In un tipico strato convoluzionale di una CNN, la struttura di ingresso può essere formata da CH canali di dimensioni uguali. Facendo riferimento ad esempio al campo della classificazione di oggetti, una immagine digitale $h \times h$ a colori può essere rappresentata tramite un modello RGB con un set di $CH = 3$ canali diversi: il primo canale (canale R) è un'immagine digitale avente $h \times h$ pixel s e corrispondente alla componente rossa dell'immagine digitale colorata, il secondo canale (canale G) è un'immagine digitale avente $h \times h$ pixel e corrispondente alla componente verde dell'immagine digitale colorata ,e il terzo canale (canale B) è un'immagine digitale avente $h \times h$ pixel e corrispondente alla componente blu dell'immagine digitale colorata.

In genere, un insieme di NF kernel è utilizzato per un singolo strato convoluzionale. In questo caso, la struttura di uscita comprenderà a sua volta NF featuremaps.

Pertanto, considerando uno scenario generico, in cui la struttura di ingresso di uno strato convoluzionale è convoluta con NF kernel, il numero NP_{CL} di parametri apprendibili nello strato (pesi w più bias) è uguale a :

$$NP_{CL} = NC*NF*(k*k+ 1), \qquad (1).$$

dove *NC* è il numero di canali di ingresso dello strato convoluzionale.

Come si può leggere ad esempio in *OverFeat : Integrated Recognition, Localization and Detection using Convolutional Networks*, di Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus e Yann LeCun, arXiv preprint arXiv: 1312.6229, pagine 1-15, 2013, un algoritmo di classificazione di oggetti efficiente - cioè , un algoritmo in grado di classificare oggetti in classi corrette con un errore di classificazione basso - basato sulle CNN di solito comprende diversi strati convoluzionali, tipicamente interfogliati con strati di sottocampionamento (ad esempio, i cosiddetti strati di max-pooling), seguiti da una sequenza di strati finali, completamente connessi ("fully-connected", cioè non convoluzionali) che agiscono come classificatori finali che forniscono in uscita come struttura di uscita un vettore di classificazione che fornisce indicazioni su una classe selezionata corrispondente tra le classi di classificazione disponibili.

Il numero NP_{FC} di parametri apprendibili (pesi più bias) di un generico strato completamente connesso avente *NU* uscite e che elabora un numero *N* di ingressi ricevuti dal livello precedente è pari a:

$$NP_{FC} = NU * (N+1) \quad (2).$$

Un aspetto molto importante di una *CNN* riguarda il modo in cui sono impostati i pesi dei kernel dei vari strati convoluzionali. L'efficienza di un algoritmo di classificazione di oggetti che sfrutta una CNN è strettamente dipendente dai valori dei pesi *w*. Se i valori dei pesi *w* non sono impostati correttamente, gli oggetti sono classificati in classi sbagliate. Al fine di impostare i pesi *w* di una CNN, la CNN è sottoposta ad una procedura di addestramento, come la cosiddetta procedura di addestramento di retropropagazione ("backpropagation") mostrata ad esempio a pagina 153 di *Neural Networks and Learning Machines*, 3/E di Simon Haykin, Prentice Hall (18/11/2008)).

La procedura di retropropagazione prevede due fasi principali: la fase di andata ("forward") e la fase di ritorno ("backward").

La fase di andata prevede fornire in ingresso come struttura dati d'ingresso alla CNN da addestrare una immagine di addestramento appartenente ad una classe nota, e quindi confrontare la corrispondente uscita - cioè, il vettore di classificazione di uscita corrispondente agli effettivi valori di peso valori - con l'uscita nota corretta – ad esempio, un vettore di classificazione obiettivo corrispondente alla classe nota corretta. Poiché la CNN non è ancora addestrata, il vettore di classificazione di uscita sarà generalmente diverso dal vettore di classificazione obiettivo. Il vettore di classificazione di uscita è quindi confrontato con il vettore di classificazione obiettivo, e viene

calcolato un corrispondente termine di perdita ("loss term"), che fornisce una quantificazione di un errore di classificazione prodotto dalla CNN (cioè una quantificazione di quanto il vettore di classificazione di uscita è diverso dal vettore di classificazione obiettivo).

Avendo C classi diverse, e fornendo un'immagine di addestramento di ingresso x appartenente a una di dette classi C alla CNN, la CNN genera un corrispondente vettore di classificazione di uscita $y(x;W)$, in cui W rappresenta l'intero insieme di pesi w inclusi nella CNN (cioè, tutti i pesi w corrispondenti a tutti i neuroni in tutti gli strati della CNN). Il vettore di classificazione di uscita $y(x;W)$ generato dalla CNN è un vettore di C elementi $y_c(x;W)$, c = 1 - C , in cui ogni elemento $y_c(x;W)$ corrisponde a una rispettiva classe $IC(c)$ e ha un valore che fornisce la probabilità che tale immagine di addestramento di ingresso appartenga a tale specifica classe $IC(c)$. Il vettore di classificazione obiettivo $t(x)$ corrispondente ad un'immagine di addestramento x appartenente alla classe $IC(c^*)$ è una matrice di C elementi $t_c(x)$, c = 1 - C, in cui $t_c(x) = 1$ per c = c * e $t_c(x) = 0$ per $c \neq c^*$.

Allo scopo di calcolare il termine di perdita sopra menzionato, si può utilizzare una funzione di perdita ("loss function"). Una funzione di perdita $L(y(x;W),t(x))$ è una funzione che dipende dall'immagine di addestramento di ingresso x e dal corrispondente vettore di classificazione di uscita $y(x;W)$. A grandi linee, una funzione di perdita è una funzione che consente di stimare l'errore di classificazione. La funzione di perdita $L(y(x;W),t(x))$ è tale che, data una specifica immagine di addestramento di x e dato uno specifico vettore di classificazione di uscita $y(x;W)$ generato dalla CNN in risposta a questa immagine di addestramento dell'ingresso x, maggiore è l'errore di classificazione corrispondente a detto vettore di classificazione di uscita $y(x;W)$, maggiore è il valore di detta funzione di perdita $L(y(x;W),t(x))$.

Ad esempio, considerando il semplice problema di addestrare una rete a due uscite per prevedere il valore delle coordinate orizzontali e verticali di un punto su un piano, la funzione di perdita per l'addestramento di questa particolare rete potrebbe essere ragionevolmente definita come la distanza euclidea tra coordinate previste ed obiettivo del punto sul piano.

In generale, la funzione di perdita deve essere sempre definita come positiva o uguale a zero e la sua scelta effettiva dipende dal problema specifico che risolva la ANN o CNN. Considerando il problema specifico della classificazione di oggetti, tra le funzioni di perdita più utilizzate vi sono la cosiddetta funzione di perdita di errore quadratico medio (Mean Square Error, "MSE"), per la quale $L(y(x;W),t(x)) = \sqrt{\frac{1}{C}\sum_{c=1}^{C}(y_c(x;W) - t_c(x))^2}$ (Lehmann, Erich L. e George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006 , pag . 51), o la cosiddetta funzione di

perdita di cross-entropia, per la quale $L(y(x;W),t(x)) = -\sum_{c=1}^{C} t_c(x)\log(y_c(x;W))$, (de Boer, Pieter- Tjerk , Kroese , Dirk P .; Mannor , Shie ; Rubinstein, Reuven Y. (Febbraio 2005),"*A tutorial on the Cross-Entropy Method*". Annals of Operations Research. 134(1). pagine 19-67).

Dopo la definizione di una corretta funzione di perdita $L(y(x;W),t(x))$, una funzione di costo J(y(w, x), t(x)) può essere definita nel modo seguente:

$$J(W;x,t(x)) = \eta L(y(x;W),t(x)) + \lambda R(W) \quad (3)$$

dove η è un numero reale positivo chiamato tasso di apprendimento, λ è un numero reale positivo chiamato tasso di decadimento, e *R(W)* è una funzione di regolarizzazione. Come è ben noto agli esperti del settore, la funzione di regolarizzazione *R(W)* è una funzione che influenza la funzione di costo $L(y(x;W),t(x))$ fornendo una regola - ad esempio, attraverso lo stabilire vincoli appropriati - su come i pesi *W* della CNN dovrebbe essere distribuiti ed impostati. La regola data dalla funzione di regolarizzazione viene aggiunta per influenzare la stima dell'errore di classificazione data dalla funzione di perdita (y(x;W), t(x)).

Ad esempio, tale regola può prevedere il vincolo dell'avere i pesi *w* che giacciono su una ipersfera (funzione di regolarizzazione del tipo L2), o il vincolo di avere i pesi *w* che sono i più piccoli possibile (funzione regolarizzazione del tipo L1).

È noto agli esperti del settore che un effetto dell'uso di una funzione di regolarizzazione *R(W)* è quello di ridurre l'effetto del sovra-adattamento ("overfitting") della CNN sulle immagini di addestramento. Una CNN è detta sovra-adattarsi a dati di addestramento se, durante la procedura di addestramento, la CNN sta imparando, insieme alle informazioni reali, anche rumore indesiderato. La regolarizzazione, che fornisce un vincolo sul modo in cui i pesi della CNN sono impostati e distribuiti, aiuta a contrastare il sovra-adattamento, rendendo più nette le frontiere tra le classi (si veda Schölkopf , Bernhard e Alexander J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002, pages 87-120).

Sono note nell'arte diverse funzioni di regolarizzazione *R(W)* sono noti nell'arte, basati sulla norma di ordine *n* dei pesi *w*, come ad esempio la funzione di regolarizzazione L0 (si veda Guo, Kaiwen, et al "Robust non-rigid motion tracking and surface reconstruction using l0 regularization". *Proceedings of the IEEE International Conference on Computer Vision*, 2015 , pagine 3083-3091).

La funzione di regolarizzazione L1 (vedi Park, Mee Young e Trevor Hastie. "L1-regularization path algorithm for generalized linear models" *Journal of the Royal Statistical Society: Series B* (*Statistical Methodology*) 69.4 (2007): 659-677) , o la funzione di regolarizzazione L2 (si

veda Bilgic, Berkin, et al. "Fast image reconstruction with L2-regularization" *Journal of Magnetic Resonance Imaging* 40.1 (2014), pages 181-191).

La fase di ritorno della procedura di addestramento prevede il calcolo del gradiente della funzione di costo J(y(w, x), t(x)) rispetto ai pesi *w* per tutti gli strati dall'ultimo strato al primo strato.

La fase di andata e la fase di retropropagazione sono ripetute per un numero molto alto di volte, utilizzando un numero molto elevato di immagini di addestramento, per esempio prese da un database di addestramento.

Immediatamente dopo ogni fase di retropropagazione, i pesi *w* possono essere aggiornati utilizzando un'operazione di discesa del gradiente in modo da minimizzare la funzione di costo (W; x,t(x)). In questo modo, l'aggiornamento dei pesi *w* viene eseguito ogni volta che una nuova immagine di addestramento del database di addestramento è fornita in ingresso alla CNN da addestrare. In alternativa, l'aggiornamento dei pesi *w* può essere effettuato mediante l'applicazione in primo luogo delle fasi di andata e ritorno a tutte le immagini di addestramento del database di addestramento per produrre una media dei gradienti prima di aggiornare i pesi *w*. Si può anche prevedere un'alternativa intermedia, in cui l'aggiornamento dei pesi *w* viene effettuato ogni volta che le fasi in avanti e indietro vengono eseguite su un sottoinsieme (denominato in gergo "mini-batch") comprendente un numero fisso di immagini di addestramento del database di addestramento.

In questo contesto, con il termine "epoca" si intende il numero di volte in cui l'intero database di addestramento è stato fornito in ingresso alla CNN durante la procedura di addestramento. Di solito, sono necessarie centinaia di epoche per fare in modo che una CNN converga alla sua massima precisione di classificazione.

I pesi *w* sono tipicamente rappresentati da valori reali ospitati nella memoria principale del sistema di elaborazione (ad esempio, computer) utilizzato per addestrare la CNN attraverso la procedura di addestramento sopra menzionata. Tali valori possono essere salvati in modo permanente, ad esempio, nella memoria di sistema del sistema di elaborazione, o a passi intermedi della procedura di addestramento (ad esempio, dopo che ogni mini-batch è stato elaborato o dopo ogni epoca della procedura di addestramento), oppure alla fine della procedura di addestramento.

La conoscenza della topologia della CNN e dei relativi pesi appresi *w* permette di rappresentare completamente una CNN addestrata. Una volta una CNN è stata addestrata, la CNN addestrata può essere utilizzata sullo stesso computer per elaborare diversi campioni di dati o può essere trasferita su un altro computer .

L'addestramento di una CNN secondo la procedura sopra descritta è un processo ad alta intensità di risorse che è praticamente possibile solo grazie alla disponibilità di risorse hardware ad hoc. Tali risorse comprendono circuiti integrati (Integrated circuits, "IC") progettati ad-hoc, o matrici di porte logiche programmabili sul campo (Field Programmable Gate Array, "FPGA"), o unità di processo grafico (Graphical Processing Unit, "GPU") dotati di un gran numero di nuclei computazionali e, in maniera più importante, tale hardware comprende miliardi di unità di memorizzazione dati in forma di delle di memoria dedicate ad ospitare i pesi *w* appresi della CNN. Ad esempio, le moderne GPU progettate specificamente per l'addestramento di CNN profonde includono decine di gigabyte di memoria veloce e dedicata per l'archiviazione permanente del gran numero di pesi *w* di una CNN (e relativi gradienti di errore al tempo di retropropagazione).

