Alma Mater Studiorum · Università di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Materia di Tesi: Analisi delle Immagini

Analisi di un modello computazionale della corteccia visiva

Tesi di Laurea di:
DAVIDE LIUZZI

Relatore: **Chiar.mo Prof**
Renato Campanini

Seconda Sessione

Anno Accademico 2009 - 2010

Alma Mater Studiorum · Università di Bologna

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Specialistica in Informatica

Materia di Tesi: Analisi delle Immagini

Analisi di un modello computazionale della corteccia visiva

Tesi di Laurea di:
DAVIDE LIUZZI

Relatore: **Chiar.mo Prof**
Renato Campanini

PAROLE CHIAVE: corteccia visiva, analisi, modello computazionale, object recognition, mammografie

Seconda Sessione

Anno Accademico 2009 - 2010

Indice

Elenco delle figure

INTRODUZIONE

Questa tesi ha lo scopo di offrire una panoramica del lavoro di ricerca di Thomas Serre[1] proponendone una possibile applicazione nel campo della radiologia, ed in particolare della mammografia - un esame del seno umano effettuato tramite una bassa dose di raggi X.

Attualmente Serre occupa il ruolo di professore assistente presso il dipartimento di scienze cognitive e linguistiche alla Brown[2]. Tra il 2000 e il 2006 è stato titolare di un dottorato di ricerca in neuroscienze computazionali presso Massachusetts Institute of Technology (MIT). In questo periodo la sua attività è stata rivolta a comprendere i meccanismi del cervello che stanno alla base del riconoscimento degli oggetti e di complesse scene visive usando una combinazione di tecniche comportamentali, fisiologiche e di elaborazione delle immagini, il tutto supportato da rigorosi modelli computazionali.

Insieme a Tomaso Poggio e ad altri colleghi del MIT, Serre ha sviluppato un modello computazionale a larga scala per il riconoscimento visivo nella corteccia.

Qui, In particolare, viene posto l'accento sulla sua tesi di dottorato intitolata: "*Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines*"[Ser06], ovvero l'apprendimento di un dizionario delle componenti morfologiche all'interno della corteccia visiva, dove viene descritto un modello quantitativo che tenta di spiegare

[1] http://serre-lab.clps.brown.edu/
[2] http://www.brown.edu/

il funzionamento dei circuiti di tipo feedforward della via visiva ventrale o occipito-temporale - a partire dalle aree corticali V1 e V2 sino a V4, IT e PFC[3].

Il modello qui descritto è un'estensione di quello originale di Riesenhuber e Poggio del 1999[RiePog99] che prende il nome di "*Hierarchical Model and X*", comunemente denominato HMAX. Inoltre, è consistente con la teoria generale dei processi visivi - estendendo il modello gerarchico di Hubel e Wiesel[HubWie59] che va dall'area visiva primaria a quella extrastriata - e tenta di spiegarne le prime centinaia di millisecondi di attività.

Una delle proprietà chiave è l'apprendimento di un dizionario generico delle componenti morfologiche dall'area corticale V2 all'IT, il quale fornisce una rappresentazione invariante ai circuiti di categorizzazione rapida presenti nelle aree superiori del cervello. Questo dizionario è appreso in maniera non supervisionata a partire da immagini naturali e costituisce un set "campione" ampio e ridondante con differenti complessità ed invarianze.

Nel Capitolo 1 si presentaranno i principali modelli gerarchici che ne costituiscono l'ossatura. Nel Capitolo 2 verrà fornita una descrizione del modello base e ci si soffermerà su quello che è il contributo originale di Serre, ovvero la creazione di un dizionario generico delle componenti morfologiche da V2 a IT. Nel Capitolo 3 il modello verrà confrontato con i neuroni, le immagini naturali, e verranno comparate le sue perfomance con quelle degli esseri umani in presenza di un task di categorizzazione rapida animale *vs.* non animale. Nel Capitolo 4 il modello verrà trattato dal punto di vista implementativo. Infatti, il modello è stato realizzato utilizzando Matlab[4] e C++.

Infine, nel Capitolo 5 si vedrà se il modello possa essere utilizzato come strumento di classificazione a prescindere dal task di categorizzazione rapida animale *vs.* non animale, come nel caso delle immagini di mammografie.

[3] V1: area visiva primaria; V2: area visiva extrastriata; IT: corteccia infero-temporale; PFC: corteccia pre-frontale.

[4] http://www.mathworks.it/

Capitolo 1

La corteccia Visiva

La corteccia visiva è composta da diverse aree che tendono ad essere organizzate gerarchicamente[FelvEs91], vedi Figura 1.1.

In genere si tende a dire che il flusso delle informazioni attraverso la corteccia visiva si può dividere in due vie: la via ventrale e la via dorsale[MUM83, DeYEss88].

Il riconoscimento degli oggetti si pensa sia mediato da quella ventrale che è organizzata in una serie di fasi neuralmente interconnesse, a partire dalla retina, passando per l'area visiva primaria, fino a quella inferotemporale[UngHax94]; fasi, che a turno giocano un ruolo chiave nel riconoscimento invariante degli oggetti[Tan96] e forniscono una sorgente di input più ampia per la corteccia prefrontale che collega la percezione alla memoria e all'azione[Mil00].

La via visiva ventrale (Figura 1.2) comincia nella corteccia visiva primaria all'interno del lobo occipitale e termina nelle aree temporali inferiori. Questa via è specializzata nell'elaborazione dell'informazione legata agli oggetti (colore, forma, dimensione, orientamento, eccetera). È anche definita la via del "cosa". Essa ha un forte legame con la memoria a lungo termine, il sistema limbico che controlla le emozioni e la via visiva dorsale o occipito-parietale

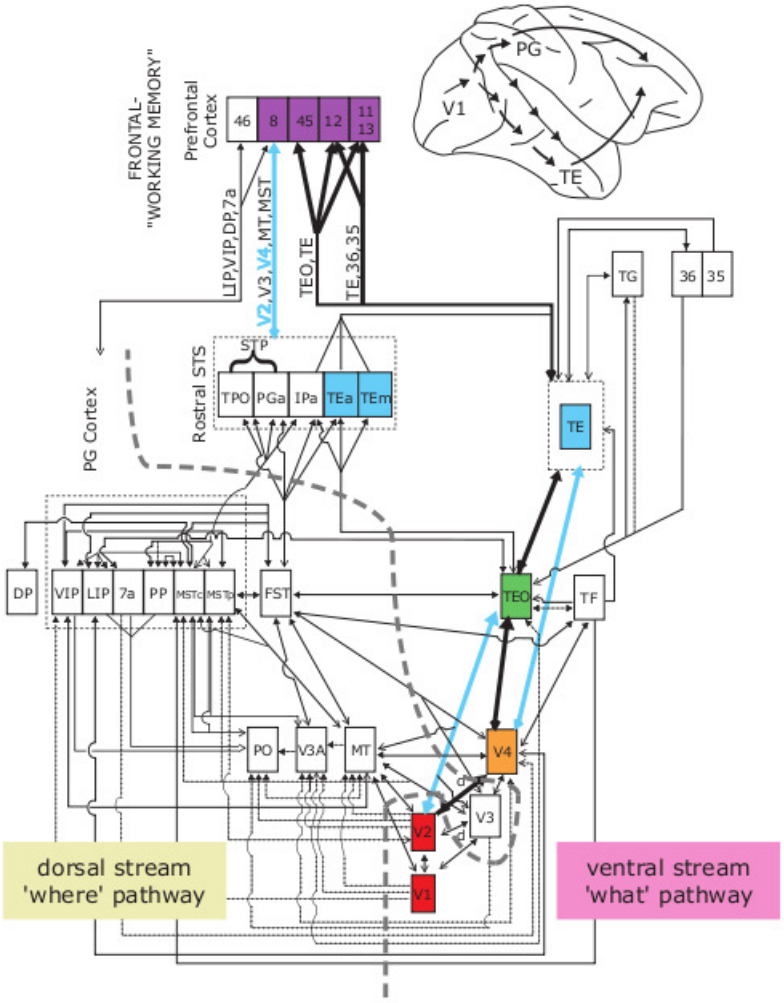


Figura 1.1: la via ventrale della corteccia visiva e il riconoscimento degli oggetti

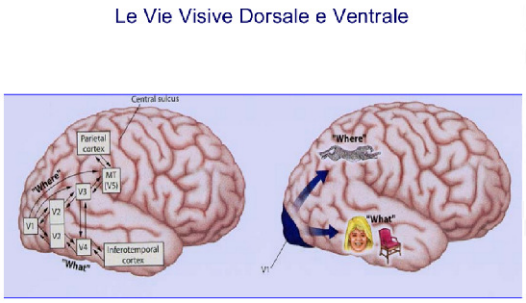


Figura 1.2: Le Vie Visive Dorsale e Ventrale

che invece si occupa del "dove".

Nel corso degli ultimi decenni, diversi studi fisiologici sui primati non umani hanno stabilito un nucleo di base consolidato dei meccanismi corticali di riconoscimento che sembrano essere ampiamente accettati e che confermano e raffinano le vecchie conoscenze della neurofisiologia. La Figura 1.3 illustra un'architettura feedforward basata sulla via ventrale che gode di queste proprietà generali.

1.1 L'organizzazione gerarchica

1.1.1 Costruzione di una rappresentazione invariante da V1 a IT

Il fatto che la via ventrale sia in grado di fornire una sorgente di input più ampia alla corteccia prefrontale suggerisce un aumento graduale sia dell'invarianza che della complessità degli stimoli preferiti dei neuroni lungo la via.

La nozione di gerarchia del processo visivo è stata introdotta da Hubel e Wiesel. Essi, in particolare, hanno descritto come, a partire dal raggruppamento di molte cellule semplici con un campo ricettivo di dimensioni ri-

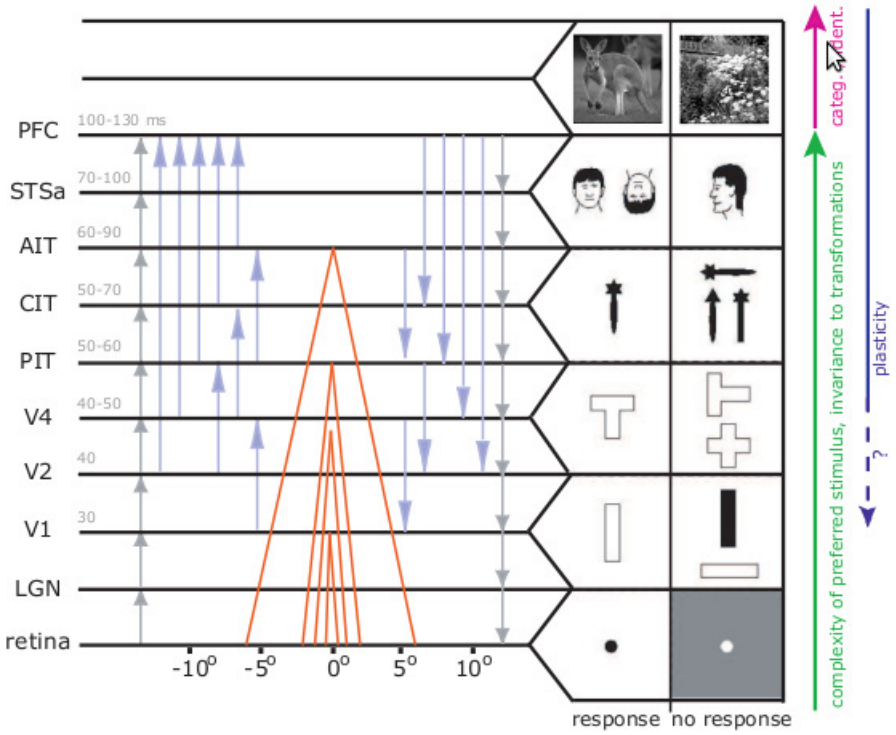


Figura 1.3: L'organizzazione della corteccia visiva basata su un nucleo di conoscenze che sono state accumulate nel corso di 30 anni

dotte in grado di rispondere efficaciemente in presenza di barre con una certa posizione e orientamento, sia possibile ottenere la risposta di una cellula complessa, la quale risponde anch'essa ad una barra ma con un qualsiasi orientamento all'interno del campo ricettivo.

Oltre V1, i neuroni lungo la via ventrale mostrano un aumento della dimensione dei propri campi ricettivi così come della complessità dei propri stimoli preferiti. Ad esempio in V2 i neuroni sono sensibili agli stimoli angolati, attraverso una combinazione non lineare delle sottounità. Oppure, in V4 vengono riconosciute le caratteristiche degli oggetti di moderata complessità, come i bordi. Infine nell'IT vengono riconosciuti i colori, l'orientamento, le texture, la direzione del movimento, le forme, eccetera.

Perciò, è chiaro che man mano che si avanza nella gerarchia la dimensione dei neuroni cresce e di conseguenza anche la dimensione dei recettori ivi applicati, con il risultato di aumentare la complessità degli stimoli preferiti. Infatti, in cima alla gerarchia della via ventrale si trova la corteccia inferotemporale anteriore (AIT) che è in grado di riconoscere stimoli complessi come le parti del corpo, le mani, le facce, eccetera. La caratteristica delle cellule dell'AIT è la robustezza nel rispondere alla variazione degli stimoli come, ad esempio, la scala o il cambio di posizione.

Oltretutto, molti neuroni mostrano specificità per certe condizioni di luce o visione degli oggetti. Questo è stato dimostrato da Logothetis [LPBP95] che nel tentativo di addestrare delle scimmie ad eseguire un task di riconoscimento con viste isolate di nuovi oggetti tridimensionali, andando ad analizzare il comportamento dell'IT, ha scoperto che la maggior parte dei neuroni si sintonizzava sulla vista di uno solo degli oggetti dell'addestramento. Inoltre, ha osservato che mentre le scimmie venivano addestrate con un oggetto nella stessa posizione e dimensione, i neuroni naturalmente esibivano un'invarianza media alla traslazione di 4° e alla scala di 2 ottave[RiePog99]. Per cui, laddove il riconoscimento invariante alla vista richieda l'esperienza visiva del nuovo oggetto, sembra che sia immediatamente presente una significativa

invarianza sia alla scala che alla posizione.

1.1.2 Oltre l'IT: I Circuiti Task-Specific

I risultati di Logothetis sono in accordo con la letteratura che suggerisce che una varietà di task di riconoscimento di oggetti possa essere realizzata attraverso una combinazione lineare di poche unità sintonizzate su specifici task.

Contrariamente trasformazioni non affini come la luminosità richiedono specifici esempi di allenamento. Questo suggerisce, in accordo con la fisiologia, che la maggioranza dei neuroni nell'IT dovrebbero esibire un range di invarianza ai cambiamenti della posizione e della scala e, inoltre, dovrebbero essere sensibili ai cambiamenti della rotazione 3-D e all'illuminazione.

Perciò, come suggerito da Poggio e Riesenhuber sarebbe utile utilizzare un dizionario generico delle componenti morfologiche[RiePog00]. Tale dizionario, da V1 all'IT, dovrebbe fornire degli input invarianti alla posizione e alla scala ai circuiti specifici dei task che si trovano oltre l'IT per generalizzare le trasformazioni non affini. Ad esempio, un circuito per il riconoscimento della posa delle facce, andrebbe costruito possibilimente in PFC, includendo diverse unità sintonizzate su differenti esempi di facce, persone, viste e condizioni di luce.

1.2 Apprendimento e plasticità

Per quanto visto sinora, si può supporre la plasticità della croteccia visiva? Sì, si può.

Parlando da un punto di vista computazionale si può supporre che l'apprendimento avvenga in tutte le aree della corteccia visiva. Ad esempio, nell'apprendimento di un oggetto con una rappresentazione ad alto livello sarà coinvolta l'area dell'IT o del PFC, viceversa se riguarda una discriminante rispetto alla direzione allora saranno coinvolte le prime fasi della ger-

archia, ad esempio V1. Ed è fortemente probabile che i cambiamenti nelle fasi ad alto livello avvengano molto più velocemente rispetto a quelle a basso livello (perché sono coinvolte molte meno unità). In questo senso per plasiticità si intende la facilità di effettuare dei cambiamenti. Infatti, alcuni studi[FRPM03] hanno dimostrato la plasticità a livello di PFC. È stato anche dimostrato[Miy88] che dopo aver addestrato degli animali per quantificare il ritardo nel riconoscimento di oggetti campioni, alcuni neuroni nell'IT sono diventati selettivi sia per lo stimolo campione che per quello di test, mentre altri lo sono diventati rispetto allo stimolo target durante il periodo di ritardo. I primi indicano una plasticità a livello di IT, mentre i secondi ne suggeriscono una ad alto livello, come il PFC.

Questi studi hanno dimostrato che la risposta agli stimoli da parte dei neuroni dipende fortemente dalle condizioni in cui avviene l'esposizione ed è maggiore laddove gli oggetti mostrati sono familiari. Questo aumenta il clustering dei neuroni che rispondono agli stimoli[EJD00].

Alla plasticità ed ad un migliore apprendimento influiscono anche una lunga esposizione visuale e l'addestramento. Questo fa sì che una maggiore esposizione al test induca un cambiamento nei neuroni dell'IT o nel PFC [LPBP95].

1.3 Il riconoscimento immediato

Sulla base degli studi comportamentali è noto che il riconoscimento è possibile per le sequenze di immagini che sono mostrate attraverso una RSVP (*rapid visual series presentation*) che non consentono un tempo sufficiente al movimento degli occhi o per un cambio di attenzione[Pot75]. In particolare è stato dimostrato che un essere umano può individuare un oggetto target inserito all'interno di una sequenza di immagini quando è presentato con una frequenza superiore ai 10 frame al secondo.

Oltre all'RSVP esiste anche l'*ultra rapid visual series categorization* o più semplicemente la categorizzazione rapida[TFM96] che ha mostrato i seguenti risultati:

- oltre agli esseri umani anche le scimmie sono in grado di eseguire una categorizzazione rapida accurata e veloce. Addirittura a scapito dell'accuratezza sono circa il 30% più veloci degli esseri umani.

- La categorizzazione rapida, oltre che con gli oggetti naturali è applicabile anche agli oggetti artificiali.

- La rimozione dell'informazione del colore durante la presentazione delle immagini non ha effetti sensibili sul riconoscimento. Introduce sono una latenza trascurabile sia negli uomini che nelle scimmie.

- Il tempo di reazione più veloce non può essere aumentato in seguito all'addestramento e alla familiarità.

- La categorizzazione rapida può avvenire anche se non si sta osservando direttamente l'oggetto, ad esempio quando appare sia vicino che lontano dalla fovea[1].

- È molto robusta rispetto alla rotazione delle immagini, sia in termini di tempo che di prestazioni.

- È possibile anche quando si abbassa il tempo di presentazione a 6.25 ms ma solo in presenza di backward masking, ovvero di una seconda immagine presentata subito dopo la prima con un ritardo di 40-50ms.

- Non richiede attenzione. Il livello delle performance rimane alto anche quando 2 immagini sono mostrate contemporaneamente, una per ogni emisfero.

[1] La fovea è una regione centrale della retina di massima acuità visiva

- Un DEEG (elettroencefalogramma differenziale) suggerisce che i task siano risolti in 150 ms[TFM96].

A questo punto si hanno alcuni capi saldi per quanto riguarda la via ventrale della corteccia visiva:

1. I neuroni, lungo la gerarchia, divengono selettivi a stimoli sempre più complessi e allo stesso tempo aumentano di invarianza. In parallelo, la dimensione dei campi ricettivi dei neuroni aumenta.

2. L'apprendimento può indurre dei veloci cambiamenti sulle proprietà di sintonizzazione dei neuroni probabilmente in tutte le aree della gerarchia e sicuramente in quelle ad alto livello, tipo IT e PFC.

3. Il processo che media il riconoscimento immediato è di tipo feedforward e non coinvolge informazioni che riguardano il colore e nemmeno l'attenzione.

1.4 I modelli di riconoscimento

I modelli per il riconoscimento degli oggetti nella corteccia visiva si dividono in 2 categorie principali: i modelli che fanno parte dell'approccio normalizzante e quelli che s basano sullo schema di replicazione totale anche noto come *invariant features*[RiePog99].

L'approccio normalizzante è quello standard nella computer vision. Tipicamente si ha un'immagine che è prima trasformata in un'immagine piramidale (è una struttura gerarchica composta di n livelli della stessa immagine a differenti risoluzioni) e poi viene scansionata su tutte le scale e posizioni utilizzando un template. Un'implementazione biologica plausibile di questo modello è il circuito di shift[OAV96]. In questo approccio i circuiti di instradamento dinamico controllano la forza delle connessioni tra i livelli di input e quelli di output spegnendo e accendendo le varie connessioni così da

estrarre una rappresentazione normalizzata all'interno della regione attesa. Un modello forse più plausibile è il gain-field[SalAbb97].

Però tutti questi modelli dipendono pesantemente dalla Back-Projection[2]. Inoltre, bisogna considerare che se da un lato certi meccanismi possano essere utilizzati all'interno della corteccia visiva (ad esempio il modello gain-field trova conferma nelle informazioni contenute in V4) dall'altro lato è che chiaro che non possono essere compatibili con i limiti fisiologici del riconoscimento immediato e della categorizzazione rapida. Oltretutto, ci sono numerosissime prove che la Back-Projection non giochi un ruolo chiave nelle prime centinaia di millisecondi del processo visivo.

Alla luce di ciò l'approccio normalizzate non è adatto allo scopo di fornire un modello invariante del riconoscimento degli oggetti da parte della corteccia visiva e della conseguente possibilità di costruire un dizionario delle componenti morfologiche.

In base a queste premesse sono stati sviluppati diversi modelli di tipo feedforward per il riconoscimento degli oggetti nella corteccia. Quelli di cui si tiene conto sono quelli che appaiono essere compatibili con la fisiologia e l'anatomia della via ventrale. Questi modelli condividono l'idea base che il sistema visivo è un processo di tipo feedforward gerarchico dove il range di invarianza e la complessità delle caratteristiche principali cresce man mano che si va verso i livelli più alti della gerarchia.

1.4.1 I Circuiti corticali e le chiavi di computazione

Come discusso in precedenza, data l'immediata selettività e sintonizzazione delle cellule all'interno di una ridottissima finestra temporale che va dai 10 ai 30 ms, i circuiti neurali devono operare solamente su pochi picchi. Questo

[2]La Back-Projection è un metodo di ricostruzione delle immagini. L'esempio più comune è sicuramente quello della TAC (Tomografia Assiale Computerizzata) dove una persona che viene bombardata dai raggi X da diversi angoli produce una serie di densità bidimensionale. Sovrapponendo queste proiezioni bidimensioni si riesce a "ricostruire" l'immagine.

suggerische che durante questo periodo i neuroni possano trasmettere solo poche informazioni.

