Università degli Studi di Padova

Facoltà di Ingegneria

Corso di laurea in ingegneria gestionale

Dipartimento di tecnica e gestione dei sistemi industriali

Tesi di laurea triennale

MODELLAZIONE DI DIAGRAMMI CCT: SVILUPPO DI UN SISTEMA A RETI NEURALI

Relatore: Ch.ma Dott.ssa REGGIANI MONICA

Correlatore: Ch.mo Ing. PAOLO FERRO

Laureando: VANESSA ZAMPA

ANNO ACCADEMICO 2010/2011

INDICE

SOMMARIO

Nel primo capitolo è stata descritta la problematica da cui prende spunto questo lavoro, cioè la necessità di prevedere, per ogni tipo di acciaio, il campo ferritico dello specifico diagramma CCT; nel secondo sono stati analizzati i vari metodi che cercano di ovviare a questa problematica, dimostrando come le reti neurali siano il metodo più promettente; nel terzo viene sviluppato il sistema a reti neurali, quindi viene fornito il programma di compilazione della rete neurale.

Lo sviluppo di un sistema a reti neurali servirà per l'apprendimento e la previsione dei valori di inizio e fine precipitazione della ferrite da parte della rete neurale e si dovrà successivamente verificare se queste previsioni si avvicinano ai valori desiderati nel caso di acciai aventi caratteristiche note alla rete neurale, ma soprattutto si vuole testarne la capacità previsionale per acciai con valori ferritici sconosciuti.

INTRODUZIONE

Questa tesi si colloca all'interno di un lavoro di gruppo avente lo scopo di sviluppare un progetto di rete neurale in grado di prevedere, per ogni tipo di acciaio, il campo ferritico dello specifico diagramma CCT (Continuous Cooling Transformation).

Il lavoro sviluppato in queste pagine, dunque, non tratterà l'intero problema, ma si focalizzerà sulla seconda fase di tale progetto, ovvero sullo sviluppo di un sistema a reti neurali.

Lo sviluppo dell'intero progetto è molto importante dal punto di vista ingegneristico. Una conoscenza delle curve CCT è infatti fondamentale per capire come evolve la struttura degli acciai in funzione del tempo e della temperatura in corrispondenza di trasformazioni a velocità di raffreddamento costanti.

Si capisce dunque come il corretto impiego di tali curve sia molto rilevante nei trattamenti termici. Esse vengono infatti utilizzate per poter stabilire quale temperatura, quale tempo e quale velocità di raffreddamento devono essere scelte per ottenere le caratteristiche di durezza, tenacità, microstruttura e lavorabilità desiderate.

Purtroppo però, la localizzazione e le forme di tali curve sono altamente sensibili alla composizione chimica dell'acciaio. Ne consegue che, eventuali fluttuazioni di quest'ultima, anche all'interno della stessa classe di acciaio, fanno si che il diagramma CCT estrapolato dalla letteratura non fornisca informazioni affidabili sulle trasformazioni dell'austenite, ossia passaggio di fase di riferimento per la costruzione di tale diagramma. Ciò che si vorrebbe, dunque, sarebbe un sistema computerizzato che fosse in grado di prevedere le curve CCT al variare delle composizioni chimiche degli acciai.

A tale proposito, sono stati nel seguito brevemente trattati i vari metodi, sviluppati da diversi studiosi, che hanno cercato di prevedere tali diagrammi nel modo più efficiente possibile.

Tra questi, quello che sembra essere più promettente è proprio il metodo basato sulle reti neurali. Quest'ultime rappresentano un'efficace soluzione ai problemi sopra descritti poiché esse, tramite la ricerca di un modello appropriato, consentono proprio la simulazione al computer dell'effetto della composizione chimica sulle proprietà richieste o l'ottimizzazione delle soluzioni realizzabili basandosi sui criteri assunti.

Esse sono impiegate in diversi campi e la loro popolarità risulta dalla fattibilità di rappresentare le relazioni che intercorrono tra le varie quantità di dati indagati senza conoscere il modello fisico del fenomeno descritto.

Perché ciò sia possibile, è necessario raccogliere un elevato quantitativo di dati, necessari per la fase di apprendimento delle reti.

Viene, poi, presentato e descritto il codice di programmazione per la compilazione della rete neurale.

È proprio su questa fase di sviluppo di un sistema a reti neurali che si è focalizzato questo lavoro di tesi.

Nelle seguenti pagine, infatti, è stata riportata la codifica delle istruzioni fornite al programma per l'addestramento della rete neurale e la successiva fase di previsione da parte di quest'ultima dei valori di interesse, cioè quelli di inizio e fine precipitazione della ferrite. La fase di previsione viene fatta prima per valori noti alla rete neurale e poi per valori ad essa sconosciuti attraverso la cross-evaluation.

CAPITOLO 1

Problematica

1.1.L'importanza dell'acciaio

L'oggetto di analisi di tale progetto è l'acciaio che gode di un vasto impiego nel campo ingegneristico.
Si definisce acciaio una lega composta principalmente da ferro e carbonio, quest'ultimo in percentuale non superiore al 2,11% poiché, oltre tale limite, le proprietà del materiale cambiano e la lega assume la denominazione di ghisa.
Si può oggi affermare che esso rappresenti la risorsa più comunemente usata nel settore dell'ingegneria, nonostante il progresso nello sviluppo dei materiali con proprietà speciali, come ad esempio la ceramica o i polimeri.
Il vantaggio dell'acciaio rispetto agli altri materiali ingegneristici, si trova in relazione alle elevate caratteristiche meccaniche che si ottengono con un grande numero di processi produttivi, di formatura e di trattamento termico.
Data la sua importanza pratica, dunque, esso è sicuramente la lega su cui più è stato fatto la sforzo per conoscerne il comportamento nei vari tipi di lavorazione, in modo da ottimizzarne la resa nei processi con cui dai processi siderurgici si passa all'oggetto finito.
Ogni qualità è stata perciò sottoposta a ricerche per garantire la lavorabilità all'utensile, la formabilità a caldo e a freddo, l'attitudine a dare getti di buona qualità, la saldabilità, la risposta al trattamento termico e così via.
Questo ha significato ottimizzare le strutture in funzione del tipo di lavorazione richiesto, a parità di composizione, oppure a parità di proprietà finali, adattare quest'ultima al tipo di lavorazione più critico nel ciclo di trasformazione.
Nota quindi la qualità dell'acciaio, è possibile attingere ad una mole spesso imponente di dati di ogni genere che lo riguardano e che servono alle industrie per impostare i processi di lavorazione.
L'importanza dell'acciaio deriva anche dal fatto che esso è la lega che copre il più vasto spettro di proprietà tra tutte quelle che hanno applicazione industriale.
Alcune di esse variano in misura relativamente modesta come la densità (intorno a 7,85 g/cm^3), il modulo elastico (tra 175 e 210 kN/mm^2) e tante altre di minore importanza.

Altre proprietà presentano invece un campo di variabilità eccezionalmente vasto, giustificato dal polimorfismo del ferro e dall'ampia possibilità di controllare le prestazioni fisico-chimiche della struttura mediante opportune alligazioni.
La resistenza meccanica (Rm) risente fortemente della condizione strutturale: allo stato ricotto o normalizzato Rm varia tra 200 e 1000 N/mm^2, per salire oltre 2000 allo stato incrudito (negli acciai inossidabili austenitici) e oltre 3500 allo stato temprato e rinvenuto (negli acciai maranging).
Le proprietà plastiche (allungamento, strizione) variano in senso inverso rispetto alla resistenza meccanica: l'allungamento raggiunge valori intorno al 35% per gli acciai da imbutitura per ridursi a entità trascurabili negli acciai a tutta tempra e la strizione raggiunge il 50-60% negli acciai inossidabili austenitici per ridursi anch'essa con l'indurimento.
La massima resistenza a fatica viene raggiunta negli acciai per molle (sono ammissibili in certe condizioni sollecitazioni di 1000-1200 N/mm^2) e negli acciai cementati e temprati per organi meccanici (oltre 850 N/mm^2).
La durezza va da valori modesti per gli acciai ricotti (si parte da 70-80 HV, durezza Vickers) per salire negli acciai temprati da utensili fino a 1100 HV; la resistenza all'usura abrasiva è direttamente legata alla durezza e va di pari passo con quest'ultima.
La resistenza alla corrosione dell'acciaio comune è assai modesta e richiede in genere delle protezioni aggiuntive (tipica la verniciatura); nel caso degli acciai inossidabili è possibile realizzare strutture resistenti in modo accettabile ad un gran numero di aggressivi chimici.

1.2.Le curve CCT

Come è stato precedentemente detto, è mediante degli opportuni trattamenti termici che è possibile far assumere alla lega quelle strutture cristalline che gli conferiscono determinate caratteristiche meccaniche e tecnologiche.
Un trattamento termico rappresenta un ciclo termico di riscaldamento effettuato in predeterminate condizioni e temperature, a cui devono seguire raffreddamenti, più o meno lenti.
Per comprendere l'effetto che il trattamento termico ha sulla struttura della lega è necessario conoscere il diagramma di stato della stessa.
Ma tale conoscenza non è tuttavia sufficiente poiché i diagrammi di stato definiscono le varie strutture di equilibrio della lega, a una determinata temperatura.

Ovvero, tale diagramma è rigorosamente applicabile solo a condizione che il raffreddamento della lega sia sufficientemente lento e non si occupa dunque della variabile tempo.

Esso, infatti, non tiene conto della velocità di raffreddamento e di quella di riscaldamento, ne' della durata di permanenza alle varie temperature.

Questi aspetti devono però essere considerati nell'applicazione di un trattamento termico.

Durante il riscaldamento o raffreddamento, infatti, si hanno delle variazioni della microstruttura. Questi fenomeni implicano la diffusione e quindi l'effetto non solo della temperatura ma anche del tempo.

A tale proposito è quindi necessario integrare le conoscenze acquisite con quelle relative all'influenza della velocità di raffreddamento/riscaldamento sulle trasformazioni indicate sul diagramma di stato, e quindi considerare l'effetto della cinetica.

Ciò risulta fondamentale poiché la velocità non influisce solo sulle temperature di transizione (che in genere saranno diverse da quelle ricavate dai diagrammi di stato), ma anche sulla natura stessa della struttura ottenuta, con la possibilità di ottenere componenti metastabili, come ad esempio la martensite, assenti nel diagramma di stato.

Bain fu il primo che cercò di considerare questo aspetto [1]. I suoi studi consistevano nel considerare inizialmente un campione di un acciaio eutettoidico, riscaldarlo sopra i 723°C così da farlo passare allo stato di austenite (una soluzione solida primaria di tipo interstiziale di carbonio nel ferro gamma che presenta un reticolo cubico a facce centrate o "CFC"), raffreddarlo successivamente al di sotto dei 723°C e lasciarlo lì per un certo tempo. Il campione veniva poi raffreddato velocemente in acqua. Bain ha eseguito tale procedimento per vari sottoraffreddamenti.

Lo studio così fatto permetteva di vedere quanta austenite si era trasformata in perlite, bainite, martensite o ferrite e quindi analizzare la microstruttura ottenuta. Si ottiene così la curva descritta in Fig. 2, chiamata anche curva di Bain o curva TTT (Temperatura – Tempo - Trasformazione).

Nei trattamenti termici più diffusi e di maggior interesse applicativo però, la trasformazione dell'austenite non avviene a temperatura costante, ma nel corso di un raffreddamento continuo.

Pertanto, nel seguire o prevedere le trasformazioni di fase, è necessario fare riferimento alle curve di trasformazione anisotermica dell'austenite, cioè alle curve CCT (Continuous – Cooling – Transformation).

Tali curve si ottengono segnando su ogni traiettoria di raffreddamento i punti di inizio e fine trasformazione dell'austenite. Si ottiene così la curva mostrata in Fig. 1.