La seguente tabella fornisce un conteggio di parametri (*W*, pesi più bias) e requisiti di memoria relativi per ciascuno dei 16 strati di un'architettura VGG esemplificata mostrata nell'articolo *Very deep convolutional networks for large scale image recognition* di Simonyan Karen e Andrew Zisserman, arXiv preprint arXiv : 1409.1556 (2104).

Strato	**Memoria**	**Parametri addestrabili**
INGRESSO: [224x224x3]	224*224*3=150K	0
CONV3-64: [224x224x64]	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64: [224x224x64]	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2:1112x112x641	112*112*64=800K	0
CONV3-128: [112x112x1281]	112*112*128=1.6M	(3*3*64)*128 = 73,728
CONV3-128: [112x112x1281]	112*112*128=1.6M	(3*3*128)*928= 147,456
POOL2: [56x56x1281]	56*56*128=400K	0
CONV3-256: [56x56x256]	56*56*256=800K	(3*3*128)*256 = 294,912
CONV3-256: (56x56x256]	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256: (56x56x256]	56*56*256=800K	(3*3*256)*256 = 589,824
POOL2: [28x28x256]	28*28*256=200K	0
CONV3-512: [28x28x512]	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]	28*28*512=400K	(3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]	28*28*512=400K	(3`3*512)*512 = 2,359,296
POOL2: [14x14x512]	1494*512=100K	0
CONV3-512: [14x14x512]	1494*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	1494*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]	1494*512=100K	(3*3*512)*512 = 2,359,296
POOL2: [7x7x512]	7*7*512=25K	0
FC: [1x1x4096]	4096	7*7*512*4096 = 102,760,448
FC: [1x1x4096]	4096	4096*4096 = 16,777,216
FC: [1x1x1000]	1000	4096*1000 = 4,096,000

La CNN include oltre 100 milioni di parametri addestrabili per un ingombro (footprint) totale di circa 132*k Megabyte (in cui k è la dimensione in byte di un parametro) di memoria per una CNN addestrata. La tabella mostra che la maggior parte dei parametri sono inclusi nel primo strato completamente connesso. Questo risultato non è sorprendente in quanto il numero di parametri del primo strato completamente connesso è proporzionale al numero di caratteristiche fornite in uscita dall'ultimo strato convoluzionale.

Al fine di migliorare la precisione della classificazione delle immagini ottenuta da una CNN, il numero di strati dovrebbe essere aumentato, ad esempio fino a 152. Un'architettura di questo tipo comporta l'addestramento di una quantità molto grande di pesi *w*, risultando così in requisiti di memoria maggiori, che possono facilmente superare la quantità di memoria disponibile sul sistema di elaborazione utilizzato per la procedura di addestramento. Per questo motivo, è noto utilizzare più GPU in parallelo per addestrare una CNN, dove ogni GPU allena un sottoinsieme della CNN (ad esempio, ogni GPU allena un sottoinsieme degli strati). Ad esempio, l'articolo *Outrageously large neural networks: The sparsely-gated mixture-of-experts layer* di Shazeer, Noam, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton e Jeff Dean, arXiv preprint arXiv: 1701.06538 (2017) descrive una CNN addestrata mediante una serie di GPU multiple disposte in cluster di elaborazione. Il numero totale di parametri di tale architettura è molto alto, pari a 137 miliardi.

Mentre le CNN sono solitamente addestrate mediante GPU di grado server per i motivi sopra citati, c'è un grande interesse nell'usare CNN precedentemente addestrate su dispositivi mobili, quali tablet o smartphone.

Nel seguito, il processo di utilizzare una CNN precedentemente addestrata su un dispositivo di elaborazione per la classificazione di immagini verrà indicato come "dispiegare" la CNN (precedentemente addestrata) su tale dispositivo di elaborazione. Il dispositivo di elaborazione su cui viene dispiegata la CNN può essere lo stesso dispositivo di elaborazione utilizzato per addestrare la CNN (ad esempio, un potente computer server) o potrebbe essere diverso (ad esempio, un PC portatile o uno smartphone). Deve essere apprezzato che le immagini di addestramento utilizzate per formare la CNN saranno generalmente diverse rispetto all'immagine che la CNN dovrebbe classificare quando dispiegata su un dispositivo di elaborazione per il suo uso normale.

Ad esempio, in un tipico scenario applicativo di classificazione di oggetti, un dispositivo mobile alimentato a batteria con funzionalità di comunicazione wireless potrebbe elaborare localmente i dati acquisiti da sensori locali tramite una CNN e prendere decisioni autonome senza comunicare con un server centrale e quindi risparmiare energia della batteria. In un altro scenario,

l'utente di un telefono cellulare potrebbe scattare una foto di un oggetto. Poi, una CNN pre-addestrata potrebbe essere utilizzata per determinare il tipo di oggetto e la marca eseguendo la CNN addestrata sulla GPU del dispositivo mobile. Infine, l'utente verrebbe reindirizzato tramite il browser web mobile ad un sito di e-shopping ove l'utente sarebbe in grado di acquistare l'oggetto, eventualmente confrontando tra le diverse opzioni possibili.

Dagli esempi sopra riportati, è chiaro che vi è un interesse ad essere in grado di dispiegare CNN addestrate su dispositivi mobili ed altri dispositivi embedded.

La presente invenzione affronta uno dei problemi principali che si presentano quando una rete addestrata viene dispiegata su smartphone ed altri dispositivi embedded.

Infatti, mentre i dispositivi mobili moderni tipicamente includono IC o GPU ad-hoc, che forniscono ad essi le capacità computazionali sufficienti per eseguire anche CNN complesse, tali dispositivi hanno una capacità di memoria di archiviazione limitata a causa di vincoli relativi al consumo di energia, al prezzo ed alle dimensioni fisiche. Allo stesso modo, la quantità di memoria di lavoro delle GPU che si trovano comunemente nei dispositivi mobili è in genere di un ordine di grandezza inferiore rispetto alla quantità di memoria di lavoro delle loro controparti server. Inoltre, nei dispositivi mobili spesso tale memoria non è memoria dedicata per l'uso esclusivo da parte della GPU, ma è condivisa dalla CPU centrale, che può utilizzare in modo permanente una parte significativa di essa per altre funzionalità di base del dispositivo.

In considerazione di quanto sopra, si può capire che le limitate capacità di memoria dei dispositivi mobili pongono un forte limite al numero massimo di parametri di una CNN, o di reti neurali generiche, quando la CNN (o una rete neurale generica) deve essere dispiegata su dispositivi mobili.

Al fine di ridurre i requisiti di memoria per dispiegare una CNN addestrata, in modo da consentire alla CNN di essere dispiegata anche in dispositivi avente una capacità di memoria con limitata, sono già state proposte diverse semplici soluzioni.

Ad esempio, secondo una prima soluzione nota (si veda Gupta, Suyog , et al. "Deep learning with limited numerical precision" *International Conference on Machine Learning,*, 2015), i parametri addestrati (che sono numeri reali) possono essere memorizzati con una precisione ridotta. Ad esempio, i parametri addestrati possono essere memorizzati secondo il formato a mezza precisione IEEE recentemente standardizzato piuttosto che con il tipico formato a precisione singola.

Un altro approccio conosciuto (si veda Han, Song, Jeff Pool, John Tran e William Dally. "Learning both weights and connections for efficient neural network" *In Advances in Neural*

Information Processing Systems, pagine 1135-1143. 2015) consiste nella semplificazione della topologia di rete (ad es, rimuovendo strati o unità dagli strati) quando la procedura di addestramento viene eseguita, fino a quando la CNN addestrata soddisfa le capacità del dispositivo mobile di destinazione su cui la CNN deve essere dispiegata.

Una possibile opzione per ridurre i requisiti di memoria di una CNN senza influire sulla sua topologia è eseguire una procedura di addestramento diretta ad aumentare la sparsità della CNN. Come è ben noto agli esperti del ramo, la sparsità di una CNN è definita come il rapporto tra il numero di parametri della CNN uguale a zero ed il numero totale di parametri della CNN (si veda *Tewarson, Reginald P. (Maggio 1973). Sparse Matrices (Part of the Mathematics in Science & Engineering series). Academic Press Inc pagine 1-13*).

Una CNN in cui un grande numero di parametri è uguale a zero, cioè con un'elevata sparsità, si dice essere sparsa. Più una CNN è sparsa, minore è la richiesta di spazio di memoria.

È stato osservato che se una procedura di addestramento viene eseguita utilizzando la funzione di regolarizzazione L1 summenzionata $R(w)$, diversi pesi w della CNN assumono valori molto bassi, vicini a zero.

L'articolo di Han Song, Jeff Pool, John Tran, e William Dally propone un approccio in tre fasi per la addestramento di topologie di rete sparse. In primo luogo, una rete è addestrata tramite retropropagazione, ma invece di impostare i pesi effettivi minimizzando qualche funzione di perdita bersaglio, viene eseguita una misura grossolana dell'importanza di ciascuna connessione. Quindi, tutte le connessioni con un indice di importanza al di sotto di una soglia vengono impostate a zero ("pruned", cioè sfoltite). Infine, la rete risultante viene addestrata con retropropagazione standard in modo tale da impostare i pesi effettivi.

In "Dropout: A simple way to Prevent Neural Networks from Overfitting" di Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky , Ilya Sutskever , Ruslan Salakhutdinov, *Journal of Machine Learning Research 15 (2014)*) viene mostrato come l'utilizzo del ritiro (dropout) migliora l'apprendimento e causa sparsificazione.

In "Sparse Convolutional Neural Networks"di Baoyuan Liu, Min Wang, Hassan Foroosh, 2015 IEEE, *Conference on Computer Vision and Pattern Recognition (CVPR)*, 7-12 giugno 2015, viene proposto un metodo di decomposizione per strati convoluzionali di una CNN, introducendo il concetto di "Rete Neurale Convoluzionale Sparsa". In particolare, convoluzioni 4-dimensionali sono trasformate in moltiplicazioni matriciali 2-dimensionali. Le matrici 2D prodotte da tale decomposizione proposta sono altamente sparse (si segnala una sparsità del 90%), implicando che esiste un'elevata ridondanza di parametri per kernel ad alta dimensione. Sulla base di questa

sparsità elevata, viene anche proposto un algoritmo di moltiplicazione di sparsità personalizzato per massimizzare gli hit della cache.

Sommario dell'invenzione

La Richiedente ha trovato che le soluzioni note sopra menzionate per ridurre l'occupazione di memoria di una CNN, o di una rete neurale generica, allo scopo di consentire a tale CNN di essere dispiegata su un dispositivo di elaborazione dotato di una scarsa capacità di memoria, non sono efficienti.

Ad esempio, la soluzione nota di Gupta, Suyog, et al, che prevede di memorizzare i parametri addestrati con una precisione ridotta, è stata osservata fornire solamente un risparmio di memoria molto scarso.

La soluzione nota di Han, Song, Jeff Pool, John Tran e William Dally che prevede di semplificare la topologia di rete quando la procedura di addestramento viene effettuata ha l'inconveniente di richiedere l'addestramento di una CNN separata avente una topologia di rete specifica (semplificata) per i dispositivi mobili. Inoltre, tale soluzione riduce anche le prestazioni della CNN o limita fortemente la dimensione della struttura dati di ingresso (ad esempio, l'immagine da classificare) che la CNN è in grado di elaborare.

Inoltre, la Richiedente ha rilevato che anche se pesi w molto bassi, ottenuti al termine di una procedura di addestramento che sfrutta la funzione di regolarizzazione L1 di cui sopra, sono impostati a zero (ad esempio, mediante un'operazione di sfoltimento basata su soglia), la prestazione della CNN sfoltita risultante è compromessa.

L'articolo di Han, Song, Jeff Pool, John Tran e William Dally rivela un approccio che offre una complessità ridotta per ottenere prestazioni migliori evitando la sovra parametrizzazione della rete. Tuttavia, questo approccio costringe l'esecuzione di un processo di raffinamento dopo la fase di apprendimento. Ciò porterà svantaggiosamente la nuova rete, meno parametrizzata, verso un minimo di energia che dipende fortemente dalla dinamica di apprendimento della rete "sovra parametrizzata". Inoltre, lo sfoltimento dei parametri viene eseguito senza tenere conto dell'attività reale che dipende dai parametri.

L'articolo di Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever , Ruslan Salakhutdinov mostra un approccio che produce una rete eccessivamente sovra parametrizzata.

Il documento di Baoyuan Liu, Min Wang, Hassan Foroosh, fornisce semplicemente risultati riguardanti la sparsità di strati convoluzionali.

In considerazione di quanto sopra, la Richiedente ha affrontato il problema di fornire una procedura di addestramento della rete neurale (e un corrispondente sistema di addestramento) che genera una rete neurale addestrata avente un ingombro di memoria ridotto (cioè, che richiede una quantità ridotta di memoria per memorizzare i suoi parametri) e allo stesso tempo è in grado di operare con prestazioni sufficientemente elevate.