Per aggirare questo problema si considerano dei "moduli computazionali" (Figura 1.3.) o unità che sono dei gruppi di cellule equivalenti con parametri e input identici provenienti da altre unità all'interno del circuito. Questi moduli diventano l'unità base del processo, piuttosto che considerare i neuroni individualmente. In questo modo le informazioni trasmesse da uno fase a quella successiva non riguardano quanto un neurone trasmette ma quanti neuroni di un particolare tipo trasmettono, cioè sono stati attivati in quella finestra temporale. In aggiunta, considerare come unità base di elaborazione questi moduli fa sì che alle cellule post-sinaptiche arrivino fino a 2n picchi di informazione a partire dagli n neuroni.

Ovviamente questa soluzione non è ottimale perché è ridondante nell'organizzazione corticale. Infatti circa 80-100 neuroni in una colonna generale sono circa 2.5 volte superiori in numero ai neuroni presenti nella fase V1[Mon57, Mon97]. Però, perifrasando Mountcastle: l'effettiva unità di operazione in un sistema distribuito di questo tipo non è il singolo neurone, ma il gruppo di cellule con proprietà simili e connesse anatomicamente. Questo suggerisce che il range dinamico in questo modo aumenta di un fattore n rispetto al singolo neurone.

1.4.1.1 Le chiavi di computazione

L'obiettivo è quello di trovare un compromesso tra la specificità e l'invarianza. Infatti, il riconoscimento deve essere in grado di discriminare finemente tra oggetti differenti o classi di oggetti mentre allo stesso tempo deve essere tollerante rispetto alle trasformazioni come scale, traslazioni, cambi di visuale, oltre a trasformazioni non rigide come le espressioni facciali e in caso di categorizzazione anche a variazioni nella forma della classe. Cioè la difficoltà principale nel riconoscimento degli oggetti sta nel migliorare questo trade-off tra la selettività e l'invarianza.

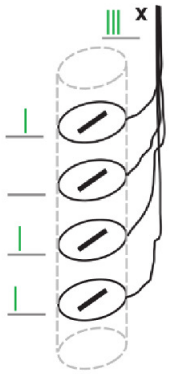


Figura 1.4: I moduli computazionali nella corteccia.

A questo proposito Poggio e Riesenhuber [RiePog99] suggeriscono che sono solamente 2 le funzionalità classificate che sono necessarie a migliorare tale compromesso: le unità semplici S e complesse C.

Le unità semplici S Un'unità semplice S esegue un'operazione di sintonizzazione (TUNING) sui propri afferenti (le cellule che le forniscono l'input) per costruire la selettività degli oggetti. L'unità S riceve gli input dalle unità organizzate retinotipicamente (è una proiezione di un'immagine nel cervello che mantiene le informazioni spaziali così come le ha percepite la retina) sintonizzate su differenti stimoli preferiti e combina queste sottounità utilizzando una funzione di sintonizzazione Gaussiana così da aumentare la selettività e la complessità di tali stimoli.

Da un punto di vista computazionale, utilizzare una funzione di sintonizzazione Gaussiana deve essere un punto chiave nell'abilità di generalizzazione nella corteccia visiva. Le reti che combinano l'attività di molte unità sintonizzate con un profilo Gaussiano su differenti esempi di addestramento hanno mostrato essere potenti schemi di apprendimento[PogGir90].

L'operazione analoga alla sintonizzazione nella computer vision è il template matching.

Le unità complesse C Un'unità complessa C esegue un'operazione di tipo MASSIMO[3] sui suoi afferenti per guadagnare invarianza rispetto alle trasformazioni di moltissimi oggetti. L'unità C riceve in input da unità, organizzate retinotopicamente come S, sintonizzate sullo stesso stimolo ma in posizioni e scale leggermente diverse. Su tali input successivamente esegue un'operazione di MASSIMO per introdurre una tolleranza alle scale e alle traslazioni.

L'esistenza di un'operazione di MASSIMO nella corteccia visiva è stata ipotizzata da Poggio e Riesenhuber [RiePog99].

1.4.1.2 Descrizione matematica delle due operazioni

Come discusso in precedenza, un aumento graduale sia della selettività che della scala (come osservato lungo la via ventrale) è critico per evitare sia l'esplosione combinatoria del numero delle unità, sia il problema di binding tra le feature. Di seguito si dà una breve approssimazione matematica delle due operazioni appena introdotte.

Si denota y come la risposta di un'unità (semplice o complessa). L'insieme degli input della cellula (ad esempio le unità pre-sinaptiche) sono denotate con l'indice $j = 1 \ldots N \in \mathbb{N}$. Quando j riceve in input un pattern di attività $\mathrm{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_N)$, la risposta di y è data da:

Come menzionato prima, per una cellula complessa, gli input x_j delle unità sono organizzati retinotopicamente (ovvero sono selezionati da una griglia di afferenti grande $m \; x \; m$ con la stessa selettività). Ad esempio, nel caso di una cellula complessa di tipo V1 sintonizzata su di una barra

[3] L'operazione di massimo non è necessario che sia esatta. Infatti, fornisce una rappresentazione invariante alla posizione e alla scala a livello di IT, con una minima perdita di performance nel processo di riconoscimento.

orizzontale, tutte le sottounità saranno sintonizzate su una barra orizzontale ma in posizioni e scale leggermente diverse.

Allo stesso modo, una descrizione idealizzata della risposta di un'unità semplice è data da:

Dove σ definisce l'adattamento della sintonizzazione dell'unità attorno ai suoi stimoli preferiti corrispondente alla forze sinaptiche w = *(w_1, ... , w_n).* Come per le cellule complesse, le sottounità delle cellule semplici sono organizzate retinotopicamente (selezionate da una griglia m x m di possibili afferenti). Ma, a differenza delle cellule complesse, le sottounità di una cellula semplice possono avere differenti selettività per accrescere la complessità dello stimolo preferito. Ad esempio, per le unità S_2 le sottounità sono cellule complesse di tipo V1 (con un ridotto range di invarianza alla posizione e alla scala) con differenti direzioni preferite. L'Eq. 1.2 si basa su di una funzione di sintonizzazione a forma di campana (Gaussiana). Per cui, la risposta dell'unità è massima ($y = 1$) quando il pattern corrente (x) matcha esattamente con il vettore dei pesi sinaptici w (ad esempio la vista frontale di una faccia) e decresce con un profilo gaussiano man mano che x diviene dissimile da w (ad esempio non appena la faccia è ruotata).

1.5 Contributo originale

È su queste basi che si sviluppa il modello proposto da Serre. Infatti, trae spunto dai modelli neurobiologici già esistenti e in particolare estende significativamente il modello HMAX di Poggio e Riesenhuber [RiePog99].

Una delle novità del modello è l'apprendimento di un dizionario generico di componenti morfologiche da V2 a IT che fornisce una ricca rappresentazione ai circuiti di categorizzazione nelle aree superiori del cervello.

L'architettura gerarchica progressivamente risulta essere più invariante rispetto alla posizione e alla scala preservando la selettività delle unità.

Questo vocabolario di unità sintonizzate è appreso a partire da un insieme di immagini naturali. L'apprendimento coinvolge tutti i livelli della gerarchia. Però, mentre i livelli inferiori sono adibiti allo sviluppo, le fasi intermedie prevedono un apprendimento non supervisionato in cui ogni unità diventa sintonizzata verso una differente sezione (patch) di un'immagine naturale. Inoltre, essendo che si fa uso di un largo numero di unità lungo tutta la gerarchia, il dizionario risulta essere ridondante di caratteristiche parziali che dividono lo stesso range di selettività e la stessa invarianza.

Come risultato di questa nuova fase di apprendimento, la nuova architettura contiene un totale di circa 10 milioni di unità sintonizzate. In cima alla gerarchia, le unità di classificazione si basano su di un dizionario di circa 6000 unità, espressione delle caratteristiche delle immagine con differenti livelli di invarianza e selettività. Questo è circa 2 o 3 volte superiore al numero di caratteristiche usate dai modelli biologici e dai sistemi di computer vision che si basano su un set di 10-100 feature al massimo

Capitolo 2

Il Modello Base

In questo capitolo viene approfondito quanto detto sinora e se ne fornisce un modello implementativo fedele.

L'elemento chiave è la costruzione di un dizionario generico delle componenti morfologiche estratto da V1 a IT per fornire una rappresentazione invariante che possa essere utilizzata dalle aree corticali superiori, come in PFC, per addestrare i circuiti task-specific al riconoscimento delle diverse categorie di oggetti.

2.1 Il Dizionario da V1 a IT

Il modello implementativo che riflette l'organizzazione generale della via ventrale della corteccia visiva è mostrato nella Figura 2.1.

Qui è descritto il tentativo di mapping tra le primitive funzionali e i livelli del modello (a destra nella figura) con le fasi corticali del sistema visivo dei primati (a sinistra nella figura). Le corrispondenze sono illustrate per mezzo dei colori. I livelli in cui intervengono le unità semplici S, rappresentate da dei dischi piani, costruiscono una complessità crescente e una rappresentazione specifica combinando la risposta di molte sottounità con differenti selettività e esibiscono una sintonizzazione di tipo gaussiana. I livelli S sono intervallati

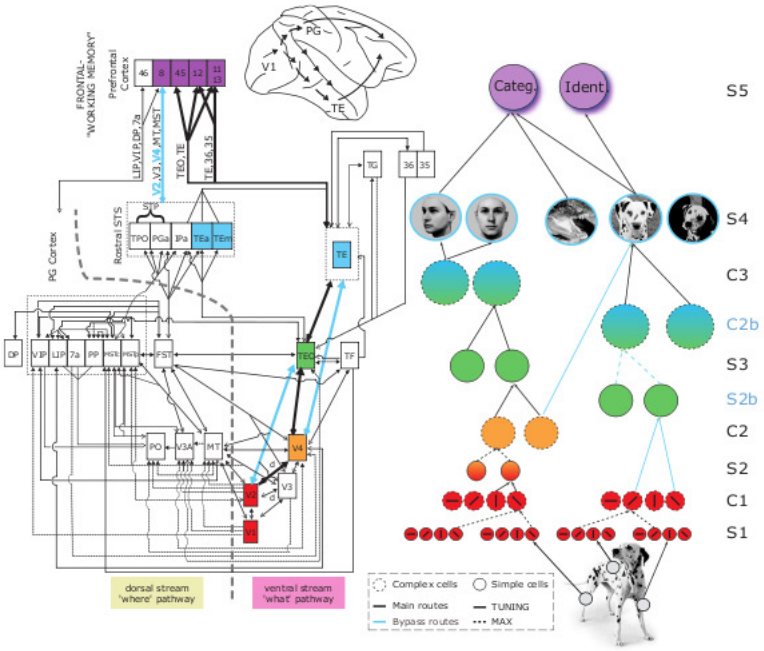


Figura 2.1: Mapping tra il modello e le aree corticali

dai livelli complessi C, rappresentati da dischi tratteggiati, i quali combinano le sottounità con selettività simili ma con una posizione e scala leggermente differenti per accrescere l'invarianza alle trasformazioni degli oggetti. L'operazione di aggregazione effettuata nei livelli C è quella di MASSIMO. Le frecce nere corrispondono alla main route che fornisce gli input all'IT. Le freccie blu illustrano la bypass route.

Quindi, lungo la gerarchia, da V1 a IT, si alternano 2 fasi così da fornire una rappresentazione degli oggetti nell'IT che viene letta al di fuori dalle aree corticali superiori per creare un ampio array di task visivi. Queste due fasi sono:

- **fasi semplici (S)**: costruiscono una complessità crescente e una rappresentazione specifica combinando la risposta di tante sottounità con differenti selettività con un'operazione di sintonizzazione (vedi Eq. 1.2.).
- **fasi complesse (C)**: costruiscono una rappresentazione invariante incrementale combinando la risposta di diverse sottounità con la stessa selettività ma con una posizione e una scala leggermente diverse con un'operazione di tipo MASSIMO (vedi Eq. 1.1).

Di seguito si esaminano le singole fasi.

2.1.1 Cellule semplici e complesse in V1

L'input al modello è un'immagine in scala di grigi. Tipicamente, l'immagine ha un range che va dai 140x140 pixel ai 256x256 pixel che corrispondono rispettivamente a circa 4° e 7° del campo visivo[1].

Le unità semplici S_1 Innanzitutto, l'immagine di input è analizzata da un array multidimensionale di unità S_1 che corrispondono alle cellule semplici in V1 di Hubel e Wiesel. Le unità seguono il modello base delle cellule semplici,

[1]Per convenzione, 1° dell'angolo visivo nel modello corrisponde ad un regione di 36x36 pixel dell'immagine in input

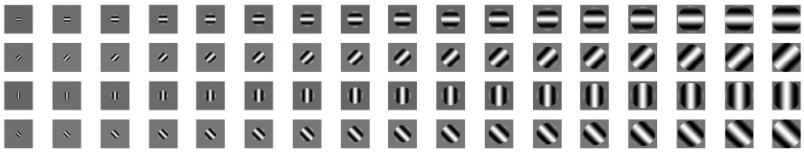


Figura 2.2: Organizzazione dei campi ricettivi di S_1

cioè sono dei filtri costituiti da sottoregioni allineate e alternativamente ON e OFF, le quali condividono un comune asse di elongazione che definisce l'orientamento preferito della cellula.

Ogni porzione del campo visivo (o meglio ogni pixel dell'immagine di input) è analizzata usando un set completo dei tipi di unità le quali devono corrispondere ad una macro colonna (l'insieme di tutti i tipi di mini colonne, ad esempio le 4 direzioni e le 2 fasi in in S_1) V1 (Figura 2.3).

Come discusso in precedenza, le unità nel modello sono simili ai moduli computazionali della corteccia, cioè un agglomerato di n cellule equivalenti con lo stesso input (ovvero che condividono lo stesso vettore w). Ogni mini colonna nel modello è cioè composta di molti moduli a differenti scale.

Le unità S_1, come le altre unità semplici all'interno del modello, eseguono un'operazione di sintonizzazione tra il pattern di input in ingresso x e il loro vettore dei pesi sinaptici w. La risposta di un'unità S_1 è massima quando x e w matchano perfettamente. Tipicamente un'alta risposta è ottenuta quando l'orientamento dello stimolo (ad esempio una barra di altezza e larghezza ottimale o un bordo) matcha perfettamente il filtro di orientamento e la risposta cala man mano che l'orientamento dello stimolo e il filtro differiscono.

Le unità complesse C_1 Il livello C_1 corrisponde alle cellule complesse striate. Ognuna delle unità complesse C_1 riceve l'output di un gruppo di unità S_1 con lo stesso orientamento preferito (e 2 fasi opposte) ma con differenti posizioni e dimensioni. L'operazione per mezzo della quale le risposte delle

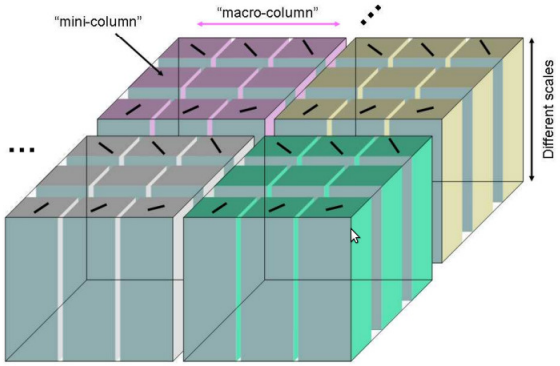


Figura 2.3: Organizzazione a colonne funzionali del modello

unità S_1 sono combinate al livello C_1 è un'operazione non lineare di tipo MASSIMO, cosicchè la risposta delle unità C_1 è determinata dal più forte di tutti i suoi input. Questa operazione garantisce un aumento della tolleranza ai cambiamenti di posizione e scala da S_1 a C_1.

Questo principio è illustrato nella Figura 2.4. In pratica, ogni unità C_1 riceve i suoi input dalle unità S_1con lo stesso orientamento preferito (ad esempio 0°) ma a differenti scale e posizioni (rimanendo all'interno di un intorno di 3x3). Quando la lettera A in figura è shiftata dalla posizione 1 alla 2(a), essa attiva S_1 in due differenti posizioni. Aggregando l'attività di tutte le unità nell'intorno, l'unità C_1diviene indifferente alla posizione dello stimolo. Stessa cosa per quanto riguarda la scala.

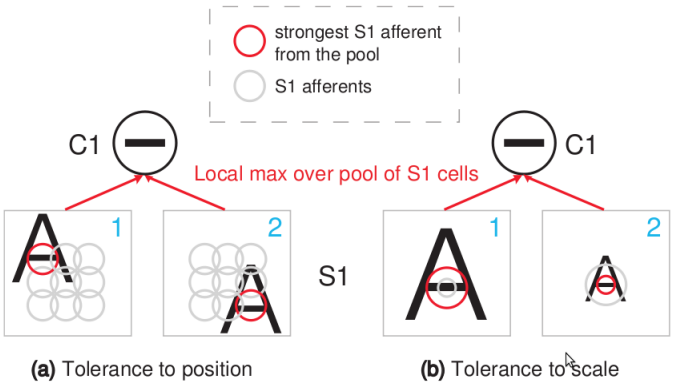


Figura 2.4: Tolleranza da S_1 a C_1

Organizzazione funzionale

Le unità S_1e C_1 La popolazione dell unità S_1 consiste di 136 tipi di unità, cioè 2 fasi x 4 direzioni x 17 dimensioni. La Figura 2.2 mostra i differenti vettori dei pesi corrispondenti ai differenti tipi di unità S_1. Matematicamente il vettore dei pesi w delle unità S_1 prende la forma da un filtro di Gabor[2] [LPBP95], che fornisce un buon modello della struttura dei campi ricettivi delle cellule semplici e può essere descritta dalla seguente equazione:

$$F(u_1, u_2) = exp\left(-\frac{(\hat{u}_1^2 + \gamma^2\hat{u}_2^2)}{2\sigma^2}\right)\ x\ cos\left(\frac{2\pi}{\lambda}\hat{u}_1 + \phi\right)$$

con

[2]Il filtro di Gabor è un filtro lineare la cui risposta all'impulso è definita da una funzione armonica moltiplicata per una funzione Gaussiana. In forza del teorema di convoluzione la trasformata di Fourier della risposta all'impulso di un filtro di Gabor risulta essere la convoluzione fra la trasformata di Fourier della funzione armonica e la trasformata di Fourier della funzione Gaussiana.

$$\hat{u}_1 = u_1 cos\theta + u_2 sin\theta$$

e

$$\hat{u}_2 = -u_1 sin\theta + u_2 cos\theta$$

I 5 paramentri, cioè, la direzione del filtro θ, l'aspect ratio (rapporto tra la larghezza e l'altezza di una figura bidimensionale) γ, l'effettiva larghezza σ, la fase ϕ e la lungezza d'onda λ, determinano le proprietà spaziali dei campi ricettivi delle unità. La sintonizzazione delle cellule semplici nella corteccia varia sostanzialmente lungo queste dimensioni. Vengono considerate 4 direzioni ($\theta = 0°, 45°, 90°, 135°$). Questa è una semplificazione ed è sufficiente a fornire invarianza alla rotazione e alla dimensione al livello S_4. Affinché le dimensioni dei campi ricettivi siano consistenti con i valori delle cellule semplici parafoveali si considerano 17 filtri con dimensioni che vanno da 7 x 7 pixel (0.2° dell'angolo visovo) a 39 x 39 (1.1°) con step di 2 pixel.

I restanti paramentri sono aggiustati manualmente cosicché le proprietà di sintonizzazione delle unità S_1 matchino strettamente quelle delle cellule semplici parafoveali di V1. Lo stesso vale per C_1.

Le unità S_2 e C_2 Al livello S_2 le unità aggregano le attività di $n = 10$ unità complesse C_1 attraverso un'operazione di sintonizzazione aumentando così la complessità degli stimoli preferiti. Nella fase C_2 le unità complesse divengono selettive per lo stesso stimolo delle sue unità afferenti S_2 aumentandone l'invarianza alla posizione e alla scala. La dimensione del campo ricettivo in C_2 è compresa tra 1.1° e 3.0°.

Le unità S_3 e C_3 A livello S_3 per ogni unità si considerano $n = 100$ unità afferenti. In questo modo aumenta la selettività e l'invarianza. Infatti,

i livelli S_3 e C_3 forniscono una rappresentazione basata sulla sintonizzazione alla forma delle componenti.

I parametri di aggregazione sono stati aggiustati manualmente in modo tale che nella fase successiva il livello S_4 esibisca proprietà di sintonizzazzione e invarianza simili a quelle delle cellule dell'AIT. La dimensione del campo ricettivo in S_3 è tra 1.2° e 3.2° mentre quella di C_3 ed S_4 copre tutto lo stimolo (da 4° x 4° a 7° x 7°).

2.1.2 Oltre V1: Caratteristiche di moderata complessità

Superato il primo livello della gerarchia, da V1, le informazioni visive sono propagate a V2, V4, e all'IT, il quale è stato dimostrato essere fondamentale nell'abilità dei primati di eseguire un riconoscimento invariante. Questo è fatto per mezzo di due strade. La strada "principale" che segue passo dopo passo la gerarchia e la strada "breve" che bypassa alcune di queste fasi (Figura 2.1). Si suppone che l'uso della strada breve possa essere d'aiuto nella creazione di un set di feature più ricco e con un vario livello di selettività ed invarianza.

Main route Nel livello S_2 le unità aggregano le attività delle unità C_1 organizzate retinotopicamente a differenti orientazioni su di un piccolo intorno (determinato dalla dimensione del vicinato). L'operazione di sintonizzazione da C_1 a S_2 permette di aumentare sia la complessità degli stimoli che la selettività delle unità.

Oltre il livello S_2, la sintonizzazione di tutte le unità S è stata appresa, in maniera non controllata dalla immagini naturali.

Nel modello ci sono circa 1000 tipi di unità S_2 che corrispondono alle differenti combinazioni di risposte di unità C_1. Inoltre, il livello S_2 è organizzato in colonne sovrapposte cosicchè una piccola parte del campo visivo è analizzato da una macro colonna che contiene n tipi di unità su tutta la scala.

Nella fase C_2 le unità aggregano le unità S_2 che sono sintonizzate sullo stesso stimolo. Le unità C_2 sono perciò selettive per lo stesso stimolo delle unità S_2. Le unità C_2 sono compatibili con le informazioni contenute in V4.

Oltre S_2 e C_2 lo stesso processo viene iterato ancora una volta per accrescere la complessità degli stimoli al livello S_3. Nella fase successiva, le unità C_3 divengono selettive alle caratteristiche moderatamente complesse di S_3 ma con un range di invarianza maggiore.

I livelli S_3 e C_3 forniscono una rappresentazione basata ampiamente sulle componenti morfologiche. I paramentri di aggregazione delle unità C_3 sono aggiustati manualmente in modo tale che nel passo successivo le unità S_4 esibiscano proprietà di sintonizzazione ed invarianza del tutto simili a quelle delle cellule dell' AIT. A questo punto il campo di ricezione di C_3 ed S_4 è intorno a 4° del campo visivo, cioè copre tutto l'intero stimolo.