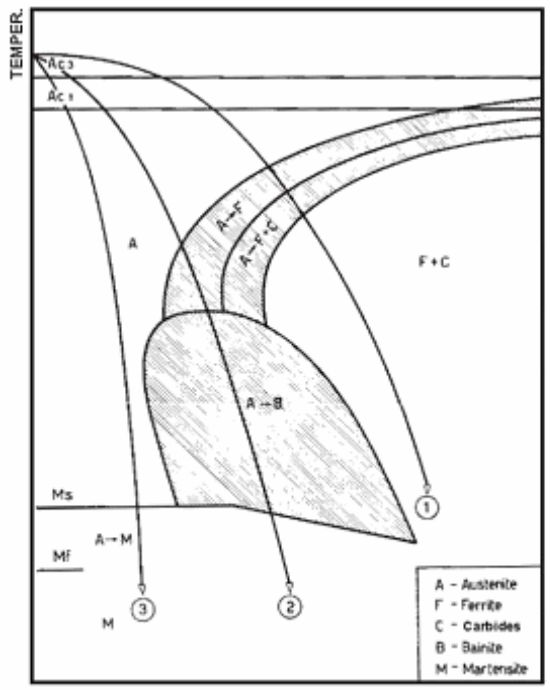

Fig. 1

Curva CCT

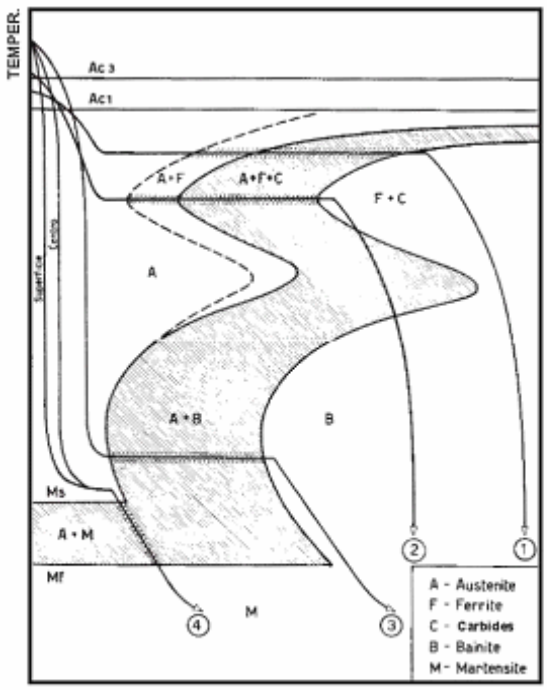

Fig. 2

Curva TTT

1.3.Cenni sui principali trattamenti termici

Ogni traiettoria di raffreddamento presente nelle curve CCT, rappresenta un particolare trattamento termico. I principali trattamenti a cui può essere sottoposto un metallo o una lega, al fine di ottenere determinate caratteristiche, sono:

- Ricottura: consiste nel riscaldamento di un acciaio al di sopra della temperatura di trasformazione di fase A_3 e ad un successivo raffreddamento lento, solitamente in forno, sufficiente per determinare nella massa metallica le trasformazioni volute. Questo trattamento termico altera la microstruttura del materiale, causando mutamenti nelle sue proprietà quali la flessibilità e la durezza. Il risultato tipico è la rimozione dei difetti della struttura cristallina, l'addolcimento del materiale e l'abbassamento della sua durezza. Al variare della temperatura di riscaldamento, della velocità di raffreddamento o dello scopo per il quale si effettua tale trattamento si hanno diversi tipi di ricottura: la ricottura completa, isotermica, incompleta, di addolcimento o rinvenimento ad alta temperatura, di omogeneizzazione o di diffusione.

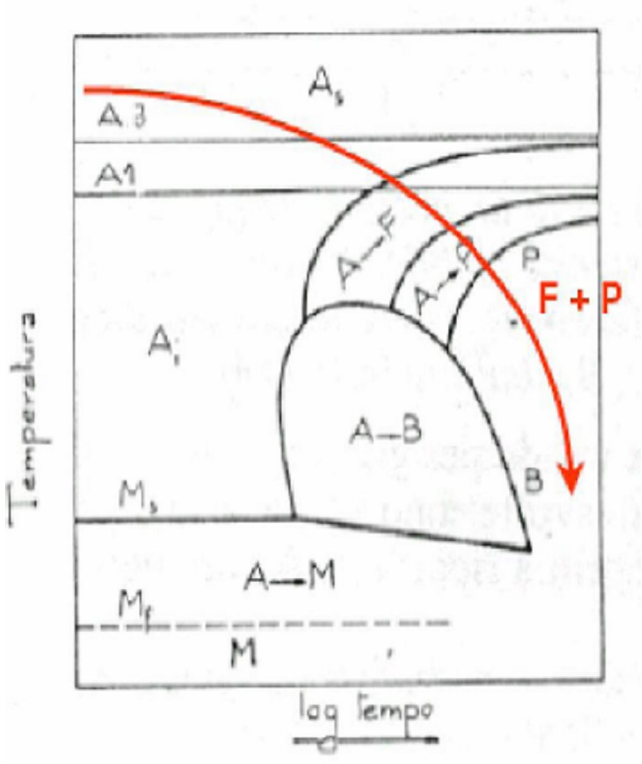

Fig. 3 Esempio velocità di raffreddamento per la ricottura in una curva CCT

- Distensione: consiste in un riscaldamento ad una temperatura notevolmente al di sotto di A_1 e permanenza a tale temperatura per un tempo conveniente. Questo trattamento riduce le tensioni interne senza alterare significativamente la durezza.

- Normalizzazione: consiste nel riscaldamento del materiale a una temperatura poco superiore ad A_3, nella permanenza a tale temperatura per un certo tempo, e nel successivo raffreddamento in aria calma. Tale processo è simile alla ricottura, ma in questo caso il raffreddamento è più rapido. A causa di questo raffreddamento più veloce, si formano dei cristalli più minuti. Si ottiene così una maggiore durezza rispetto ad un acciaio ricotto e un miglioramento della duttilità, grazie all'affinazione del grano.

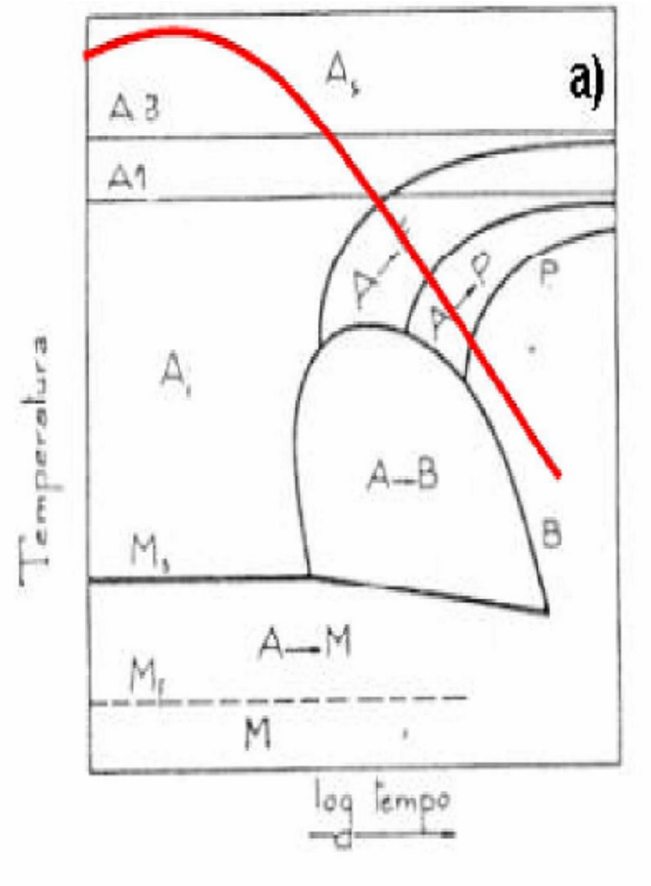

Fig. 4 Esempio velocità di raffreddamento per la normalizzazione in una curva CCT

- Tempra: consiste nel brusco raffreddamento fino a temperatura ambiente di un materiale, dopo averlo portato ad alta temperatura. L'elevata velocità di raffreddamento inibisce i processi diffusivi necessari alla stabilizzazione termodinamica, consentendo così l'immediata trasformazione in martensite. Un monocristallo così trattato ha resistenza meccanica maggiore rispetto al monocristallo raffreddato lentamente. In relazione al tipo di acciaio e alle dimensioni dei pezzi da temprare, verrà scelto il mezzo di spegnimento più adatto: acqua, olio o aria. Le modalità di raffreddamento differenziano i vari tipi di tempra, che sono: tempra diretta, interrotta, scalare martensitica, isoterma bainitica.

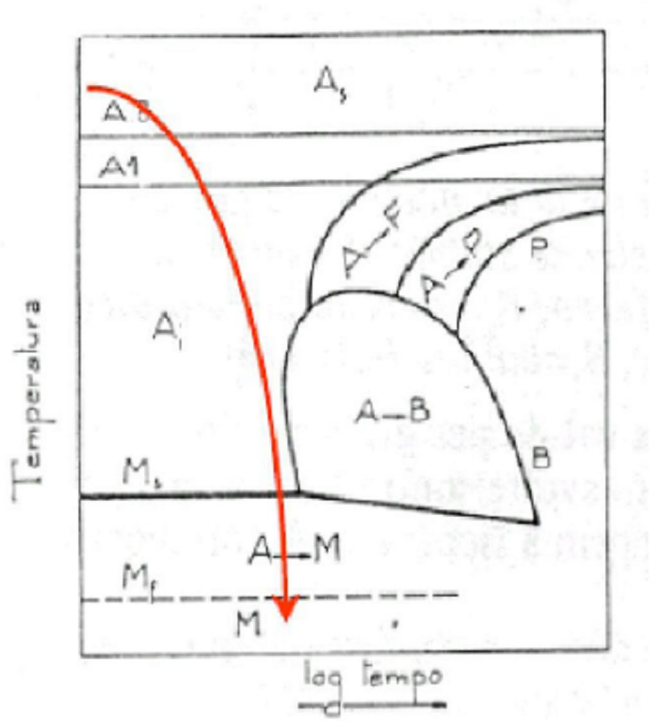

Fig. 5 Esempio velocità di raffreddamento per la tempra in una curva CCT

- Rinvenimento: esso comprende un riscaldamento ad una temperatura inferiore ad A_1, un mantenimento per un certo tempo a questa temperatura ed infine un raffreddamento in un mezzo appropriato fino a temperatura ambiente. Tale trattamento viene eseguito subito dopo la tempra poiché allo stato temprato l'acciaio presenta un'elevata durezza e basse caratteristiche di tenacità. È necessario quindi ricorrere ad un successivo trattamento che ne modifichi, più o meno profondamente, la struttura martensitica di tempra annullandone le tensioni e la fragilità. La temperatura di rinvenimento va scelta in modo da ottenere il miglior compromesso tra le caratteristiche di durezza e di tenacità.

1.4.Effetto degli elementi in lega

Oltre a presentare dunque, come detto prima, delle buone proprietà meccaniche, un'ulteriore caratteristica dell'acciaio è la presenza di altri elementi in lega, oltre al carbonio. Essi possono essere naturalmente presenti nell'acciaio, oppure aggiunti in esso per conferire caratteristiche specifiche, determinando così alcune modifiche delle proprietà chimico/fisiche del materiale: il fosforo ad esempio e lo zolfo riducono la tenacità dell'acciaio, il molibdeno aumenta la temprabilità e la resistenza a caldo, il cromo aumenta anch'esso la temprabilità ed anche la resistenza all'usura, e così via per altri elementi.

Tali elementi però, influenzano le curve TTT e CCT. Quest'ultimi, infatti, modificano sia la forma, sia la posizione di tali curve. In particolare:

- tutti gli elementi in lega, ad eccezione del cobalto, provocano degli spostamenti delle curve di inizio e di fine trasformazione verso tempi più lunghi, quindi verso destra, rispetto agli acciai al solo carbonio.
- inoltre, tutti gli elementi in lega tranne il cobalto spostano verso il basso la temperatura M_s e M_f; ciò può causare la presenza di austenite alla temperatura ambiente.

1.5.Le problematiche delle curve CCT

Questa rappresenta dunque la problematica da cui nasce lo scopo di questo lavoro.
Se infatti da uno stesso acciaio, tramite l'immersione in bagni diversi, si ottengono acciai con diverse composizioni chimiche e i diagrammi CCT risultano essere sensibili a ciò, ne consegue che è impossibile produrre sufficienti diagrammi per l'uso generalizzato.
Ovvero, i diagrammi presenti in letteratura non riescono a fornire delle adeguate informazioni per ogni tipo di acciaio. Poiché però, la conoscenza di tali diagrammi risulta essere fondamentale per l'applicazione dell'adeguato trattamento termico, non essendo disponibili in letteratura, questi si dovrebbero ricavare sperimentalmente. Tale operazione però risulta essere molto dispendiosa in termini di tempo e di denaro. Inoltre, non riuscirebbe a soddisfare la sempre più crescente domanda di prodotti, non essendo un metodo in grado di ottimizzare la selezione di un vasto insieme di materiali.
Per risolvere tale problema, sarebbe infatti auspicabile poter avere un modello che, una volta data la specifica composizione chimica, fosse in grado di prevedere la corrispondente curva CCT.
Una soluzione di questo tipo infatti, consentirebbe così di ovviare a tutti i problemi sopra descritti, ovvero, permetterebbe di fornire dati affidabili, ad un costo contenuto e in un tempo ragionevole.
È da qui dunque che questo lavoro prende spunto, poiché è volto a trovare il modello che riesca ad ottimizzare tale previsione, nel modo più immediato ed efficiente possibile.

CAPITOLO 2

Stato dell'arte

Come è stato precedentemente detto, i diagrammi CCT sono molto importanti per la determinazione della struttura del materiale, dopo un trattamento termico. A tale proposito, sono stati intrapresi molti studi per cercare il metodo più efficiente per la previsione di tali diagrammi, necessari per la produzione industriale. Vengono qui sotto riportati i modelli più significativi sviluppati finora.