La Richiedente ha trovato che questo può essere raggiunto modificando opportunamente la funzione di regolarizzazione della funzione di costo usata per addestrare una rete neurale.

A questo proposito, la richiedente ha ideato una funzione di regolarizzazione che promuove valori bassi dei pesi nella rete neurale senza ridurre le prestazioni della rete stessa tenendo conto di un tasso di variazione dell'uscita della rete neurale causato dalle variazioni dei pesi.

Un aspetto della presente invenzione riguarda un metodo.

Secondo una forma di realizzazione della presente invenzione, il metodo comprende fornire una rete neurale avente un insieme di pesi e configurata per ricevere una struttura di dati di ingresso per generare un vettore di uscita corrispondente secondo valori di detto insieme di pesi.

Secondo una forma di realizzazione della presente invenzione, il metodo comprende inoltre addestrare la rete neurale per ottenere una rete neurale addestrata.

Secondo una forma di realizzazione della presente invenzione, detto addestrare comprende impostare valori dell'insieme di pesi mediante un algoritmo di discesa del gradiente che sfrutta una funzione di costo comprendente un termine di perdita e un termine di regolarizzazione.

Secondo una forma di realizzazione della presente invenzione, il metodo comprende inoltre dispiegare la rete neurale addestrata su un dispositivo attraverso una rete di comunicazione.

Secondo una forma di realizzazione della presente invenzione, il metodo comprende inoltre usare la rete neurale addestrata dispiegata sul dispositivo.

Secondo una forma di realizzazione della presente invenzione, il termine di regolarizzazione è basato su un tasso di cambiamento di elementi del vettore di uscita causato da variazioni dell'insieme di valori di pesi.

Secondo una forma di realizzazione della presente invenzione, detto termine di regolarizzazione è basato su una somma di penalità ciascuna penalizzante un corrispondente peso dell'insieme di pesi, ciascuna penalità essendo basata sul prodotto di un primo fattore ed un secondo fattore.

Secondo una forma di realizzazione della presente invenzione, detto primo fattore è basato su una potenza di detto peso corrispondente, in particolare un quadrato del peso corrispondente.

Secondo una forma di realizzazione della presente invenzione, detto secondo fattore è basato

su una funzione di quanto è sensibile il vettore di uscita ad una variazione nel peso corrispondente.

Secondo una forma di realizzazione della presente invenzione, detta funzione corrisponde alla media di valori assoluti di derivate di elementi di uscita del vettore di uscita rispetto al peso.

Secondo una forma di realizzazione della presente invenzione, detto addestrare comprende, per ciascuna tra una pluralità di strutture di dati di ingresso di addestramento, confrontare il vettore di uscita generato dalla rete neurale secondo detta struttura di dati di ingresso di addestramento con un corrispondente vettore di uscita obiettivo che ha solo un elemento diverso da zero.

Secondo una forma di realizzazione della presente invenzione, detta funzione corrisponde al valore assoluto della derivata, rispetto al peso, di un elemento del vettore di uscita corrispondente a detto un elemento diverso da zero del vettore di uscita obiettivo.

Secondo una forma di realizzazione della presente invenzione, detto addestrare comprende calcolare un corrispondente peso aggiornato da ciascun peso dell'insieme di pesi sottraendo da detto peso:

- un primo termine basato sulla derivata del termine di perdita rispetto ai pesi, e
- un secondo termine basato sul prodotto tra detto peso e un'ulteriore funzione, detta ulteriore funzione essendo uguale a uno meno detta funzione se detta funzione non è maggiore di uno, ed essendo uguale a zero se detta funzione è maggiore di uno.

Secondo una forma di realizzazione della presente invenzione, detto addestrare comprende inoltre impostare a zero pesi dell'insieme di pesi aventi un valore inferiore a una soglia corrispondente.

Secondo una forma di realizzazione della presente invenzione, il metodo comprende inoltre impostare detta soglia ad uno selezionato tra:

- un valore di soglia basato sulla media dei pesi diversi da zero;
- un valore di soglia tale che un rapporto tra una prima dimensione di insieme ed una seconda dimensione di insieme sia uguale a una costante, in cui detta prima dimensione di insieme è il numero di pesi diversi da zero i cui valori assoluti sono minori o uguali a detto valore di soglia e detta seconda dimensione di insieme è uguale al numero di tutti i pesi diversi da zero.

Secondo una forma di realizzazione della presente invenzione, detto dispiegare la rete neurale addestrata su un dispositivo comprende inviare pesi diversi da zero dalla rete neurale addestrata al dispositivo attraverso detta rete di comunicazione.

Secondo una forma di realizzazione della presente invenzione, detto utilizzare la rete neurale addestrata dispiegata sul dispositivo comprende usare la rete neurale addestrata dispiegata con un'applicazione per una classificazione di oggetti visuale in esecuzione sul dispositivo.

Secondo una forma di realizzazione della presente invenzione, detto dispositivo è un dispositivo mobile.

Secondo una forma di realizzazione della presente invenzione, detto dispositivo è un dispositivo di elaborazione di un sistema di controllo di un veicolo a guida autonoma.

Breve descrizione dei disegni

Queste ed altre caratteristiche e vantaggi della presente invenzione saranno rese evidenti dalla seguente descrizione di alcune sue forme di realizzazione esemplificative e non limitative, da leggere unitamente ai disegni allegati, in cui:

la **Figura 1** illustra una porzione di una CNN atta ad essere utilizzata in una procedura di classificazione di oggetti in cui possono essere applicati i concetti secondo forme di realizzazione della presente invenzione;

la **Figura 2** è un diagramma di flusso che illustra in termini di blocchi funzionali una procedura di addestramento diretta ad impostare i pesi degli strati della CNN di **Figura 1** secondo una forma di realizzazione della presente invenzione;

le **Figure 3 - 5** sono diagrammi di flusso che illustrano in termini di blocchi funzionali le fasi principali di sottoprocedure del procedimento di addestramento di **Figura 2** secondo una forma di realizzazione della presente invenzione;

la **Figura 6** illustra in termini di blocchi funzionali molto semplificati uno scenario di applicazione esemplificativo della soluzione secondo forme di realizzazione della presente invenzione.

Descrizione dettagliata di forme di realizzazione esemplificative dell'invenzione

La Figura 1 illustra una porzione di una CNN **100** atta ad essere utilizzata in una procedura di classificazione di oggetti in cui possono essere applicati i concetti secondo forme di realizzazione della presente invenzione.

La CNN **100** è configurata per ricevere come ingresso un'immagine digitale **x** (immagine di ingresso) raffigurante un oggetto, e per selezionare una classe appropriata per l'oggetto rappresentato nell'immagine di ingresso **x** tra una pluralità di classi di immagini *C* predefinite *IC(c)* ($c = 1, 2, \ldots, C$), come ad esempio:

- *IC (1)* = classe di immagine *persona ;*

- *IC (2)* = classe di immagine *gatto;*
- *IC (3)* = classe di immagine *cane ;*
- *IC (4)* = classe di immagine *auto ;*
- *IC (5)* = classe di immagine *casa* ,
- ...

A tale scopo, la CNN **100** è progettata per elaborare l'immagine di ingresso **x** al fine di generare un corrispondente vettore di classificazione che fornisce un'indicazione circa una classe di immagine selezionata *IC(c)* tra quelle predefinite disponibili. Il vettore di classificazione sarà indicato con *y(x;W)*. Qui, *W* rappresenta l'insieme di pesi *w* della CNN $\{w_1, \ldots, w_K\}$, dove si presume che la CNN abbia un totale di *K* pesi. La CNN è organizzata in strati, e lo strato l-esimo contiene un sottoinsieme dei pesi $W_l \subseteq W$.

Ad esempio, il vettore di classificazione *y(x;W)* comprende *C* elementi $y_c(x;W)$, ciascuno corrispondente ad una classe di immagine *IC(c)* ed avente un valore che indica la probabilità che l'immagine di ingresso *x* rappresenti un oggetto appartenente a tale classe di immagine *IC(c)*. Detto valore può o rappresentare direttamente tale probabilità o può essere successivamente trasformato in tale probabilità applicando una mappatura, quale la funzione softmax, si veda Bridle, JS (1990). "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters", in*Advances in neural information processing systems* (pagine 211-217).

Al fine di semplificare la descrizione, la CNN **100** di **Figura 1** è atta ad elaborare un'immagine di ingresso avente un singolo canale (ad esempio un'immagine in scala di grigi), in cui ciascuno strato della CNN **100** prevede un kernel singolo, ed in cui ogni strato fornisce in uscita in singolo canale. Tuttavia, considerazioni simili si applicano se l'immagine di ingresso ha più di un canale e se viene utilizzato più di un kernel in almeno alcuni livelli.

L'immagine di ingresso *x* è un'immagine digitale avente $h \times h$ pixel (ad esempio, *h* può essere uguale a 112). Considerazioni simili si applicano se l'immagine di ingresso *x* ha una risoluzione diversa (cioè include un diverso numero di pixel). A questo proposito, anche se si è fatto riferimento ad un'immagine di ingresso quadrata, considerazioni simili si applicano nel caso in cui l'immagine di ingresso abbia una forma diversa, ad esempio una forma rettangolare.

La CNN **100** comprende una sequenza ordinata di *L* strati **120(l)** (l = 1, 2, ... L) , con lo strato generico **120(l)** della sequenza che è configurato per :

- ricevere dallo strato precedente **120(l-1)** della sequenza una corrispondente struttura di

ingresso **110(l-1)** comprendente un corrispondente insieme di $h(l-1) \times h(l-1)$ caratteristiche;

- elaborare detta struttura di ingresso ricevuta **110(l-1)** sfruttando un kernel **130(l)** comprendente un corrispondente insieme di $k(l) \times k(l)$ pesi w e

- generare una corrispondente struttura di uscita **110(l)** comprendente un corrispondente insieme di $h(l)-k(l)+1 \times h(l)-k(l)+1$ caratteristiche.

Il primo strato **120(1)** della CNN **100** è configurato per ricevere come struttura di ingresso l'immagine di ingresso **x** comprendente un corrispondente insieme di pixel $h(0) \times h(0)$.

Ciascuno strato **120(l)** della sequenza è uno strato convoluzionale configurato per eseguire una procedura di convoluzione per generare una struttura di uscita **110(l)** dalla struttura di ingresso **110(l-1)** ricevuta usando il kernel **130(l)** come è ben noto agli esperti del settore.

Alcuni degli strati convoluzionali **120(l)** della sequenza possono essere seguiti da un corrispondente strato di tipo max-pooling (non illustrato), che è configurato per eseguire una procedura di sotto-campionamento diretta a generare una versione sotto-campionata della struttura **110(l)** ricevuta dallo strato convoluzionale **120(l)**. La procedura di sotto-campionamento prevede di muovere una finestra di selezione mobile sulla struttura **110(l)** al fine di selezionare insiemi corrispondenti di caratteristiche e generare per ciascun insieme selezionato di caratteristiche una corrispondente caratteristica avente il valore più alto tra quelli dell'insieme selezionato di caratteristiche. Lo scopo di questa procedura di sotto-campionamento è quello di consentire un certo grado di invarianza alla traslazione e di ridurre i requisiti computazionali per i livelli **120(l)** seguenti della sequenza. Considerazioni simili si applicano se la procedura di sotto-campionamento viene eseguita in un modo diverso, come ad esempio calcolando la media tra i valori dell'insieme selezionato di caratteristiche.

La CNN **100** comprende inoltre r strati addizionali **150(1)**, **150(2)**, ..., **150(r)** di tipo completamente connesso, cioè, strati non convoluzionali strati progettati per generare strutture di uscita dalle strutture di ingresso, in cui ciascun valore di uscita della struttura di uscita è una funzione di tutti i valori di ingresso della struttura di ingresso. Gli strati aggiuntivi **150(1)**, **150(2)**, ..., **150(r)** agiscono come classificatori finali con un numero di neuroni di uscita uguale al numero di possibili classi di immagini predefinite $IC(c)$, in modo tale che ciascun neurone di uscita sia associato a uno specifico tra le classi di immagini predefinite $IC(c)$.

Il primo strato aggiuntivo **150(1)** è progettato per ricevere come ingresso la struttura di uscita **110(L)** generata dall'ultimo strato **120(L)**, mentre l'ultimo strato aggiuntivo **150(r)** è progettato per generare come struttura di uscita il vettore di classificazione $y(x; W)$.

La **Figura 2** è un diagramma di flusso che illustra in termini di blocchi funzionali una

procedura di addestramento **200** diretta ad impostare i pesi W_1 degli strati della CNN **100** secondo una forma di realizzazione della presente invenzione.

A questo proposito, si deve notare che mentre nella presente descrizione si farà riferimento a CNN (cioè reti neurali convoluzionali), la procedura di addestramento **200** secondo una forma di realizzazione della presente invenzione può essere applicata direttamente a qualsiasi tipo di reti neurali non convoluzionali.

La procedura di addestramento **200** comprende una sequenza di tre sotto-procedure principali **210**, **220**, **230**.