Bypass route Nonostante la via principale costituisca la maggiore sorgente di input per l'IT[Tan96], quella bypass rimane una sorgente significativa di input. Ad esempio, alcuni studi hanno mostrato che in presenza di lesioni in V2, si verifica solamente un lieve indebolimento nei task di discriminazione fine e questo suggerisce che la strada bypass debba giocare un ruolo importante.

Nel modello, una strada di tipo bypass corrisponde ad una proiezione dai livelli C_1 a quelli S_{2b} e poi C_{2b}. Le unità S_{2b} combinano la risposta delle unità C_1 proprio come le unità S_2, anche se la dimensione del campo di ricezione è più larga (almeno 2 o 3 volte). Lo stesso vale per il numero di afferenti a S_{2b} che passa da 10 a 100.

L'effetto di saltare una fase da C_1 a S_{2b} non risulta solamente nei livelli C_{2b} che risultano essere più selettivi rispetto alle altre unità allo stesso livello lungo la gerarchia (C_3) ma soprattutto esibiscono un minor range di invarianza alla posizione e alla scala.

Per un sistema visivo che si basa sia sulla main route che su quella bypass

i vantaggi sono evidenti. Oltre all'ovvia robustezza alle lesioni, si suppone che la bypass route possa aiutare a fornire un più ricco vocabolario di unità sintonizzate sulla forma con differenti livelli di complessità ed invarianza. Inoltre, mentre il livello delle performance sui vari task di categorizzazione dei singoli livelli del modello (cioè C_2 vs. C_{2b} vs. C_3) sono molto simili, la loro combinazione fornisce un guadagno significativo in termini di performance.

2.1.3 Riconoscimento invariante in IT

Come più volte sottolineato, l'IT è ritenuto giocare un ruolo chiave nell'abilità dei primati nell'eseguire un riconoscimento invariante degli oggetti[Tan96]. Basandosi sullo studio delle lesioni, si è soliti suddividere l'IT in 2 sotto regioni: l'area infero-temporale posteriore (PIT) e quella anteriore (AIT) che sono coestensivi con TE e TEO (che insieme costituiscono l'IT)[LogShe96]. Cioè, la selettività dei neuroni in TEO e V4 è simile, anche se i campi ricettivi sono più larghi in TEO.

Quindi, in tutto il percorso dalla parte posteriore del TEO a quella più anteriore del TE la dimensione del campo ricettivo aumenta significativamente così da coprire gran parte del campo visivo (fino a 30-50° in TE). Questo incremento è accompagnato da un significativo aumento della complessità degli stimoli preferiti (da semplici stimoli a volti e/o mani). È perciò probabile che le unità S_3 debbano sovrapporsi tra TEO e TE (le unità S_3 rivecono l'input da molte cellule tipo V3 con una invarianza limitata). Le unità C_3 che coprono un range di invarianza di più o meno 2° in traslazione e 2 ottave in scala si trovano con maggiore probabilità nella parte anteriore del TE. Infatti, i neuroni che rispondono alle parti degli oggetti, ad esempio gli occhi [Per92], devono corrispondere a unità S_{2b} o C_{2b} mentre i neuroni che richiedono la presenza simultanea di parti multiple di una faccia [PerOra93] devono corrispondere a S_3, C_3 o S_4.

2.2 Apprendimento del dizionario

2.2.1 Le correlazioni dell'apprendimento

L'esperienza visiva, durante e prima dello sviluppo (nelle aree inferiori della corteccia), insieme ai fattori genetici determina la connettività e le proprietà delle cellule. Si assume che l'apprendimento giochi un ruolo chiave nel determinare le connessioni e il vettore dei pesi sinaptici delle unità del modello.

Più specificatamente, si suggerisce che le proprietà di sintonizzazione delle cellule semplici, nei vari livelli della gerarchia, corrispondano ad apprendere quali combinazioni di feature appaiono più frequentemente nelle immagini. Questo è equivalente ad apprendere un dizionario di pattern di immagini che appaiono con un'alta probabilità. Per cui, il cablaggio del livello S dipende dalla correlazione tra le feature presenti nello stesso istante all'interno dell'immagine.

La sintonizzazione delle cellule complesse, d'altro canto, deve riflettere la capacità proveniente dall'esperienza visiva di associare trasformazioni frequenti nel tempo (traslazioni, scale) di specifiche feature complesse codificate da cellule semplici. Il cablaggio dei livelli C riflette invece l'apprendimento delle correlazioni lungo il tempo.

Questo apprendimento nei livelli S e C è presente nel mondo visivo. Attualmente è ancora poco chiaro se questi due tipi di apprendimento richiedano differenti tipi di "regole" sinaptiche oppure no.

2.2.2 La regola dell'apprendimento

L'obiettivo della fase dell'apprendimento è determinare la selettività delle unità S, e più precisamente definire i tipi base in ognuna della macro colonne. Si suggerisce che queste macro colonne o mappe di feature costituiscano il dizionario base delle componenti morfologiche con le unità che sono sintoniz-

zate sulle caratteristiche delle immagini che occorrono con un'alta probabilità nella immagini naturali.

Nel modello, si assume che la fase di apprendimento non sia supervionata e debba verificarsi durante la fase dello sviluppo. È probabile che le nuove feature debbano essere apprese dopo questo apprendimento iniziale durante l'età adulta (certamente a livello dell'IT e anche in alcune aree inferiori). I risultati suggeriscono che è possibile eseguire un riconoscimento di oggetti invariante a partire da un insieme di unità sintonizzate sulla forma, apprese senza supervisione da un set generico di immagini naturali non collegate a nessun task di categorizzazione.

L'apprendimento nel modello è sequenziale, ad esempio i livelli sono addestrati uno dopo l'altro (tutte le immagini da un database sono presentate durante l'addestramento ad ogni singolo livello) a partire da quello più basso (S_2 e S_{2B}) e procedendo fino al top (S_3).

Durante questa fase dello sviluppo i pesi (w_1, ... , w_n), cioè gli stimoli preferiti delle unità S all'interno di ogni mini colonna che sono condivise tra tutte le macro colonne del livello, sono appresi sequenzialmente partendo da w_1 fino ad w_n. Alla k-esima immagine nella presentazione una macro colonna (che corrisponde ad una particolare porzione del campo vivisvo e scala) è selezionata a random e l'unità w_k di questa colonna è "imprinted". Ad esempio, l'unità salva nel suo vettore dei pesi sinaptici il pattern corrente dell'attività dei suoi input afferenti in risposta alla parte dell'immagine naturale che cade all'interno del campo ricettivo. Questo è fatto settando w_k ad essere uguale al pattern corrente dell'attività pre-sinaptica x. Come risultato, l'immagine sovrapposta x che cade all'interno del campo di ricezione di w_k diviene il suo stimolo preferito. Dopo questo processo di imprinting, l'unità è matura.

Durante la fase di apprendimento, si assume che l'immagine si muova cosicchè la selettività delle unità w_k sia generalizzata all'interno della stessa mini colonna in unità a scale differenti e tra macro colonne a unità a differenti locazioni nel campo visivo.

Apprendere tutti gli n tipi di unità S all'interno di un livello richiede di esporre il modello con n immagini. Il database delle immagini che è stato utilizzato contiene una larga varietà di immagini naturali. Il dizionario delle componenti morfologiche è appreso durante questa fase. Si vedrà in seguito che lo stesso dizionario potrà essere utilizzato per irrobustire e rendere invariante il riconoscimento di differenti categorie di oggetti.

Superata questa fase solo i circuiti task specific dall'IT al PFC richiedono un apprendimento per il riconoscimento di oggetti specifici e categorie di oggetti.

2.3 Costruzione dei circuiti task-specific

Si assume che una particolare routine sia settata da qualche parte oltre l'IT, possibilimente nel PFC, ma l'esatta locazione dipende dal task. In uno stato passivo (ovvero non ci sono task visivi in azione) ci deve essere un task di default (ad esempio: "cosa è questo?"). Qui si pensa ad una routine come ad una particolare unità di classificazione tipo PFC che combina le attività di poche centinaia di unità S_4 sintonizzate ad esempio su di un oggetto target. Mentre l'apprendimento nel modello da S_2 a S_4 è guidato dagli stimoli (non supervisionato), le unità di classificazione di tipo PFC sono addestrate in maniera supervisionata.

Unità S_4 sintonizzate sulla vista Quando un nuovo task è appreso, le unità S_4, che corrispondono alle cellule sintonizzate sulla vista dell'AIT, divengono selettive verso specifici esempi di training set. Questo in accordo con la specificità dei neuroni dell'IT alla vista di certi oggetti e condizioni luminose.

Nel modello proposto, durante l'addestramento dei circuiti, una piccola frazione, circa il 25% del set di oggetti dell'addestramento è salvato al livello delle unità S_4. Così come le unità nelle fasi preliminari si sintonizzano sulle

patch delle immagini naturali, così le unità S_4 si sintonizzano alla vista degli oggetti target salvando nei loro vettori dei pesi sinaptici il preciso pattern delle attività dei suoi afferenti durante la presentazione degli oggetti.

È importante precisare che usando la fase S_4 si migliorano complessivamente le performance del modello. Ragionevolmente buoni risultati possono essere ottenuti anche senza una fase S_4. Questo perché, performance invarianti nel riconoscimento possono essere ottenute anche per mezzo di un classificatore lineare che usa direttamente una parte del dizionario delle componenti morfologiche. In particolare, è stato utilizzato il classificatore SVM che attualmente compete con i migliori sistemi di computer vision quando si usano le feature del modello C_{2b}.

Unità di classificazione tipo PFC Il concetto di un classificatore lineare che prende i suoi input generalmente da poche unità sintonizzate sugli esempi è uno schema di apprendimento potente che è molto vicino alle RBF, che sono tra le migliori in termini di apprendimento nella generalizzazione. Simulazioni eseguite hanno mostrato la plausibilità di questo schema per il riconoscimento visivo e per la consistenza quantitativa dei dati. In particolare, Poggio suggerisce che l'ampia sintonizzazione delle unità nell'IT debba essere la chiave nel fornire buone proprietà di generalizzazione delle unità di classificazione oltre l'IT.

In maniera interessante, un recente studio di Hung del 2005 dimostra che un classificatore lineare può invece estrarre con una grande accuratezza e in tempo estremamente breve l'identità degli oggetti, le categorie e altre informazioni dall'attività di circa 100 neuorni all'interno dell'IT.

Nel modello, la risposta delle unità di classificazione che prendono in input un vettore dei pesi $c = (c_1, \ldots, c_n)$ è data da:

$$f(x) = \sum_i c_i K(x^i, x) \tag{2.1}$$

dove

$$K(x^i, x) = \exp\left(-\frac{1}{2\sigma^2}\sum_{j=1}^{n}(x_j^i - x_j)^2\right)$$

caratterizza l'attività dell'i-esima unità S_4, sintonizzata sull'esempio di addestramento x^i, in risposta all'immagine di input x ed è stata ottenuta rimpiazzando il vettore dei pesi w nell'equazione 1.2 con x^i ($w = x^i$). L'indice i indica l'immagine nel set di addestramento e j indica l'indice dell'unità pre-sinaptica. Un apprendimento supervisionato in questa fase comporta di aggiustare il vettore c così da minimizzare l'errore di classificazione E sul set delle immagini di addestramento, cioè:

$$E = \sum_{i=1}^{l} \|f(x^i) - y^i\|^2 + R(f) \tag{2.2}$$

dove y^i è l'etichetta (0/1) dell'esempio di addestramento x^i e $f(x^i)$ corrisponde alla risposta dell'unità di classificazione x^i. $R(f) = \lambda\|f\|$ è un termine di regolarizzazione che rafforza l'appuntimento di f e che potrebbe essere omesso per semplicità. Di conseguenza, la minimizzazione di E corrisponde a minimizzare l'errore del classificatore sul set di allenamento.

Nella corrente implementazione del modello, un'unità di classificazione di tipo PFC è addestrata per ogni task di categorizzazione. Ad esempio, affinché il modello sia abile a riconoscere tutti gli oggetti presenti nel database Caltech-101, sono state addestrate 101 differenti unità f^h di tipo PFC con differenti vettori dei pesi c^h, uno per ognuno degli oggetti *vs.* il resto. Per una nuova immagine, l'etichetta h dell'unità f^h con il massimo output tra tutte le unità è considerata la risposta finale del modello. Alternativamente, il modello può essere testato con task di presenza/assenza di animali confrontando l'output di un'unità $f^o \in [0, 1]$ di tipo PFC allenate sul task animale *vs.* non animale ad una soglia fissata $f^O \leq \theta$.

A questo punto, si potrebbe pensare che addestrare il modello per tutte

le possibili categorie nel mondo porti a un numero intrattabile di unità. Non è il caso. Infatti, come si è detto in precedenza, è possibile saltare la fase S_4 e mantenere un alto livello di performance. Allenare il modello per riconoscere un numero plausibile di oggetti diversi (non più di 30.000) vorrebbe dire aggiungere circa 3 milioni di unità S_4(assumento circa 100 mila unità S_4 per classe). Il numero di neuroni in AIT è circa 15 milioni per ogni emisfero. Ad un livello simile al PFC il numero delle unità di classificazione richiesto sarebbe veramente piccolo, circa 100 mila unità. È fortemente probabile che nella corteccia visiva le stesse unità evolverebbero in differenti task di categorizzazione.

La chiave nel modello è dunque l'uso di un dizionario generico delle componenti morfologiche comuni alla maggior parte degli oggetti che permette di prevenire l'esplosione nel numero delle unità necessarie al riconoscimento.

Capitolo 3

Risultati del modello

In questo capitolo, innanzitutto, viene mostrato come il modello riesca a duplicare le proprietà di sintonizzazione dei neuroni nelle varie aree del cervello (ad esempio V1, V4, ed IT). In particolare, in accordo con le informazioni presenti in V4 che riguardano le risposte dei neuroni alla combinazione di semplici stimoli (due barre) all'interno dei campi ricettivi delle unità S_2 e C_2, mostra una sintonizzazione verso i bordi consistente con V4.

Successivamente, viene mostrato come il modello, oltre a duplicare le proprietà dei neuroni in presenza di stimoli artificiali, riesca a gestire il riconoscimento di oggetti nel mondo reale, estendendo così la competitività con i migliori sistemi di computer vision.

Infine, viene descritto il confronto tra le performance del modello e quelle degli esseri umani alle prese con un task di riconoscimento rapido animale *vs.* non animale che probabilmente non prevede l'intervendo della Back-Projection. I risultati suggeriscono che il modello eguaglia le performance umane estremamente bene quando il ritardo tra lo stimolo e la maschera è di circa 50 ms. Questo suggerisce che la Back-Projection non giochi un ruolo chiave quando il ritardo è all'interno di questo range, e il modello quindi fornisce una descrizione soddisfacente del percorso di tipo feedforward.

3.1 Confronto con i neuroni

In questa sezione viene mostrato come i livelli del modello possano essere mappati con le aree corticali. In particolare viene fornito il parallelismo tra il modello e le informazioni provenienti dalle aree V1, V4 ed IT (TE).

3.1.1 V1 e il modello

I parametri delle unità S_1 e C_1 sono stati aggiustati manualmente cosicchè la dimensione dei campi ricettivi, la frequenza di sintonizzazione e l'ampiezza dell'orientamento coprano il range delle cellule semplici e complesse di V1 quando soggette ad uno stimolo standard.

3.1.1.1 Metodi

Orientamento La sintonizzazione dell'orientamento è valutata in questo modo: seguendo [DYH82], si è sovrapposta un'onda sinusoidale graticolare con frequenza ottimale sul campo ricettivo di ogni unità in 36 differenti direzioni (per un totale di 180° del campo visivo, con step di 5°). Per ogni unità e direzione, si è registrata la risposta massima (lungo la posizione) per stimare la curva di sintonizzazione dell'unità e calcolarne il bandwidth a metà ampiezza.

Frequenze spaziali La selettività della frequenza spaziale di ogni unità è stata valutata sovrapponendo la funzione onda sinusoidale graticolare alle varie frequenze (posizioni) sul campo ricettivo dell'unità. Per ogni frequenza la risposta massima dell'unità è stata utilizzata per adattare la curva di sintonizzazione. Il bandwidth di selettività della frequenza spaziale è stato calcolando dividendo la somma delle frequenze nei più alti passaggi della curva a metà ampiezza per i più bassi passaggi allo stesso livello. Prendendo il $\log_2$ di questo rapporto si ottiene:

$$bandwidth = \log_2 \frac{high\,cut}{low\,cut} \tag{3.1}$$

Risultati Le unità del modello sembrano catturare bene le proprietà di sintonizzazione di un insieme di cellule parafoveali (V1). In aggiunta, in accordo con la fisiologia, le unità del modello esibiscono il seguente trend:

1. Una correlazione positiva tra la dimensione del campo ricettivo e l'ampiezza delle frequenze;

2. Una correlazione negativa tra la dimensione del campo ricettivo e i picchi di frequenza (posizioni);

3. Un ampliamento del bandwidth da S_1 a C_1 (circa il 20%).

3.1.2 V4 e il modello

Qui vengono confrontate le proprietà di sintonizzazione delle unità C_2 con quelle delle cellule in V4. Come detto in precedenza la sintonizzazione di S_2 e C_2 avviene in maniera non supervisionata dalle immagini naturali. Durante questa fase le unità divengono sintonizzate alle caratteristiche delle immagini che appaiono con più alta probabilità.

Qui si analizza la loro selettività agli stimoli standard, mostrando anche che le unità C_2 esibiscono delle proprietà di sintonizzazione che sono in accordo con le informazioni provenienti da V4.

Innanzitutto, si mostra la risposta del modello in presenza di bordi e successivamente si analizza l'interazione di 2 stimoli semplici (barre) all'interno del campo ricettivo di S_2 in assenza di attenzione.

3.1.2.1 Conformità dei bordi

Per provare ad avere una descrizione quantitativa delle cellule V4, [PasCon01]hanno considerato uno spazio di forme 2D parametrizzato di moderata complessità

(Figura 4.1). L'insieme degli stimoli è stato generato combinando sistematicamente elementi bordo convessi e concavi per produrre semplici componenti chiuse con bordi condivisi. Con questo insieme di stimoli essi hanno quantificato la risposta delle cellule V4 (che rispondono agli stimoli complessi).

Pasupathy e Connor hanno trovato che lo spazio della conformità dei bordi meglio caratterizza l'insieme di 109 risposte di V4. Da questi risultati, essi hanno suggerito una rappresentazione basata sulle parti di forme complesse in V4, dove le parti sono pattern di bordi definiti dalla curvatura e dalla posizione relativa al resto dell'oggetto.

Per guardare alla plausibilità dell'architettura del modello e le regole di apprendimento associate si sono eseguite "registrazioni" simili sulle sue unità C_2.

Facendo un confronto tra un neurone in V4 e un'unità C_2 entrambe sembrano essere sintonizzate ad una curvatura concava sulla destra ed esibiscono simili pattern di risposte alle differenti forme. L'unità del modello è stata presa dal set delle 109 unità C_2. Le differenze tra il modello e il neurone sono evidenti perché qui non sono previste procedure di adattamento per apprendere i pesi delle unità del modello: semplicemente l'unità è stata selezionata dalla popolazione delle 109 unità; la sua sintonizzazione è stata appresa dalle immagini naturali. L'organizzazione del campo ricettivo delle unità C_2 è dettata dalla teoria vista sinora: La selettività alla forme delle unità è ereditata dai suoi afferenti S_2 che sono tutti sintonizzati sullo stesso stimolo preferito ma su posizioni e scale differenti (garantendo la tolleranza). Ogni afferente S_2 stesso riceve i suoi input dalle cellule complesse.

Applicando la tecnica su tutte le 109 unità si ha che ognuna di esse apprende il pattern di input da un porzione dell'immagine naturale. La bontà dell'adattamento è stata valutata calcolando il coefficiente di correlazione tra le risposte neurali e le risposte predette dalle funzioni di sintonizzazione. La popolazione delle unità del modello risultante esibisce una sintonizzazione che è spiegata meglio dallo spazio di conformità dei bordi che è una caratteristica

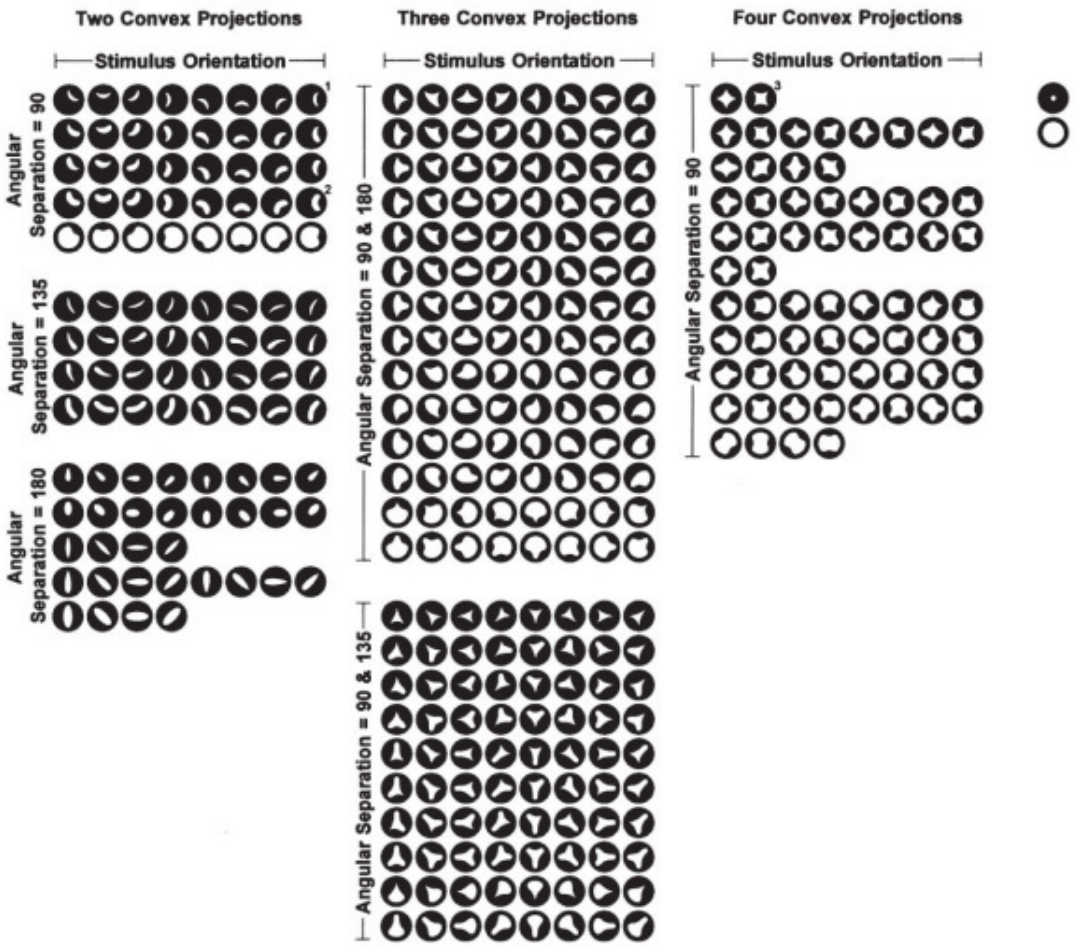


Figura 3.1: L'insieme degli stimoli

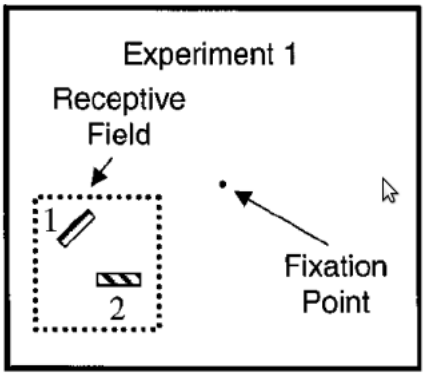


Figura 3.2: L'esperimento con 2 stimoli

della popolazione di neuroni in V4 come riportato da Pasupathy e Connor.