2.1.Modelli empirici

Un primo metodo possibile è quello di ricavare tale diagramma sperimentalmente, ovvero tracciarlo usando un test di dilatometro su un simulatore caldo ed un'analisi metallografica [2,3].
Tale metodo, operativamente, consiste nel lavorare opportunamente i campioni fino ad ottenere le specifiche desiderate: ad esempio, modellarli fino a raggiungere un diametro di 8 mm ed una lunghezza di 12 mm. I campioni così ottenuti, poi, vanno riscaldati fino ad una stabilita temperatura, con una certa velocità e mantenuti successivamente a tale temperatura per un dato tempo. Successivamente, essi vengono raffreddati fino ad una precisa temperatura, con diverse velocità. Durante i vari processi di raffreddamento, vengono registrate sul simulatore, le dilatazioni dei campioni da parte del dilatometro. Quando tutti i campioni sono stati presi in esame, per ognuno di essi vengono analizzate le microstrutture, ottenute tramite un microscopio ottico, al fine di determinare la loro composizione di fase.
Ciò che si ottiene tramite un'analisi con il dilatometro, è mostrato nella figura sottostante (Fig. 1). Ogni punto di flesso mostra la corrispondente temperatura a cui avviene una trasformazione di fase.
Da tale figura è evidente però anche qual è la problematica di tale metodo. Si vede, infatti, come nella curva di dilatazione mostrata in Fig. 1(a), sia difficile determinare tutte le temperature di trasformazione, poiché il punto di flesso "b" non risulta molto chiaro.
In questo caso dunque, tale metodo non garantisce di tracciare accuratamente il diagramma CCT.

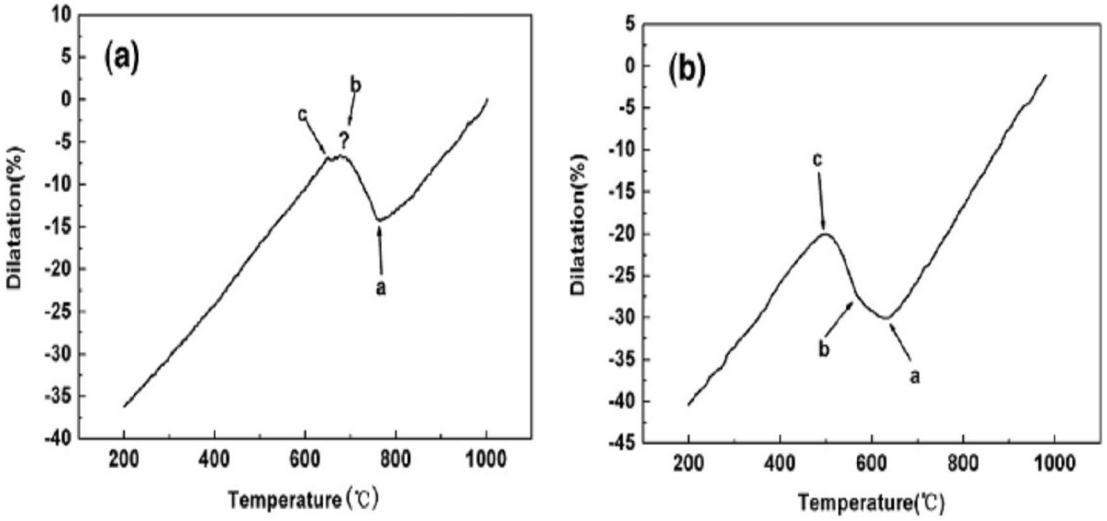

Fig. 1 Curve di dilatazione. (a) Campione 1, raffreddato alla velocità di 0.1 °C/s che mostra un vertice non chiaro; (b) Campione 8, raffreddato alla velocità di 20 °C/s che mostra chiari vertici.

Viene poi qui sotto mostrata l'analisi della microstruttura dei due campioni presi in esame, che è utile per decifrare le varie composizioni di fase, ottenute con il raffreddamento.

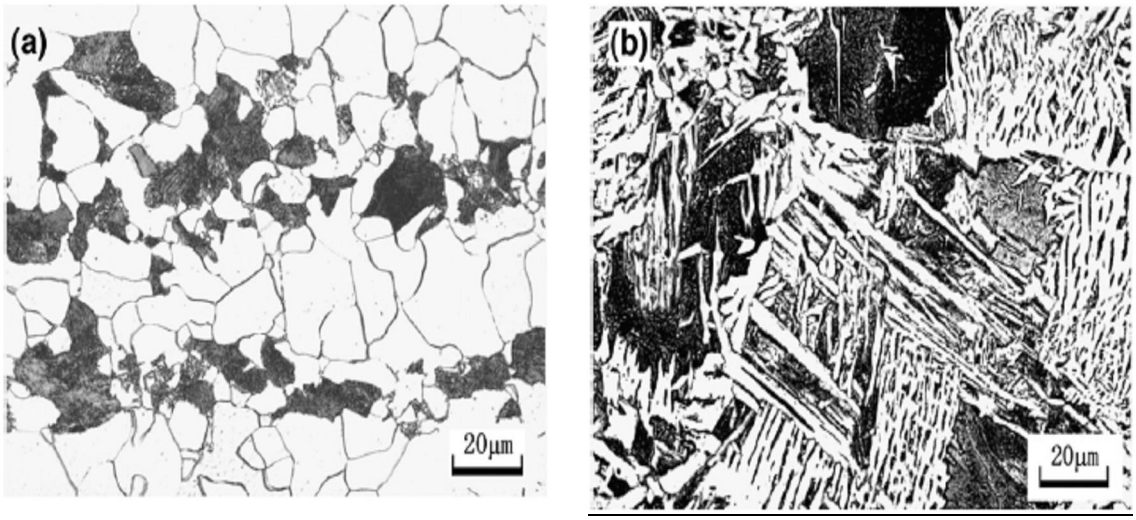

Fig. 2 (a) Metallografia del campione 1 che mostra una microstruttura costituita da ferrite e perlite; (b) metallografia del campione 8 che mostra una microstruttura costituita da perlite e struttura Widmanstatten.

La raccolta di tutti questi dati permette così di ottenere una tabella che sia in grado di riassumere, per ogni campione e per ogni velocità di raffreddamento, le temperature nelle quali si ottengono determinate microstrutture. Una tabella così fatta è rappresentata qui sotto.

No.	Heating rate (°C/s)	Cooling rate (°C/s)	Microstructure	Transformation temperature of A→F		Transformation temperature of A→P	
				Initial temperature	Finishing temperature	Initial temperature	Finishing temperature
1	5.0	0.1	F+P(banding distribution)	775	663	663	645
2	5.0	0.2	F+P(banding distribution)	767	651	651	638
3	5.0	0.5	F+P	762	645	645	628
4	5.0	1.0	F+P+W	751	642	642	617
5	5.0	2.0	W+F (reticular)+ P(small quantity)	737	632	632	595
6	5.0	5.0	W+F+P	717	614	614	582
7	5.0	10.0	W+F+P	681	577	577	535
8	5.0	20.0	W+P	630	565	565	489

Tabella 1 Microstruttura, tipi di trasformazioni e temperature per tutti i campioni nel test.

Questo metodo, però, poiché non è in grado di tracciare accuratamente i diagrammi CCT per certi tipi di acciai ed anche perché risulta essere un metodo molto dispendioso di tempo e costoso, non può essere preso di riferimento per un utilizzo su ampia scala.

2.2.Modelli analitici

Altri tentativi includono l'utilizzo di modelli matematici dei processi che avvengono nell'acciaio durante il raffreddamento o dipendenze empiriche sviluppate dopo molti esperimenti.

Questi tentativi, consentono di ottenere le curve CCT, ricavandole dalle curve TTT.

Per fare ciò si fa ricorso alla regola dell'additività di Scheil [4]. L'assunzione di tale legge è che una generica storia termica possa essere descritta come combinazione di un numero sufficientemente ampio di passi di reazioni isotermiche. Tale legge è espressa nella seguente formula:

$$\int_0^{t_x} \frac{dt}{\tau_x(T)} = 1$$

Dove $\tau_x(T)$ è il tempo impiegato per la frazione di volume trasformato per raggiungere *x* sotto una temperatura isotermica T, e t_x il tempo per raggiungere *x* sotto raffreddamento continuo.

Graficamente, tale regola è mostrata nella figura sottostante (Fig. 3). In quest'ultima si vede che il range di temperatura è diviso in una serie di piccoli finiti passi. Mantenere l'intervallo di tempo Δt_i sufficientemente breve, garantisce che le condizioni siano isoterme su ogni intervallo di tempo. Si è supposto che in ogni passo temporale, venga prodotta una tale trasformazione che si verifica come nel diagramma isotermo, alla stessa temperatura.

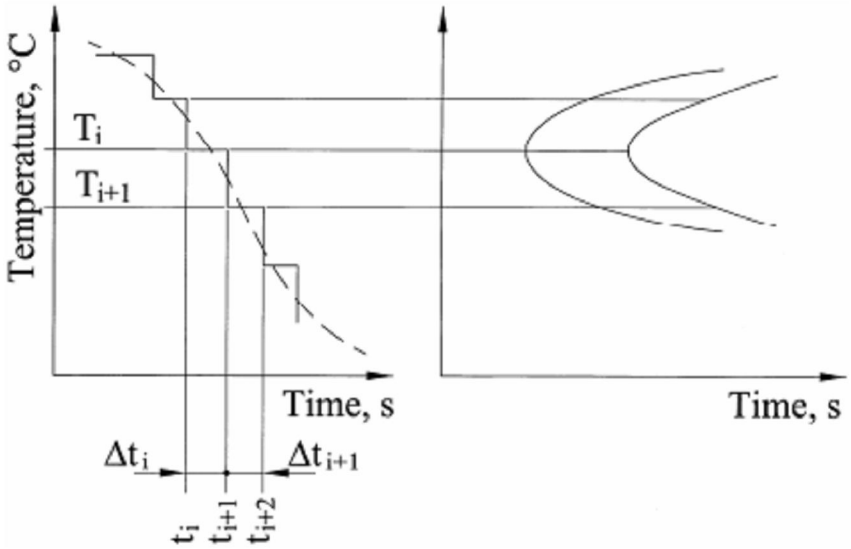

Fig. 3 Predizione della microstruttura da una curva di raffreddamento e diagramma TTT

Fa ricorso a tale regola, la legge di Johnson-Mehl-Avrami-Kolmogorov [5], che rappresenta il modello più comunemente usato per descrivere le trasformazioni cinetiche di tipo diffusivo. Tale legge definisce la frazione di volume Φ del costituente prodotto al tempo t, per una trasformazione isotermica, alla temperatura T, nel seguente modo:

$$\phi(t) = 1 - e^{-kt^n} \tag{1}$$

Dove *k* ed *n* sono parametri empirici che dipendono dalla temperatura T, tenuta durante tutta la trasformazione.

Per quanto riguarda invece le trasformazioni "martensitiche", ovvero prive di diffusione, si impiegano leggi simili a quella proposta da Koistinen-Marburger [6].

Avrami, con la sua legge, pose la prima limitazione alla regola dell'additività poiché dimostrò la sua validità solo quando la velocità di nucleazione è proporzionale alla velocità di crescita, ovvero solo in una condizione "isocinetica".

Matematicamente, ciò si traduce nel fatto che il fattore *n* dell'equazione (1) non dipenda dalla temperatura. Se questa condizione non viene verificata, dunque, la regola dell'additività non è più valida in un senso strettamente matematico.
A tale proposito, la valida applicazione di tale legge è stata successivamente estesa da Cahn [4], integrando anche la velocità nella relazione, nella seguente forma:

$$\dot{\phi} = \hat{j}(\theta)\,\hat{h}(\phi) \tag{2}$$

Dove il punto indica una derivata nel tempo e θ rappresenta la temperatura. Questa relazione implica una condizione di "generale isocinetica".
Il risultato raggiunto da Cahn permette di predire con accuratezza una vasta gamma di reazioni non isotermiche utilizzando la legge dell'additività, ampliando così le condizioni di applicazione della stessa.
Anche se l'argomentazione di Cahn è valida per quanto riguarda il rispetto del ruolo dell'additività nelle condizioni di generale isocinetica, egli ha inoltre dichiarato che la regola vale per una classe ancora più ampia di relazioni cinetiche. In particolare, Cahn affermò che la regola dell'additività può essere applicata ad ogni equazione cinetica che si presenti nella seguente forma:

$$\dot{\phi} = g(\phi, \theta) \tag{3}$$

Tuttavia, nonostante la riuscita applicazione della regola di additività negli ultimi quarant'anni, è stato dimostrato che tale equazione non può essere applicata a tutte le relazioni cinetiche che si presentano in quella forma, chiamate anche relazioni indipendenti dalla velocità. Tramite degli esempi analitici, si è potuto quindi verificare la non veridicità dell'ultima conclusione a cui era arrivato Cahn, ovvero che tutte le reazioni indipendenti dalla velocità seguissero la regola dell'additività.
Molti studiosi dunque, cercarono dei modelli alternativi, proprio per cercare di ovviare alle limitazioni di tale regola.
Tra questi si ha il ricorso a modelli analitici [7], che siano sempre in accordo con la regola dell'additività, oppure l'utilizzo, grazie allo studio degli studiosi Fried e Gurtin, di un principio di micro-bilanciamento [5,8], che porta a una relazione differenziale cinetica.
Qualsiasi tentativo di tipo matematico però risulta applicabile solo in certe condizioni, e non è in grado di soddisfare appieno gli obiettivi desiderati.