La prima sotto-procedura **210** è diretta ad addestrare inizialmente la CNN **100** secondo una procedura standard di addestramento a retropropagazione che sfrutta una funzione di costo J(W; x, t(x)) - in cui *t(x)* è il vettore di classificazione obiettivo *t(x)* corrispondente all'immagine di ingresso *x* - senza alcuna funzione di regolarizzazione.

La seconda sotto-procedura **220** è diretta ad addestrare ulteriormente la CNN **100** che è stata sottoposta alla procedura di addestramento della prima sotto-procedura **210** con una procedura di addestramento di retropropagazione che sfrutta una funzione di costo *J(W; x, t(x))* avente una funzione di regolarizzazione *R(W; x)* secondo una forma di realizzazione della presente invenzione. Come verrà descritto in maggior dettaglio nel seguito della presente descrizione, secondo una forma di realizzazione della presente invenzione la funzione di regolarizzazione *R(W; x)* dipende sia dai pesi *W* sia dall'immagine di ingresso *x*, a differenza delle funzioni regolarizzazione *R(W)* già note, che dipendono solo dal insieme di pesi *W*.

La terza sotto-procedura **230** è diretta ad addestrare ulteriormente la CNN **100** che è stata sottoposta alla procedura di addestramento della seconda sotto-procedura **220** con una procedura di addestramento a retropropagazione che sfrutta la stessa funzione di costo *J(W; x, t(x))* usata nella seconda sotto-procedura **220**. Tuttavia, a differenza delle prime due sotto-procedure **210**, **220**, secondo una forma di realizzazione della presente invenzione, durante la terza sotto-procedura **230** viene anche eseguita un'operazione di sfoltimento ("pruning"), diretta ad impostare a zero quei pesi *w* che hanno un valore inferiore a una soglia corrispondente.

Scopo della prima sotto-procedura **210** (che sfrutta una funzione di costo senza funzione di regolarizzazione) è accelerare il processo e raggiungere i risultati in modo più rapido. Tuttavia, secondo un'ulteriore forma di realizzazione della presente invenzione, la prima sotto-procedura **210** può essere saltata. Pertanto, secondo questa ulteriore forma di realizzazione della presente invenzione, la procedura di addestramento **200** può iniziare direttamente con la seconda sotto-procedura **220** (che sfrutta una funzione di costo comprendente la funzione di regolarizzazione

secondo una forma di realizzazione della presente invenzione).

Saltare la prima sotto-procedura **210** può essere utile nel caso in cui la procedura di addestramento **200** sia applicata ad una CNN già addestrata. Questo è un caso abbastanza comune nel campo dell'apprendimento profondo ("Deep Learning"), in cui il punto di partenza è spesso una CNN già addestrata, che viene poi ulteriormente addestrata per adattarla a classi diverse. Un altro caso in cui la prima sotto-procedura **210** potrebbe essere vantaggiosamente saltata è quando la CNN da addestrare è stata appena inizializzata con pesi casuali. In questo caso, tuttavia, può essere utile impostare il parametro di tasso di decadimento λ (vedere l'equazione (3)) ad un valore inferiore rispetto al solito.

Le sotto-procedure **210**, **220**, **230** verranno ora descritte in maggior dettaglio facendo riferimento alle **Figure 3-5**. Le tre sotto-procedure **210**, **220**, **230** prevedono l'aggiornamento dei pesi *W* ogni volta che le fasi in avanti e indietro sono state eseguite su un insieme *TI(p)* (p = 1, 2, 3 ...) di immagini di ingresso x (in questo caso denominato anche "immagini di addestramento di ingresso") appartenente ad un database di addestramento. Ogni set *TI(p)* contiene $|TI(p)|$ immagini.

Si deve notare che nel seguito della descrizione si assume che i pesi *W* siano aggiornati ogni volta che le fasi di andata e di ritorno sono state eseguite su una singola immagine di addestramento di ingresso del database di addestramento, generando un gradiente della funzione di costo (cioè, l'insieme *TI(p)* contiene solo una singola immagine). Tuttavia, considerazioni simili si applicano nel caso in cui i pesi *W* sono aggiornati dopo l'elaborazione di un insieme generico *TI(p)* di immagini di addestramento, nel qual caso un gradiente per ciascuna delle immagini di addestramento in *TI(p)* viene calcolato e i pesi in *W* vengono aggiornati per mezzo del gradiente medio attraverso TI(p). Ne consegue che le considerazioni si applicano anche nel caso in cui sia presente un singolo insieme *TI(p)* di immagini di addestramento di ingresso comprendente tutte le immagini di addestramento del database di addestramento, in modo che i pesi *W* siano aggiornati ogni volta che sono state eseguite fasi di andata e ritorno su tutte le immagini di addestramento del database di addestramento.

La **Figura 3** è un diagramma di flusso che illustra in termini di blocchi funzionali le fasi principali della sotto-procedura **210**.

La prima fase della sotto-procedura **210** prevede di selezionare da un insieme *TI(p)* di immagini di addestramento di ingresso appartenente al database addestramento una immagine di addestramento di ingresso x, e fornire tale immagine di addestramento di ingresso **x** come struttura di dati di ingresso alla CNN **100** da addestrare (blocco **310**). Il database di addestramento comprende immagini di addestramento note, vale a dire, immagini per le quali è noto quale sia la

classe $IC(c = c^*)$ a cui appartiene l'oggetto raffigurano in essa tra le classi disponibili $IC(c)$.

Nella successiva fase (blocco **320**), la CNN **100** elabora l'immagine di addestramento di ingresso x ricevuta in modo da generare un corrispondente vettore di classificazione di uscita $y(x; W)$.

Come già accennato in precedenza, avendo C classi diverse $IC(c)$, il vettore di classificazione di uscita $y(x; W)$ è un vettore comprendente C elementi $y_c(x; W)$ ciascuno corrispondente ad una classe di immagini $IC(c)$ ed avere un valore che o rappresenta la probabilità che l'immagine di ingresso x mostri un oggetto appartenente a quella classe di immagini $IC(c)$, o che possa essere trasformata in tale probabilità applicando una mappatura come una funzione softmax.

Nella fase successiva (blocco **330**), viene calcolata una funzione di costo $J(W; x, t(x)) = \eta L(y(x; W), t(x))$ (si veda equazione 3). Per calcolare la funzione di costo $J(W; x, t(x))$, può essere usata una qualsiasi funzione di perdita funzione nota L(y(x; W), t(x)), come la funzione di perdita MSE, per cui

$$L(y(x;W),t(x)) = \frac{1}{C}\sum_{c=1}^{C}(y_c(x;W) - t_c(x))^2 \quad (4)$$

o la funzione di perdita ad entropia incrociata ("cross entropy"), per cui

$$L\big(y(x;W),t(x)\big) = -\sum_{c=1}^{C} t_c(x)\log\big(y_c(x;W)\big) \quad (5)$$

in cui $t(x)$ è il vettore di classificazione corrispondente ad una immagine di addestramento di ingresso x appartenente alla classe $IC(c^*)$ e comprende C elementi $t_c(x)$, c = 1, …, C , in cui $t_c(x) = 1$ per $c = c^*$ e $t_c(x) = 0$ per $c \neq c^*$.

Le fasi corrispondenti ai blocchi **310**, **320**, **330** corrispondono alla porzione di andata della sotto-procedura **210**.

La porzione di ritorno della sotto-procedura **210** (blocco **340**) prevede il calcolo del gradiente della funzione di costo $J(W; x, t(x))$ rispetto ai pesi W per tutti gli strati in modo ricorsivo dall'ultimo strato al primo strato secondo la seguente equazione:

$$\nabla_W J(W;x,t(x)) = \left[\frac{\partial J(W;x,t(x))}{\partial w_1}, \quad \frac{\partial J(W;x,t(x))}{\partial w_2}, \quad \ldots\right] \quad (6)$$

Successivamente, l'insieme di pesi *W* della CNN **100** è aggiornato (blocco **370**). L'aggiornamento dei pesi *W* viene effettuato utilizzando un'operazione di discesa del gradiente al fine di minimizzare la funzione di costo *J(W; x, t(x))* secondo la seguente equazione:

$$w^{p+1} = w^p - \frac{\partial J(W;x,t(x))}{\partial w}, \quad (7)$$

dove w^p è il peso generico della matrice W^p di pesi *w* della CNN **100** in cui sono utilizzate le immagini di addestramento *x* del p-esimo insieme *TI(p)* vengono utilizzati, e w^{p+1} è il peso generico aggiornato della CNN **100**.

A questo punto, viene effettuato un controllo (blocco **380**) per verificare se l'immagine considerata *x* è l'ultima nel set di addestramento o meno.

Nel caso in cui l'immagine considerata non sia l'ultima (blocco di uscita **N** del blocco **380**), una prossima immagine di addestramento *x* viene selezionata (blocco **390**) e tutte le operazioni precedentemente descritte sono eseguite utilizzando questa nuova immagine di addestramento (ritorno al blocco **310**), producendo una versione della CNN **100** avente i pesi aggiornati W^{p+2}.

Nel caso in cui l'immagine considerata sia l'ultima (ramo di uscita **Y** del blocco **380**), significa che sono state utilizzate tutte le immagini di addestramento del database di addestramento. In questo caso, si dice che è passata un'epoca ("epoch") della fase di addestramento.

La sotto-procedura **210** qui descritta può essere ripetuta per diverse epoche, come ad esempio decine di epoche, al fine di fornire alla CNN **100** l'intero database di addestramento più volte.

La **Figura 4** è un diagramma di flusso che illustra in termini di blocchi funzionali le fasi principali della sotto-procedura **220** secondo una forma di realizzazione della presente invenzione.

Secondo una forma di realizzazione della presente invenzione, la sotto-procedura **220** è sostanzialmente uguale alla sotto-procedura **210** precedentemente descritta nella **Figura 3**, con la differenza principale che la funzione di costo *J(W; x, t(x))* usata è differente, compresa una funzione di regolarizzazione R(W; x) che dipende dall'immagine di ingresso *x* e dal vettore di pesi *W*.

La prima fase della sotto-procedura **220** prevede la selezione di un'immagine di addestramento di ingresso *x* e la fornitura di tale immagine di addestramento di ingresso come struttura dati di ingresso alla CNN **100** da addestrare (blocco **410**).

Nella fase successiva (blocco **420**), la CNN **100** elabora la immagine di addestramento di

ingresso ricevuta **x** allo scopo di generare un corrispondente vettore di classificazione di uscita *y*.

Secondo una forma di realizzazione della presente invenzione, l'addestramento è basato sulla seguente funzione di costo:

$$J(W; x, t(x)) = \eta L(y(x; W), t(x)) + \lambda R(W; x) \quad (8)$$

(si veda equazione 3). Questa funzione di costo comprende una funzione di regolarizzazione *R(W; x)* secondo una forma di realizzazione della presente invenzione.

La funzione di regolarizzazione R(W; x) è una funzione dell'immagine di ingresso *x* e del vettore di pesi *W*. Secondo una forma di realizzazione della presente invenzione, la funzione di regolarizzazione promuove valori bassi dei pesi nella CNN **100** ogniqualvolta ciò possa essere ottenuto senza ridurre le prestazioni della CNN **100**. La funzione di regolarizzazione *R(W; x)* secondo una forma di realizzazione della presente invenzione influenza la funzione di costo *J(W; x, t(x))* fornendo una penalità selettiva ad ogni peso della CNN **100**.

La funzione di regolarizzazione è una somma di costi (o penalità) assegnati a ciascun peso in proporzione ad un prodotto che comprende due fattori. Il primo fattore è il valore del peso al quadrato; il secondo fattore è una funzione di come è sensibile l'uscita ad un cambiamento nel peso, cioè, il secondo fattore fornisce una quantificazione di quanto varia l'uscita rispetto ad una variazione del peso. La penalità risultante sarà grande se entrambi i fattori sono grandi: ciò si verificherà quando il peso ha un valore elevato e la sensibilità al cambiamento è bassa. Viceversa, la penalità sarà piccola o pari a zero quando il peso ha un valore basso o la sensibilità al cambiamento è alta. La funzione di regolarizzazione è continua rispetto al peso *w* e assume valori di penalità non negativi. In una forma di realizzazione, la funzione di regolarizzazione è una somma delle penalità da tutti i pesi in *W*:

$$R(W; x) = \frac{1}{2} \sum_{w \in W} w^2 H\big(S(w; x, W)\big) \quad (9)$$

Qui, i due fattori discussi sopra sono il peso al quadrato w^2 e la funzione *H* della sensibilità *S*. Le funzioni *H* ed *S* saranno definite di seguito.