Per i metodi utilizzati per la valutazione della selettività delle unità ai bordi fare riferimento a[PasCon01].

3.1.2.2 Interazione di due stimoli

Si guarda alla risposta delle unità C_2 in presenza di stimoli con barra singola o doppia come in[RCD99](Figura 4.2.). In [RCD99], Reynolds ha registrato la risposta dei singoli neuroni V4 in 2 diverse condizioni sperimentali, ad esempio quando le scimmie fanno attenzione o all'interno o all'esterno del campo ricettivo dei neuroni. Il modello feedforward di Serre non tiene conto dei meccanismi legati all'attenzione. Perciò di seguito si guarda solamente alla condizione per la quale la scimmia è lontana dal campo ricettivo così che l'attenzione non ha effetto sulla risposta dei neuroni. In questa condizione, Reynolds ha trovato che l'aggiunta di un secondo stimolo presentato all'interno del campo ricettivo di un neurone V4 causa che la risposta del neurone sposta in avanti quella del secondo stimolo.

Metodi Come in [RCD99] si sono utilizzati 16 stimoli. Siccome il modello non tiene conto delle cellule sensitive, invece di utilizzare gli stimoli composti da tutte le combinazione di 4 barre orientate (0°, 45°, 90°, 135°) presentati in 4 colori (red, blue, green, yellow) si sono presentate 16 barre orientate in scala di grigi. Così sono state eseguite un numero eguale di misure sulle unità del modello e sui neuroni V4.

Allo stesso modo gli stimoli possono apparire in una di 2 possibili posizioni all'interno del campo ricettivo delle unità: per definizione, lo stimolo che appare nella posizione 1 è lo stimolo di riferimento (scelto tra i 16 possibili stimoli). Come in Reynolds può capitare che lo stimolo di riferimento possa essere scelto come stimolo preferito dell'unità, oppure quello debole o quello intermedio. Lo stimolo che appare nella posizione 2, lo stimolo "sonda", è stato selezionato a random dallo stesso set di 16 stimoli come per Reynolds.

Mentre l'identità dello stimolo di prova varia ad ogni run, l'identità di quello di riferimento è fissa per tutta la sessione di registrazione dell'esperimento. Su una data prova vengono testate 3 condizioni:

1. lo stimolo di riferimento appare in posizione 1 da solo;
2. lo stimolo "sonda" appare in posizione 2 da solo;
3. entrambi gli stimoli compaiono contemporaneamente rispettivamente in posizione 1 e 2.

La risposta di ogni unità viene normalizzata dividendo tutte le risposte per quella massima tra tutte le condizioni. Come Raynolds sono stati calcolati diversi indici:

- un indice di selettività SE_i:

$$SE_i = PROBE_i - REF_i$$

  dove $PROBE_i$ è la risposta normalizzata dell'unità allo stimolo di riferimento e REF_i la risposta normalizzata dell'unità all'i-esima prova.

Questo è stato fatto per tutte le prove. L'indice di selettività ha un range compreso tra -1 e +1, con valori negativi che stanno ad indicare che lo stimolo di riferimento ha suscitato la risposta più forte, un valore di 0 indica la stessa risposta tra lo stimolo di riferimento e quello di test, e un valore positivo indica che è stato lo stimolo di test a suscitare la risposta più forte.

- un indice di interazione sensoriale SI_i:

$$SI_i = PAIR_i - REF_i$$

dove $PAIR_i$ è la risposta normalizzata alla coppia composta dallo stimolo di riferimento e l'i-esimo stimolo di test. È nello stesso range di SE. Un valore negativo indica che la risposta alla coppia degli stimoli è minore rispetto alla risposta allo stimolo di riferimento. Un valore pari a 0 indica che l'aggiunta dello stimolo di test non ha avuto effetti sulla risposta dei neuroni. Un valore positivo indica che l'aggiunta dello stimolo di test aumenta la risposta dei neuroni.

3.1.3 TE e il Modello

Una delle operazioni chiave del modello originale era la sua abilità a duplicare le proprietà di sintonizzazione ed invarianza delle unità sintonizzate sulla vista di TE/AIT. Per assicurare che il modello presente rimanga consistente con ciò, si sono testate le unità S_4 del modello utilizzando come stimolo una clip. Come in [RiePog99] si sono usati 80 stimoli (20 target, 60 distrattori) dei 200 usati da Logothetis.

Per valutare il grado di invarianza alle trasformazioni dello stimolo, si è usato un paradigma simile a quello usato da Poggio e Riesenhuber nel quale uno stimolo target trasformato (scalato o ruotato in profondità) è considerato propriamente riconosciuto in certe condizioni di presentazione se l'unità S_4 sintonizzata sullo stimolo target originale (dimensione e vista di

default) risponde con maggiore intensità alla sua presentazione piuttosto che alla presentazione di uno stimolo distrattore.

Per misurare le proprità di invarianza delle unità S_4 alla traslazione, si è addestrata un'unità per ognuna dei 20 stimoli target, presentati nel centro dell'immagine. Per la durata del test, si è confrontata la risposta di ciascuna unità ai suoi stimoli preferiti in 8 possibili dimensioni. Per esaminare l'invarianza alla posizione, si è addestrata un'unità per ognuno dei 20 stimoli target ad una vista di riferimento (denotata 0°, posizionata al centro dell'immagine di input). Per la durata del test si è testata la risposta di ogni unità S_4 con i suoi stimoli preferiti a differenti orientazioni ($\pm$ 50° da quello di riferimento con step di 4°).

Risultati Si è confermato che il range di invarianza delle unità S_4 è all'interno del range delle unità sintonizzate sulla vista in AIT [RiePog99]. Le unità S_4 mostrano un'invarianza media alla posizione di $\pm$ 2° e un'invarianza alla scala di $\pm$ 1 ottava. I dettagli dell'esperimento possono essere trovati in [SerRie04].

3.2 Performance con le immagini naturali

Affinchè una teoria del riconoscimento degli oggetti nella corteccia sia di successo, dovrebbe essere abile ad eseguire un riconoscimento invariante e robusto nel mondo reale.

Le immagini che vengono utilizzate non sono segmentate e sia la fase di apprendimento che quella di riconoscimento hanno a che fare con il caos (ovvero gli oggetti target sono presenti insieme ad altri oggetti). Si mostra che non solo il modello può duplicare le proprietà di sintonizzazione dei neuroni in varie aree del cervello quando è sottoposto a stimoli artificiali, ma è in grado di gestire il riconoscimento degli oggetti del mondo reale.

3.2.1 Il database Caltech-101

È stato utilizzato il database Caltech-101 che contiene immagini di oggetti organizzati in 101 differenti categorie dove ciascuna contiene tra le 40 e le 800 immagini con una media di 50. La dimensione di ogni immaginne è di circa 300x200 pixel ma per velocizzare il proceso di apprendimento le immagini sono state scalate di circa la metà (140 pixel di altezza). Il database contiene immagini di molte categorie differenti di oggetti con una grossa variazione nelle forme, illuminazione, dimensione, ecceterea. Alcuni degli oggetti sono altamente deformabili, ad esempio gli animali appaiono i qualsiasi posa.

Il set delle immagini usate per guidare e testare il modello non è segmentato (gli oggetti sono embeddati nel clutter) e sono convertite in scala di grigio.

La parte centrale del modello, il dizionario corrispondente alle unità da V4 a TEO usato in questo esperimento è stato ottenuto con la procedura descritta in precedenza, ovvero tramite apprendimento non supervisionato. Questa fase setta gli stimoli preferiti delle unità S nei diversi livelli del modello. Come risultato, le unità divengono sintonizzate alle feature delle immagini che occorrono con una più alta probabilità nelle immagini naturali. Durante questa fase di apprendimento non supervisionato, il modello è esposto a poche centinaia di immagini naturali casuali, non correlate a nessun task di categorizzazione. Il dizionario risultante delle feature è generico nel senso che può supportare il riconoscimento di una larga varietà di categorie di oggetti. Come discusso in precedenza, questo dizionario è ridondante e contiene feature di varia complessità.

Per addestrare il modello ad eseguire differenti task di categorizzazione si sono addestrati i circuiti task-specific (in S_4 o PFC). Questo è stato fatto eseguendo una divisione random, tra le immagini usate per l'addestramento e quelle usate per testare il modello, su ogni insieme di immagini. Ovvero, per ogni categoria si è selezionato un numero variabile di immagini N_{tr} per l'addestramento e fino a $N_{te} = 50$ per il testing (prese dalle immagini rimanenti).

Questa procedura chiamata "*leave out*" ([DLG96]) fornisce una buona stima degli errori di classificazione.

Come si è visto prima i circuiti task-specific sono allenati in maniera supervisionata in 2 differenti fasi, per ogni categoria di oggetti:

1. Esempi tipici (circa il 25% del set di addestramento) presi dall'insieme degli oggetti target sono salvati nelle unità S_4, che così forniscono una rappresentazione olistica e specifica della vista degli oggetti familiari a livello di AIT.

2. Nel livello PFC un'unità di classificazione è addestrata (ad esempio i suoi pesi sinaptici sono aggiustati manualmente) così da minimizzare l'errore di classificazione (Eq. 2.2).

Durante il periodo dell'addestramento, si calcola l'errore dell'unità di classificazione sul set delle immagini di test (non usate durante l'allenamento). Per ogni categoria di immagini si genera una curva ROC[1] e si stima l'area sotto la curva come misura delle performance del modello [FFP04]. La procedura è reiterata 10 volte per ogni categoria e al termine si prende la media delle performance.

In [SWP05] si mostra che la dimensione del set di addestramento può essere ulteriorimente ridotta e che ragionevolemente buone performance possono essere ottenute utilizzando un numero ridotto di esempi (3-6).

Per una spiegazione più completa fare riferimento a [SWP05, Ser06].

3.2.2 Approssimazione delle chiavi di computazione

Qui si testa la robustezza del modello alle "semplificazioni.

Per prima cosa si testa quanto è critico l'uso di risposte "analogico-continue" delle singole unità.

[1] http://it.wikipedia.org/wiki/Receiver_operating_characteristic

Mentre si pensa che nelle aree di basso livello siano necessari molti livelli di quantizzazione, come suggerito in precedenza, il numero n di unità in un modello computazionale deve decrescere lungo la gerarchia con unità nei livelli alti che si comportano essenzialmente come degli switch (ON/OFF). Perciò qui si testa la robustezza del modello binarizzando la risposta delle unità in un livello intermedio del modello, il livello C_{2b} (corrispondente a TEO) che dà input alle unità S_4 (unità sintonizzate sulla vista in TE).

Inoltre, si testa anche la robustezza del modello a diverse approssimazioni di sintonizzazione a livello S_4.

Tutti gli esperimenti sono stati eseguiti su un sotto insieme del database Caltech-101.

Prima si seleziona la categoria delle immagini che contiene almeno 150 esempi così da poter eseguire dei run random con 100 immagini di addestramento e 50 di test selezionate a caso per ogni run. Quest porta ad avere 5 categorie (facce, animali, motocicli, aerei e orologi) ed un insieme di distrattori dalla categoria degli sfondi.

Per testare le dipendenze del modello sull'uso dei valori analogici *vs.* binari si confrontano le performance del modello standard (valori analogici) con un'implementazione che si basa su unità binarie di risposta in uno dei livelli intermedi (C_{2b}) che forniscono gli input alle unità S_4. Prima si calcola la soglia di risposta θ per ognula delle unità C_{2b} così che l'unità corrispondente sia attiva su P% dell'intero set di addestramento e inattiva sul restante 100-P%. Si è sperimentato con differenti valori di P, ad esempio P=10%, 30%, 60%. Le performance del modello sono valutate con due differenti paradigmi:

1. un paradigma "target presente/assente" (chance livello 50%) per ognuna delle 5 classi prese separatamente, ad esempio ogni task di riconoscimento è stato valutato come un problema di classificazione binaria indipendente. I distrattori sono campionati a random dallo stesso set separato dei distrattori (la categoria dei background).

2. un paradigma di scelta forzata N-alternative (dove N=5 è il numero totale di classi con un livello di chance del 20%). Ad esempio, ogni immagine presentata è stata classificata dal modello come appartenenete ad una delle 5 possibili categorie.

In Figura 3.3 e 3.4 sono mostrati i risultati della simulazione per P=30%. Quando il numero di afferenti è largo più di 100, la perdita di performance indotta dalla binarizzazione delle unità diviene trascurabile. Notare che si sono trovati i risultati qualitativamente simili per P=10% e P=60% e si è osservata una larga caduta delle performance per valori più alti di P.

Questi risultati mostrano che il modello non si basa criticamente sulla computazione esatta ma può basarsi sulle approssimazioni. Questo suggerische che le performance ottenute con operazioni "idealizzate" possono generare operazioni di approssimazioni eseguite in circuiti biologicamente e fisiologicamente plausibili. I risultati delle simulazioni danno supporto all'ipotesi iniziale che la dimensione dei modelli computazionali deve decrescere lungo una gerarchia con unità nei livelli superiori che si comportano essenzialmente come degli switch.

Concludendo, si è visto che:

- L'implementazione del modello è in grado di gestire un riconoscimento invariante di molte categorie differenti di oggetti con uno stesso dizionario di base

- Il modello esegue decisamente bene semplici task di riconoscimento così come scelte forzate.

- Il modello non sembra dipendere dall'esattezza delle chiavi di computazione almeno nei livelli alti e varie approssimazioni possono supportare un riconoscimento robusto ed invariante. In particolare, perché lungo la gerarchia le unità ricevano più input l'operazione di tuning non deve necessitare di essere esatta.

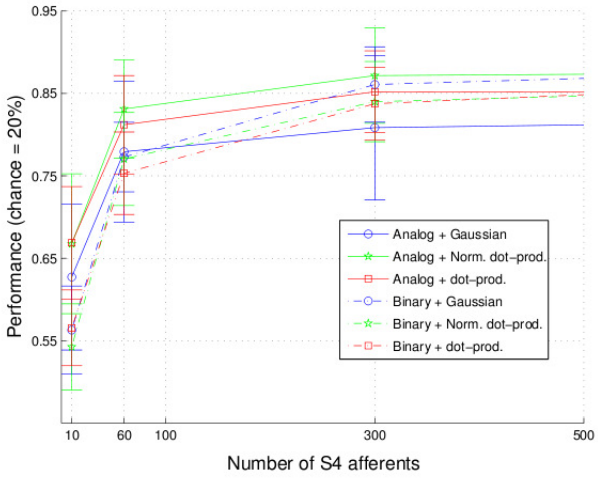


Figura 3.3: La robustezza del modello a livello S_4(N-alternative)

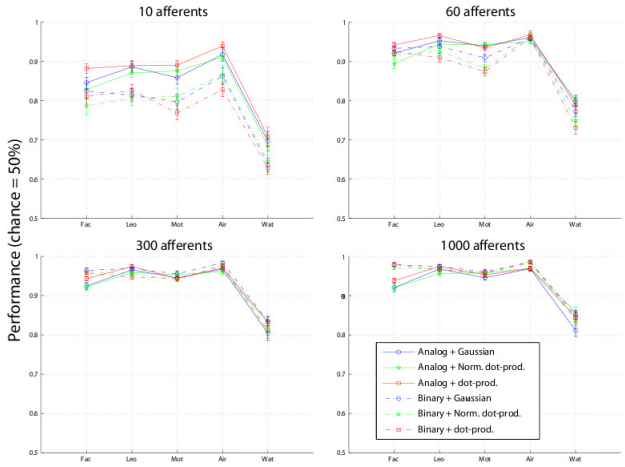


Figura 3.4: La robustezza del modello a livello S_4(presenza/assenza)

- Si è trovato che nelle fasi superiori del modello, una classificazione delle risposte non è necessaria e che la risposta delle unità binarie (on/off) deve essere sufficiente a supportare un riconoscimento robusto ed invariante.

- La capacità del modello di gestire il riconoscimento di una varietà di oggetti del mondo reale fornisce un'altra dimostrazione della plausibilità di questa classe di modelli.

- Infine si è visto che l'insieme delle feature morfologiche che sono passate al classificatore finale è davvero ridondante.

3.3 Confronto con le performance umane

In questo paragrafo viene mostrato che una specifica implementazione [RiePog99] di una classe di teorie feedforward per il riconoscimento degli oggetti - che estende la gerarchia delle cellule semplici → complesse di Hubel e Wiesel che tiene conto dei limiti anatomici e fisiologici - può riprodurre le performance umane su task di riconoscimento animale *vs.* non animale.

Si è visto che il modello è in grado di riconoscere bene immagini complesse e quando testato sulle immagini naturali, esso riesce a competere con lo stato dell'arte della computer vision su moltissimi task di categorizzazione. È perciò naturale chiedersi se il modello sia in grado di duplicare le performance umane nei task di riconoscimento complesso.

Finora si è visto che il modello presentato estende quello originale (HMAX) introducendo un apprendimento non supervisionato della sintonizzazione delle unità corrispondenti alle cellule V4 e a PIT su di un set di immagini naturali non correlate ad alcun task. Si è visto anche che le unità divengono sintonizzate alle attività neurali indotte dalle immagini naturali creando quello che è un dizionario di feature universale e ridondante che è invariante rispetto alla traslazione e alla scala e può supportare il riconoscimento di molti

differenti categorie di oggetti. Quando testato sulle immagini naturali del mondo reale, il modello compete con i migliori sistemi di computer vision su moltissimi task di categorizzazione. Questo è sorprendente, dati i limiti biologici specifici che la teoria soddisfa.

Sulla base di tutto questo, perciò è naturale chiedersi se un tale modello di tipo feedforward possa essere in grado di duplicare le performance degli esseri umani con task di riconoscimento natuale e complesso.

Normalmente, la visione quotidiana include effetti top-down che devono essere mediati da una vasta Back-Projection corticale[Bul01]. La Back-Projection deve controllare i circuiti e le routine, ad esempio in PFC, che estraggono in maniera guidata dai task l'informazione dalle aree visive inferiori (ad esempio l'oggetto nella scena è un animale? quanto è grande?)[HKPD05]. Essi in aggiunta possono influenzare sia le aree inferiori che l'IT durante o prima del task. La richiesta chiave dei modelli feedforward, così come nel modello presentato qui, è che i primi 150 ms della percezione visiva non coinvolgano significative dinamiche di tipo feedback.

Si è visto un paradigma sperimentale nel quale il riconoscimento è veloce e la Back-Projection corticale è inattiva. Il paradigma che si usa per confrontare le performance umane con quelle del modello è la categorizzazione degli oggetti ultra rapida. Il task è un riconoscimento classico animale *vs.* non animale. Gli animali nelle scene naturali costituiscono una classe di stimoli impegnativa dovuta a una grande varietà di forme, dimensioni, posizioni, eccetera.

Si usa un protocollo di "mascheramento posteriore" (backward-masking) che consiste nell'aggiungere un rumore all'immagine della durata di 80 ms. Studi precedenti hanno suggerito che un backward mask può interrompere un processo visivo [KXFP01] e bloccare la Back-Projecetion [BacMac05]. Per variare la difficoltà del task e prevenire che gli osservatori umani e il modello si basino sui segnali di basso livello, si usano 4 insiemi di categorie di immagini bilanciate (150 animali e 150 distrattori), ognuno corrispondente

ad una particolare distanza visiva dalla camera.

Prima che il modello possa essere testato sui task di categorizzazione scelti, esso deve essere addestrato. L'unico task specifico per l'addestramento richiesto coinvolge i circuiti nei livelli superiori del modello, ad esempio, il classificatore lineare possibilmente in PFC che "guarda" all'attività delle diverse centinaia di unità S_4[HKPD05]. Tale classificatore è allenato su un task specifico (animale *vs.* non animale) in modo supervisionato. Questa fase richiede un numero relativamente piccolo di esempi (100). Il classificatore è stato addestrato usando n split random sull'intero database delle immagini. In un dato run, metà delle immagini sono selezionate a random per l' addestramento e l'altra metà è usata per testare il modello.

Il modello è stato confrontato con gli osservatori umani in 3 differenti esperimenti.

Esperimento 1 In questo esperimento, si sono replicati i precedenti risultati psicofisici [BacMac05] per testare l'influenza della maschera sul processo visivo in 4 diverse condizioni sperimentale, ad esempio, quando la maschera segue l'immagine target:

- A) senza alcun ritardo (immediate-task);
- B) con un breve intervallo di 30 ms all'interno dello stimolo (ISI);
- C) con un ISI di 60 ms;
- D) mai (no-mask condition).

Per tutte e 4 le condizioni, la presentazione del target è stata fissata a 20 ms, e le performance nelle condizioni A e D stabiliscono il limite inferiore e superiore nelle performance umane (in assenza di movimento degli occhi). Un confronto tra le performance degli osservatori umani (n = 21) e il modello è mostrato in Figura 3.5.

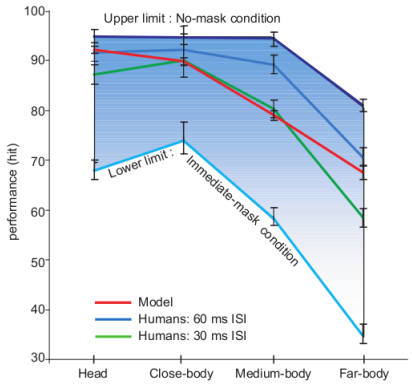


Figura 3.5: Confronto tra gli osservatori umani e il modello con differenti condizioni

Si è trovato che l'accuratezza (hits) degli osservatori umani è ottima all'interno del range ottenuto in precedenza (condizioni A e D). Le performance dei soggetti hanno raggiunto un livello massimo nella condizione C (ad eccezione di quando l'animale era camuffato nella scena). Come ci si aspettava il ritardo tra lo stimolo e l'inizio della maschera modula il livello delle performance degli osservatori migliorando il riconoscimento di base nella condizione A sino al massimo nella condizione D.