2.3.JMatPro

Nonostante i modelli descritti nel paragrafo precedente dimostrano che è possibile calcolare i diagrammi TTT e CCT con una certa accuratezza, si è visto però che ciò è vero solo per acciai basso legati. Questi modelli dunque risultano essere di successo per quest'ultimi, ma limitativi per gli acciai alto legati. Questo software nasce dunque con lo scopo di ampliare la gamma di acciai su cui poter applicare dei metodi predittivi [9].

Quest'ultimo, acronimo di Java-based Material Properties, è un software che può fornire molte delle proprietà del materiale che vengono richieste dai processi di simulazione.

Esso può essere considerato un ulteriore tipo di modello analitico, poiché utilizza il modello di Kirkaldy. Quest'ultimo è stato scelto come base per i nuovi calcoli, in quanto vi è un insieme di parametri di input richiesti, chiaramente identificabili e che possono essere facilmente calcolati. Esso presenta anche parametri empirici che possono essere sistemati facilmente e sono controllabili.

Come detto prima, dunque, il successo di tale modello è dovuto al fatto che esso è in grado di predire le trasformazioni di fase per diversi tipi di acciai, inclusi sia quelli medio che alto legati. È in grado inoltre di fornire un'accurata descrizione di tutte le principali trasformazioni di fase che si hanno, così come il calcolo delle proprietà delle diverse fasi che si formano durante i processi di trattamento termico.

Oltre poi a calcolare le curve TTT e CCT, tale modello è in grado di calcolare una vasta gamma di proprietà fisiche, termo-fisiche, tra cui la forza di snervamento e quella di deformazione.

Tale pacchetto software consente dunque la modellazione di complesse leghe commerciali e le loro proprietà caratteristiche. Esso è in grado di calcolare il diagramma di fase degli acciai, utilizzando delle equazioni matematiche. Da questo se ne deduce che questo metodo, anche se migliore dei metodi descritti precedentemente, presenta ancora delle limitazioni.

Le leggi su cui si basa, infatti, sono applicabili anch'esse solo sotto determinate condizioni e, talvolta, i risultati che si ottengono si discostano abbastanza dai risultati sperimentali.

Anche questo metodo, dunque, non risulta essere adatto ad essere impiegato per un uso su larga scala.

2.4.Il modello delle reti neurali

Dai paragrafi precedenti si è visto come i metodi sviluppati finora, al fine di prevedere le curve di raffreddamento, presentino molte limitazioni nell'applicazione: i metodi sperimentali sono molto costosi in termini di tempo e di denaro e non sempre sono in grado di tracciare correttamente le curve; le equazioni su cui si basano i modelli analitici sono applicabili solo in certe condizioni e quindi sono valide solo per un piccolo gruppo di acciai; l'utilizzo di JMatPro presenta molte volte dei risultati molto diversi dai reali valori degli stessi parametri ed inoltre anch'esso non si può applicare ad ogni tipo di acciaio perché si basa su leggi matematiche che presentano le stesse problematiche dei metodi analitici.

Tutti questi motivi rendono dunque tali modelli non adatti ad un utilizzo su ampia scala. Poiché però questo rappresenta proprio il nostro obiettivo, la nostra analisi non ha considerato i metodi descritti prima, ma si è focalizzata sull'utilizzo delle reti neurali.

Questo metodo si è molto sviluppato negli ultimi decenni ed ha avuto largo impiego anche in diversi campi. Nella metallurgia, sembra essere il modo più promettente per descrivere i complicati processi di trasformazione dell'acciaio durante un trattamento termico, proprio perché è in grado di rappresentare le relazioni tra le quantità di dati indagati senza conoscere il modello fisico del fenomeno descritto [10, 11, 12]. I risultati forniti dalla rete neurale molto spesso presentano una maggiore compatibilità con i dati empirici che con i risultati ottenuti grazie alle interrelazioni empiriche o ai modelli matematici dei processi analizzati. Questo metodo infatti imita la funzione del cervello umano che è in grado di imparare dagli esperimenti, di associare, predire o prendere decisioni razionali. È proprio da ciò che deriva il nome "neurale", ovvero in grado di simulare la funzione dei neuroni presenti nel cervello.

Allo stato attuale, infatti, a differenza delle macchine, l'uomo è un ottimo esempio di "sistema" in grado di elaborare informazioni. Tali elaborazioni, come ogni processo cognitivo, hanno sede nel cervello, una complessa struttura neurobiologica, attualmente decifrata in modo piuttosto accurato per quanto riguarda gli aspetti anatomici.

Tale struttura è composta da una cellula denominata neurone, che costituisce un "mattone elementare" e caratterizza tutte le strutture cerebrali.

Nel cervello umano sono presenti tipicamente oltre 100 miliardi di neuroni, ciascuno interconnesso a circa altri 10000. E' opinione condivisa da molti ricercatori che proprio tali interconnessioni siano alla base dell'elaborazione dell'informazione a livello cerebrale; diversi compiti cognitivi, infatti, sono caratterizzati da pattern di interconnessioni neuronali diversi.

Noto dunque ciò, ci si è chiesti se dal comportamento dei neuroni all'interno della struttura cerebrale, si potessero trarre degli utili suggerimenti e ispirazioni per la costruzione di macchine in grado di replicare compiti connotati da una forte componente di elaborazione, attualmente difficilmente realizzabili negli attuali calcolatori.

Ecco dunque che per i modelli artificiali è stata seguita una metafora simile a quella del cervello umano: sono stati studiati diversi tipi di neuroni e diverse architetture associandovi le modalità di elaborazione concepite per implementare un determinato compito cognitivo.

Fu così che vennero realizzati i primi neuroni artificiali e questi vennero poi sviluppati al fine di renderli in grado di soddisfare sempre più esigenze reali.

Al giorno d'oggi vengono utilizzati vari criteri per catalogare e raggruppare diversi tipi di reti. In genere, si possono distinguere in due grandi famiglie: reti feed-back e reti feed-forward.

Nelle reti di tipo feed-back i neuroni di uno strato possono ricevere il loro input da qualsiasi neurone che faccia parte dello stesso strato oppure che appartenga ad uno degli strati seguenti o precedenti; come input, il neurone, potrebbe usare addirittura il suo stesso output.

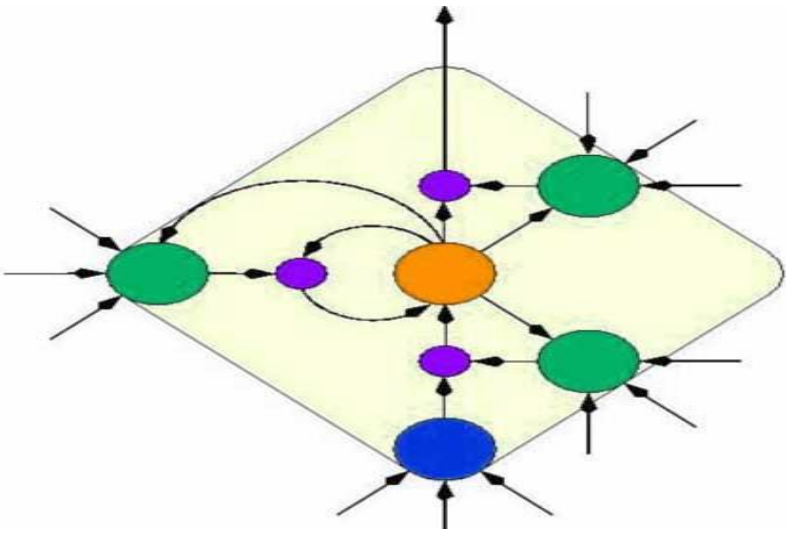

Fig. 4 Esempio di rete neurale di tipo feed-back

Una rete feed-forward invece, è una rete neurale che non ha connessioni di retroazione.

In questo tipo di rete, i neuroni prendono il loro input solo dallo strato precedente ed inviano il loro output solo allo strato seguente, come si vede in Fig. 5.

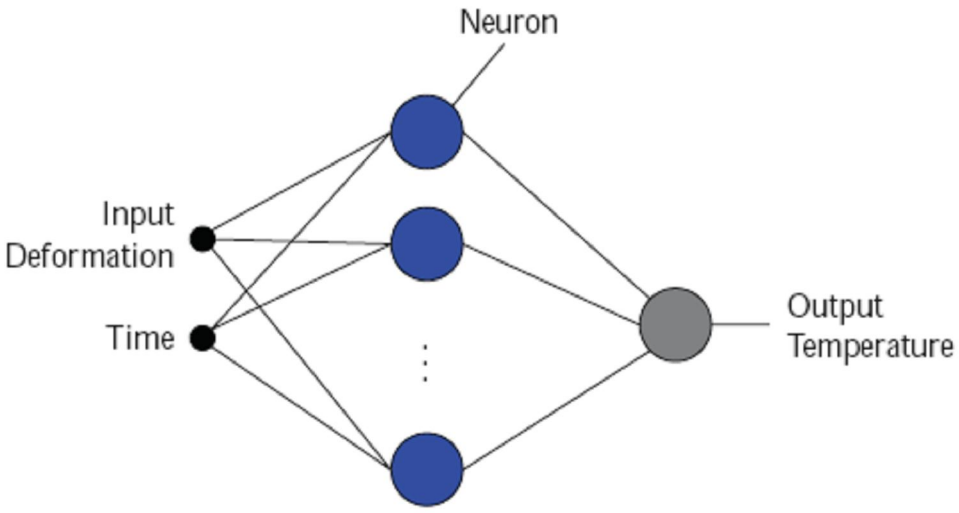

Fig. 5 Esempio di rete neurale di tipo feed-forward

Neuroni dunque dello stesso strato non sono tra loro connessi. A causa di ciò, queste reti, calcolano un risultato molto rapidamente. Un risultato in una rete di questo tipo è ottenuto nel seguente modo.

Prima i neuroni di input calcolano il loro valore di output basato sul corrente input. Dopo che tutti i neuroni hanno completato simultaneamente questo compito, allora nel successivo strato (quello intermedio), ognuno dei neuroni calcola il proprio output. Ogni neurone intermedio ottiene il risultato tenendo conto dei segnali provenienti da tutti i neuroni dello strato di input. Infatti il neurone intermedio è collegato con una distinta connessione (dall'appropriato "peso") ad ognuno dei neuroni dello strato di input.

In genere, quindi, gli output sono differenti per ognuno dei neuroni intermedi. Quando tutti i neuroni intermedi hanno implementato il loro risultato, quelli di output (o del successivo strato intermedio) calcolano il loro output basando sempre sulla somma pesata dei segnali provenienti da tutti i neuroni intermedi.

Dato che ogni neurone deve essere dotato di una connessione diretta (con il relativo "peso") con ognuno dei neuroni dello strato precedente, la rete feed-forward ha bisogno di memorizzare una notevole quantità di pesi (molto meno rispetto alle reti feed-back).

Utilizzando un appropriato metodo di apprendimento o learning mode si produce un processo di modifica graduale dei pesi fino a raggiungere la configurazione ottimale.

Il processo di modifica dei pesi in maniera ordinata prende il nome di training. Esso viene svolto sfruttando un metodo di apprendimento. Ci sono due categorie di metodi di apprendimento: supervised (sorvegliato) e unsupervised (non sorvegliato). Nel metodo di tipo supervised, sono utilizzati dei supervisori durante il training per dire se il corrispettivo output generato è corretto oppure no. Nel caso dell'unsupervised, non è

presente nessun tipo di controllo. Nel caso dei modelli supervised, dopo aver abbozzato la rete, bisogna attuare il training costruendo il training set, cioè una lista di coppie: un valore di input e un valore di output detto “pattern”. Il pattern è il risultato che la rete deve produrre in corrispondenza di quel determinato input. Ognuna delle coppie è chiamata “fact” (fatto). E’ importante che tutte le informazioni che la rete deve imparare siano nel training set e cioè che i fact siano numerosi e vari tali da poter rappresentare tutto il range di situazioni possibili. Il training set è fornito alla rete: essa analizzerà sequenzialmente un fact alla volta. Per ogni fact la rete preleva l’input e produce un suo output. Quest’ultimo viene confrontato con il valore del pattern, ovvero con il desiderato output. Se c’è una differenza, i pesi delle connessioni vengono modificati per diminuire il valore dell’errore. La rete poi analizza la successiva coppia fino all’ultima e continuerà a fare ciò finché, anche solo una di queste, produce errore. In genere la rete deve analizzare numerose volte i fact prima di apprendere totalmente il training set. In alcuni casi il ciclo non termina mai e probabilmente sarà necessario rivedere l’architettura della rete. Quando invece il training è completo la rete è pronta per l’utilizzo; infatti agli input risponde con i corretti e desiderati output.