Allo scopo di introdurre una tale funzione di regolarizzazione, è necessario definire una funzione *S(w; x, W)*, in seguito denominata "funzione di sensibilità". Questa funzione quantifica il tasso di variazione dell'uscita della CNN **100** - ad esempio, il vettore di classificazione di uscita y*(x;*

W) - causato da variazioni del peso *w*. La funzione di sensibilità può essere definita come la media delle derivate assolute degli elementi $y_c(x; W)$ del vettore di classificazione *y(x; W)* rispetto al peso *w*, a condizione che il vettore di pesi sia *W* e l'immagine di ingresso sia *x*

$$S(w; x, W) = \frac{1}{C}\sum_{c=1}^{C}\left|\frac{\partial y_c(x;W)}{\partial w}\right| \tag{10}$$

Data una specifica configurazione di peso W^*, in cui il peso in considerazione è w^*, maggiore è il valore della sensibilità $S(w^*; x, W^*)$, maggiore è la variazione del vettore di classificazione di uscita $y(x; W^*)$ in presenza di variazioni del peso *w* nel nelle vicinanze di w^*. Quindi, se il valore di $S(w^*; x, W^*)$ è grande, allora un cambiamento nel peso w^* può portare a cambiamenti sostanziali nel vettore di classificazione di uscita *y(x;W)*. Viceversa, se il valore di $S(w^*; x, W^*)$ è piccolo, un cambiamento nel peso w^* ha un piccolo impatto sul vettore di classificazione di uscita.

Si deve osservare che *S(w;x,W)* è una misura *locale*, nel senso che un peso può avere valori di sensibilità diversi a due valori di peso diversi $w = a$ e $w = b$.

Secondo una forma di realizzazione della presente invenzione, la funzione di regolarizzazione *R(W; x)* è una funzione la cui derivata rispetto al peso *w* è uguale a:

$$\frac{\partial R(W;x)}{\partial w} = wH\big(S(w; x, W)\big) \tag{11}$$

dove

$$H\big(S(w; x, W)\big) = \begin{cases} 1 - S(w; x, W) & 0 < S(w; x, W) \leq 1 \\ 0 & S(w; x, W) > 1 \end{cases} \tag{12}$$

In vista di quanto sopra, la prossima fase della procedura secondo una forma di realizzazione della presente invenzione (blocco **440**) prevede il calcolo della derivata della funzione di costo *J(W; x, t(x))* rispetto al peso *w* secondo la seguente equazione:

$$\begin{aligned} \frac{\partial J\big(W; x, t(x)\big)}{\partial w} &= \eta\frac{\partial L\big(y(x; W), t(x)\big)}{\partial w} + \lambda\frac{\partial R(W; x)}{\partial w} \\ &= \eta\frac{\partial L(y(x;W),t(x))}{\partial w} + \lambda wH\big(S(w; x, W)\big) \end{aligned} \tag{13}$$

L'aggiornamento di ciascun peso w nel set di pesi W viene effettuato utilizzando un'operazione di discesa del gradiente al fine di minimizzare la funzione di costo $J(W; x, t(x))$ secondo la seguente equazione

$$w^{p+1} = w^p - \eta \frac{\partial L(y(x;W^p),t(x))}{\partial w} - \lambda w^p H\big(S(w^p; x, W^p)\big) \qquad (14)$$

Dove W^p è il vettore dei pesi w della CNN **100** in cui sono utilizzate le immagini di addestramento x del p-esimo insieme $TI(p)$, w^p è il peso generico della matrice di pesi W^p, e w^{p+1} è il generico peso aggiornato w della CNN **100**. A questo punto, i pesi vengono aggiornati in W^{p+1} (blocco **470**).

Secondo l'equazione (14), il peso aggiornato w^{p+1} è ottenuto dal peso precedente w^p. Tale operazione equivale a sottrarre un primo termine proporzionale alla derivata della funzione di perdita (cioè , proporzionale all'errore di classificazione prodotto dalla CNN **100**), e sottraendo un secondo termine che può variare da 0 ($H(s(w; W)) = 0$) a λw^p ($H(s(w; W)) = 1$). La logica è la seguente. Se l'uscita della rete è insensibile ai cambiamenti di un dato peso w^p, allora il valore di S è piccolo, e di conseguenza il valore di H è vicino a 1; la regolarizzazione contribuisce quindi a modificare w^p verso lo zero di una quantità di circa λw^p. Al contrario, se l'uscita della rete è molto sensibile ad una variazione di un dato peso, allora il valore di S è alto, e di conseguenza il valore di H è piccolo o nullo; quindi, anche il termine $\lambda w^p H(S(w^p; x, W^p))$ è piccolo o pari a zero e la regolarizzazione non contribuisce a modificare il peso. Il risultato di questo processo è spingere verso lo zero tutti quei pesi che non sono utili per la classificazione di oggetti.

Da questa discussione dovrebbe essere chiaro che il fattore di regolarizzazione λ dovrebbe avere un valore nell'intervallo tra 0 e 1. Poiché la sensibilità è una misura locale, è vantaggioso che il fattore di regolarizzazione λ sia più vicino a zero che ad uno; in questo caso, qualsiasi peso con un'alta sensibilità viene modificato verso lo zero di una piccola frazione, in modo tale che un nuovo valore di sensibilità possa essere calcolato rispetto al nuovo valore del peso. Se il valore di sensibilità diventa grande dopo alcune iterazioni della discesa del gradiente, allora l'aggiornamento dovuto alla regolarizzazione si interromperà. Se il valore di sensibilità di un dato peso non diventa mai grande nel corso delle iterazioni della discesa gradiente, quindi il valore di questo peso è probabile che si avvicini allo zero.

A questo punto, viene eseguito un controllo (blocco **480**) per verificare se l'immagine di addestramento considerata x è l'ultima o no.

Nel caso in cui l'immagine di addestramento considerata x non sia l'ultima (blocco di uscita **N** del blocco **480**), viene selezionata una immagine di addestramento x successiva (blocco **490**) e tutte le operazioni precedentemente descritte vengono eseguite usando questa immagine di addestramento (ritornare al blocco **410**) fornendo in ingresso ad essa una versione della CNN **100** avente i pesi aggiornati W^{p+1}.

Nel caso in cui l'immagine di addestramento x considerata è l'ultima (ramo d'uscita **Y** del blocco **480**), significa che tutte le immagini di addestramento del database addestramento sono state utilizzate. In questo caso, si dice che è passata un'epoca della fase di addestramento.

Come nel caso della sotto-procedura **210**, anche la sotto-procedura **220** qui descritta può essere ripetuta per diverse epoche, come ad esempio decine di epoche, al fine di fornire alla CNN **100** l'intero database di addestramento più volte.

La richiedente ha riscontrato che minimizzando la funzione di costo $J(W; x, t(x)) = \eta L(y(x; W), t(x)) + \lambda R(W; x)$, un numero significativo di i pesi w assumono valori vicini allo zero. I valori assoluti di altri pesi rimangono molto più grandi, ovvero quelli la cui sensibilità è rimasta elevata verso la fine del processo di addestramento. Dovrebbe essere chiaro che in una somma pesata di pixel (o caratteristiche), in cui alcuni pesi sono molto piccoli e altri pesi sono relativamente grandi, i termini che hanno pesi molto piccoli come fattori contribuiscono poco alla somma. Inoltre, se quei pesi molto piccoli sono sostituiti da zero, la somma pesata risultante è una buona approssimazione della somma originale ottenuta con tutti i pesi inalterati.

Questa osservazione suggerisce una separazione dei pesi in due sotto-insiemi disgiunti, uno contenente i pesi piccoli ed uno contenente i pesi relativamente grandi. Tale separazione può essere ottenuta confrontando i valori assoluti dei pesi con una soglia.

Ciò ha portato la Richiedente a sviluppare un metodo di *pruning*, cioè, un metodo che imposta esattamente a zero tutti quei pesi cui valori assoluti sono sotto una certa soglia.

La **Figura 5** è un diagramma di flusso che illustra in termini di blocchi funzionali le fasi principali della sotto-procedura **230** che prevede il pruning summenzionato secondo una forma di realizzazione della presente invenzione.

La prima fase della sotto-procedura **230** prevede di impostare (blocco **510**) una soglia di pruning TH da utilizzare per impostare a zero quei pesi w che hanno assunto un valore sufficientemente basso.

Empiricamente è stato osservato che, in assenza di una procedura di pruning, i pesi W della CNN **100** sono distribuiti secondo una distribuzione gaussiana di media zero e deviazione standard σ. In una tale distribuzione, la probabilità che un peso assuma un valore assoluto più piccolo di una

data soglia *TH* è una funzione di σ e *TH*. Nel caso alternativo in cui alcuni pesi siano stati impostati a zero ed una fase di aggiornamento come quella di Equazione (11) e del blocco **470** sia stata successivamente applicata, si può osservare che alcuni pesi rimangono esattamente a zero mentre gli altri pesi si raggruppano attorno ad un valore positivo e ad un valore negativo. In questo caso alternativo, la distribuzione di tutti i pesi non ha una semplice forma parametrica. La Richiedente ha quindi sviluppato una procedura non parametrica al fine di eseguire l'ulteriore pruning dei pesi.

Osservando gli istogrammi dei valori assoluti |w| di quei pesi che sono diversi da zero, la Richiedente ha trovato che la distribuzione dei valori assoluti di questi pesi è approssimativamente unimodale, cioè ha un picco, e che questo picco si verifica vicino al valore assoluto medio dei pesi diversi da zero. Di conseguenza, la soglia che determina quali pesi saranno impostati a zero può essere vantaggiosamente proporzionale a questo valore assoluto medio. In alternativa, la soglia può essere impostata in modo da separare i pesi diversi da zero in due insiemi il cui numero di elementi ha una proporzione costante.

Secondo una forma di realizzazione esemplificativa della presente invenzione, la soglia di potatura è impostata a:

$$TH = \theta \underset{w \in W, w \neq 0}{\text{mean}} |w| \tag{15}$$

in cui il valore medio viene calcolato considerando tutti quei pesi della CNN **100** che hanno valori diversi da zero, e θ è una costante positiva minore di 1, ad esempio $\theta = 1/10$.

Secondo una seconda forma di realizzazione esemplificativa della presente invenzione, la soglia di potatura è impostata a

$$TH, such\ that\ P(|w| \leq TH) = \theta \tag{16}$$

in cui $P(|w| \leq TH)$ indica il rapporto tra due dimensioni di insiemi: uno è il numero di pesi diversi da zero i cui valori assoluti sono minori o uguali a *TH*, e uno è il numero di tutti i pesi diversi da zero.

La fase successiva (blocco **520**) prevede l'azzeramento dei pesi w della CNN **100** il cui valore assoluto è inferiore alla soglia di pruning *TH*.

Poiché la funzione di sensibilità *S(w, x, W)* utilizzata per il calcolo del gradiente del termine di regolarizzazione *R(W; x)* è una misura locale, la sua validità è assicurata per variazioni di *W*

molto piccole. Poiché il processo di pruning forza alcuni pesi allo zero, la soglia di pruning è vantaggiosamente scelta per minimizzare questo tipo di perturbazione dovuta al pruning stesso.

A questo punto, la CNN **100** è addestrata per un'epoca (blocco **530**) utilizzando una procedura di addestramento uguale a quella corrispondente alla sotto-procedura **220**.

Deve essere previsto un certo calo di prestazioni (ad esempio con rispetto alla precisione di classificazione) come effetto collaterale del pruning. Un metodo completo include quindi un criterio di arresto; ad esempio, il processo può essere interrotto quando le prestazioni raggiungono un limite inferiore predefinito. Tale meccanismo è descritto di seguito.

L'accuratezza della classificazione può essere definita rispetto ad un insieme di immagini, chiamato insieme di validazione. L'insieme di validazione normalmente non contiene nessuna delle immagini utilizzate per l'addestramento. Allo stesso modo dell'insieme di addestramento, anche ogni immagine dell'insieme di validazione deve essere equipaggiata con una classe obiettivo, in modo tale che l'immagine di validazione x abbia un vettore di classificazione obiettivo $t(x)$. La precisione di classificazione per l'insieme di validazione VS può ora essere definita come una frazione di 100%, in cui il numeratore è il numero di immagini classificate correttamente dalla CNN e il denominatore è il numero di immagini contenute nell'insieme di validazione. L'accuratezza della classificazione è chiaramente funzione dei pesi W nella CNN.

Il criterio di arresto può ora essere definito. Si supponga che sia stata fissata una prestazione di riferimento, ad esempio misurando la prestazione di classificazione $A(W^F)$ per una CNN precedentemente addestrata che non è stata sottoposta a pruning (questa prestazione può essere ragionevolmente considerata come un limite superiore alle prestazioni di una CNN sottoposta a pruning sparso). Prima dell'avvio dell'addestramento, è possibile impostare una perdita di prestazioni accettabile, ad esempio $a = 0{,}01$, corrispondente ad una perdita di prestazioni dell'1%.

Dopo che la CNN **100** è stata addestrata per un'epoca, il cui risultato sono i W, le sue prestazioni possono essere confrontate con le prestazioni di riferimento (blocco **540**). Se

$$A(W) < (1 - a) \cdot A(W^F) \tag{17}$$

allora l'addestramento viene interrotto e vengono mantenuti i pesi risultanti dalla precedente epoca di addestramento (ramo di uscita **Y** del blocco **540**). In caso contrario, i pesi e il valore delle prestazioni di classificazione vengono memorizzati, e può essere avviata una nuova epoca di addestramento (ramo di uscita **N** del blocco **540**, ritornare al blocco **510**).