L'area ombreggiata corrispondente alla condizione C definisce il range delle performance di riconoscimento ammissibili che dovrebbero corrispondere al sistema visivo umano nella sua modalità feedforward. Le performance del modello sono ottime all'interno di quest'area e duplicano le prestazioni umane con un ISI compreso tra i 30 ms e 60 ms. Questo implica che con un ISI all'interno di questo range, la Back-Projection non gioca un ruolo significativo e che il modello può fornire una descrizione soddisfacende della percorso feedforward.

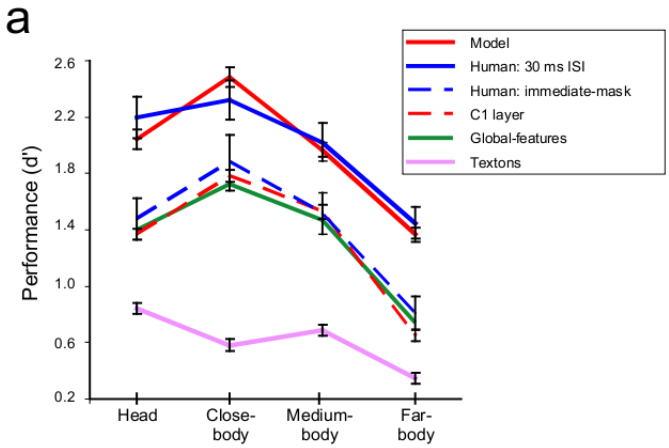


Figura 3.6: Confronto con l'accuratezza umana.

Esperimento 2 Nell'esperimento 2 si è raffinato il confronto tra il modello e gli osseravatori umani testando i soggetti (n= 24) sulla singola condizione C.

Per tenere conto sia delle risposte dello stimolo target che di quello distrattore si introduce una misura della sensibilità (dalla teoria di riconoscimento dei segnali [MacCre91]) *d'*, che è la differenza standardizzata tra le medie degli hit e la distribuzioni dei falsi allarmi di ogni osservatore (la frequenza degli errori e degli hit dovrebbe essere simile).

Come mostrato in Figura 3.5 il comportamento umano è simile a quello del modello: per tutte e 4 le categorie di immagini, il livello delle performance non mostra differenze significative. Le performance del modello, quindi, risultano essere rimarcabili.

Esperimento 3 Infine, nell'esperimento 3, si è misurato l'effetto della rotazione delle immagini (90°, 180°).

Recenti studi comportamentali [GKT05], suggeriscono che i task di categorizzazione animale possono essere fatti a diverse orientazioni dell'immagine,

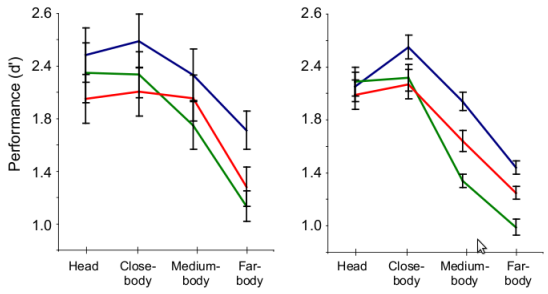


Figura 3.7: Confronto sulla rotazione delle immagini

cioè fornendo un interessante test per il modello. Come mostrato in Figura 3.5, il livello delle performance sia degli osservatori umani che del modello è piuttosto robusto alla rotazione dell'immagine (ad eccezione della condizione di oggetto lontano per la quale la scena del background è probabile che influenzi le performance). Le misure di accuratezza ottenute per gli osservatori umani e per il modello sono compatibili con i risultati precedenti [GKT05].

La robustezza del modello è particolarmente rimarcabile per il fatto che esso non è stato ri-addestrato prima di essere testato sulle immagini ruotate (è improbabile che i soggetti umani abbiano un'esperienza estesa con immagini ruotate di animali).

Il fatto che un modello di tipo feedforward - fedele all'anatomia e fisiologia della corteccia visiva - raggiunga un livello di accuratezza comparabile con gli esseri umani su un task di riconoscimento difficile pone una questione intrigante: quali sono i task di puro riconoscimento visivo che realmente hanno bisogno di circuiti feedback e di coinvolgere la Back-Projection?

Capitolo 4

L'implementazione

Il pacchetto contenente l'implementazione del modello è denominato PNAS[1]. Questo pacchetto va ad inserirsi all'interno di FHLib, ovvero di una libreria di feature gerarchiche multiscala creata da Jim Mutch[2]. Si tratta di un framework generale che permette la definizione di gerarchie feedforward aventi un numero arbitrario di livelli.

PNAS utilizza una Support Vector Machine (SVM) come classificatore top-level[3]. Le SVM sono un insieme di metodi per un apprendimento supervisionato che analizza dati e riconosce pattern, usati per la classificazione e l'analisi delle regressioni. Una SVM standard prende un insieme di dati in input, e predice, per ognuno degli input, a quale delle due possibili classi l'input è membro. Questo fa delle SVM un classificatore binario lineare non probabilistico.

Nel caso specifico si utilizza una particolare implementazione: SVMLight. SVMLight[4] è stato sviluppato da Thorsten Joachims.

In ogni caso, è sempre possibile utilizzare altri classificatori, come ad

[1] `http://cbcl.mit.edu/software-datasets/serre/SerreOlivaPoggioPNAS07/index.htm`

[2] `http://www.mit.edu/~jmutch/fhlib/`

[3] http://en.wikipedia.org/wiki/Support_vector_machine

[4] `http://www.cs.cornell.edu/People/tj/svm_light/`

esempio quelli lineari basati sui minimi quadrati. Tale scelta è fatta andando a modificare la variabile *doSVM*.

La funzione principale di PNAS è *replicatePNAS_ CV()* che richiede 3 parametri: *splitStarNum*, *splitEndNum* e *useSavedModel*. I primi 2 specificano il numero di split che devono essere eseguiti durante l'esecuzione del modello e come devono essere numerati. Ad esempio, *replicatePNAS_ CV(1,10)* esegue 10 split numerati da 1 a 10.

```
function replicatePNAS_CV(splitStartNum, splitEndNum, useSavedModel)
```

Ciascuno split sceglie casualmente un sottoinsieme di immagini su cui operare.

replicatePNAS_ CV() utilizza una validazione incrociata[5] per ottenere i migliori settaggi per i parametri dei classificatori, indipendentemente da quello che verrà utilizzato, in quanto entrambi danno dei risultati similari. Questo è fatto per ogni split utilizzando gli insiemi delle immagini di test, di training e di validazione.

I passi principali nell'esecuzione del modello sono:

1. si esegue il modello su tutto il set delle immagini (tipicamente 1200) producendo una matrice di dimensioni 1200x6000. Come si può notare ad ogni immagine è associata una riga della matrice e per ciascuna di essa si hanno 6000 colonne corrispondenti al numero di feature che vengono estratte dai livelli definiti in *levelsToExtractResp*, che sono 4.

2. per ogni split, il set delle 1200 immagini viene suddiviso in 3 sottoinsiemi: uno di test (contenente $\sim 1/2$ del totale), uno di training ($\sim 1/4$) e uno di validazione ($\sim 1/4$):

[5] La validazione incrociata è un metodo statistico per validare un modello predittivo. Preso un campione di dati, esso viene suddiviso in sottoinsiemi alcuni dei quali vengono usati per la costruzione del modello (insiemi di allenamento, training sets) e gli altri da confronare con le predizioni del modello (insiemi di validazione, validation sets). Mediando la qualità delle predizioni tra i vari insiemi di validazione dà una misura dell'accuratezza delle predizioni.

(a) Per ogni valore di ogni singolo parametro, si addestra il classificatore sull'insieme di training e lo si testa su quello di validazione in accordo con la validazione incrociata.

(b) Si prende il valore che dà il migliore risultato. Usando questo valore, si addestra il classificatore sull'insieme costituito da quello di training e di validazione e lo si testa su quello di test. I risultati vengono infine salvati.

(c) si ripetono i passi (a) e (b) per tutti i parametri.

3. Si calcolano i risultati globali in base al classificatore che si è scelto di utilizzare.

Ora si andrà a vedere nel dettaglio il pacchetto PNAS. Le componenti principali di PNAS sono *replicatePNAS_CV()* e *featuresNatural_newGRBF()*. Mentre si è detto che *replicatePNAS_CV()* rappresenta il programma principale che simula il funzionamento del modello, *featuresNatural_newGRBF()* è la funzione che genera a partire da un set di immagini generico il dizionario delle feature, che verrà poi utilizzato dal programma per la classificazione delle immagini.

4.1 replicatePNAS_CV

```
function replicatePNAS_CV(splitStartNum, splitEndNum, useSavedModel)
```

Innanzitutto, si consideri il parametro *useSavedModel*, che è opzionale. Il suo utilizzo è legato al fatto che l'esecuzione del modello sull'intero set di immagini richiede molto tempo. Si pensi che su di un computer monoprocessore il tempo di elaborazione di un set di 100 immagini è calcolato mediamente in 2 ore. Per cui può essere utile un'ottimizzazione che consiste nel lanciare l'esecuzione specificando il paramentro *useSavedModel*, i cui valori possono essere 0 o 1. Se è uguale a 0, i risultati sono salvati

in "replicatePNAS_CV\replicatePNAS_model.mat", ovvero in un file con estensione MATLAB che rappresenta il file contenente le feature estratte. Invece se si vuole utilizzare un file di feature generato in un run precedente *useSavedModel* deve essere uguale ad 1. Il valore di default è 0.

Inizializzazione dei parametri

In entrambi i casi di utilizzo di *useSavedModel* i parametri che vengono utilizzati sono gli stessi. Si parte con l'inizializzazione delle variabili utili all'esecuzione del modello.

```
splitNums = splitStartNum : splitEndNum;
levelsToExtractResp = {'c1' 'c2' 'c3' 'c2b'};
numFeaturesToUse = [1500 1500 1500 1500];
```

dove *splitNums* indica il numero di split da eseguire durante il modello e *levelToExtractResp* rappresenta i livelli nei quali estrarre le informazioni per la costruzione della risposta. Come si può vedere contiene tutti i livelli complessi. Infatti, come si è visto nei capitoli precedenti, è all'interno di questi livelli, per mezzo dell'operazione di MASSIMO, che vengono estratte le informazioni (le feature) sintonizzate verso un determinato stimolo. Si ricordi che la sintonizzazione avviene in ogni livello semplice S che precede un livello complesso C.

numFeaturesToUse sta ad indicare il numero di informazioni, cioè caratteristiche, da estrarre da ciascun livello. Si può notare che la somma delle feature è 6000. Questo valore è lo stesso che si è visto nell'introduzione a questo capitolo, ovvero ciascuna immagine al termine dell'esecuzione ha associata una matrice contenente 6000 feature. Come si vedrà in seguito, questa matrice di feature è strettamente correlata al numero di feature contenute nel dizionario. Ad esempio, nel caso della classificazione di immagini di mammografie il numero di feature da estrarre per ogni livello passa da 1500 a 62.

Segue l'inizializzazione dei parametri relativi al classificatore SVM o lineare con kernel Radial Basis Function (RBF). Di default sono utilizzati entrambi, ma è sempre possibile decidere quali utilizzare giocando sulle variabili *doRBF* e *doSVM*. Se le 2 variabili valgono 1 i rispettivi classificatori vengono eseguiti all'interno degli split, altrimenti no.

```
% Parametri per regularized least-squares con RBF kernel
doRBF = 1;
lambdas = [1e-3 5e-3 1e-4 1e-5 1e-6 1e-7];
sigmas  = [5 1 0.5 0.1 0.05];

% Parametri per SVM
doSVM = 1;
doTrgSVM = 1;
doTstSVM = 1;
doWriteTrg = 1;
doWriteTst = 1;
```

Qui vengono settati i valori che le due funzioni di approssimazione utilizzano nella fase di classificazione. Questi valori vengono definiti manualmente in modo tale da simulare il comportamento della corteccia visiva.

Nel caso di una classificazione che utilizza una funzione di approssimazione lineare come quella dei quadrati minimi (regolarizzata) RLS con kernel RBF sono necessari 2 parametri: lambda(λ) e sigma (σ), dove λ rappresenta il parametro di classificazione o regolarizzazione mentre σ rappresenta il "bandwidth" ovvero l'ampiezza della funzione gaussiana associata.

Nel caso del classificatore SVM invece basta utilizzare un solo parametro: *paramCs*, che rappresenta il vettore dei costi C i quali controllano il tradeoff tra il massimizzare il margine della risposta durante l'addestramento e minmizzare l'errore. Piccoli valori di C tendono a enfatizzare il margine ignorando i valori errati nell'insieme di addestramento, mentre valori grandi di C possono tendere ad adattarsi su di tale insieme.

Quindi, C crea una sorta di "margine soft" che permette una "misclassificazione". Incrementando il valore di C aumenta il peso dei punti di misclassificazione e questo forza la creazione di un modello più accurato.

All'inizializzazione dei parametri dei 2 tipi di classificatori seguono le configurazioni del repository delle immagini, degli eseguibili del classificatore SVM e della root directory del modello:

```
SAVE_REL_PATH = fullfile('.', 'replicatePNAS_CV');
SVM_REL_PATH = '../svm_light';
IMAGE_DIR = fullfile(BASE_PATH, 'images');
```

Inizializzazione del modello

L'avvio dell'esecuzione del modello è preceduta dall'inizializzazione delle strutture che conterranno le informazioni estratte dall'insieme delle immagini.

Innanzitutto, si verifica se la funzione è stata chiamata con il parametro *useSavedModel,* e in tal caso con quale valore. Questo è importante perché nel caso in cui sia stato chiamato con 0 allora il file di log relativo al run corrente andrà a sostituire quello già esistente; invece se chiamato con valore 1 si andrà ad aggiungere in fondo allo stesso file. Come detto in precedenza l'utilizzo del parametro *useSavedModel* è utile nel caso in cui si scelga di ottimizzare i tempi dell'elaborazione utilizzando gli split di un run precedente. Ovviamente, questo si può nel caso in cui si stia utilizzando lo stesso set di immagini.

```
if (useSavedModel == 0)
  %overwrite
        logFileHandle = fopen(fullfile(SAVE_REL_PATH, logFile), 'wt');
else
  %append
        logFileHandle = fopen(fullfile(SAVE_REL_PATH, logFile), 'at');
end
```

Segue la dichiarazione di *splitStartTimes* che è un array di 0 di dimensione *numSplits* x 1: [0;0;0;0;0;0;0;0;0;0]. Esso rappresenta il tempo di esecuzione impiegato da ciascuno split.

Successivamente, vengono create le strutture utilizzate dal classificatore SVM, sia nel caso del training che del test. In entrambi i casi verranno utilizzate per i successivi calcoli dei risultati globali al termine dell'esecuzione.

Tra le variabili si noti la presenza di *svm_train_Dprime* che verrà utilizzata in seguito per la scelta del valore del parametro che dà il risultato migliore. Utilizzando questo valore, si addestrerà il classificatore sull'insieme costituito da quello di training e di validazione e lo si testerà su quello di test.

```
splitStartTimes = zeros(numSplits, 1);
%Variabili per salvare i risultati del classificatore SVM
svm_train_accuracies = zeros(numSplits,numDistances,length(paramCs));
svm_train_hits = zeros(numSplits,numDistances,length(paramCs));
svm_train_misses = zeros(numSplits,numDistances,length(paramCs));
svm_train_CRs = zeros(numSplits,numDistances,length(paramCs));
svm_train_FAs = zeros(numSplits,numDistances,length(paramCs));
svm_train_Dprime = zeros(numSplits,numDistances,length(paramCs));
svm_test_hits = zeros(numSplits,numDistances);
svm_test_misses = zeros(numSplits,numDistances);
svm_test_CRs = zeros(numSplits,numDistances);
```

Quanto detto nel caso di SVM vale anche per RBF, con la differenza che nel caso delle variabili che inizializzano le strutture per il training si ha che la dimensione aumenta, perché utilizzati i due parametri: λ e σ.

Le variabili *chosenC* (per SVM), *chosenLambda* e *chosenSigma* (per il classificatore RLS con kernel RBF) vengono create e inizializzate a 0, con una dimensione pari al numero di split attesi. Questo viene fatto perché dopo aver concluso l'esecuzione del modello ogni classificatore per ogni split sceglie un singolo valore dei parametri C, sigma e lambda da utilizzare per l'addestramento sull'insieme costituito da quello di traning e validazione e il testing su quello di test.

```
chosenC = zeros(numSplits, 1);
chosenLambda = zeros(numSplits, 1);
chosenSigma = zeros(numSplits, 1);
```

Il passo successivo è quello di recuperare il set completo delle immagini che verranno utilizzate durante l'esecuzione del modello è di suddividerle. La suddivisione è effettuata in base alla distanza del soggetto dell'immagine: testa, corpo, distanza media e lontana. In questo modo le immagini vengono suddivise in 4 insiemi. Ciascun insieme a sua volta è suddiviso in 3 sottoinsiemi: uno contenente le immagini target, uno contenente le immagini che vengono utilizzate come distrattore animale e un altro contentente distrattori artificiali. Come è possibile vedere dal codice che segue la suddivisione delle immagini viene fatta in base al nome del file. Attualmente il modello è ottimizzato per la gestione di immagini in formato JPG.

```
fileNames = dir (fullfile(IMAGE_DIR, '*.jpg'));
filesHeadTgt = dir(fullfile(IMAGE_DIR, 'H_*.jpg'));
filesHeadDistractArti = dir(fullfile(IMAGE_DIR, 'Hda_*.jpg'));
filesHeadDistractNatu = dir(fullfile(IMAGE_DIR, 'Hdn_*.jpg'));
filesBodyTgt = dir(fullfile(IMAGE_DIR, 'B_*.jpg'));
filesBodyDistractArti = dir(fullfile(IMAGE_DIR, 'Bda_*.jpg'));
filesBodyDistractNatu = dir(fullfile(IMAGE_DIR, 'Bdn_*.jpg'));
filesMedTgt = dir(fullfile(IMAGE_DIR, 'M_*.jpg'));
filesMedDistractArti = dir(fullfile(IMAGE_DIR, 'Mda_*.jpg'));
filesMedDistractNatu = dir(fullfile(IMAGE_DIR, 'Mdn_*.jpg'));
filesFarTgt = dir(fullfile(IMAGE_DIR, 'F_*.jpg'));
```

Come si può vedere le immagini che costituiscono l'insieme dei distrattori (naturali e artificiali) sono caratterizzate dalla presenza del carattere "d" in posizione 2 nel nome del file.

Per una gestione più efficiente si utilizzano 3 ulteriori variabili: *allFileList* che è una matrice che contiene la tipologia delle immagine (target, distrattore naturale o artificiale); *isAnimFile* che è un array che contiene 1 laddove l'immagine rappresenta un animale e quindi, come si può osservare, è in corrispondenza delle immagini target all'interno di *allFileList*; *fileDist* che indica la distanza delle immagini.

```
allFileList = {
 'filesHeadTgt' 'filesHeadDistractArti' 'fileHeadDistractNatu'
```

```
 'filesBodyTgt' 'filesBodyDistractArti' 'fileBodyDistractNatu'
 'filesMedTgt' 'filesMedDistractArti' 'filesMedDistractNatu'
 'filesFarTgt' 'filesFarDistractArti' 'filesFarDistractNatu' };
isAnimFile = [1 0 0 1 0 0 1 0 0 1 0 0];
fileDist   = [1 1 1 2 2 2 3 3 3 4 4 4];
```

Qui termina la fase di inizializzazione del modello e comincia il run vero e proprio.

L'esecuzione del modello

La prima operazione che viene eseguita è la creazione delle variabili *imageType* e *allImages* che sono 2 array inizialmente vuoti e che al termine di un ciclo che utilizza come indice i tipi differenti di immagini conterranno rispettivamente tutti i tipi di immagini e tutte le immagini stesse.

```
imageType = [];
allImages = [];
```

il passo successivo è quello di caricare la configurazione del modello di tipo feedforward sviluppato da Serre e colleghi.

```
config = FHConfig_PNAS07;
config = FHSetupConfig(config);
```

Su questo punto si tornerà in seguito. Per il momento è sufficiente considerare che la funzione *FHConfig_PNAS07()* è una funzione di libreria messa a disposizione dal framework FHLib, il cui scopo è quello di fornire la configurazione del modello. Tale configurazione, per poter essere utilizzata, deve essere validata e finalizzata da *FHSetupConfig()*.

Quel che fa *FHSetupConfig()* è inizializzare il set dei parametri definiti per ciascun livello del modello (*config*). Nel caso specifico serve per validare la configurazione che è l'insieme dei parametri che definiscono quanti livelli ci sono nella gerarchia e che tipo di filtri vengono utilizzati per ognuno di

essi. Il risultato della validazione è contenuto in *config* che è in pratica la struttura che rappresenta il modello stesso.

Fino a questo momento si è visto che prima di procedere all'elaborazione delle immagini il modello ha bisogno di essere inizializzato in tutte le sue componenti. L'ultima componente fondamentale di cui necessità il modello è il dizionario delle feature, contenuto in *lib*. Tale libreria si vedrà in seguito essere generata dalla funzione *featuresNatural_ newGRBF()*.

```
load(featureFile, 'lib');
```

A questo punto si sono ottenute sia la configurazione del modello (*config*) che il dizionario delle caratteristiche (*lib*) che verranno utilizzati per l'elaborazione delle immagini e la loro successiva classificazione. Si può partire con il run.

```
responseCells = zeros(length(allImages), sum(numFeaturesToUse));
whichFeatures = cell(length(levelsToExtractResp), 1);
```

responceCells è una matrice di dimensioni *allImages* x *sum(numFeaturesToUse)* (che vale 6000 in quanto ci sono 4 livelli e per ciascuno di essi si estraggono 1500 feature). Sarà questa struttura che conterrà il risultato globale del modello al termine del run. Invece *whichFeatures* è una matrice che conterrà il numero di feature estratte.