I più popolari algoritmi di apprendimento per le reti feed-forward sono: Perceptron, Adaline, Madeline, Cognitron, Neo-Cognitron, Competition, Boltzman, Harmony, Counter Propagation e Back Propagation. Tutte queste tecniche si assomigliano abbastanza tra loro e danno il nome a differenti sottomodelli di reti feed-forward. Tutti gli algoritmi di apprendimento hanno un identico obbiettivo: ottenere una configurazione interna di pesi che permette alla rete feed-forward di svolgere una determinata funzione, ovvero di completare con successo l’addestramento previsto dal training set. Perciò, allorquando durante il training si verificano degli errori, dovranno essere modificati opportunamente tutti i pesi delle connessioni della rete. La necessaria variazione di peso viene calcolata applicando delle rigorose regole matematiche basate sulla teoria di Hebb o su alcune sue varianti di tale regola, come la regola Delta e Back Propagation.

Questo lavoro si è incentrato sullo sviluppo di tale modello, per la previsione della regione ferritica di acciai con composizioni chimiche diverse.

L’impiego di tale metodo, non presentando le limitazioni delle altre metodologie presenti, dovrebbe essere in grado di predire il profilo di curva della zona ferritica, dopo aver inserito una qualsiasi composizione chimica non nota. Con questo progetto di gruppo, si vuole vedere se ciò è possibile, poiché in questo caso la scarsa presenza di dati in letteratura, potrebbe costituire un impedimento alla riuscita del metodo. Nel caso avesse una buona riuscita, però, questo rappresenterebbe il modo più efficiente e rapido per poter conoscere le curve di raffreddamento. Esso infatti ridurrebbe

notevolmente gli elevati costi dovuti alla sperimentazione, e il tempo che essa richiede, fornendo nel contempo dei risultati affidabili.

Nel capitolo seguente viene presentato e descritto lo sviluppo di un sistema a reti neurali, con l'obiettivo di prevedere i tempi di inizio e fine precipitazione della ferrite.

CAPITOLO 3

Sviluppo di un sistema a reti neurali

Viene, ora, presentata la fase di sviluppo di un sistema a reti neurali che segue la fase di raccolta dei dati.

La raccolta dei dati degli acciai si era basata sull'utilizzo dei diagrammi CCT che si presentano come in figura (Fig. 1).

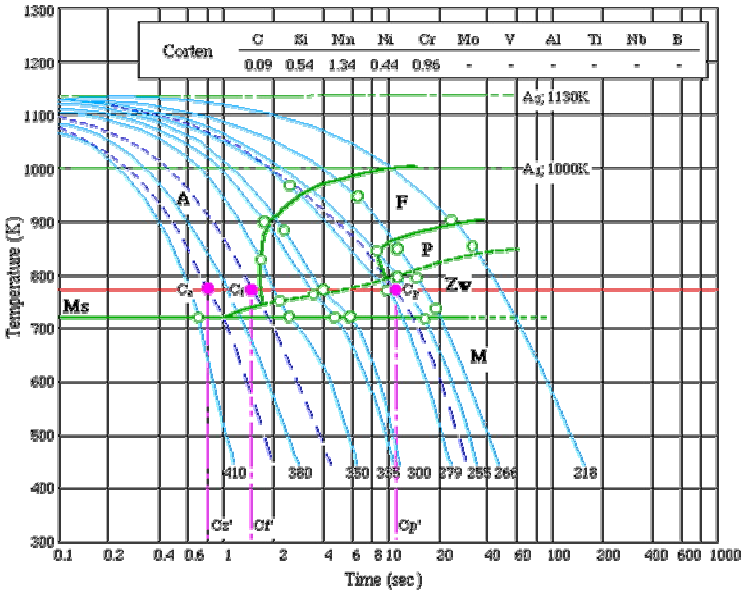

Fig. 1

Questo aveva portato alla costruzione di una tabella Excel che si presenta come di seguito riportato (Tabella 1).

Nome	Composiz. chimica (C, Si, N…)	T. di austenitizz. [K]	T di rif. [K]	A_{c3}	A_{c1}	t di raffredd. [s]	F_s	F_f
Hi-Z		1623	773	1098	1013	20,71	21,96	24,35
						33,64	30,03	35,75
						107,18	54,71	95,17
						174,11	83,37	151
						237,84	94,37	194,5
						400	196,2	334,8

Tabella 1

In tale tabella sono, dunque, rappresentati la composizione chimica dell'acciaio (C, Si, Mn, P, S, Cu, Ni, Cr, Mo, V, Al, Ti, N, Nb, B, Mg), i punti caratteristici di esso (temperatura di austenitizzazione, temperatura di raffreddamento, punti critici A_1 e A_3, tempo di raffreddamento), e ciò che si vorrebbe ottenere, ovvero i punti di inizio e fine precipitazione della ferrite.

La fase di raccoglimento dei dati degli acciai diventa, quindi, l'input per il lavoro di sviluppo di un sistema a reti neurali. Questo sistema viene realizzato con l'impiego della libreria Shark, che è una libreria orientata agli oggetti per la progettazione di sistemi adattativi. Essa comprende i metodi per l'ottimizzazione di obiettivi singoli e multipli, come metodi basati su kernel, reti neurali, e altre tecniche di apprendimento automatico.

3.1.La libreria Shark

Shark è una libreria in linguaggio C++ per la progettazione e l'ottimizzazione di sistemi adattativi. Essa serve come strumento per le applicazioni del mondo reale e la ricerca base nell'intelligenza computazionale e nell'apprendimento automatico. La libreria fornisce metodi per l'ottimizzazione di obiettivi singoli o multipli, in particolare algoritmi evolutivi e gradient-based, metodi di apprendimento kernel-based, reti neurali, e molte altre tecniche di apprendimento automatico. I suoi principali criteri di progetto sono la flessibilità e la velocità. In questo ambito si restringe la descrizione di Shark ai componenti di interesse, cioè quelli relativi alle reti neurali, sebbene la libreria contenga molte altre funzioni supplementari.

3.1.1.Metodi di regressione e classificazione

I metodi di interesse per la costruzione delle reti neurali sono quelli di regressione e classificazione (Regression and Classification Methods - ReClaM). L'obiettivo del modulo ReClaM è quello di fornire gli algoritmi di apprendimento automatico per la classificazione e la regressione in una struttura unificata, modulare. Esso è costruito come un kit di costruzione, dove i principali componenti elementari sono modelli di elaborazione di dati adattativi, funzioni di errore e algoritmi di ottimizzazione.
Si vede l'organizzazione delle classi base di ReClaM con un'illustrazione del flusso di informazioni (Fig. 2).

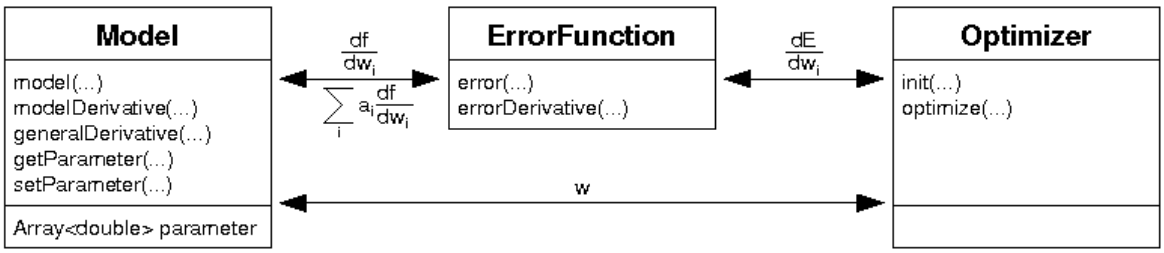

Fig. 2

Un problema è delineato da un modello che definisce una famiglia di ipotesi candidate, e una funzione che regolarizza il possibile errore per minimizzarlo. Questo problema viene di solito risolto con un algoritmo di ottimizzazione (iterativo), che adatta i parametri del modello allo scopo di minimizzare la funzione di errore valutata sulla raccolta di dati considerata. Le funzioni di errore aggiuntive e le raccolte di dati possono quindi essere usate per testare le performance risultanti. Questa struttura chiara rende ReClaM facile da usare ed estendere.

ReClaM si focalizza sui metodi kernel e le reti neurali. Questo modulo offre una varietà di modelli di rete predefiniti che includono le reti feed-forward e quelle recurrent multi-layer perceptron, le reti radial basis function, e CMACs. Diversi algoritmi di ottimizzazione gradient-based includono il metodo del gradiente coniugato, l'algoritmo BFGS, e l'Rprop.

3.1.2.Riconoscimenti

Il progetto Shark partì da M. Kreutz che scrisse i componenti base come LinAlg, Array, e Rng oltre a EALib. Poi B. Sendhoff si unì al progetto, che fu fuso con la libreria ReClaM di C. Igel. In seguito molte altre persone contribuirono al programma, in particolare (in ordine alfabetico) R. Alberts, T. Bücher, A. W. Dietrich, che inventò il nome Shark, T. Glassmachers, che estese la libreria ReClaM, M. Hüsken, T. Okabe, che scrisse il MOO- EALib, S. Roth, P. Stagge, T. Suttorp, M. Toussaint, e T. Voß.

Lo sviluppo di Shark è supportato dai seguenti istituti e compagnie: Institut for Neuroinformatik, NISYS e Europe Honda Research Institute; i cui loghi vengono di seguito riportati (Fig. 3).

Fig. 3

3.2.Le reti neurali

Si è visto come ReClaM proponga alcuni tipi predefiniti di reti. Queste reti coprono una vasta varietà di applicazioni. Ogni volta che questi prototipi risultano inappropriati possono servire come classi base o punti di partenza per implementazioni perfezionate.

La seguente tabella (Tabella 2) elenca alcune proprietà:

- Nome - Il nome delle classi che implementano la rete;
- Tipo – Ci sono alcuni tipi base di rete, i tipi possibili sono Feed Forward, Recurrent e Radial Basis Function;
- Funzioni di attivazione – Ogni neurone ha una funzione di attivazione che determina come i valori di input si propagheranno attraverso la rete. Le funzioni di attivazione dei neuroni degli strati hidden e degli strati output possono essere poste separatamente;
- Misure di errore – Alcune reti propongono funzioni di errore integrate, che sono usate solo per l'apprendimento.

Name	Type	Activation Functions		Training Error Measure
		Hidden Neurons	Output Neurons	
FFNet	Feed Forward	$1/(1+e^{-a})$	$1/(1+e^{-a})$	none
MSEFFNet	Feed Forward	$1/(1+e^{-a})$	$1/(1+e^{-a})$	Mean Squared Error
LinOutFFNet	Feed Forward	$1/(1+e^{-a})$	a	none
LinOutMSEBFFNet	Feed Forward	$1/(1+e^{-a})$	a	Mean Squared Error
TanhNet	Feed Forward	$2/[(1+e^{-2a})-1]$	$2/[(1+e^{-2a})-1]$	Squared Error
LinearOutputTanhNet	Feed Forward	$2/[(1+e^{-2a})-1]$	a	Squared Error
ProbenNet	Feed Forward	a/(1+lal)	a	Mean Squared Error
ProbenBNet	Feed Forward	a/(1+lal)	a	Mean Squared Error
MSERNNet	Recurrent	$1/(1+e^{-a})$	$1/(1+e^{-a})$	Mean Squared Error
RBFNet	Radial Basis Function	special	special	none
MSERBFNet	Radial Basis Function	special	special	Mean Squared Error

Tabella 2

3.3.La rete neurale feed forward

Dei vari tipi di rete neurale offerti da ReClaM viene presa in considerazione quella di tipo feed-forward, in quanto reputata più adatta ai fini della predizione del profilo di curva della zona ferritica.
In quest'ambito la classe FFNet offre le funzioni per creare e lavorare con una rete feed-forward. Insieme con la classe di misura dell'errore e quella dell'algoritmo di ottimizzazione si può usare la classe della rete feed-forward per produrre il proprio strumento di ottimizzazione.

3.3.1.Struttura di una rete neurale feed-forward

Una rete neurale è un modello matematico/informatico di calcolo, costituito da un gruppo di interconnessioni di informazioni e processi che utilizzano un approccio di connessionismo di calcolo. Gli elementi che vengono interconnessi sono definiti neuroni artificiali, proprio perchè si rifanno ai neuroni biologici. Nella maggior parte dei casi una rete neurale è un sistema adattativo che cambia la sua struttura basata su informazioni esterne o interne che scorrono attraverso la rete durante la fase di apprendimento.
In termini pratici le reti neurali sono strutture non-lineari di dati statistici organizzati come strumenti di modellazione. Esse possono essere utilizzate, come si è precedentemente visto, per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare.
Una rete neurale riceve segnali esterni su uno strato di nodi (unità di elaborazione) d'ingresso, ciascuno dei quali è collegato con numerosi nodi interni, organizzati in più livelli. Ogni nodo elabora i segnali ricevuti e trasmette il risultato a nodi successivi.
Tutto questo viene schematicamente rappresentato nell'illustrazione sotto riportata (Fig. 4).