In una forma di realizzazione alternativa della presente invenzione, la funzione di sensibilità

ha una definizione diversa. In questo caso, la derivata assoluta media è sostituita dalla derivata assoluta solo per quel elemento del vettore di classificazione di uscita corrispondente alla classe dell'immagine obiettivo. Questa funzione di sensibilità alternativa può quindi essere scritta come

$$S(w;x,W) = \sum_{c=1}^{C} t_c(x) \left|\frac{\partial y_c(x;W)}{\partial w}\right| \quad (18)$$

poiché il vettore di classificazione obiettivo *t(x)* ha il valore 1 nell'elemento che corrisponde alla classe corretta, mentre tutti gli altri elementi sono pari a 0.

Rispetto alle soluzioni note, la procedura di addestramento sopra descritta **200** in accordo con le forme di realizzazione della presente invenzione consente di ottenere una CNN **100** avente un numero ridotto di parametri (pesi *w*) diversi da zero, e quindi richiede una occupazione di memoria ridotta.

La ridotta occupazione di memoria della CNN **100** addestrata con la procedura di addestramento secondo le forme di realizzazione della presente invenzione è particolarmente vantaggiosa in diversi scenari di applicazione.

Ad esempio, una CNN con occupazione di memoria ridotta può essere vantaggiosamente dispiegata su dispositivi con capacità di memoria di archiviazione limitata a causa di vincoli correlati al consumo di energia, al prezzo e alla dimensione fisica, come negli scenari in cui la CNN viene dispiegata su dispositivi mobili dotati di un'applicazione per la classificazione visuale di oggetti.

Inoltre, una CNN con una occupazione di memoria ridotta può essere vantaggiosamente dispiegata su dispositivi mediante una rete di comunicazione, come una rete di comunicazione mobile con capacità di larghezza di banda ridotta. Ad esempio, in alcuni scenari è richiesto un dispiegamento remoto periodico di CNN complesse sul dispositivo, come ad esempio per scopi di aggiornamento di rete (ad esempio, per l'aggiornamento dei pesi *W* di una CNN dispiegata su un dispositivo di elaborazione di un sistema di controllo di un veicolo a guida autonoma che richiede un aggiornamento periodico per migliorare le prestazioni di guida automatica).

La **Figura 6** illustra in termini di blocchi funzionali molto semplificati uno scenario di applicazione esemplificativo della soluzione secondo forme di realizzazione della presente invenzione. Lo scenario di applicazione considerato prevede l'addestramento di una CNN su un modulo server **610** e quindi il dispiegamento della CNN addestrata su un dispositivo mobile **620** che esegue un'applicazione per scopi di classificazione di oggetti.

La CNN (non addestrata), identificata in figura con il riferimento **100'**, viene fornita al modulo server **610** per essere addestrata. Come schematizzato nella **Figura 6**, il modulo server **610** comprende un motore di addestramento **630**, quale un processo in esecuzione su un processore, un processore, un oggetto, un eseguibile, un thread di un eseguibile, un programma e / o un'applicazione software da dispositivo di calcolo che implementa la procedura di addestramento **200** secondo la forma di realizzazione dell'invenzione. Il motore di addestramento **630** è ulteriormente accoppiato con un database di addestramento **640** comprendente un set di dati di addestramento di immagini di addestramento appartenenti a rispettive classi conosciute che saranno utilizzate dal motore di addestramento **630** per addestrare la CNN **100'** usando la procedura di addestramento **200** secondo la forma di realizzazione dell'invenzione ed ottenendo una corrispondente CNN **100"** addestrata.

Il set di pesi *W* della CNN **100"** addestrata è quindi trasmesso al dispositivo mobile **620** mediante una rete di comunicazione **650**, come una rete di comunicazione mobile.

Il dispiegamento della CNN **100"** addestrata viene effettuato memorizzando il set di pesi *W* in un'unità di memoria **660** del dispositivo mobile **620**.

A questo punto, la CNN **100"** addestrata che è stata dispiegata può essere utilizzata da un motore di classificazione **670** del dispositivo mobile **620**, quale un processo in esecuzione su un processore, un processore, un oggetto, un eseguibile, un thread di un eseguibile, un programma e / o un'applicazione software da dispositivo di calcolo presso il dispositivo mobile **620** per implementare la classificazione di immagini *x*, come per esempio immagini acquisite da un modulo di fotocamera **680** del dispositivo mobile **620**, e generare un corrispondente vettore di classificazione *y(x; W)*.

La procedura di addestramento proposta **200** secondo le forme di realizzazione dell'invenzione sopra descritta è stata confrontata con il metodo proposto da *Han* et Al. su un certo numero di architetture di reti neurali ben note e su una serie di diversi insiemi di dati di immagini. Per ciascun modello addestrato, la sparsità risultante ed il corrispondente footprint (impronta) di memoria sono stati misurati ipotizzando che ogni parametro (peso *w*) sia stato memorizzato come un numero float a precisione singola di 4 byte secondo lo standard IEEE-754. Dai confronti seguenti verrà mostrato che la procedura di addestramento **200** secondo le forme di realizzazione dell'invenzione, migliora la sparsità della rete e quindi riduce il footprint del modello dispiegato senza comprometterne le prestazioni.

Primo risultato sperimentale. LeNet300 (MNIST)

La procedura di addestramento **200** secondo la forma di realizzazione descritta della presente invenzione è stata confrontata sul set di dati MNIST usando l'architettura di Lenet-300. Il set di dati MNIST è costituito da 28 x 28 immagini in scala di grigi a 8 bit di cifre scritte a mano (C = 10 classi). Il database consiste in 50000 campioni di addestramento e 10000 campioni di prova.

L'architettura LeNet-300 è una semplice rete completamente connessa progettata per il riconoscimento dei caratteri scritti a mano e consiste in tre livelli completamente connessi con 300, 100 e 10 neuroni. La CNN è stata addestrata sia usando la procedura di addestramento **200** secondo la forma di realizzazione della presente invenzione in cui la funzione di sensibilità è quella descritta nell'equazione 10, sia usando la procedura di addestramento **200** secondo la forma di realizzazione dell'invenzione in cui la funzione di sensibilità è quella descritta nell'equazione 18.

La seguente tabella mostra, per ogni livello dell'architettura di rete LeNet-300, il numero originale di parametri e il numero di parametri rimanenti per ciascun metodo considerato (le cifre più basse indicano topologie di rete più sparse e quindi requisiti di memoria inferiori). Su circa 266k parametri nell'architettura originale, il metodo proposto da *Han* et al. è in grado di ridurre tale cifra a circa 21k parametri. I risultati mostrano che la forma di realizzazione dell'invenzione corrispondente all'equazione 10 raggiunge una migliore sparsità, tuttavia la forma di realizzazione dell'invenzione corrispondente all'equazione 18 raggiunge un errore complessivo inferiore (circa lo 0,1% nei nostri esperimenti) e richiede meno tempo di addestramento (addestramento da 1k a 2,5k iterazioni, rispettivamente).

Strati	**LeNet300** [# parametri]	**Han et al.** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri\ LeNet300}100\right]\%$	**Procedura di addestramento proposta** $\left[\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri\ LeNet300}100\right]\right]\%$	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η**=0.1** λ**=10^{-5}** θ**=0.1**	η**=0.1** λ**=10^{-4}** θ**=0.1**
fc1 - 300	235K	8%	2.25%	4.78%
fc2 - 100	30K	9%	11.93%	24.75%

fc3 - 10	1K	26%	69.3%	73.8%

	LeNet300	**Han et al.**	**Procedura di addestramento proposta**	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η=0.1 λ=10^{-5} θ=0.1	**η=0.1 λ=10^{-4} θ=0.1**
Dimensione Totale [# parametri]	266K	21.76K	9.55K	19.39K
errore [%]	1.8%	1.6%	1.65%	1.56%

Secondo risultato sperimentale. LeNet5 (MNIST)

L'esperimento precedente è stato ripetuto con il database MNIST sull'architettura di rete LeNet5. L'architettura LeNet5 è una semplice rete convoluzionale per il riconoscimento di caratteri nascosti che consiste in due livelli convoluzionali e due livelli completamente connessi, per un totale di circa 431k parametri apprendibili.

La rete LeNet5 viene addestrata sia usando la procedura di addestramento **200** secondo la forma di realizzazione della presente invenzione in cui la funzione di sensibilità è quella descritta nell'equazione 10 sia usando la procedura di addestramento **200** secondo la forma di realizzazione dell'invenzione in cui la funzione di sensibilità è quella descritta nell'equazione 18. Le prestazioni della rete addestrata vengono quindi confrontate nella seguente tabella con le prestazioni di *Han* et al. La tabella mostra andamenti simili ai risultati mostrati per l'architettura LeNet-300. Dei 430k parametri dell'architettura originale, il metodo proposto da *Han* et al. è in grado di ridurre tale cifra a circa 37k parametri. La procedura di addestramento proposta **200** nelle sue due differenti forme di realizzazione è in grado di ridurre tale figura a soli 9k e 11k parametri per le forme di realizzazione corrispondenti alle equazioni 10 e 18, rispettivamente. I risultati mostrano che la forma di realizzazione dell'invenzione secondo l'equazione 10 raggiunge una migliore sparsità, tuttavia la

forma di realizzazione dell'invenzione secondo l'equazione 18 raggiunge lo stesso errore.

Strati	**LeNet5** [#parametri]	**Han et al.** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri LeNet5}100\right]\%$	**Procedura di addestramento proposta** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri LeNet5}100\right]\%$	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η**=0.1** $\lambda = 10^{-4}$ θ**=0.1**	η**=0.1** $\lambda = 10^{-4}$ θ**=0.1**
conv1	0.5K	66%	67.6%	72.6%
conv2	25K	12%	11.76%	12.04%
fc3	400K	8%	0.90%	1.26%
fc4	5K	19%	31.04%	37.42%

	LeNet5	**Han et al.**	**Procedura di addestramento proposta**	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η**=0.1** $\lambda = 10^{-4}$ θ**=0.1**	η**=0.1** $\lambda = 10^{-4}$ θ**=0.1**
Dimesnione totale [# parametri]	431K	36.28K	8.43K	10.28K
errore[%]	0.8%	0.8%	0.8%	0.8%

Terzo risultato sperimentale LeNet5 (Fashion MNIST)

L'esperimento precedente è stato ripetuto sul set di dati Fashion-MNIST (https://github.com/zalandoresearch/fashion-mnist). Tale set di dati consiste in immagini

monocromatiche di dimensioni 28x28 come nel set di dati MNIST originale. Tuttavia, le immagini contengono oggetti come borse, vestiti , scarpe, ecc.

L'obiettivo di questo ulteriore esperimento è valutare se la procedura di addestramento proposta **200** è anche in grado di analizzare le reti che operano su segnali più densi. In effetti, il set di dati MNIST è un set di dati sparsi, ovvero un numero elevato di pixel delle immagini di ingresso ha un valore uguale a 0. È noto che i segnali sparsi tendono a promuovere la sparsità della rete durante l'addestramento. Al contrario, il database Fashion-MNIST è notevolmente meno sparso, quindi ottenere un'architettura di rete sparsa è più difficile. Il set di dati MNIST è piuttosto sparso in quanto i pixel hanno valori attorno a 0 (pixel nero) e 255 (pixel bianchi). Al contrario, il set di dati Fashion-MNIST ha molte più informazioni lungo i valori intermedi di intensità dei pixel che rappresentano diverse sfumature di grigio.

La tabella seguente mostra le tendenze simili ai risultati mostrati per l'architettura LeNet-5 quando è stata addestrata sul set di dati MNIST originale. Dei 430k parametri dell'architettura originale, il metodo proposto da Han et al. è in grado di ridurre tale cifra a circa 46k parametri. La procedura di addestramento proposta **200** nelle sue due differenti forme di realizzazione è in grado di ridurre tale cifra a soli 37k e 61k parametri per le forme di realizzazione corrispondenti alle equazioni 10 e 18, rispettivamente. I risultati mostrano che la forma di realizzazione corrispondente all'equazione 10 consente di ottenere migliore sparsità, tuttavia la forma di realizzazione corrispondente all'equazione 18 ottiene lo stesso errore. Questo esperimento conferma che le immagini che sono meno sparse producono reti che hanno parametri naturalmente più significativi. Tuttavia, la procedura di addestramento proposta **200** è ancora in grado di aumentare la sparsità della rete anche con segnali non sparsi.

Strati	**LeNet5** [#parametri]	**Han et al.** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri LeNet5}100\right]\%$	**Procedura di addestramento proposta** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri LeNet5}100\right]\%$	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η=0.1 $\lambda = 10^{-5}$ θ=0.1	η=0.1 $\lambda = 10^{-5}$ θ=0.1

conv1	0.5K	73%	76.2%	84.4%
conv2	25K	30.32%	32.56%	49.26%
fc3	400K	8.63%	6.50%	11.2%
fc4	5K	56.86%	44.02%	65.32%

	LeNet5	**Han et al.**	**Procedura di addestramento proposta**	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			η**=0.1** λ **= 10^{-5}** θ**=0.1**	η**=0.1** λ **= 10^{-5}** θ**=0.1**
Dimensione totale [# parametri]	431K	45.24K	36.72K	60.72K
errore[%]	8.8%	8.5%	8.5%	8.5%

Quarto risultato sperimentale VGG-16 (ImageNet)

Come esperimento finale, l'esperimento precedente è stato ripetuto con il database Imagenet sull'architettura di rete VGG-16.