Detto ciò, segue un doppio ciclo for che per ogni immagine (*length(allImages)*) prima ne estrae le caratteristiche principali creando quello che viene chiamato lo stream, attraverso la funzione di libreria *FHCreateStream()* e poi per ogni livello (*lenght(levelsToExtractResp))* attraverso la funzione *FHGetResponses()* ne estrae il vettore risposta dallo stream.

```
for curImageNum = 1 : length(allImages)
 stream = FHCreateStream(config, lib,fullfile(IMAGE_DIR,
     allImages(curImageNum).name), topLevel);

 outputVector = [];
 for curLevel =  1 : length(levelsToExtractResp)
     tempResponses = FHGetResponses(config, lib, stream,
         levelsToExtractResp(curLevel));

 if (curImageNum == 1)
  numFeatures = length(tempResponses);
  if numFeatures < numFeaturesToUse(curLevel)
       numFeaturesToUse(curLevel) = numFeatures;
  end
  tempVec = [ones(1, numFeaturesToUse(curLevel)) zeros(1,
      numFeatures -  numFeaturesToUse(curLevel))];

  tempVec = tempVec(randperm(length(tempVec)));
  whichFeatures{curLevel} = tempVec;
 end % if (curImageNum == 1)

 tempResponses = tempResponses(find(whichFeatures{curLevel}));
 outputVector = [outputVector; tempResponses];
end % for curLevel =  1 : length(levelsToExtractResp)
```

In particolare *FHGetResponses()* prende come parametri: c, lib, s, levels, rispettivamente la configurazione del modello, la libreria di features, lo stream ottenuto dall'immagine appena elaborata da *FHCreateStream()* e il livello corrente. Quel che fa è creare un vettore di dimensioni *unitCount* che conterrà la risposta di tutte le unità di s per tutti i valori di *scaleScount*.

Ora, si consideri il seguente pezzo di codice:

```
...
 if (curImageNum == 1)
  numFeatures = length(tempResponses);
  if numFeatures < numFeaturesToUse(curLevel)
       numFeaturesToUse(curLevel) = numFeatures;
  end
  tempVec = [ones(1, numFeaturesToUse(curLevel)) zeros(1,
      numFeatures - numFeaturesToUse(curLevel))];
  tempVec = tempVec(randperm(length(tempVec)));
  whichFeatures{curLevel} = tempVec;
 end % if (curImageNum == 1)
```

come si può vedere viene eseguito solamente una volta e in particolare con la prima immagine. Questo viene fatto perché in questo modo vengono selezionate a caso quali features devono essere estratte da tutte le immagini successive per ogni livello.

Al termine del doppio ciclo for si sono estratte, per tutti i livelli complessi (C), tutte le feature da tutte le immagini che costituiscono il set completo. Tali caratteristiche sono contenute nella variabile *responseCells*. Ogni riga di *responseCells* rappresenta la risposta del modello ad una singola immagine.

```
responseCells(curImageNum,1:length(outputVector)) = outputVector;
```

Il fatto che non si possa fare direttamente *responseCells* = *outputVector* dipende dal fatto che la lunghezza di *outputVector* potrebbe non coincidere con il numero determinato staticamente di colonne di *responceCell*. Questo, ad esempio, potrebbe accadere se si volessero 3000 feature per il livello S3, ma in realtà ce n'è sono solamente 2000. Difatti, come si vedrà nel caso dell'elaborazione di immagini mammografiche, il numero di feature da estrarre cala drasticamente sotto le centinaia di feature. Questo dipende esclusivamente dal numero di feature che il dizionario è stato in grado di estrarre dall'insieme generico di immagini.

Arrivati a questo punto il run del modello si può considerare terminato e si può passare alla fase di classificazione.

L'esecuzione dei singoli split

L'esecuzione degli split richiesti è racchiusa nel seguente ciclo for:

```
for curSplitNum = 1 : numSplits
 %DO SOMETHING
end % for curSplitNum = 1 : numSplits
```

Come nel caso del run del modello l'esecuzione dei singoli split è preceduta dall'inizializzazione delle variabili relative.

Il primo passo da fare è quello di suddividere l'intero insieme di immagini in training e test. Quindi per ogni tipo si trovano gli indici delle immagini che sono del tipo corrente (*curType*) e le si suddivide nel seguente modo: circa una metà va a finire nell'insieme di training e l'altra metà nell'insieme di test. A sua volta l'insieme di training viene diviso in un insieme di validazione e di training. Viene utilizzato il vettore *tempRandVec* per indicare se un'immagine appartenga ai sottoinsiemi di test, di training o di validazione, rispettivamente indicati con 0,1 e 2. Come si può osservare viene eseguita anche un'operazione di permutazione degli elementi di tempRandVec in modo tale da garantire che ogni split scelga casualmente il sottoinsieme di immagini su cui operare. Ovviamente in questo modo può accadere che le stesse immagini vengano utilizzate più volte. Questo garantisce un grado maggiore di affidabilità dei risultati.

```
for curType = 1 : length(allFileList)
 indicesCurType = find(imageType == curType);
 numCurType = length(indicesCurType);
    numTrgImages = floor(numCurType / 2); % 50% del totale
 numTstImages = numCurType - numTrgImages;  % 1/4 del totale
 numValImages = floor(numTrgImages / 2); % 1/4 del totale
 numTrgImages = numTrgImages - numValImages;

 tempRandVec = [zeros(1, numTstImages) ones(1, numTrgImages) 2*
     ones(1, numValImages)];
 tempRandVec = tempRandVec(randperm(numCurType));

 isTrgImage(indicesCurType(tempRandVec == 2)) = 2;  isTrgImage(
     indicesCurType(tempRandVec == 1)) = 1;  isTrgImage(
     indicesCurType(tempRandVec == 0)) = 0;
end
```

Al termine del ciclo in *isTrgImage* si avrà che tutte le immagini sono state suddivise nei sottoinsiemi di training, test e validazione.

A questo punto si procede con la suddivisione delle immagini per tipo. Allo stesso modo viene diviso il vettore dei tipi e quello delle risposte ottenuto come risultato dall'esecuzione del modello (sempre in base ai tipi di

immagini). Queste numerose suddivisioni in sottoinsiemi torneranno utili in seguito.

```
comImages = allImages((isTrgImage == 1) | (isTrgImage == 2));
valImages = allImages(isTrgImage == 2);
trgImages = allImages(isTrgImage == 1);
tstImages = allImages(isTrgImage == 0);
comImagesType = imageType((isTrgImage == 1) | (isTrgImage == 2));
valImagesType = imageType(isTrgImage == 2);
trgImagesType = imageType(isTrgImage == 1);
tstImagesType = imageType(isTrgImage == 0);
comData = responseCells((isTrgImage == 1) | (isTrgImage == 2), :);
valData = responseCells((isTrgImage == 2), :);
trgData = responseCells((isTrgImage == 1), :);
tstData = responseCells((isTrgImage == 0), :);
```

Prima di procedere con l'addestramento del classificatore SVM è necessario definire quattro vettori detti di "Ground Truth" ovvero di "Grandi Verità" (GT). Con GT ci si riferisce a quel processo in cui ad esempio un pixel su di un'immagine satellitare è confrontato con quello che è la realtà (in quell'istante) con lo scopo di verificare il contesto del pixel sull'immagine. Nel caso di un'immagine classificata, permette una classificazione supervisionata in modo tale da aiutare a determinare l'accuratezza della classificazione minimizzando gli errori. Quindi sono da intendersi come una sorta di analisi parallela ed indipendente dei dati con lo scopo di validare i risultati ottenuti con la classificazione mediante l'uso di SVM.

Dopo aver inizializato i vettori delle grandi verità li si riempe. Questo viene fatto andando a verificare se l'immagine corrispondente è un distrattore o meno, ovvero verificando se nel nome del file è contenuto il carattere 'd'. In questo modo si sono discriminate le tipologie di immagini, cioè positive e distrattori (animali e artificiali).

```
tstWasPositiveGT = zeros(length(tstImages), 1);
trgWasPositiveGT = zeros(length(trgImages), 1);
valWasPositiveGT = zeros(length(valImages), 1);
comWasPositiveGT = zeros(length(comImages), 1);
```

```
for i = 1 : length(tstImages)
 tstWasPositiveGT(i) = (strcmp(tstImages(i).name(2), 'd') ~= 1);
end
for i = 1 : length(trgImages)
 trgWasPositiveGT(i) = (strcmp(trgImages(i).name(2), 'd') ~= 1);
end
for i = 1 : length(valImages)
 valWasPositiveGT(i) = (strcmp(valImages(i).name(2), 'd') ~= 1);
end
for i = 1 : length(comImages)
 comWasPositiveGT(i) = (strcmp(comImages(i).name(2), 'd') ~= 1);
end
```

Al termine di ciò si avrà che i vettori delle grandi verità conterranno 1 laddove l'immagine contiene un animale e 0 se contiene un distrattore naturale o artificiale che sia.

Arrivati a questo punto c'è la parte di addestramento sul vettore delle feature prodotto dal modello contenuto in *responceCells*. Questa fase è seguita dalla classificazione.

Addestramento

Si consideri il caso in cui ad essere utilizzato è solo il classificatore SVM. Si ricordi che in precedenza è stato detto che i risultati in caso di RLS con kernel RBF sono del tutto simili.

La prima cosa che viene fatta è quella di provare tutti i valori del parametro C (*paramCs*) sugli insiemi di training e di validazione, e poi usare i risultati per ottenere il valore unico migliore. Nel dettaglio questo viene fatto, per ogni split, all'interno del seguente ciclo for:

```
for curParamC = 1 : length(paramCs)
 paramCs = [0.002 0.005 0.01 0.05 0.1 0.5 1.0 5.0]

 % TRAIN AND RUN SVM ON TRAINING SET
 wasPositive = replicatePNAS_runSVM(SVM_REL_PATH, SAVE_REL_PATH,
      paramCs(curParamC), splitNums(curSplitNum), trgImages,
     valImages, trgData, valData,  trgWasPositiveGT,
```

```
        valWasPositiveGT, tempDoWriteTrg, tempDoWriteTst, doTrgSVM,
        doTstSVM, doSVMQuietly, doWriteLog, logFileHandle, 0);
...
```

Come si può vedere il passaggio principale riguarda la funzione *replicatePNAS_runSVM()* che tra i parametri prende il valore corrente di C, lo split attuale, gli insiemi di training e validazione contenenti i sottoinsiemi di immagini scelte a caso e le risposte delle rispettive unità contenute in trgData e valData, i vettori GT, eccetera.

L'obiettivo della funzione *replicatePNAS_runSVM()* è quello di creare e riempire il file di traning per SVM. All'interno di questo file è contenuto il numero di esempi positivi sul totale delle immagini analizzate per quanto riguarda le immagini di training. La stessa cosa viene fatta nel caso del file di testing che viene generato a runtime. Questo file conterrà il numero di esempi positivi sul totale dell'insieme di testing. Al termine dell'esecuzione della funzione wasPositive conterrà il numero di esempi positivi pari alla dimensione di valImages.

A questo punto avendo il numero di esempi positivi sia nell'insieme di training che di test si procede con il reale training e testing del classificatore SVM. Questo viene fatto utilizzando i seguenti due comandi contenuti all'interno delle stringhe *trainString* e *testString*:

```
trainString = sprintf('%s %s -c %.6f -z c -t 0 %s %s', fullfile(
    SVM_REL_PATH, 'svm_learn'), quietStringTrg, curParamC,
    trgFileName, modelFileName); % e.g. "c:|dir|svm_learn -z c -t
     0 svm_train.txt model.txt"

testString  = sprintf('%s %s %s %s %s', fullfile(SVM_REL_PATH, '
    svm_classify'), quietStringTst, tstFileName, modelFileName,
    outputFileName);
```

Come si può vedere vengono utilizzati i due file binari che compongono il classificatore. Si tratta di *svm_learn* e *svm_classify*[6].

[6]Sul sito http://svmlight.joachims.org/ è possibile ottenere i sorgenti C dei due eseguibili.

L'esecuzione di svm_learn e svm_classify avviene per ogni valore di c all'interno di ciascuno split. L'esecuzione consiste nel generare due file rappresentanti rispettivamente il modello e l'output dello stesso. A questa fase segue l'aggregazione dei risultati che vengono raccolti nella variabile *wasPositive,* la quale conterrà 1 se e solo se il valore corrispondente contenuto nel file di output è ≥ 0.

Questo viene fatto per tutti i valori di c all'interno del singolo split. E per ciascun valore di c ne viene calcolata l'accuratezza. L'accuratezza viene calcolata tenendo conto dei valori di hit (successo), miss (insuccesso), di FA)falsi positivi), e di CR (frequenza di classificazione). Di questo si discuterà in seguito all'interno di questo capitolo.

A questo punto che si sono testati i vari valori dei parametri sui set di training e di test, si sceglie quello che ha dato la risposta massima e lo si usa per testare l'insieme di test. Tale valore è scelto nel modo seguente:

```
% Choose a param value
if (strcmp(metricToUse, 'dprime') == 1)
 tempDprimes = mean(squeeze(svm_train_Dprime(curSplitNum, :, :)));
 chosenParamIndex = find(tempDprimes == max(tempDprimes));
chosenC(curSplitNum) = paramCs(chosenParamIndex(1));
end
```

Come si può vedere la scelta è fatta in base ai valori di *svm_train_Dprime* che è un vettore contenente i valori di *tempDprime* ottenuto nel seguente modo:

```
tempDprime  = norminv(tempHitrate, 0, 1) - norminv(tempFArate, 0, 1);
```

Ovvero, come la differenza tra l'inverso della funzione di distribuzione normale standard (norminv [7]) calcolata su *tempHitrate* e *tempFArate.*

Scelto tale valore del parametro lo si testa sugli insiemi di training e test, ovvero chiamando la funzione *replicatePNAS_runSVM()* specificando il

[7] http://en.wikipedia.org/wiki/Normal_distribution

paramentro *chosenC(curSplitNum)* che sta ad indicare il valore del parametro che è stato selezionato e il parametro *isFinalized*. Sui valori che saranno contenuti dentro *wasPositive* al termine dell'esecuzione delle funzione si andrà ad eseguire i calcoli riguardanti l'accuratezza, la percentuale di successo, il FA rate che sta ad indicare la frequenza dei falsi positivi e il valore della sensibilità (D') calcolato come abbiamo appena visto.

Ultimo passo è quello di calcolare i risultati globali. Quest'operazione viene eseguita dalla funzione *calcResults_SVM_chosen()*. Un esempio di un run che ha utilizzato 10 split:

```
USING SVM: OVERALL RESULTS

HEAD Dist: accu was 78 +- 0.9% (hit rate = 91 +- 1.3%, FA rate =
     34 +- 1.8%, D' = 1.79 += 0.07)

BODY Dist: accu was 84 +- 0.8% (hit rate = 82 +- 0.8%, FA rate =
     15 +- 1.1%, D' = 1.98 += 0.06)

MED Dist: accu was 80 +- 1.3% (hit rate = 76 +- 1.9%, FA rate =
    16 +- 1.6%, D' = 1.72 += 0.09)

FAR Dist: accu was 67 +- 1.0% (hit rate = 58 +- 1.8%, FA rate =
    25 +- 0.9%, D' = 0.89 += 0.06)

Chosen Cs were: 0.050 0.010 0.100 0.010 0.010 0.005 0.010 0.010
    0.002 0.050
```

Sull'analisi dei risultati si tornerà nel dettaglio nel prossimo paragrafo.

4.2 featuresNatural_newGRBF()

È la funzione che si occupa di generare il dizionario delle feature che viene utilizzato dal modello. Tale dizionario è generato a partire da un insieme generico di immagini.

L'obiettivo principale di questa funzione è quello di estrarre le feature di tipo S_2, S_{2b} e S_3 da tale insieme di immagini nel tentativo di simulare quella che è l'esperienza visiva umana.

Infatti, come si è detto nei capitoli precedenti, durante il riconoscimento degli oggetti è proprio a partire da quelle aree corticali che coincidono con il livello S_2 che avviene la raccolta di tali feature, man mano sempre più complesse, che andranno a costituire quello che è il dizionario delle caratteristiche. Dizionario che sarà poi reso disponibile nelle aree superiori della corteccia visiva, ad esempio PFC, per la classificazione degli oggetti. Per cui anche nell'implementazione della funzione che ne simula la costruzione si parte dal livello S_2.

In base a tale libreria il modello sarà in grado di decidere se un'immagine può essere classificata positivamente o negativamente.

Inizializzazione

Prima di procedere con la generazione del dizionario è necessario deciderne il comportamento. Ad esempio bisogna specificare qual è il numero di feature da estrarre per ogni immagine; il numero di immagini che si andranno a processare; il numero di input per una feature in S_2, S_{2b} e S_3; la scala minima e massima; la dimensione delle immagini prese in input e altro ancora. Per quanto riguarda i parametri *numImagesToFeaturesS** essi stanno ad indicare il numero di unità, appartenenti al livello precedente, che devono essere coinvolte nell'estrazione. Infatti, come detto nei capitoli precedenti ogni livello semplice effettua un'operazione di sintonizzazione utilizzando come input le unità afferenti del livello precedente.

```
numFeaturesPerImage = 2;
numImagesToSample = 1000;
numInputsToFeatureS2 = 10;
numInputsToFeatureS3 = 100;
numInputsToFeatureS2b = 100;
minScale = 1;
maxScale = 3;
imageSize = 256;
```

Esecuzione

In *replicatePNAS_CV()* la prima operazione che viene eseguita è quella di caricare la configurazione del modello all'interno della variabile *config*, che risulta essere una matrice di 1x9 celle ciascuna delle quali contiene i campi che caratterizzano un particolare livello del modello: 'image', 's1', 'c1', 's2b', 's2', 'c2b', 'c2', 's3', 'c3'.

```
config = FHConfig_PNAS07;
```

Particolare rilevanza l'assume il primo livello, quello denominato: 'image'. All'interno di tale livello sono contenute le informazioni che riguardano direttamente l'immagine e quindi la dimensione e soprattutto il numero di scale da utilizzare per la costruzione delle immagini piramidali.

Il passo successivo è quello di validare la configurazione che è stata appena caricata, in modo da renderla disponibile per l'elaborazione. La configurazione è costituita da un set di parametri che definisce quanti livelli ci sono nel modello e i filtri che verranno utilizzati per l'elaborazione. La configurazione risultante validata può essere poi utilizzata come parametro di input.

```
config = FHSetupConfig(config);
```

Per avere un'idea di come avviene la validazione si consideri l'elaborazione del primo livello. Se il campo *class* di *filters* è uguale ad "image" viene eseguito l'handler della funzione associata, che nel caso specifico è: *FHFilterImage().* Si tratta di un filtro speciale che crea il primo livello dello stream dall'immagine di input. Essa è responsabile della conversione dell'immagine in livelli di grigio (nel caso in cui l'immagine di input sia a colori), di un ridimensionamento appropriato nel caso in cui l'immagine supera i 256x256 pixel, e della generazione dell'immagine piramidale multiscala. Il filtro "image" è sempre il primo filtro nella configurazione del modello ed ha i seguenti parametri:

- *size*: la dimensione dell'immagine in scala 1 (100%) espressa in pixel.
- *sizeMode*: Opzionale. È una stringa che determina come interpretare il parametro size. Il valore di default è 'short', che significa che l'immagine di input sarà scalata così che la lungezza del lato più lungo diventi pari a size. Per le altre opzioni fare riferimento alla funzione *GLResizeGray()* che è la funzione che si occupa del resize dell'immagine.
- *scaleCount*: Il numero di scale per l'immagine piramidale.
- *scaleBase*, *scaleRoot*: Insieme determinano il valore di moltiplicazione tra le scale. Il fattore di scala sarà: (*scaleBase* ^ (1 / *scaleRoot*)). I settaggi di default sono: scaleBase = 2, scaleRoot = 3 or 4.

Al termine dell'esecuzione di *FHFilterImage()*, chiamata su 'image', si ha che il primo livello dello stream dell'immagine è stato creato.

```
filterNo: 1
name: 'image'
class: 'image'
prevFilterNo: 0
sizeMode: 'fit'
size: [256 256]
scaleCount: 11
scaleBase: 2
scaleRoot: 4
```

Il passo successivo è quello di eseguire i calcoli su di questo livello. Lo si fa andando a rieseguire *FHFilterImage()* ma questa volta in modalità: 'filterCalc'.

```
filterNo: 1
name: 'image'
class: 'image'
prevFilterNo: 0
izeMode: 'fit'
size: [256 256]
scaleCount: 11
```

```
scaleBase: 2
scaleRoot: 4
classPath: 'image'
classNo: 0
imageScales: [1 2 3 4 5 6 7 8 9 10 11]
scaleFactors: [1 1.1892 1.4142 1.6818 2.0000 2.3784 2.8284
    3.3636 4.0000 4.7568 5.6569]
xyMargin: 0
xyFactor: 1
locCount: []
```

Questo viene fatto per tutti quanti i livelli definiti in *config* che si ricordi essere 9. Quello che cambia sono le funzioni che vengono utilizzate. Infatti per ciascun livello, andando a vedere nella configurazione iniziale di *config*, si può notare che è specificata una determinata classe. Ad esempio, il livello 2 denominato 's1' ha come classe associata: 'ndp'. Pertanto, verrà gestita dall'handler opportuno che in questo caso corrisponde alla funzione *FHFIlterNDP()*.

Pertanto, si può osservare che ogni funzione, in questa fase, viene eseguita prima nella modalità 'filter' con lo scopo di inizializzare il livello dello stream relativo e successivamente nella modalità 'filterCalc' che esegue le operazioni necessarie per riempire i campi della struttura appena creata.

Gli handler associati alle classi dei livelli del modello sono i seguenti:

- *FHFilterNDP()*: 's1'. È un filtro s1 che esegue il prodotto scalare normalizzato. Un filtro 'ndp' calcola, utilizzando il prodotto scalare, i match tra le patch dei pixel e un template 2D fissato. Fa riferimento a FHFilterS1().

- *FHFilterMAX()*: 'c1'. È un filtro per i livelli complessi di tipo massimo. Un filtro 'max' trova la funzione di risposta massima nel suo insieme di aggregazione.

- *FHFilterGRBFNorm()*: 's2b' e 's3'. È una funzione Gaussiana RBF normalizzata per filtri semplici. Un filtro 'grbfnorm' calcola la differenza tra le patch e i prototipi utilizzando una funzione gaussiana RBF.

Essa prende il parametro sigma che è la deviazione standard. Questo filtro differesce da 'grbf' in quanto sigma è moltiplicato per la norma L2 del prototipo.

- *FHFilterGMax()*: 'c2b' e 'c3'. È un filtro massimo di tipo globale. Esso trova la risposta massima per ogni feature.

- *FHFilterS()*: 's2'. È un filtro semplice di base. Calcola la risposta delle patch delle unità all'interno di un livello precedente all'applicazione di patch (feature) composte dalle stesse unità, salvate in un dizionario di feature. Utilizza i seguenti paramentri:

  - xsSizes: la dimensione delle patch salvate nelle dimensioni x e y. Ci possono essere più patch salvate.
  - xyStep: Opzionale. La dimensione dello step, o fattore di sottocampionamento, nelle dimensioni x e y. Ad esempio, un valore di 2 significa che un'unità sarà calcolata solamente ogni seconda posizione nel livello precedente. Il livello risultante, grossolanamente, dimezzerà di un numero linerare le posizioni per ogni scala. Di default vale 1 (no sottocampionamento).
  - xySpacing: Opzionale. Controlla l'aumento di spazione, nelle dimensioni x e y, tra le unità in una patch prototipale. Di default vale 1 che significa che le patch sono composte di unità adiacenti. Un valore di 2 significa che le unità in una patch prototipale saranno incluse ogni 2 posizioni; in questo caso una patch 4x4 comprirà in realtà 7x7 posizioni (3 delle quali saranno ignorate).
  - scaleSize, scaleStep; scaleSpacing.