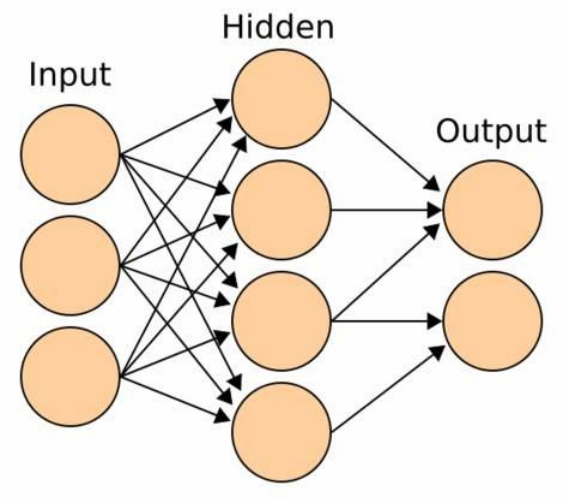

Fig. 4 Una rete neurale artificiale è un'interconnessione di un gruppo di nodi chiamati neuroni

Si considera in particolare una rete neurale feed-ferward.

Queste reti si basano principalmente sulla simulazione di neuroni artificiali opportunamente collegati. I suddetti neuroni ricevono in ingresso degli stimoli e li elaborano. L'elaborazione può essere anche molto sofisticata, ma in un caso semplice si può pensare che i singoli ingressi vengano moltiplicati per un opportuno valore detto peso, il risultato delle moltiplicazioni viene sommato e se la somma supera una certa soglia il neurone si attiva attivando la sua uscita. Il peso indica l'efficacia della linea di ingresso e serve a quantificarne l'importanza: un ingresso molto importante avrà un peso elevato, mentre un ingresso poco utile all'elaborazione avrà un peso inferiore. Quanto appena descritto lo si può vedere nella figura sottostante (Fig. 5).

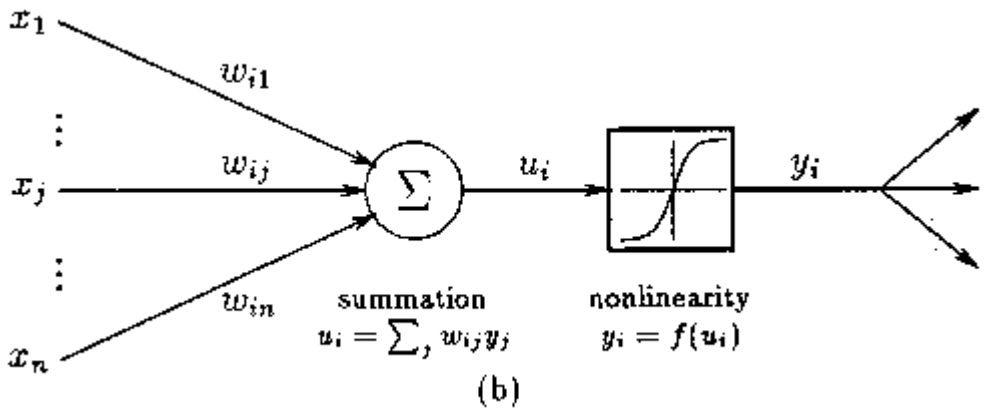

Fig. 5

Andando più nel dettaglio si può vedere un esempio che viene di seguito illustrato (Fig. 6).

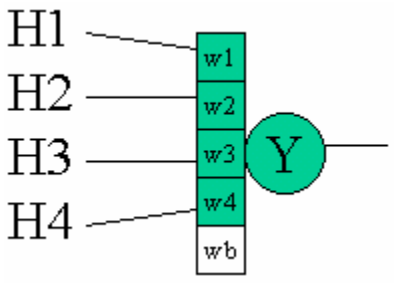

Fig. 6

In questa figura il neurone Y ha 4 ingressi (sinapsi o connessioni) a cui arrivano i segnali H1, …, H4 ciascuno associato ad un peso (w1, …, w4). I pesi sono un fattore moltiplicativo applicato al segnale di ingresso e possono essere positivi o negativi (sinapsi eccitatoria o inibitoria), maggiori o minori di 1 (sinapsi amplificatrice o attenuatrice). In più ogni neurone ha un peso di bias (wb) che serve per regolare il punto di lavoro del neurone stesso. Questa sinapsi si considera collegata ad un'unità fittizia che genera un segnale sempre pari a 1, e pertanto non è necessario rappresentarla nel disegno. Si definisce attivazione interna del neurone (chiamata A) la somma pesata di tutti i segnali di ingresso:

$$A = H1 \cdot w1 + H2 \cdot w2 + H3 \cdot w3 + H4 \cdot w4 + wb$$

Il segnale di uscita dal neurone è invece chiamato attività ed è calcolato applicando una funzione di trasferimento. L'uscita del nostro neurone si ottiene perciò con:

$$Y = 1/(1+e^{-a})$$

A seconda del valore dei pesi, al variare dei valori in ingresso l'uscita del neurone riproduce una diversa funzione matematica.

Si può pensare che se due neuroni comunicano tra loro utilizzando maggiormente alcune connessioni, allora tali connessioni avranno un peso maggiore fino a che non si creeranno delle connessioni tra l'ingresso e l'uscita della rete che sfruttano "percorsi preferenziali". Tuttavia è sbagliato pensare che la rete finisca col produrre un unico percorso di connessione: tutte le combinazioni infatti avranno un certo peso, e quindi contribuiscono al collegamento ingresso/uscita.
Il modello nella figura sottostante (Fig. 7) rappresenta una classica rete neurale pienamente connessa.

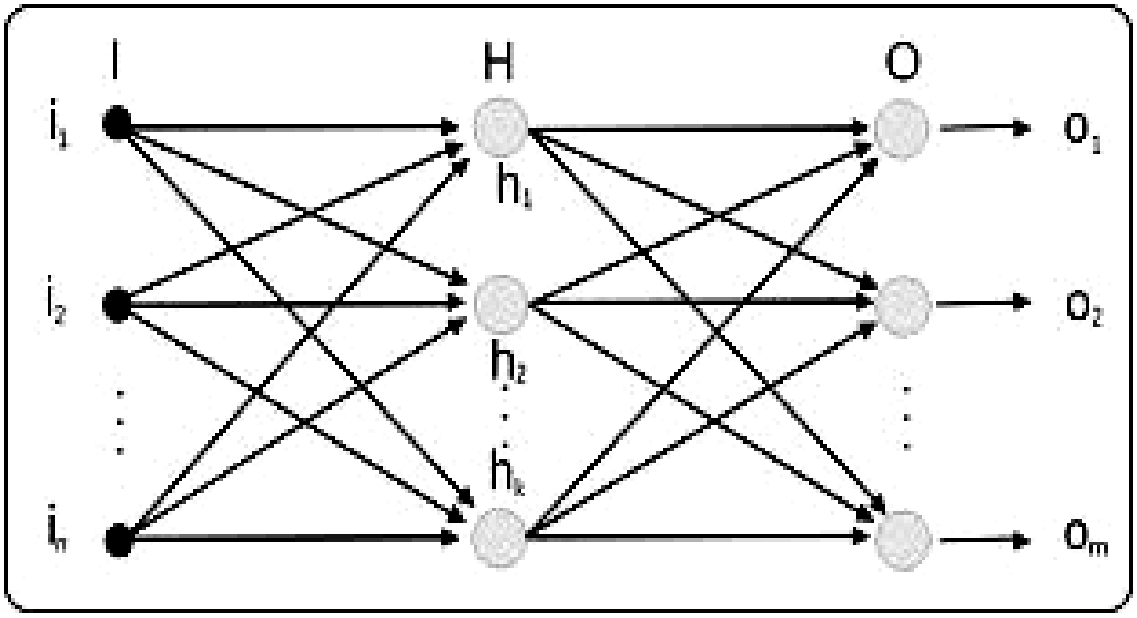

Fig. 7

I singoli neuroni vengono collegati alla schiera di neuroni successivi, in modo da formare una rete di neuroni. Normalmente una rete è formata da tre strati. Nel primo si hanno gli ingressi (Input - I), questo strato si preoccupa di trattare gli ingressi in modo da adeguarli alle richieste dei neuroni. Se i segnali in ingresso sono già trattati può anche non esserci. Il secondo strato è quello nascosto (Hidden – H), si preoccupa dell'elaborazione vera e propria e può essere composto anche da più colonne di neuroni. Il terzo strato è quello di uscita (Output - O) e si preoccupa di raccogliere i risultati ed adattarli alle richieste del blocco successivo della rete neurale. Queste reti possono essere anche molto complesse e coinvolgere migliaia di neuroni e decine di migliaia di connessioni.

Per costruire la struttura di una rete neurale multistrato si possono inserire N strati Hidden; vi sono però alcune dimostrazioni che mostrano che con uno o due strati di Hidden si ottiene una stessa efficace generalizzazione da una rete rispetto a quella con più strati Hidden. L'efficacia di generalizzare di una rete neurale multistrato dipende dall'addestramento che ha ricevuto e dal fatto di essere riuscita o meno ad entrare in un minimo locale buono.

Le neural network per come sono costruite lavorano in parallelo e sono quindi in grado di trattare molti dati. I modelli prodotti però, anche se molto efficienti, non sono spiegabili in linguaggio simbolico umano: i risultati vengono accettati “così come sono”, da cui anche la definizione inglese delle reti neurali come “black box”: in altre parole, a differenza di un sistema algoritmico dove si può esaminare passo passo il percorso che dall’input genera l’output, una rete neurale è in grado di generare un risultato

valido, o comunque con un'alta probabilità di essere accettabile, ma non è possibile spiegare come e perché tale risultato sia stato generato.

3.4.Implementazione del programma per lo sviluppo della rete neurale

Quanto si è finora visto consente di addentrasi nella fase di implementazione del programma per lo sviluppo della rete neurale .

Innanzitutto i dati, precedentemente raccolti in Excel, sono stati poi raggruppati in base alla composizione chimica, in modo da ottenere un sottogruppo il più omogeneo possibile per facilitare l'apprendimento della rete neurale, pertanto si sono tolti quegli acciai che presentavano degli elementi in lega che la maggior parte degli acciai non avevano, e sono stati quindi tolti anche quei corrispondenti elementi caratterizzanti la composizione chimica.
I dati sono dunque stati raccolti come riportato nella tabella sottostante (Tabella 3).

Nome acciaio	C[wt%]	Si[wt%]	Mn[wt%]	P[wt%]	S[wt%]	Cu[wt%]	Ni[wt%]	Cr[wt%]
FTW58	0,170	0,380	1,310	0,000	0,000	0,000	0,040	0,000
FTW58	0,170	0,380	1,310	0,000	0,000	0,000	0,040	0,000
…	…	…	…	…	…	…	…	…

Mo[wt%]	V[wt%]	Al[wt%]	Ta[K]	Trif[K]	traff[s]	A1[K]	A3[K]	Fs[s]	Ff[s]
0,010	0,040	0,000	1623	773	6,00	953	1073	5,17	8,30
0,010	0,040	0,000	1623	773	8,94	953	1073	5,37	11,62
…	…	…	…	…	…	…	…	…	…

Tabella 3

Infine i dati così raggruppati sono stati esportati in csv per facilitarne la lettura da parte del programma di implementazione della rete neurale sviluppato nell'ambiente di programmazione C++.

I dati esportati in csv si presentano come segue:

Nome acciaio,C,Si,Mn,P,S,Cu,Ni,Cr,Mo,V,Al,Temperatura di austenitizzazione [K],Temperatura di riferimento [K],Tempo di raffreddamento [s],A1 [K],A3 [K],Fs[s],Ff[s]

FTW58,0.170,0.380,1.310,0.000,0.000,0.000,0.040,0.000,0.010,0.040,0.000,1623,773,6.00,953,1073,5.17,8.30

FTW58,0.170,0.380,1.310,0.000,0.000,0.000,0.040,0.000,0.010,0.040,0.000,1623,773,8.94,953,1073,5.37,11.62
...

Queste informazioni sono state necessarie per codificare le prime istruzioni fornite al programma, in quanto definiscono la struttura di ogni acciaio considerato.
Viene di seguito riportata la prima parte del codice:

```
struct SteelData{
  std::string steelName;
  std::vector<double> componentPercentage;
  double austenitizationTemperature;
  double referenceTemperature;
  double coolingTime;
  double A1;
  double A3;
  double Fs;
  double Ff;
};
```

La struttura `SteelData` è una collezione di variabili di uno o più tipi, relative ad ogni acciaio, le quali sono:

- la stringa `steelName`, rappresentativa del nome dell'acciaio (nell'esempio FTW58);
- il vettore `componentPercentage`, relativo alla composizione percentuale (nell'esempio C=0.17%, Si=0.38%, Mn=1.31%, …);
- la temperatura di austenitizzazione, T_a (nell'esempio 1623K);
- la temperatura di riferimento relativa al raffreddamento, T_{rif} (nell'esempio 773K);
- le temperature relative ai punti critici A_1 e A_3 (nell'esempio A_1=953K e A_3=1073K);
- i tempi di inizio e fine precipitazione della ferrite (nell'esempio Fs=5.17s e

Ff=8.303s).

I passi successivi codificano le istruzioni che saranno necessarie alla rete neurale nell'implementazione Shark, con l'obiettivo di istruirla all'apprendimento e alla successiva predizione delle informazioni desiderate.

