UNIVERSITÀ DEGLI STUDI DI TERAMO

Convolutional Neural Networks

Prof. ssa Romina Eramo
Università degli Studi di Teramo
Dipartimento di Scienze della Comunicazione
reramo@unite.it

Le Reti Neurali Convoluzionali (CNN)

- Limitazioni dei Modelli Classici:
  - Selezione manuale delle caratteristiche.
  - Incapacità di sfruttare la struttura degli input.
- Vantaggi delle CNN:
  - Apprendimento end-to-end: generano rappresentazioni e classificano.
  - Sfruttano la struttura intrinseca degli input (es. immagini, serie temporali).
- Storia:
  - Origini: dal Perceptron (Rosenblatt) all'architettura del 1998.
  - Svolta: AlexNet (2012) grazie a miglioramenti computazionali.

L'importanza della struttura negli input

- Differenze dai modelli tradizionali:
  - I modelli tradizionali trattano le caratteristiche come indipendenti.
  - L'ordine delle caratteristiche è irrilevante per i modelli classici.
- Punto di forza delle CNN:
  - Sensibili all'ordine e alla correlazione spaziale dei dati.
  - Ideali per input strutturati (es. immagini con pixel adiacenti correlati).

Esperimento con il dataset MNIST

- Due versioni di MNIST:
  - Dataset originale: cifre leggibili.
  - Dataset confuso: ordine dei pixel riorganizzato (figura).

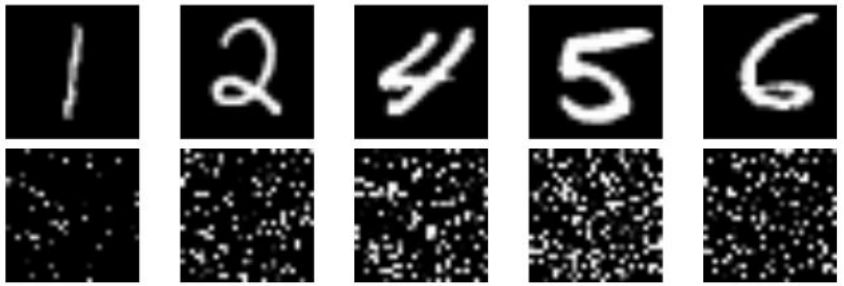


Figure 5-1: Example MNIST digits (top) and scrambled versions of the same digits (bottom)

- Prestazioni dei modelli tradizionali:
  - Trattano entrambi i dataset allo stesso modo ⟶ stesse prestazioni.
- Prestazioni delle CNN:
  - Funzionano bene sul dataset originale.
  - Prestazioni degradano sul dataset confuso ⟶ non riescono a sfruttare la struttura locale.

La relazione tra struttura e prestazioni

- Le CNN sfruttano la struttura locale per interpretare gli input.
- Dataset confusi ⟶ struttura distrutta ⟶ degradazione delle prestazioni delle CNN.
- Implicazioni per il design:
  - Le CNN imitano il modo in cui il cervello umano interpreta immagini.
  - Applicabili quando la struttura è significativa (es. immagini, video, dati volumetrici).

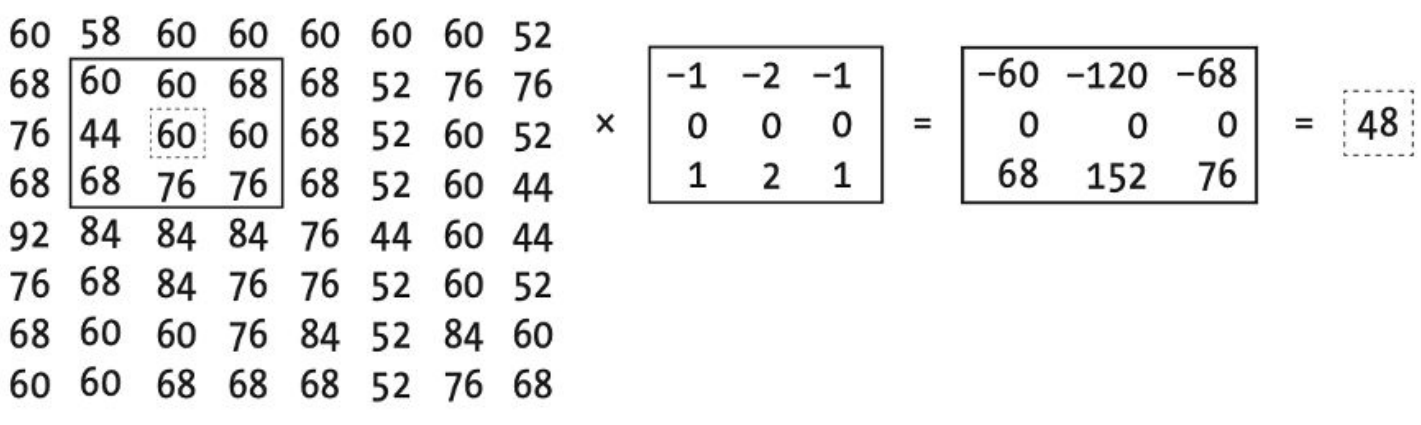
Cosa vedi nel dipinto?

Figure 5-2: Van Gogh's bedroom in Arles, 1889 (public domain)

Cos'è la Convoluzione?