- *FHFilterC()*: 'c2'. È un filtro di base per i livelli complessi. Un filtro di questo tipo aggrega le unità con lo stesso tipo di feature nel livello precedente all'interno di un range locale di posizioni e scale. Esse hanno tutti i suguenti paramentri:

- xySize: la dimensione del range di aggregazione nelle dimensioni x e y. Ad esempio, un valore di 10 significa che un'unità è stata calcolata su di un pool di 10x10 patch di unità nel livello precedente.
- xyStep: Opzionale. La dimensione dello step, o il fattore di sottocampionamento, nelle dimensioni x e y. Per esempio, un valore di 5 significa che un'unità dovrebbe essere computata ogni 5 posizioni nel livello precedente. Il livello risultata avrebbe così circa 1/5 delle posizioni per ogni scala del livello precedente. Di default vale 1 (no sottocampionamento).
- scaleSize: Opzionale. La dimensione del range di aggregazione nella dimensione di scala. Per esempio, un valore di 2 significa che un'unità è calcolata aggregando le unità in 2 scale adiacenti nel livello precedente. Da notare che il valore di xySize si applica solo sulla più piccola scala del range di pool; ad una più alta scala sono meno corrispondenti le posizioni sulle quali fare l'aggregazione. Di default vale 1.
- scaleStep: Opzionale. La dimensione dello step, o fattore di sottocampionamento, nella dimensione della scala. Funziona similarmente a xyStep. Di default vale 1 (no sottocampionamento).

Fatto questo è necessario registrare tale configurazione in *lib*, cioè la struttura che rappresenta il dizionario. È in essa che verranno effettivamente salvate le feature estratte. Infatti, config non contiene feature nè staticamente nè apprese. Queste sono salvate separatamente in lib.

```
lib = FHSetupLibrary(config);
```

La modalità di registrazione che viene utilizzata da *FHSetupLibrary()* è identica a quella che si è vista sopra nel caso della creazione dello stream, cioè

della struttura del modello. La differenza sta nel fatto che alla funzione *FHCallFilter()* che si occupa di associare l'handler opportuno viene passata la parola chiave 'dict' al posto di 'filter' o 'filterCalc'.

```
lib.dicts{i} = FHCallFilter('dict', false, class, pf, c.filters{
    i}, pd, struct, dict);
```

dove dict{i} sta ad indicare il livello corrente che viene elaborato ed aggiunto alla libreria lib e suggerisce il fatto che è creato un dizionario per ogni livello di config.

Un accorgimento che viene utilizzato per rendere il tutto stocasticamente indipendente è quello di eseguire un rimescolamento degli indici delle immagini utilizzando una distribuzione discreta uniforme.

Dizionario di feature S_2, S_3, S_{2b}

Come si è detto nell'introduzione di questo paragrafo, l'estrazione delle delle caratteristiche avviene in tutti i livelli semplici del modello, ovvero S_2, S_3, S_{2b}.

Supponendo di avere 1000 immagini e avendo supposto di estrarre 2 feature per ciascuna di esse nel livello S_2 ci si ritrova ad estrarre circa 2000 feature. Affinchè il tutto avvenga nella maniera più casuale possibile il campionamento viene preceduto dall'esecuzione della funzione *GLInitRand()*. Questo dovrebbe assicurare una differente sequenza ogni volta.

La prima fase del campionamento delle feature all'interno di S_2 consiste nel campionare la prima immagine dell'insieme. In questo modo viene creato il dizionario, rappresentato dalla struttura dict che verrà successivamente utilizzato dalla funzione *FHCombineDict()*.

Il campionamento viene eseguito dalla funzione di libreria *FHSampleFeatures()* i cui parametri sono: la configurazione del modello (config); la libreria (lib) ovvero quello che costituisce il dizionario completo contenente

i template; lo stream o come nel caso specifico l'immagine; la dimensione dell'immagine che è stata impostata a 256 pixel; il livello ('s2'); il numero di feature per immagine (2); la scala. *FHSampleFeatures()* è utilizzata per la creazione di un dizionario (parziale) di feature, campionato su di una singola immagine per un determinato livello s.

```
dict = FHSampleFeatures(config, lib,
fixImage(fullfile(IMAGE_DIR, allImages(chosenImages(1)).name),
    imageSize),
's2', numFeaturesPerImage, minScale, maxScale);
```

dove lo stream dell'immagine rappresentato da allImages(chosenImages(1)).name è costituito dai livelli di grigio dell'immagine stessa.

Segue, il campionamento di tutte le immagini restanti. Come si può vedere il ciclo, che parte dalla seconda immagine crea un dizionario temporaneo in cui si va ad inserire il risultato del campionamento sull'immagine corrispondente all'indice. Successivamente viene fatto un append tra il dizionario (dict) e quello appena generato contenuto in newDict.

```
for index = 2 : numImagesToSample
 newDict = FHSampleFeatures(config, lib,fixImage(fullfile(
     IMAGE_DIR, allImages(chosenImages(index)).name), imageSize),
      's2', numFeaturesPerImage, minScale, maxScale);
 dict = FHCombineDicts(config, 's2', dict, newDict);
end
lib = FHSetDict(config, lib, 's2', dict);
```

Alla fine in dict saranno contenute tutte le feature estratte in corrispondenza del livello S_2.

La funzione *FHCombineDicts()*, che può essere utilizzata per combinare più dizionari in uno più grande contenente le feature di tutte le immagini analizzate, deve essere seguita dalla funzione *FHSetDict()*. Questo è fatto perché affinché sia possibile utilizzare tale dizionario è necessario registrarlo. Nel caso specifico dict viene registrato in lib e da questo momento in avanti può essere liberamente utilizzato per le elaborazioni.

Da notare che al termine del campionamento all'interno di ogni livello del dizionario ('s2', 's3', 's2b') viene eseguita la funzione *FHSparcifyDict()*. Lo scopo di questa funzione è quello di passare da una rappresentazione delle feature a basso livello ad un ad alto livello utilizzabile da un classificatore lineare come nel caso di SVM.

Il funzionamento di FHSparcifyDict() si può dividere in 2 fasi distinte:

1. Fase di Choose: consiste nel selezionare tutte le unità il cui valore è nullo.

2. Fase di Sparcify: elimina i valori nulli selezionati nella fase di choose e ordina gli elementi rimanenti, ovvero i valori delle unità campionate.

FHSampleFeatures

I passi principali sono:

- Innanzitutto si determina, attraverso la funzione *FHFilters()*, se il livello appartiene alla classe dei livelli semplici (s).

- Nel caso in cui lo stream, ovvero l'immagine, non sia stato creato lo si fa utilizzando la funzione *FHCreateStream()* che crea lo stream a partire da un'immagine. Per stream si intende la struttura che contiene le unità che saranno computate su tutti i livelli del modello. Le operazione principali di *FHCreateStream()* sono:

  - Innanzitutto viene letta l'immagine e nel caso la sua dimensione non sia conforme viene "aggiustata" dalla funzione *GLResizeGray()* che fa sì che la dimensione dell'immagine sia 256x256 pixel.
  - Viene inizializzata la struttura dei livelli.
  - Viene creato il livello dell'immagine che nella gerarchia del modello corrisponde al primo livello. Questo consiste nel creare la versione piramidale multiscala dell'immagine in base al valore di

scaleCount. Ovvero a partire dalla dimensione massima che come detto è 256x256 si cicla sull'immagine e per ogni ciclo si crea un livello dell'immagine caratterizzato da un determinato numero di coppie posizione/scala rappresentato da locCount. Da considerare che ad ogni giro ho un'immagine scalata di dimensioni sempre più piccole (da 256x256 fino a 45x45 considerando 11 scale) e di conseguenza diminisce anche il numero di unità coinvolte.

- Successivamente vengono creati i livelli superiori. Il procedimento è simile a quello appena visto, ovvero ciclando sul numero di scale si ottiene il numero di coppie posizione/scala uniche. Da notare che ad ogni scala diminuisce il range possibile fino ad arrivare ad 1.
- Infine vengono popolate le strutture dei livelli. Questo viene fatto dalla funzione *FHComputeLevels()*. il cui scopo è quello di calcolare il numero di differenti tipi di unità (contenuto in unitCount); il numero totale di unità in questo livello dato dal prodotto di locCount e typeCount; dire se il livello è stato allocato o meno (allocated), e infine se il livello è stato computato si otterà un array di celle contenenti matrici 3D. Una matrice per una scala Z avrà dimensione (typeCount x yCount(Z) x xCount(Z)). Questo rappresenta il numero di unità.
  Quando viene chiamata la funzione FHComputeLevels viene calcolato il ilvello specificato che è livello precedente. Avere come riferimento il livello precedente all'interno dello stream è necessario in quanto il campionamento lo si fa sulle unità afferenti ovvero quelle unità che si trovano all'interno dell'intorno di pooling del livello precedente.
- Per cui nello stream risultante, il valore delle unità calcolate per il livello L, la scala Z, la posizione X, Y, di tipo T, si troverà in: S.levels{L}.units{Z}(T,X,Y).

- Successivamente, vengono calcolati i valori di *unitValues, sizeIndex, sampleScales, sampleXs, sampleYs*. Questi sono campi che sono contenuti nel dizionario. È fatto utilizzando la funzione *FHSampleFeatures_MEX()* che si occupa del campionamento delle singole feature. In particolare:

  - Richiama la funzione *SampleFeature()* passando come parametri: prevLevel, sizeIndex, scale, x, y, unitValues Dove prevLevel è il livello precedente; sizeIndex è calcolato sul valore di SizeCount del filtro; scale è scelta a random (GLRand()) tra la scala minima e massima; e unitValues è preso dalla configurazione e costituisce il numero di unità che sono state effettivamente campionate.

- Attraverso la funzione *FHOrigCoords()* vengono calcolati i valori di imageScales, origXs e origYs. *FHOrigCoords()* converte coordinate specifiche per i livelli in un sistema di coordinate cartesiane. FHOrigCoords() utilizza i parametri sampleScale, sampleXs, sampleYs ottenuti come risultato della funzione FHSampleFeatures_MEX().

- Al termine questi valori vengono registrati nel dizionario (*dict*).

Quanto detto per il campionamento del livello S_2 vale anche per S_3 e S_{2b}. La differenza consiste nella quantità di feature da estrarre per ogni immagine e questo aumenta drasticamente il tempo di elaborazione.

Al termine del campionamento dei 3 livelli si ottiene quello che è il dizionario delle caratteristiche generato a partire dal livello S_2 e che verrà poi utilizzato per simulare il comportamento delle alte aree corticali come IT e PFC per la classificazione vera e propria.

4.3 Analisi dei risultati

Finora si è analizzato nel dettaglio qual è il funzionamento del dizionario delle feature e del modello. Rimane da analizzare i risultati ottenuti a seguito di un run completo.

Tornando alla funzione *replicatePNAS_ CV()*, al termine dell'esecuzione del classificatore che si è scelto di utilizzare il passo successivo è quello di calcolarne l'accuratezza il cui valore tiene conto delle frequenze di hit, di miss, di FA (la frequenza dei falsi positivi) e di CR (frequenza di classificazione) rappresentati rispettivamente dalle variabili *numHits*, *numMiss*, *numFAs* e *numCRs* che sono inizializzate a vettori di dimensione pari a *numDistances*.

Il riempimento è affidato al seguente ciclo for:

```
for tempIndex = 1 : length(wasPositive)
 if(valWasPositiveGT(tempIndex) && wasPositive(tempIndex))
     numHits(fileDist(valImagesType(tempIndex))) = numHits(
     fileDist(valImagesType(tempIndex))) + 1;

 elseif(valWasPositiveGT(tempIndex) && ~wasPositive(tempIndex))
      numMiss(fileDist(valImagesType(tempIndex))) = numMiss(
     fileDist(valImagesType(tempIndex))) + 1;

 elseif(~valWasPositiveGT(tempIndex) && wasPositive(tempIndex))
      numFAs(fileDist(valImagesType(tempIndex))) = numFAs(
     fileDist(valImagesType(tempIndex)))  + 1;

 elseif(~valWasPositiveGT(tempIndex) && ~wasPositive(tempIndex))
       numCRs(fileDist(valImagesType(tempIndex))) =  numCRs(
     fileDist(valImagesType(tempIndex)))  + 1;
 end
end
```

Come si può vedere, si ottiene un hit quando in corrispondenza dell'indice *temIndex* è presente il valore 1 sia in *valWasPositiveGT* sia in *wasPositive*, ovvero si tratta di un'immagine di training. Altrimenti, si tratta di un miss nel caso in cui *valWasPositive* in corrispondenza dell'indice vale 1 ma *wasPositive* no, ovvero pur essendo un'immagine animale non è stata considerata

come tale dal classificatore. Stesso discorso vale per *numFAs* in cui *valWasPositiveGT* è 0 e *wasPositive* è 1 e *numCRs* che in entrambi i casi vale 0.

Su questi valori vengono eseguiti i calcoli per ottenere i valori per l'accuratezza, l'hitrate, il missrate, e le frequenze di FA e di CR per ogni valore di c e questo viene fatto per tutti i valori e per tutti gli split. Al termine di ciò si può passare al calcolo dei risultati globali. Quest'operazione viene eseguita dalla funzione *calcResults_SVM_chosen()*.

```
for curSplitNum = 1 : numSplits
 tempAccus(curSplitNum) = accuracies(curSplitNum, curDist);

 tempHits(curSplitNum) = hits(curSplitNum, curDist) /
(hits(curSplitNum, curDist) + misses(curSplitNum, curDist));

 tempFAs(curSplitNum) = FAs(curSplitNum, curDist) /
(FAs(curSplitNum, curDist) + CRs(curSplitNum, curDist));

 if (tempHits(curSplitNum) == 1.0)
  tempHits(curSplitNum) = tempHits(curSplitNum) - 1/N;

 elseif (tempHits(curSplitNum) == 0.0)
  tempHits(curSplitNum) = tempHits(curSplitNum) + 1/N;
 end

 if (tempFAs(curSplitNum) == 1.0)
  tempFAs(curSplitNum) = tempFAs(curSplitNum) - 1/N;
 elseif (tempFAs(curSplitNum) == 0.0)
  tempFAs(curSplitNum) = tempFAs(curSplitNum) + 1/N;
 end
 tempDprime(curSplitNum) = norminv(tempHits(curSplitNum)) -
     norminv(tempFAs(curSplitNum));

end
```

Come si può osservare per ogni split i valori vengono calcolati nel seguente modo:

- L'accuratezza è calcolata prendendo il valore corrispondente allo split e alla distanza correnti.

- l'hit è calcolato dividento il valore dell'hit corrente per il totale della somma degli hit e miss
- La stessa operazione viene eseguita anche per il valore di FA.
- segue il calcolo della misura di sensibilità (D') ottenuta come la sottrazione il valore della distribuzione cumulativa normale inversa calcolata sul *tempHits* dello split corrente e il valore di *tempFAs*. D' è la misura delle performace che, per ogni osservatore, combinano sia l'hit rate, cioè, la proporzione di immagini di animali correttamente classificate dall'osservatore, e i falsi positivi, cioè, la proporzione di immagini non animali non correttamente classificate.

Di seguito è riportato il risultato di un run eseguito sull'intero set di 1200 immagini considerando 10 split.

```
USING SVM: OVERALL RESULTS
HEAD Dist: accu was 78 +- 0.9% (hit rate = 91 +- 1.3%, FA rate =
     34 +- 1.8%, D' = 1.79 += 0.07)

BODY Dist: accu was 84 +- 0.8% (hit rate = 82 +- 0.8%, FA rate =
     15 +- 1.1%, D' = 1.98 += 0.06)

MED Dist: accu was 80 +- 1.3% (hit rate = 76 +- 1.9%, FA rate =
    16 +- 1.6%, D' = 1.72 += 0.09)

FAR Dist: accu was 67 +- 1.0% (hit rate = 58 +- 1.8%, FA rate =
    25 +- 0.9%, D' = 0.89 += 0.06)

Chosen Cs were: 0.050 0.010 0.100 0.010 0.010 0.005 0.010 0.010
    0.002 0.050
```

Come si può osservare ciascun valore è corredato dalla rispettiva deviazione standard.

Ora si cercherà di dare un'interpretazione ai risultati mostrati.

Innanzitutto, si consideri il valore dell'accuratezza. Esso è ottenuto, come già detto come la media delle accuratezze dei singoli split. E ciascuna accuratezza nel modo seguente.

```
numPosGT = numHits(curDist) + numMiss(curDist);
numNegGT = numFAs(curDist) + numCRs(curDist);

tempAccu = (numHits(curDist) + numCRs(curDist)) / (numPosGT + numNegGT);
```

Ovvero, l'accuratezza è calcolata come il rapporto tra il numero di successi sul numero totale.

Per cui, considerando il fatto che vengono testate contemporaneamente 4 classi di immagini, in base alla distanza, avere dei risultati per quanto riguarda l'accuratezza che vanno da 70% circa in su è un buon risultato se confrontato con le performance degli esseri umani, come si è visto nel capitolo precedente.

Infatti, in test successivi in sono state testate classi singole di immagini l'accuratezza risulta essere migliore, per il semplice fatto che per ogni split scegliendo casualmente le immagini di training e di test in ogni caso si ricade nella stessa classe di immagini.

Inoltre, essendo che ciascuno split sceglie a caso su quali immagini operare avendo un numero così elevato di immagini divise in maniera pressoché identica i valori sia dell'accuratezza che degli hit e miss possono variare da run a run. Per cui per avere un dato medio converrebbe eseguire il run, ad esempio, n = 20 volte.

Per quanto riguarda il valore dei falsi positivi c'è da notare che in alcuni casi tocca un valore pari a circa la metà del campione testato.

Capitolo 5

Contributo originale

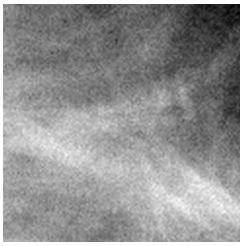
Finora si è visto che il modello è in grado di simulare il riconoscimento degli oggetti da parte della corteccia visiva umana, eguagliandone le performance almeno per quanto riguarda i task di riconoscimento animale *vs.* non animale.

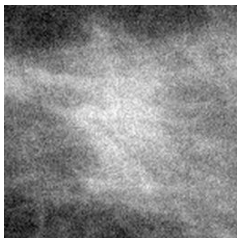
Ora, quello che si vorrebbe verificare è se possa essere generalizzato in modo tale da poterlo utilizzare su qualsiasi task di riconoscimento. È possibile fare ciò? Ed in particolare potrebbe essere in grado di classificare immagini mammografiche?

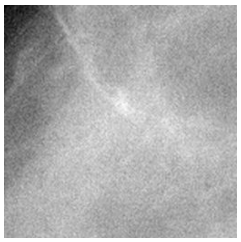
Per poterlo dimostrare, si è proceduto in un passo intermedio. Si è voluto verificare se il modello potesse essere utilizzato per la classificazione di immagini generiche, ad esempio automobili e volti umani, all'interno di condizione ben determinate. Solo successivamente sono stati preparati dei test specifici atti a verificare se esso fosse in grado di distinguere le differenti classi di immagini mammografiche testate sia singolarmente che contemporaneamente.

Nel caso delle immagini di mammografie le classi sono 3: normali ovvero che non contengono nessun segno di calcinoma; vere positive che presentano il calcinoma; e i falsi positivi.

Innanzitutto, a differenza del modello originale che utilizza immagini in formato JPG si è scelto di utilizzare immagini in formato TIFF. La ragione


(a) Normlae


(b) Veri Positivi


(c) Falsi Positivi

principale di questa scelta sta nel fatto che il formato TIFF permette di mantenere l'intera qualità dell'immagine, anche se a scapito di una maggiore dimensione dei file. L'utilizzo di un formato differente da JPG è consentito dalla funzione *rgb2gray()* che prende un'immagine a colori a 8 o 16 bit e la converte in una in livelli di grigio. In secondo luogo, la scelta del formato TIFF è fatta in quanto le immagini rispettano lo standard DICOM (Digital Imaging and COmmunications in Medicine, immagini e comunicazione digitali in medicina)[1] che definisce i criteri per la comunicazione, la visualizzazione, l'archiviazione e la stampa di informazioni di tipo biomedico quali ad esempio immagini radiologiche, come appunto le mammografie.

All'interno di ogni test, concluse le fasi preliminari, si è proceduto con l'addestramento del dizionario relativo, il quale essendo predisposto all'estrazione delle caratteristiche comuni a partire da un set generico di immagini non ha previsto sostanziali modifiche. Infatti, si è andati a modificare principalmente il valore delle feature da estrarre da ogni immagine. La possibilità di avere più dizionari contemporaneamente è data dal fatto che ognuno di essi è rappresentato da un file con estensione MATLAB e quindi, rinominandoli opportunatamente, è possibile averne quanti se ne vuole.

Segue la descrizione nel dettaglio dei test condotti. Ciascun test è stato eseguito 20 volte. Il motivo di ciò dipende dal fatto che un run del modello è costituito da un numero variabile di split e ciascuno di essi seleziona casualmente un sottoinsieme di immagini sia per l'addestramento che per la

[1] http://it.wikipedia.org/wiki/DICOM

successiva classificazione. Questo comporta il fatto che il livello di accuratezza di uno split possa essere totalmente differente da quello di un altro perché ad esempio quello split ha operato su più distrattori che target. Per cui, l'esecuzione di un numero elevato di run successivi, ha come obiettivo quello di normalizzare i risultati prodotti.

Test 1

In questo primo test, l'obiettivo è quello di verificare se, a partire da un insieme di immagini generiche di autovetture, il modello addestrato su di un suo sottoinsieme sia in grado di classificare le immagini restanti. Si considerano le seguenti condizioni:

- a) Si hanno solo le immagini target.
- b) Alle immagini target sono aggiunti un ristretto numero di distrattori.
- c) Si hanno solo distrattori.

Per fare ciò, in accordo con gli esperimenti condotti da Serre e colleghi, si è suddiviso l'insieme delle immagini in 2 sottoinsiemi di training e di test, contententi ciascuno circa la metà delle immagini. In questo modo si è potuto procedere con l'addestramento del dizionario sul solo insieme di training.

Una volta ottenuto il dizionario si è proceduto con la classificazione sull'insieme di test.

Nel caso del condizione a) i risultati sono stati i seguenti:

```
BSpos Dist: accu was 100 +- 0.0% (hit rate = 97 +- 0.0%, FA rate
     = 3 +- 0.0%, D' = 3.80 += 0.00)
```

Come si può osservare nel caso in cui nell'insieme di test siano contenute solo immagini di automobili la misura dell'accuratezza è del 100% e l'hit rate si attesta intorno al 97%. Questo vuol dire che il classificatore è stato in grado di individuare con esattezza tutte le immagini target.