La rete è addestrata mediante un opportuno algoritmo (tipicamente la back propagation che è appunto un algoritmo di apprendimento supervisionato). Si parte con i pesi impostati a piccoli valori casuali e si sottopongono ad essa degli esempi di ingressi e uscite corretti; si applica poi un procedimento di correzione dei pesi per piccoli passi fino ad ottenere un'uscita effettiva che per ogni esempio non si discosti oltre una certa percentuale dall'uscita campione usata per l'addestramento. Per fare questo si usa la regola delta generalizzata: in sostanza per ogni esempio fornito in ingresso si valuta l'errore in uscita rispetto al valore desiderato e si calcola un delta del neurone che serve poi per modificare i diversi pesi. Ogni peso viene modificato di una quantità pari al delta del neurone per il segnale in arrivo su quella connessione e il tutto ancora moltiplicato per un fattore di apprendimento (compreso tra 0,1 e 0,9) scelto a priori. Nel calcolo dell'errore si parte dall'uscita e si procede verso gli ingressi, è inoltre importantissimo calcolare il delta di tutti i neuroni della rete prima di iniziare a modificare i pesi.
Se l'addestramento ha successo la rete impara a riconoscere la relazione incognita che lega le variabili d'ingresso a quelle d'uscita, ed è quindi in grado di fare previsioni anche laddove l'uscita non è nota a priori, in altri termini, l'obiettivo finale dell'apprendimento supervisionato è la previsione del valore dell'uscita per ogni valore valido dell'ingresso, basandosi soltanto su un numero limitato di esempi di corrispondenza (vale a dire, coppie di valori input-output). Per fare ciò la rete deve essere dotata di un'adeguata capacità di generalizzazione, con riferimento a casi ad essa ignoti. L'insieme dei campioni non va scelto a caso, anzi la scelta è un passo di fondamentale importanza per il corretto addestramento; bisogna cioè prendere per quanto possibile campioni distribuiti omogeneamente in tutto lo spazio del problema, e devono anche essere abbastanza fitti da seguire e descrivere tutte le caratteristiche della funzione voluta.

Per l'addestramento globale di una rete si procede nel seguente modo:

- si impostano i pesi a piccoli valori casuali
- per ogni esempio
    1. si presenta l'esempio alla rete e la si esegue per determinarne l'uscita

2. si calcolano i delta di tutte le unità
3. si modificano i pesi

- se il massimo errore commesso da una qualunque unità di uscita durante un qualsiasi esempio è maggiore di quello ammesso si torna al punto 2).

Si vede, a questo punto, la prima parte di codice che viene di seguito riportato e che fornisce l'input per la fase di apprendimento, o training, della neural network:

```
  //------ reading the data sets for TRAINING
  // argv[1] is the first argument. MUST be a csv file
  ParseSteelData trainingData(argv[1]);
  // Scommentare per vedere in uscita i dati di training
  // cout << "Training data\n";
  cout << trainingData;

  cout << "Steel Number: " << trainingData.getNumberPatterns()
<< endl;

  // istanze (uno per grafico)
  const unsigned numberTrainingPatterns =
trainingData.getNumberPatterns();
  // sono le colonne (numero di dati di input per ogni grafico)
  const unsigned numberFeatures =
trainingData.getNumberFeatures();

  // build the data structure for Shark
  Array<double> trainInput(numberTrainingPatterns,
numberFeatures);
  Array<double> trainTarget(numberTrainingPatterns, 1);

  // choose what you want to learn
  trainingData.fillFs(trainInput, trainTarget);
  //trainingData.fillFf(trainInput, trainTarget);

  // output the trainInput and trainTarget
  writeArray(trainInput, cout);
  writeArray(trainTarget, cout);
```

Relativamente a questa fase di training, è stato creato un primo file di tipo csv chiamato `argv[1]` contenuto nell' oggetto `trainingData` appartenente alla classe `ParseSteelData`. Questo file contiene le matrici `trainInput` e `trainTarget`. La prima matrice è un array bidimensionale avente un certo numero di righe chiamate `numberTrainPatterns`, in numero uguale alle righe del file csv di raccolta dei dati, e un numero di colonne chiamate `numberFeatures` in numero uguale alle colonne del file csv di raccolta dei dati tranne quelle necessarie a far apprendere la neural network (in questo caso Fs o Ff). La seconda matrice ha lo stesso numero di righe `numberTrainPatterns` della prima matrice, ma un' unica colonna relativa a quelle informazioni del file csv di raccolta dei dati necessarie a far apprendere la neural network (nel nostro caso spedifico Fs o Ff). Tramite la funzione `fill` viene pertanto estratta dalla matrice `trainInput` l'informazione Fs o Ff per il training della rete neurale e riportata nella matrice `trainTarget`.

La seconda parte di codice sottoriportato fornisce l'input per la successiva fase di evaluating, in cui la neural network sarà chiamata ad indovinare i valori di interesse (in questo caso specifico Fs o Ff):

```
  //----- read the data sets for EVALUATING
  ParseSteelData evaluatingData(argv[2]);
  // Scommentare per vedere in uscita i dati di evaluating
  // cout << "Evaluating data\n";
  // cout << evaluatingData;

  const unsigned numberEvaluatingPatterns =
evaluatingData.getNumberPatterns();
  const unsigned numberEvalFeatures =
evaluatingData.getNumberFeatures();

  if (numberEvalFeatures != numberFeatures)
    cout << "Something is wrong: training " << numberFeatures <<
"  features, evaluating " << numberEvalFeatures << "
numberEvalFeatures\n";

  // build the data structure for Shark
  Array<double> valInput(numberEvaluatingPatterns,
numberFeatures);
```

```
Array<double> valTarget(numberEvaluatingPatterns, 1);

// choose what you want to guess
evaluatingData.fillFs(valInput, valTarget);
// evaluatingData.fillFf(valInput, valTarget);

// output the trainInput and trainTarget
// writeArray(valInput, cout);
// writeArray(valTarget, cout);
```

Relativamente a questa fase di evaluating, è stato creato un primo file di tipo csv chiamato `argv[2]` contenuto nell' oggetto `EvaluatingData` appartenente alla classe `ParseSteelData`. Questo file contiene le matrici `valInput` e `valTarget`. La prima matrice è un array bidimensionale avente un certo numero di righe chiamate `numberEvaluatingPatterns`, in numero uguale alle righe del file csv di raccolta dei dati, e un numero di colonne chiamate `numberFeatures` in numero uguale alle colonne del file csv di raccolta dei dati tranne quelle necessarie alla neural network per indovinare le informazioni di interesse (in questo caso specifico Fs o Ff). La seconda matrice ha lo stesso numero di righe `numberEvaluatingPatterns` della prima matrice, ma un' unica colonna relativa a quelle informazioni del file csv di raccolta dei dati necessarie a far prevedere la neural network (in questo caso spedifico Fs o Ff). Tramite la funzione `fill` viene, pertanto, estratta dalla matrice `valInput` l'informazione Fs o Ff per l'evaluating della rete neurale e riportata nella matrice `valTarget`. Prima della funzione `fill` viene però eseguito il costrutto `if` per la verifica dell'uguaglianza del numero di colonne tra la prima matrice della fase di training e quella della fase di evaluating.

Infine si ha il codice di programmazione che implementa la vera e propria parte sulle reti neurali Shark di tipo feed-forward. Di seguito viene riportato il suddetto codice:

```
// define topology unsigned
unsigned inputs = numberFeatures;
unsigned firstHiddenLayer = numberFeatures + 1;
//  unsigned secondHiddenLayer = 7;
unsigned outputs = 1;
```

```
  Array<int> con;
  createConnectionMatrix(con, inputs,
                         firstHiddenLayer, outputs);

  // feed-forward neural network
  MyNet net(inputs, outputs, con), netMin(inputs, outputs, con);

  // error function
  MeanSquaredError error;

  // optimizer
  IRpropPlus optimizer;
  optimizer.init(net);

  // initialize the weights uniformly between -0.1 and 0.1
  net.initWeights(-0.1, 0.1);

  std::ofstream trajectory("trajectory");
  // training loop
  unsigned numberOfLearningCycles = 3000;
  for (int t = 1; t <= numberOfLearningCycles; t++) {
    // train the network
    optimizer.optimize(net, error, trainInput, trainTarget);

        err = error.error(net, trainInput, trainTarget);

     // write results
    trajectory <<   t << "\t" << error.error(net, trainInput,
trainTarget) << "\t" << error.error(net, valInput, valTarget) <<
std::endl;
  }

  trajectory.close();

// output network as source code FFNetSource(std::cout, inputs,
outputs, con, net.getWeights(), "tanh(#)", "#", 10);
```

```
  std::ofstream previsione("previsione");
  Array<double> in(numberFeatures), out(1);
  for (int t = 0; t < numberEvaluatingPatterns; t++) {

     for (int j= 0; j < numberFeatures; j++)
        in(j) = valInput(t, j);
     net.model(in, out);
     previsione << "Input: ";
     for (int j = 0; j < numberFeatures; j++)
        previsione << valInput(t,j) << " ";
     previsione << "Output:\t" << valTarget(t) << " " << out(0)
<< endl;
  }
```

In questa parte di codice viene definita la topologia del modello della rete neurale. Innanzitutto vengono impostati gli `inputs`, cioè gli ingressi, che sono uguali alle colonne `numberFeatures`; poi vengono definiti i `firstHiddenLayer`, cioè i nodi dello strato nascosto; infine vengono forniti gli `outputs`, cioè le uscite, che in questo caso sono in numero pari a uno, che fa riferimento a Fs o Ff. Poi viene creata la matrice di connessione con la funzione `createConnectionMatrix` che ha come argomento quattro parametri per specificare la connettività. A questo punto viene fornito il modello della feed-forward neural network con le funzioni `MyNet net` e `netMin`, la funzione di errore con `MeanSquaredError error`, l'ottimizzatore con `IRpropPlus optimizer` e vengono inizializzati i pesi con il costrutto `net.initWeights`. Il passo successivo prevede una parte di codice relativa all'apprendimento della rete neurale. Viene, quindi, creato un file `trajectory` e definito un certo numero di cicli `numberOfLearningCycles`, che vengono svolti attraverso il costrutto `for`. Questi cicli sono il numero di volte che si vuole che la rete neurale venga implementata per l'apprendimento, in modo da abbassare l'errore. Il ciclo for, quindi, fa si che vengano eseguite le istruzioni di ottimizzazione e viene calcolato l'errore per il numero di cicli desiderato. I valori dell'errore rilevati sia per l'apprendimento che per la previsione vengono, dunque, forniti in uscita. Viene, poi, chiuso questo file con `trajectory.close`. Infine l'ultima parte di codice è relativa alla previsione dell'output che la rete neurale riesce a fare sulla base dei dati forniti in input a lei noti. Pertanto viene creato un file `previsione`, che prevede un ciclo `for`, il quale viene eseguito per il numero di righe `numberEvaluatingPatterns`, cioè per

ogni acciaio. Questo ciclo prevede inoltre al suo interno un ciclo `for` che viene eseguito per il numero di colonne `numberFeatures` e consente di fornire gli input alla rete neurale; la variabile oggetto `net` alla quale viene applicata la funzione `model`, che consente, fornendo gli input, la previsione dell'output, cioè fa si che il modello della rete neurale venga applicato ai dati in input; un altro ciclo `for` che viene eseguito sempre per il numero di colonne `numberFeatures` e serve a fornire in uscita gli input che sono stati dati alla rete neurale per la previsione, e gli output che corrispondono a quello voluto e quello ottenuto dalla previsione.
Dalla compilazione di questo codice si ottengono, dunque, due file: un file trajectory che riporta l'errore per ogni ciclo eseguito, e un file previsione che fornisce l'output desiderato e quello previsto dalla rete neurale sulla base di dati ad essa noti per ogni acciaio.

Le precedenti parti di codice, messe insieme, servono, dunque, per la previsione dell'output Fs o Ff sulla base di dati noti alla rete neurale.