Il set di dati ImageNet è costituito da immagini a colori 224x224 a 24 bit di 1000 diversi tipi di oggetti (C = 1000 classi). Il database consiste in 1M di campioni di addestramento e di 100k campioni di prova. Per quanto riguarda il set di dati MNIST, il set di dati ImageNet rappresenta problemi di riconoscimento di oggetti su vasta scala più pratici.

L'architettura VGG-16 è un'architettura di rete convoluzionale molto più complessa per il riconoscimento di oggetti generici che consiste in 13 strati convoluzionali e tre livelli completamente connessi come mostrato di seguito nella tabella, per un totale di circa 138 milioni di parametri apprendibili. Riguardo al numero di parametri, la VGG-16 ha circa 250 volte più parametri rispetto all'architettura LeNet5, e rappresenta più da vicino la scala delle reti impiegate

praticamente per risolvere problemi di riconoscimento di oggetti su larga scala.

La rete VGG-16 è stata addestrata con la procedura di addestramento proposta **200** sia secondo la forma di realizzazione corrispondente all'equazione 10 sia secondo la forma di realizzazione corrispondente all'equazione 18. Le prestazioni della rete addestrata in entrambi i casi sono quindi confrontate con quella di *Han* et al. nella seguente tabella. La tabella mostra tendenze simili ai risultati mostrati per l'architettura LeNet-300. Sui 138M parametri dell'architettura originale, il metodo proposto da *Han* et al. è in grado di ridurre tale cifra a circa 10,3M parametri. La procedura di addestramento proposta **200** nelle sue due differenti forme di realizzazione è in grado di ridurre tale cifra a circa 11,3M e 9,8 M parametri. Infine, l'esperimento mostra che la procedura di addestramento proposta **200** è in grado di ridurre l'errore di rete di circa l'1% aumentando allo stesso tempo la sparsità della rete.

Strati	**VGG-16** [#parametri]	**Han et al.** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri VGG-16} 100\right]\%$	**Proposed training procedure** $\left[\frac{\#\ parametri\ rimanenti}{\#\ parametri VGG-16} 100\right]\%$	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			$\eta=10^{-3}$ $\lambda=10^{-6}$ $\theta=0.1$	$\eta=10^{-3}$ $\lambda=10^{-5}$ $\theta=0.1$
conv1_1	2K	58%	97.80%	96.35%
conv1_2	37K	22%	90.47%	80.87%
conv2_1	74K	34%	87.81%	81.49%
conv2_2	148K	36%	84.96%	81.41%
conv3_1	295K	53%	83.44%	77.68%
conv3_2	590K	24%	81.92%	71.81%
conv3_3	590K	42%	80.85%	69.25%
conv4_1	1M	32%	71.07%	62.03%
conv4_2	2M	27%	62.96%	51.2%
conv4_3	2M	34%	62.34%	51.91%
conv5_1	2M	35%	60.47%	57.09%

conv5_2	2M	29%	59.66%	57.08%
conv5_3	2M	36%	59.63%	47.75%
fc6	103M	4%	1.08%	1.13%
fc7	17M	4%	6.27%	8.35%
fc8	4M	23%	35.43%	14.81%

	VGG-16	**Han et al.**	**Procedura di addestramento proposta**	
			Forma di realizzazione in accordo con la equazione 10	**Forma di realizzazione in accordo con la equazione 18**
			$\eta=10^{-3}$ $\lambda = 10^{-6}$ $\theta=0.1$	$\eta=10^{-3}$ $\lambda = 10^{-5}$ $\theta=0.1$
Dimensione totale [# parametri]	138M	10.35M	11.34M	9.77M
Errore [Top1% – Top5%]	31.50%-11.32%	31.34%-10.88%	29.29%-9.8%	30.92%-10.06%

La descrizione precedente presenta e discute in dettaglio diverse forme di realizzazione della presente invenzione; tuttavia, sono possibili diverse modifiche alle forme di realizzazione descritte, nonché diverse forme di realizzazione dell'invenzione, senza allontanarsi dall'ambito definito dalle rivendicazioni allegate.

* * * * *

RIVENDICAZIONI

1. Un metodo, comprendente:

- fornire una rete neurale avente un insieme di pesi (**W**) e configurata per ricevere una struttura di dati di ingresso (**x**) per generare un vettore di uscita corrispondente (**y(x,W)**) in accordo con valori di detto insieme di pesi;

- addestrare (**200**) la rete neurale (**100**) per ottenere una rete neurale addestrata (**100''**), detto addestrare comprendendo impostare valori dell'insieme di pesi mediante un algoritmo di discesa del gradiente che sfrutta una funzione di costo comprendente un termine di perdita e un termine di regolarizzazione;

- dispiegare la rete neurale addestrata (**100''**) su un dispositivo attraverso una rete di comunicazione (**650**);

- usare la rete neurale addestrata (**100''**) dispiegata sul dispositivo,

caratterizzato dal fatto che

il termine di regolarizzazione è basato su un tasso di cambiamento di elementi del vettore di uscita causato da variazioni dell'insieme di valori di pesi.

2. Il metodo di rivendicazione 1, in cui detto termine di regolarizzazione è basato su una somma di penalità ciascuna penalizzante un corrispondente peso dell'insieme di pesi, ciascuna penalità essendo basata sul prodotto di un primo fattore ed un secondo fattore, in cui:

- detto primo fattore è basato su una potenza di detto peso corrispondente, in particolare un quadrato del peso corrispondente.

- detto secondo fattore è basato su una funzione di quanto è sensibile il vettore di uscita ad una variazione nel peso corrispondente.

3. Il metodo di rivendicazione 2, in cui detta funzione corrisponde alla media di valori assoluti di derivate di elementi di uscita del vettore di uscita rispetto al peso.

4. Il metodo di rivendicazione 2, in cui:

- detto addestrare comprende, per ciascuna tra una pluralità di strutture di dati di ingresso di addestramento, confrontare il vettore di uscita generato dalla rete neurale in accordo con detta struttura di dati di ingresso di addestramento con un corrispondente vettore di uscita obiettivo che ha solo un elemento diverso da zero, e

- detta funzione corrisponde al valore assoluto della derivata, rispetto al peso, di un elemento

del vettore di uscita corrispondente a detto un elemento diverso da zero del vettore di uscita obiettivo.

5. Il metodo delle rivendicazione da 2 a 4, in cui detto addestrare (**200**) comprende calcolare (**440**, **470**) un corrispondente peso aggiornato da ciascun peso dell'insieme di pesi sottraendo da detto peso:

- un primo termine basato sulla derivata del termine di perdita rispetto ai pesi, e
- un secondo termine basato sul prodotto tra detto peso e un'ulteriore funzione, detta ulteriore funzione essendo uguale a uno meno detta funzione se detta funzione non è maggiore di uno, ed essendo uguale a zero se detta funzione è maggiore di uno.

6. Il metodo di una qualunque tra le rivendicazioni precedenti, in cui detto addestrare (**200**) comprende inoltre impostare a zero (**520**) pesi dell'insieme di pesi aventi un valore inferiore a una soglia corrispondente.

7. Il metodo di rivendicazione 6, comprendente inoltre impostare detta soglia ad uno selezionato tra:

- un valore di soglia basato sulla media dei pesi diversi da zero;
- un valore di soglia tale che un rapporto tra una prima dimensione di insieme ed una seconda dimensione di insieme sia uguale a una costante, in cui detta prima dimensione di insieme è il numero di pesi diversi da zero i cui valori assoluti sono minori o uguali a detto valore di soglia e detta seconda dimensione di insieme è uguale al numero di tutti i pesi diversi da zero.

8. Il metodo di una qualunque tra le rivendicazioni precedenti, in cui detto dispiegare la rete neurale addestrata su un dispositivo (**620**) comprende inviare pesi diversi da zero dalla rete neurale addestrata al dispositivo attraverso detta rete di comunicazione (**650**).

9. Il metodo di una qualunque tra le rivendicazioni precedenti, in cui detto utilizzare la rete neurale addestrata (**100''**) dispiegata sul dispositivo comprende usare la rete neurale addestrata (**100''**) dispiegata con un'applicazione per una classificazione di oggetti visuale in esecuzione sul dispositivo.

10. Il metodo di una qualunque tra le rivendicazioni precedenti, in cui detto dispositivo è un

dispositivo mobile.

11. Il metodo di una qualunque tra le rivendicazioni da 1 a 8, in cui detto dispositivo è un dispositivo di elaborazione di un sistema di controllo di un veicolo a guida autonoma.