Come ci si spiega il fatto che nonostante ci sia un'accuratezza del 100% non lo sia anche l'hit rate e addirittura la frequenza dei falsi positivi sia diversa da 0? Questo è dovuto al fatto che per evitare che il valore delle derivata prima tenda all'infinito si applica una correzione convenzionale che consiste nel calcolare i valori di hit rate e FA rate come segue:

```
tempHitrate = 1 / (numPosGT + numNegGT);
tempFArate = 1 / (numPosGT + numNegGT);
```

Si osservi cosa succede nella condizione b) se si utilizza lo stesso dizionario addestrato sulle immagini di automobili con l'aggiunta di distrattori rappresentati da dei volti di persona. Si ottengono i seguenti risultati:

```
BSpos Dist: accu was 97 +- 0.4% (hit rate = 97 +- 0.0%, FA rate
    = 97 +- 0.0%, D' = 0.00 += 0.00)
```

Si può notare che il livello di accuratezza nel complesso cala di poco rispetto alla condizione a). Quello che varia in maniera sostanziale è il valore dei falsi positivi che sfiora il 100%. Questo suggerisce il fatto che nell'atto di riconoscere i distrattori come immagini contenenti un'automobile il classificatore, correttamente, le considera dei falsi positivi. Questo è accaduto perché si è forzata l'individuazione dei distrattori rinominando apportunamente le immagini relative.

Infine, si consideri la condizione c) in cui invece l'insieme delle immagini di test è rappresentato dai soli distrattori, anche se a differenza della condizione b) in questo caso non si sono rinominate le immagini.

Seguono i risultati:

```
BSpos Dist: accu was 100 +- 0.0%  (hit rate = 94 +- 0.0%, FA
    rate = 6 +- 0.0%, D' = 3.13 += 0.00)
```

In questo caso, senza aver forzato l'individuazione, i risultati non sono confortanti perché il modello non è in grado di riconoscere nessuna delle immagini sottoposte come distrattore. Questo suggerisce la seguente considerazione: probabilmente il fatto di utilizzare un numero esiguo di feature (10) estratte per ogni immagine non consente di ottenere un dizionario che sia realmente discriminante. La soluzione potrebbe essere quella di aumentare il numero di feature.

Test 2

Per confermare quanto osservato con il primo test si va ad addestrare un secondo dizionario su delle immagini appartenenti alla classe dei volti umani. Allo stesso modo si procede dividendo l'insieme in 2 sottoinsiemi di training e di test contenenti ciascuno circa la metà delle immagini.

Questa volta a differenza del precedente test si prova ad aumentare il numero delle feature passando da 10 a 50 caratteristiche estratte per ogni immagine.

Si consideri la condizione a). I risultati sono i seguenti:

```
BSpos Dist: accu was 100 +- 0.0% (hit rate = 94 +- 0.0%, FA rate
     = 6 +- 0.0%, D' = 3.10 += 0.00)
```

Come si può osservare nella condizione a) i risultati sono ottimi, anche se di poco inferiori a quelli dell'test precedente. In ogni caso il livello di accuratezza è del 100% e questo sta ad indicare che il classificatore ha riconosciuto correttamente l'intero set di immagini sottoposto.

Ora, si consideri la condizione b) ovvero quella in cui l'insieme di test oltre alle immagini target contiene anche un numero esiguo di distrattori e si ricordi che anche in questo caso i distrattori sono stati rinominati per forza la loro individuazione da parte del classificatore.

```
BSpos Dist: accu was 90 +- 1.7% (hit rate = 88 +- 0.0%, FA rate
    = 72 +- 10.0%, D' = 0.46 += 0.31)
```

Come nel caso dell'test 1 nella condizione b) si ha che sia le percentuali di hit rate che dei falsi positivi sono individuate correttamente.

Infine, si osservi cosa succede nella condizione c) dove i distrattori non sono stati rinominati:

```
BSpos Dist: accu was 100 +- 0.0% (hit rate = 86 +- 0.0%, FA rate
     = 14 +- 0.0%, D' = 2.14 += 0.00)
```

In questa condizione si ripresentano gli stessi risultati del precedente test. Ovvero, nel caso in cui siano presenti solo distrattori i quali non siano stati opportunatamente rinominati, il classificatore non è in grado di identificarli come tali inficiando il risultato finale.

In questi primi 2 esperimenti si è visto che il modello e la rispettiva implementazione possono essere utilizzati per la classificazione di immagini generiche, anche se con risultati non sempre soddisfacenti soprattutto nel caso in cui l'insieme delle immagini di test sia composto esclusivamente da distrattori non trattati. Infatti in questo caso, invece di rispondere con una percentuale dei falsi positivi elevata il modello considera tali immagini come target. Probabilmente si avrebbe una risposta più accurata se in primo luogo si disponesse di insiemi consistenti di immagini, come nel caso del task animale *vs.* non animale e se venissero estratte un maggior numero di feature.

Da considerare che l'aumento del numero delle caratteristiche da estrarre per ogni immagine è esoso dal punto di vista computazionale.

Negli esperimenti che seguono, tenendo conto delle considerazioni fatte fino a questo momento, si andranno ad osservare quali sono le prestazioni del modello nell'ambito delle immagini mammografiche con l'obiettivo di capire se il modello possa effettivamente tornare utile allo scopo.

Test 3

In questo test, l'obiettivo è quello di verificare se il modello addestrato su di un sottoinsieme di immagini mamografiche vere positive, sia in grado di classificare correttamente le restanti immagini.

Innanzitutto, tenendo conto delle considerazioni fatte nei precedenti esperimenti, si è generato il dizionario estraendo 100 feature per ciascuna immagine. Una volta ottenuto il dizionario si è proceduto con la classificazione sull'insieme di test.

I risultati di questo test nella condizione a) sono i seguenti:

```
BSpos Dist: accu was 100 +- 0.0% (hit rate = 88 +- 0.0%, FA rate
     = 12 +- 0.0%, D' = 2.30 += 0.00)
```

Come si può vedere l'accuratezza è del 100 %, mentre l'hit rate è del 88% e l'FA rate è del 12%. Una prima osservazione che si può fare è quella che anche se l'accuratezza è identica a quella degli esperimenti 1 e 2 l'hit rate risulta essere lievemente inferiore, ma essendo un risultato medio è di poco conto.

Ora si supponga di essere nella condizione b). Come nei casi precedenti si sceglie di forzare la classificazione. I risultati sono stati i seguenti:

```
BSpos Dist: accu was 62 +- 4.2% (hit rate = 68 +- 3.8%, FA rate
    = 60 +- 7.6%, D' = 0.20 += 0.10)
```

Si può osservare che inserendo dei distrattori la percentuale dell'accuratezza diminuisce attestandosi al 62% mentre l'hit rate e i falsi positivi sono rispettivamente del 68% e del 60%. Questo indica che anche in questo test se si inseriscono dei distrattori queste vengono individuate se e solo se esse sono state precedentemente trattate.

Si considerei l'ultima condizione c), cioè quella in cui tutte le immagini dell'insieme sono costituite da distrattori. Si ottengono i seguenti risultati:

```
BSpos Dist: accu was 100 +- 0.0% (hit rate = 75 +- 0.0%, FA rate
     = 25 +- 0.0%, D' = 1.35 += 0.00)
```

Da considerare che le percentuali sono inferiori rispetto agli esperimenti precedenti perché sono state considerate per comodità un minor numero di immagini.

Quindi, anche in questo caso si può concludere che il modello non è in grado di discrimare correttamente tra le immagini target e i distrattori costituiti dai falsi positivi nonostante si sia decuplicato il numero di feature estratte.

Quanto detto all'interno di questo test vale anche nel caso dei falsi positivi e delle immagini normali.

Test 4

Nei precedenti esperimenti si è visto che considerando singolaremente le classi di immagini si ottengono ottimi risultati solo nel caso in cui l'obiettivo della classificazione sia l'immagine target, mentre si ottengono risultati negativi altrimenti. Per concludere la panoramica, in questo e nel prossimo test si andrà a verificare cosa accade se si considerano tutte e 3 le classi di immagini mammografiche contemporaneamente.

Per fare ciò è stato, innanzitutto, addestrato nuovamente il dizionario delle caratteristiche comuni allo stesso modo degli esperimenti precedenti. Ovvero si è andati a suddividere l'insieme delle immagini in 2 sottoinsiemi di training e di test. Allo stesso modo si è proceduto con l'addestramento del dizionario sull'insieme di training e successimante con la classificazione su quello di test. I risultati sono i seguenti:

```
NORMAL Dist: accu was 100 +- 0.0% (hit rate = 97 +- 0.0%, FA
    rate =  3 +- 0.0%, D' = 3.89 += 0.00)

BSPOS Dist: accu was 100 +- 0.0% (hit rate = 97 +- 0.0%, FA rate
     = 3 +- 0.0%, D' = 3.89 += 0.00)

BS Dist: accu was 100 +- 0.0% (hit rate = 97 +- 0.0%, FA rate =
    3 +- 0.0%, D' = 3.89 += 0.00)
```

Come si può vedere in questo i livelli di accuratezza delle classi sono ottimi con un hit rate del 97%. Questi risultati suggeriscono che se le immagini vengono considerate senza fare distinzione tra target e distrattori, i risultati sono ottimi. E qual è lo scopo di questo procedimento se esso non è in grado di discriminare le immagini? Allo stato attuale dell'implementazione del modello suggerisce che esso possa essere utilizzato come strumento di conferma.

Alla luce di quanto detto, si può concludere che il modello così come è stato realizzato non è sufficientemente adatto all'individuazione di una determinata classe di oggetti se testato contemporaneamente ad altre classi distrattori.

Tirando le somme degli esperimenti condotti si può desumere che il modello opportunamente addestrato su di un set di immagini generiche in determinate condizione risulta essere valido e con livelli di accuratezza particolarmente alti, se utilizzato al di fuori di un task di riconoscimento animale *vs.* non animale. Invece, nel caso in cui ad essere testate sullo stesso dizionario di feature siano la classe delle immagini distrattori i risultati non sono buoni perché il classificatore non è in grado di riconoscerne la differenza.

Perciò, si può concludere che il modello attualmente possa essere utilizzato come supporto alla classificazione di singole classi senza considerare distrattori. O per lo meno bisognere considerare l'ipotesi di, a partire da un numero cospicuo di immagini, estrarre un numero considerevole di feature da ogni immagine creando così un dizionario fortemente orientato alla classe di appartenza. In questo modo le immagini dovrebbero essere classicate adeguatamente.

Capitolo 6

CONCLUSIONI

In questo elaborato si è potuto analizzare nel dettaglio il modello sviluppato da Thomas Serre e colleghi. Si tratta di un modello quantitativo - che è consistente con le teorie dei processi visivi che estendono il modello gerarchico[HubWie59] - di tipo feedforward della via ventrale della corteccia visiva che comprende le aree corticali da V1, V2 sino all'IT e al PFC.

Una delle chiavi portanti di questo modello è l'apprendimento di un dizionario generico delle componenti morfologiche a partire dall'area V2 (che all'interno dell'implementazione corrisponde al livello semplice S_2) all'IT, il cui obbiettivo è quello di fornire una rappresentazione invariante ai circuiti di categorizzazione rapida che si trovano nelle aree superiori del cervello (rappresentate dal classificatore lineare SVM). Questo dizionario delle unità sintonizzate sulla forma è appreso in maniera non supervisionata dalle immagini naturali. E a livello implementativo corrisponde alla funzione *featuresNatural_ GRBF()*.

La natura quantitativa del modello è stata descritta nel Capitolo 3 dove è stato confrontato prima con il comportamento dei neuroni mostrando come il modello riesca a duplicarne le proprietà di sintonizzazione nelle varie aree del cervello (ad esempio V1, V4, ed IT). Successivamente, si è visto come, oltre a duplicare le proprietà dei neuroni in presenza di stimoli artificiali, sia

in grado di gestire il riconoscimento di oggetti nel mondo reale, estendendo così la competitività con i migliori sistemi di computer vision. Ed infine, è stato mostrato il confronto tra le performance del modello e quelle degli esseri umani nel caso di task di riconoscimento rapido animale *vs.* non animale. I risultati hanno evidenziato che è in grado di predirre il livello delle performance umane.

Presi insieme, questo modello e la teoria su cui esso si basa, costituiscono lo scheletro di una teoria di successo del "riconoscimento immediato" nella corteccia visiva.

Terminata l'analisi si è cercato di verificare se fosse possibile adattare questo modello generale della corteccia visiva ad un caso specifico: il riconoscimento di immagini di mammografie.

L'obiettivo è stato quello di verificare se tale implementazione del modello fosse in grado di classificare con un livello di accuratezza accettabile una determinata classe di oggetti. Tale dizionario è stato ottenuto a partire da un insieme di immagini di diversa natura: "normali" che non presentano alcuna forma di calcinoma; "veri positivi" che presentano effettivamente cellule cancerogene; "falsi positivi".

I risultati non sono stati omogenei. Infatti, se da un lato i test che hanno coinvolto o un'unica classe di immagini target o tutte e 3, senza considerare distrattori, i risultati sono stati buoni con percentuali di accuratezza elevate, dall'altro lato laddove si è tentata una classificazione che tenesse conto della presenza di immagini distrattori i risultati sono stati negativi.

In base a quanto mostrato nel corso degli esperimenti, si può concludere che il modello per come è pensato e implementato attualmente non è adatto alla classificazione di questa categoria di immagini. Questo è dovuto a diversi fattori principali:

- l'obiettivo primario del modello è quello di verificare le prestazioni nel caso di un task di riconoscimento immediato animale *vs.* non animale

e non quello di individuare una generica classe di immagini, anche se come si è visto questo è possibile.

- Esso è ancora in una fase embrionale, in quanto si basa su di una teoria, quella del riconoscimento degli oggetti da parte della corteccia visiva, che non è ancora totalmente definita.

Sviluppi Futuri

Alla luce di quanto detto il modello lascia ampi margini di miglioramento sia a livello di implementazione che della teoria su cui esso si basa. Alcuni dei possibili sviluppi futuri sono:

- Nella presente implementazione del modello la sintonizzazione delle cellule semplici in V1 è cablata principalmente sull'orientamento delle unità. Ci si aspetterebbe che un apprendimento non supervisionato dalle immagini naturali fosse ricettivo oltre che verso l'orientamento anche e soprattutto verso stimoli più complessi come ad esempio le unità stesse.

- Dovrebbero essere inclusi meccanismi per il riconosicmento del colore e della stereoscopia[1].

  - La presente implementazione è in grado di gestire unicamente immagini in livelli di grigio. L'introduzione di meccanismi per il colore calza bene con il fatto che l'informazione sul colore non sembra avere un impatto sulle performance nei task di riconoscimento rapido [DRF00] e soprattutto con il fatto che i fenomeni più complessi coinvolgono sempre la componente colore e perciò bisognerebbe considerarne l'integrazione.

[1] La stereoscopia è una tecnica di realizzazione e visione di immagini, disegni, fotografie e filmati, finalizzata a trasmettere una illusione di tridimensionalità, analoga a quella generata dalla visione binoculare del sistema visivo umano.

  - La stereoscopia può potenzialmente giocare un ruolo nell'apprendimento non supervisionato aiutando la segmentazione tra l'oggetto e il background.

- Un altro possibile sviluppo potrebbe essere l'inclusione della Back-Projection che però comporterebbe la ridefinizione di tutta la teoria in quanto in questa architettura se n'è supposta l'assenza. La sua introduzione sarebbe dovuta al fatto che mentre task di riconoscimento anche complessi come la categorizzazione degli oggetti nelle immagini naturali possono essere eseguiti all'interno di una finestra temporale di circa 150 ms, esistono task di riconoscimento non immediato che necessitano di un tempo maggiore: alcuni task richiedono probabilmente ricorsioni delle predizioni, verifiche e appunto i circuiti di Back-Projection.

- L'introduzione di meccanismi di elaborazione delle immagini come l'edge detection. Questo potrebbe consentire una classificazione più efficiente in quanto il numero delle unità da considerare per ogni immagine calerebbe drasticamente.

Bibliografia

[Ser06] T. Serre, *Learning a Dictionary of Shape-Components in Visual Cortex: Comparison with Neurons, Humans and Machines*, 2006.

[HubWie59] D.H. Hubel and T.N. Wiesel, *Receptive fields of single neurons in the cat's striate visual cortex, J. Phys.*, 148:574–591, 1959.

[RiePog99] M. Riesenhuber and T. Poggio, *Hierarchical models of object recognition in cortex*, Nat. Neurosci., 2:1019–1025, 1999a.

[FelvEs91] D.J. Felleman and D.C. van Essen, *Distributed hierarchical processing in the primate cere- bral cortex*, Cereb. Cortex, 1:1–47, 1991.

[RiePog00] M. Riesenhuber and T. Poggio, *Models of object recognition*, Nat.Neurosci.Supp., 3:1199– 1204, 2000.

[MUM83] M. Mishkin, L.G. Ungerleider, and K.A. Macko, *Object vision and spatial vision: Two cortical pathways*, Trends Neurosci., 1983.

[DeYEss88] E. A. DeYoe and D. C. Van Essen, *Concurrent processing streams in monkey visual cortex*, Trends. Neurosci., 11(5):219–26, May 1988.

[UngHax94] L.G. Ungerleider and J.V. Haxby, *'What' and 'where' in the human brain*, Curr. Op. Neuro- biol., 4:157–165, 1994.

[Tan96] K. Tanaka,*Inferotemporal cortex and object vision*, Ann. Rev. Neurosci., 19:109–139, 1996.

[Mil00] E.K. Miller, *The prefrontal cortex and cognitive control*, Nat. Rev. Neurosci., 1:59–65, 2000.

[FRPM03] D.J. Freedman, M. Riesenhuber, T. Poggio, and E.K. Miller, emphComparison of primate pre- frontal and inferior temporal cortices during visual categorization, J. Neurosc., 415:5235– 5246, 2003.

[Miy88] Y. Miyashita, emphNeuronal correlate of visual associative long-term memory in the primate temporal cortex, Nature, 335:817–820, 1988.

[EJD00] C.A. Erickson, B. Jagadeesh, and R. Desimone,emphClustering of perirhinal neurons with simi- lar properties following visual experience in adult monkeys, Nat. Neurosci., 3:1143–1148, 2000.

[Pot75] M.C. Potter, emphMeaning in visual search Science, 187:565–566, 1975.

[TFM96] S.J. Thorpe, D. Fize, and C. Marlot,emphSpeed of processing in the human visual system , Nature, 381:520–522, 1996.

[OAV96] B.A. Olshausen, C.H. Anderson, and D.C. Van Essen,emphA neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information, J. Neurosci., 13(11):4700–19, November 1993

[SalAbb97] E. Salinas and L. F. Abbott, emphInvariant visual responses from attentional gain fields, J. Neu- rophys., 77:3267–3272, 1997.

[Mon57] V.B. Mountcastle, emphModality and topographic properties of single neurons of cat's somatic sensory cortex,J. Neurophys., 20:408–434, 1957.

[Mon97] V.B. Mountcastle, emphThe columnar organization of the neocortex, Brain, 120(Part 4):701–22, 1997.

[PogGir90] T. Poggio and F. Girosi, emphNetworks for approximation and learning,Proc. IEEE, 78(9), 1990.

[LogShe96] N.K. Logothetis and D.L. Sheinberg, emphVisual object recognition, Ann. Rev. Neurosci., 19: 577–621, 1996.

[SWP05] T. Serre, L. Wolf, and T. Poggio,emphObject recognition with features inspired by visual cortex , In IEEE Computer Society Press, editor, Proc. IEEE Conf. on Comput. Vision and Pattern Recognit., San Diego, 2005b.

[FTIC92] I. Fujita, K. Tanaka, M. Ito, and K. Cheng,emphColumns for visual features of objects in monkey inferotemporal cortex, Nature, 360:343–346, 1992.

[DYH82] R.L. DeValois, E.W. Yund, and N. Hepler, emphThe orientation and direction selectivity of cells in macaque visual cortex, Vis. Res., 22:531–544, 1982b.

[SFV76] P. H. Schiller, B. L. Finlay, and S. F. Volman,emphQuantitative studies of single-cell properties in monkey striate cortex II. Orientation specificity and ocular dominance, J. Neurophysiol., 39(6):1334–51, 1976c.

[PasCon01] A. Pasupathy and C.E Connor,emphShape representation in area V4: Position-specific tuning for boundary conformation, J. Neurophys., 86(5):2505–2519, 2001.

[RCD99] J. H. Reynolds, L. Chelazzi, and R. Desimone, emphCompetitive mechanisms subserve attention in macaque areas V2 and V4,J. Neurosci., 19:1736–1753, 1999.

[SerRie04] T. Serre and M. Riesenhuber, emphRealistic modeling of simple and complex cell tuning in the HMAX model, and implications for invariant object recognition in cortex, AI Memo 2004-017 / CBCL Memo 239, MIT, Cambridge, MA, 2004.

[DLG96] L. Devroye, G. Laszlo, and G. Lugosi, emphA probabilistic theory of pattern recognition Springer- Verlag, New York, 1996.

[FFP04] L. Fei-Fei, R. Fergus, and P. Perona, emph Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories, In Proc. IEEE CVPR, Workshop on Generative-Model Based Vision, 2004.

[Bul01] J. Bullier,emphIntegrated model of visual processing, Brain Res. Rev., 36, 2001.

[HKPD05] C. Hung, G. Kreiman, T. Poggio, and J. DiCarlo,emphFast read-out of object identity from macaque inferior temporal cortex, Science, 310:863–866, November 2005.

[KXFP01] C. Keysers, D. K. Xiao, P. Foldi´ k, and D. I. Perrett,emphThe speed of sight, J. Cogn. Neurosci.,¨ a 13:90–101, 2001.

[BacMac05] N. Bacon-Mace, M.J. Mace, M. Fabre-Thorpe, and S.J. Thorpe,emphThe time course of visual processing: backward masking and natural scene categorisation, Vis. Res., 45:1459–1469, 2005.

[MacCre91] N. A. Macmillan and C. D. Creelman,emphDetection Theory: A User's Guide, Cambridge Uni- versity Press, 1991.

[GKT05] R. Guyonneau, H. Kirchner, and S.J. Thorpe,emphAnimals roll around the clock: The rotation invariance of ultra-rapid visual processing, Eur. Conf. on Visual Proc., 2005.

[LPBP95] Logothetis, N. K., J. Pauls, H. H. Bülthoff and T. Poggio: emphView-dependent object recognition by monkeys, Current Biology 4, 401-414 (1994)

[DRF00] A. Delorme, G. Richard, and M. Fabre-Thorpe. emphUltra-rapid categorisation of natural im- ages does not rely on colour: A study in monkeys and humans Vis. Res., 40:2187–2200, 2000.