Il passo successivo, che rappresenta l'obiettivo principale di questa tesi è la predizione dei valori di interesse, in questo caso Fs o Ff, da parte della neural network sulla base di valori non noti ad essa. Viene, perciò, riportata la parte di codice per quello che viene chiamato cross-evaluation, cioè la previsione di valori che caratterizzano l'output sconosciuti alla rete neurale, dandone però l'input solo nella fase di previsione:

```
  std::ofstream previsione("previsione");

  // cycle for each steel

  int steelsNumber = 897;
  for (int currentSteel = 0; currentSteel < steelsNumber;
++currentSteel) {

    //------ reading the data sets for TRAINING
    // argv[1] is the first argument. MUST be a csv file
    ParseSteelData trainingData(argv[1]);
    // Scommentare per vedere in uscita i dati di training
    // cout << "Training data\n";
    cout << trainingData;
```

```
    cout << "Steel Number: " << trainingData.getNumberPatterns()
<< endl;

    // istanze (uno per grafico)
    const        unsigned        numberTrainingPatterns        =
trainingData.getNumberPatterns();
    // sono le colonne (numero di dati di input per ogni
grafico)
    const          unsigned          numberFeatures          =
trainingData.getNumberFeatures();

    if (numberTrainingPatterns != steelsNumber) {
       cout << "Sorry! We have a wrong number of steels
here!!!\n";
       exit(0);
    }

    // build the data structure for Shark
    Array<double>           trainInput(numberTrainingPatterns-1,
numberFeatures);
    Array<double> trainTarget(numberTrainingPatterns-1, 1);

    // build the data structure for Shark
    Array<double> valInput(1, numberFeatures);
    Array<double> valTarget(1, 1);

    // choose what you want to learn
    trainingData.fillFsEverythings(trainInput,      trainTarget,
valInput, valTarget, currentSteel);
    //trainingData.fillFfEverythings(trainInput,    trainTarget,
valInput, valTarget, currentSteel);

    // define topology unsigned
    unsigned inputs = numberFeatures;
    unsigned firstHiddenLayer = numberFeatures + 1;
    //  unsigned secondHiddenLayer = 7;
    unsigned outputs = 1;
```

```
    Array<int> con;
    createConnectionMatrix(con, inputs,
                           firstHiddenLayer, outputs);

    // feed-forward neural network
    MyNet net(inputs, outputs, con), netMin(inputs, outputs,
con);

    // error function
    MeanSquaredError error;

    // optimizer
    IRpropPlus optimizer;
    optimizer.init(net);

    // initialize the weights uniformly between -0.1 and 0.1
    net.initWeights(-0.1, 0.1);

    std::ofstream trajectory("trajectory");
    // training loop
    // MODIFICARE QUESTO:
    unsigned numberOfLearningCycles = 3000;
    for (int t = 1; t <= numberOfLearningCycles; t++) {
      // train the network
      optimizer.optimize(net, error, trainInput, trainTarget);

       err = error.error(net, trainInput, trainTarget);


      // write results
      trajectory <<  t << "\t" << error.error(net, trainInput,
trainTarget)
                 << "\t" << error.error(net, valInput, valTarget)
<< std::endl;
    }
    trajectory.close();
```

```
     // output network as source code FFNetSource(std::cout,
inputs, outputs, con, net.getWeights(), "tanh(#)", "#", 10);

    Array<double> in(numberFeatures), out(1);
    for (int j= 0; j < numberFeatures; j++)
        in(j) = valInput(0, j);
    net.model(in, out);
    previsione << "Input: ";
    for (int j = 0; j < numberFeatures; j++)
       previsione << valInput(0,j) << " ";
    previsione << "Output:\t" << valTarget(0) << " " << out(0)
<< endl;
  }

 previsione.close();
```

Questo codice presenta, all'inizio, la creazione del file `previsione`. Poi viene definito un ciclo `for` che verrà eseguito per ogni singolo acciaio, questo ciclo fa si che venga presa una riga per volta relativa ad un acciaio e venga tolta nella fase di apprendimento, riga che è relativa al valore Fs o Ff che si vuole prevedere e che verrà, poi, fornita come input alla rete neurale nella fase di previsione, però con l'esclusione del valore di interesse Fs o Ff, così da ottenere la previsione di un dato sconosciuto alla rete neurale. All'interno di questo ciclo viene creato un file di tipo csv chiamato `argv[1]` contenuto nell'oggetto `trainingData` appartenente alla classe `ParseSteelData`. Questo file contiene le matrici `trainInput` e `trainTarget`. La prima matrice è un array bidimensionale avente un certo numero di righe chiamate `numberTrainPatterns` corrispondenti alle righe del file csv di raccolta dei dati con l'esclusione di una riga, e un numero di colonne chiamate `numberFeatures` in numero uguale alle colonne del file csv di raccolta dei dati tranne quelle necessarie a far apprendere la neural network (in questo caso Fs o Ff). La seconda matrice ha lo stesso numero di righe `numberTrainPatterns` della prima matrice, ma un' unica colonna relativa a quelle informazioni del file csv di raccolta dei dati necessarie a far apprendere la neural network (in questo caso Fs o Ff). Si hanno, inoltre, le matrici `valInput` e `valTarget`. La prima matrice è un array avente una sola riga e un numero di colonne chiamate `numberFeatures` in numero uguale alle colonne del file csv di raccolta dei dati tranne quelle necessarie a far apprendere la neural network (in

questo caso specifico Fs o Ff). La seconda matrice ha una sola riga e una sola colonna relativa a quell'informazione del file csv di raccolta dei dati necessaria alla previsione da parte neural network (in questo caso spedifico Fs o Ff). È stata così costruita la struttura dei dati per Shark. Si sceglie, poi, ciò che si vuole far apprendere alla rete neurale tramite la funzione `fill`. Viene, dunque, definita la topologia del modello della rete neurale. Per prima cosa vengono impostati gli `inputs`, cioè gli ingressi, che sono uguali alle colonne `numberFeatures`; poi vengono definiti i `firstHiddenLayer`, cioè i nodi dello strato nascosto; infine vengono forniti gli `outputs`, cioè le uscite, che in questo caso sono in numero pari a uno, che fa riferimento a Fs o Ff. Poi viene creata la matrice di connessione con la funzione `createConnectionMatrix` che ha come argomento quattro parametri per specificare la connettività. A questo punto viene fornito il modello della feed-forward neural network con le funzioni `MyNet net` e `netMin`, la funzione di errore con `MeanSquaredError error`, l'ottimizzatore con `IRpropPlus optimizer` e vengono inizializzati i pesi con il costrutto `net.initWeights`. Il passo successivo prevede una parte di codice relativa all'apprendimento della rete neurale. Viene, quindi, creato un file `trajectory`, e definito un certo numero di cicli `numberOfLearningCycles`, che vengono svolti attraverso il costrutto `for`. Questi cicli sono il numero di volte che si vuole che la rete neurale venga implementata per l'apprendimento, in modo da abbassare l'errore. Il ciclo for, quindi, fa si che vengano eseguite le istruzioni di ottimizzazione e viene calcolato l'errore per il numero di cicli desiderato. I valori dell'errore rilevati sia per l'apprendimento che per la previsione vengono, dunque, forniti in uscita. Viene, quindi, chiuso questo file con `trajectory.close`. Infine, si ha un ciclo `for`, il quale viene eseguito per il numero di colonne `numberFeatures` e consente di fornire gli input alla rete neurale; la variabile oggetto `net` alla quale viene applicata la funzione `model`, che consente, fornendo gli input, la previsione dell'output, cioè fa si che il modello della rete neurale venga applicato ai dati in input; un altro ciclo `for` che viene eseguito sempre per il numero di colonne `numberFeatures` e serve a fornire in uscita gli input che sono stati dati alla rete neurale per la previsione, e gli output che corrispondono a quello voluto e quello ottenuto dalla previsione. Con questo si chiude il ciclo for iniziale e poi si chiude il file relativo alla previsione con `previsione.close`.

Si vede come, in questo caso, nel quale si vuole ottenere la previsione di un valore sconosciuto alla rete neurale e non più la previsione di un valore noto, il file previsione sia creato all'inizio, e venga svolto per ogni acciaio un certo numero di cicli; il funzionamento consiste, in definitiva, nell'andare a togliere una riga per volta dal file csv, per vedere se la rete neurale riesce, dato l'input di tale acciaio nella sola fase di

previsione (cioè date tutte le caratteristiche dell'acciaio tranne quella di interesse Fs o Ff), a prevedere un output che non conosce.

In definitiva, si è visto il codice di programmazione in linguaggio C++ per lo sviluppo di un sistema basato sulle reti neurali di tipo feed-forward, codice che utilizza le funzioni fornite dalla libreria Shark, e che viene sviluppato per due casi diversi. Nel primo caso si vuole far apprendere e prevedere alla rete neurale dei dati noti. Infine, nel secondo caso viene fornito il codice per la cross-evaluation, cioè si vuole far apprendere la rete neurale su dati noti e poi far prevedere ad essa dei dati sconosciuti. Quest'ultimo caso è relativo allo scopo finale della tesi, che è appunto quello di prevedere i tempi di inizio e fine precipitazione della ferrite di acciai che presentano determinate caratteristiche (quali la composizione chimica), e per i quali questi valori non sono noti.

CONCLUSIONI

In questo lavoro è stata dunque presentata la problematica da cui si è preso spunto. Sono stati descritti i metodi che si sono proposti di trovare un modello efficiente in grado di prevedere le curve CCT per un acciaio con una qualsiasi composizione chimica. Si è dimostrato come tali metodi non consentano di fare ciò, a causa delle numerose limitazioni descritte. Per questi motivi, si è deciso di sviluppare un sistema di neurale per la previsione di tali curve. Si è pensato di focalizzare l'attenzione su questo metodo proprio perché non presenta le limitazioni dei metodi precedenti ed anche perché, precedenti studi hanno potuto dimostrare la sua validità.

Il metodo delle reti neurali trattato in questo progetto di ricerca, dunque, rappresenta una promettente modalità per predire i diagrammi di trasformazione anisoterma a partire dalla semplice conoscenza della composizione chimica dell'acciaio preso in esame.

È di comune interesse verificare se esso è in grado di approssimare correttamente tali curve per qualsiasi tipo di acciaio.

Come già menzionato in introduzione, questo lavoro di tesi si è focalizzato sullo sviluppo di un sistema a reti neurali, successivo ad un lavoro di strutturazione dei dati di ingresso. Sarà, dunque, attraverso un altro lavoro che il funzionamento del sistema sviluppato potrà essere verificato.

Ciò che si è concluso con tale lavoro, pertanto, è lo sviluppo del programma di implementazione della rete neurale, che attraverso i dati raccolti, forniti in input, serve per la previsione dei tempi di inizio e fine precipitazione della ferrite.

Questo rappresenta dunque la conclusione di tale lavoro, ma è il punto di partenza per il successivo, che utilizza, come già detto, i dati raccolti e il sistema a reti neurali sviluppato.

BIBLIOGRAFIA

[1] William D.Callister Jr, *Scienze e Ingegneria dei Materiali. Una Introduzione*; EDISES

[2] Guang Xu, Lun Wan, Shengfu Yu, Li Liu e Feng Luo, 2008, " A new method for accurate plotting continuous cooling transformation curves", *Materials Letters*, vol. 62, p. 3978-3980;

[3] Yoon-Jun Kim, L. Scott Chumbley e Brian Gleeson,2007, "Continuous cooling transfmation in cast duplex stainless steels CD3MN and CD3MWCuN", *ASM International*, vol. 1059-9495;

[4] J.S. Ye, H.B. Chang e T.Y. Hsu (Xu Zuyao), 2003, "On the Application of the Additivity Rule in Pearlitic Transformation in Low Alloy Steels", *Metallurgical and Materials transactions*, vol. 34A;

[5] Mark Lusk e Herng-Jeng Jou, 1997, "On the rule of additivity in phase transformation kinetics", *Metallurgical and Materials transactions*, vol. 28A;

[6] Victor D. Fachinotti, Alberto Cardona e Andr´es A. Anca, 2005, "Solid-State microstructure evolution in steels", *Mecanica Computacional*, vol. XXIV, A. Larreteguy Editor;

[7] F. Liu , C. Yang, G. Yang e Y. Zhou, 2007, "Additivity rule, isothermal and non-isothermal transformations on the basis of an analytical transformation model", *Acta Materialia*, vol.55, p.5255-5267

[8] M. Lusk, G. Krauss e H.-J. Jou, 1995, "A Balance Principle Approach for Modeling Phase Transformation Kinetics", *Journal de physique IV*, vol.C8, p.279-284;

[9] Jean-Philippe Schillé1*, Zhanli Guo1, Nigel Saunders2, A Peter Miodownik2 e Rongshan Qin1, *SimPro'08*, 2008, Ranchi, INDIA;

[10] J.Trzaska e L.A.Dobrzanski, 2007, “Modelling of CCT diagrams for engineering and constructional steels”, Materials Processing Technology, vol. 192-193, p. 504-510;

[11] L.A. Dobrza´nski e J. Trzaska, 2004, “Application of neural networks to forecasting the CCT diagrams”, *Journal of Materials Processing Technology* , vol.157–158, p.107–113;

[12] Leszek A. Dobrzanski e Jacek Trzaska , 2004, “Application of neural networks for the prediction of continuous cooling transformation diagrams”, *Computational Materials Science*, vol. 30, p.251–259.

SITOGRAFIA

Rete neurale, “http://it.wikipedia.org/wiki/Rete_neurale”.

Shark library, “http://shark-project.sourceforge.net/”.

RINGRAZIAMENTI

Ringrazio i miei genitori per i sacrifici fatti per farmi studiare, e per il sostegno e l'incoraggiamento che mi hanno dato.
Ringrazio mie sorelle per la loro presenza, la loro vicinanza e il loro sostegno.
Ringrazio il mio fratellino per la gioia e l'entusiasmo trasmessomi.
Ringrazio Nicola per la sua vicinanza e l'aiuto datomi.
Da ultimo, ma più importante, ringrazio il Signore con Maria, che mi accompagnano sempre e che sono il fulcro della mia vita.