FIG.1

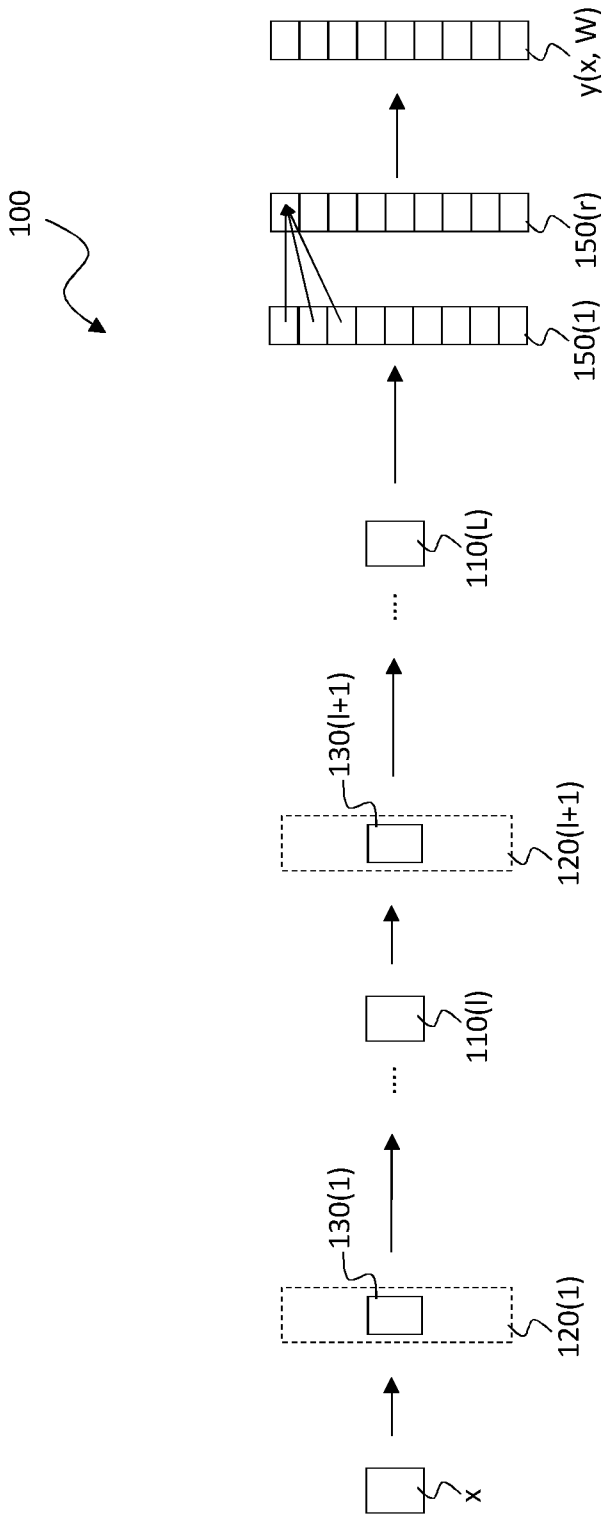


FIG.2

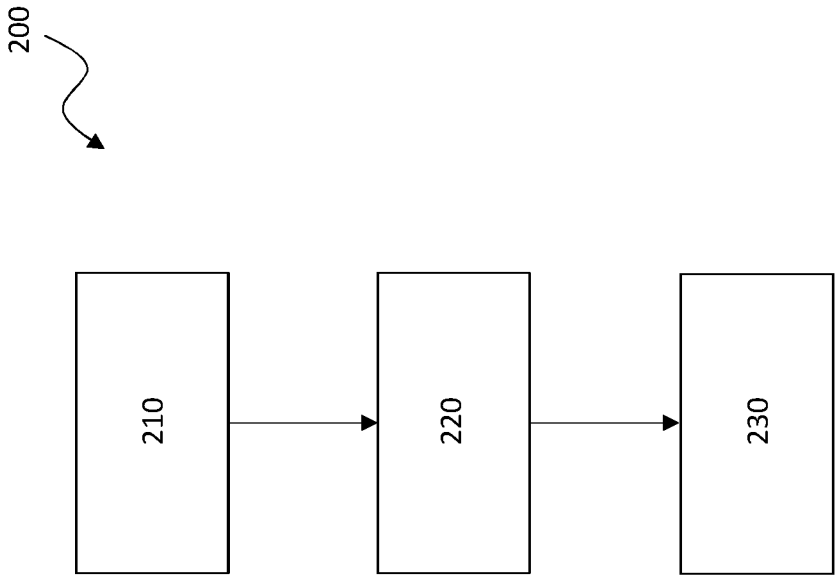


FIG.3

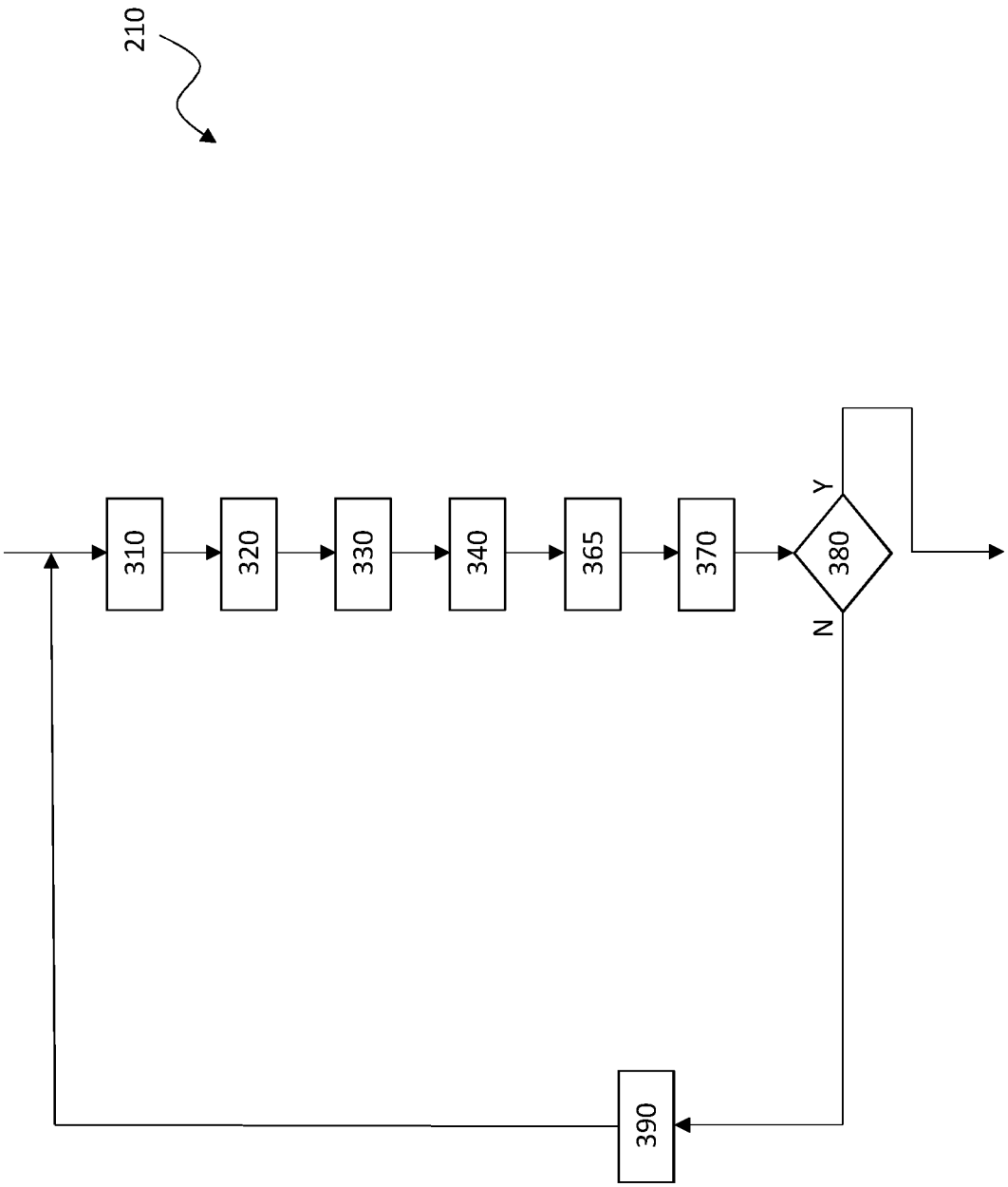


FIG.4

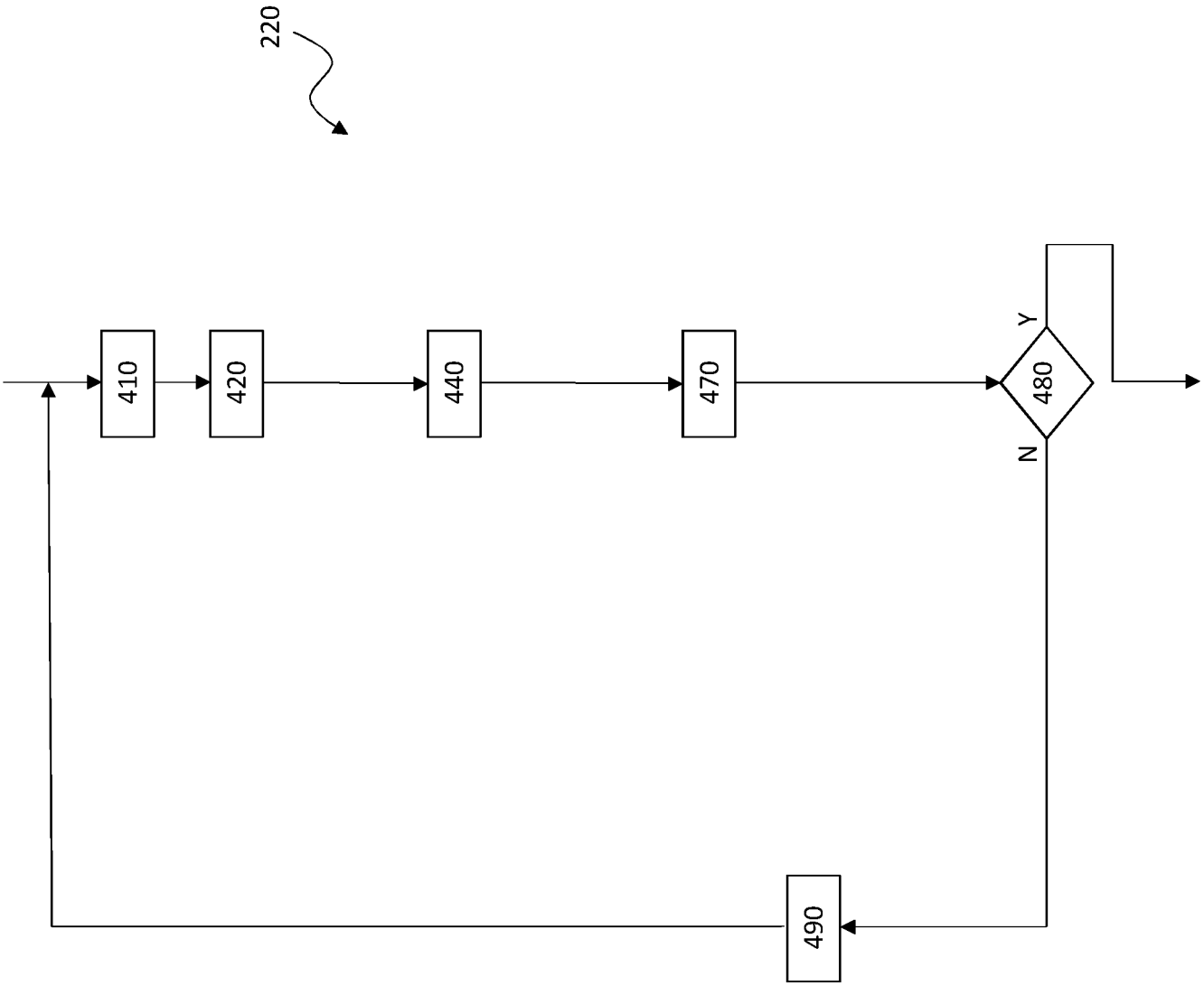
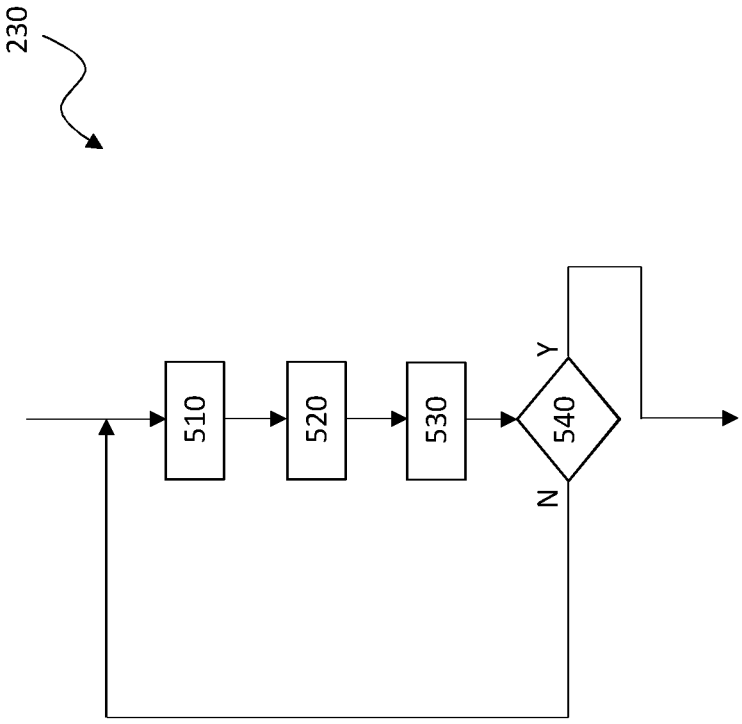


FIG.5



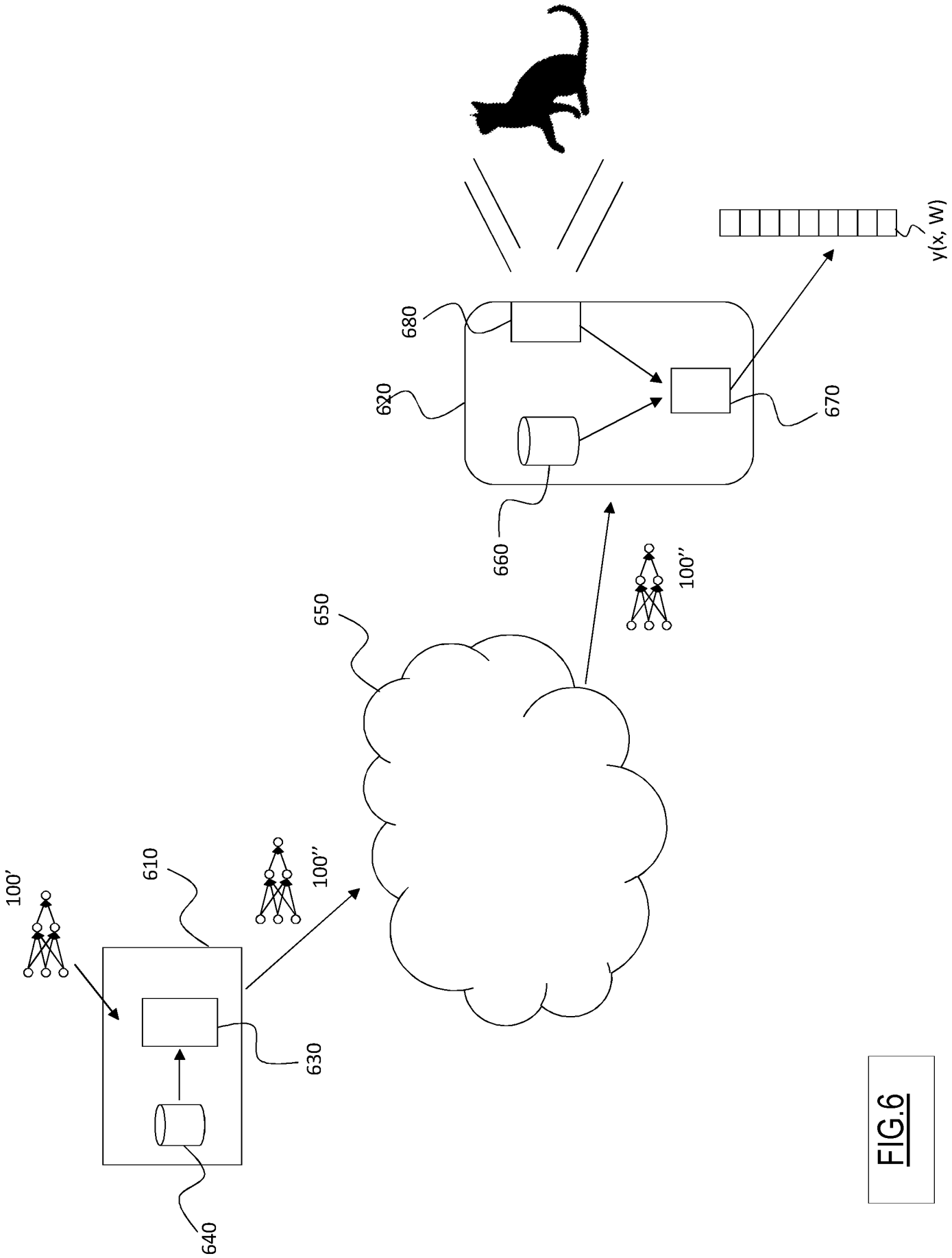


FIG.6