- Operazione matematica essenziale per le immagini digitali.
- Definizione semplificata:
  - Un piccolo quadrato (kernel) scorre sull'immagine.
  - In ogni posizione:
    - Moltiplicazione tra valori di pixel e kernel.
    - Somma dei risultati → Nuovo valore del pixel di output.
- Processo di convoluzione:
  - Si ripete riga per riga e colonna per colonna fino a coprire l'intera immagine.

Come funziona la Convoluzione?



- **Componenti:**
  - Griglia di pixel dell'immagine (es. valori da 0 a 255 per scala di grigi).
  - Kernel (es. griglia 3×3).
- **Passaggi:**
  - Moltiplica valori dei pixel per quelli del kernel.
  - Somma i risultati per ottenere il valore del pixel di output (es. 60 → 48).
  - Ripeti scorrendo il kernel sull'immagine.
- **Risultato:** Una nuova immagine convoluta.

Applicazione: filtraggio delle immagini con convoluzione

La convoluzione può migliorare o trasformare un'immagine:

- **Sfocatura:** Kernel con valori uniformi.
- **Rilevamento bordi orizzontali:** Kernel specifico (Figura 5-3).
- **Rilevamento bordi verticali:** Kernel ruotato di 90°.

Esempio: La figura mostra:

- Immagine originale (Gold Hill).
- Versioni convolute con diversi kernel

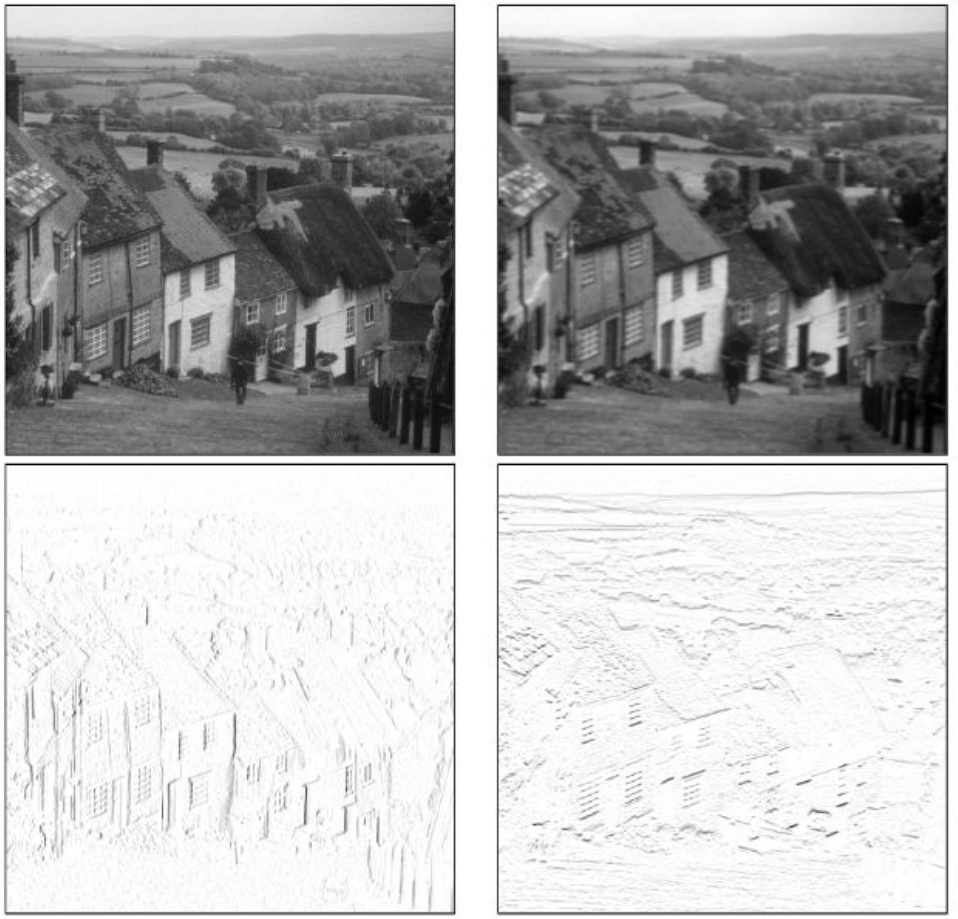


Figure 5-4: Convolution kernels in action

Perché la convoluzione è importante?

- Ogni kernel evidenzia aspetti distinti dell'immagine:
  - Bordi, orientamenti, texture, colori.
- Rilevanza per le CNN:
  - Le CNN apprendono automaticamente i kernel rilevanti per classificare immagini.
  - Processo simile al sistema visivo umano (es. rilevamento di bordi in V1).
- Ruolo della convoluzione nelle CNN:
  - Estrarre struttura rilevante dagli input.
  - Costruire nuove rappresentazioni delle immagini.
- Prossimo Passo:
  - Come le CNN utilizzano la convoluzione per classificare immagini in modo end-to-end.

Architettura delle CNN

- **Reti tradizionali**:
  - Basate su strati completamente connessi (o densi).
  - Flusso dati: input ⟶ nodi ⟶ output.
- **CNN**:
  - Struttura flessibile con più tipi di livelli:
    - **Strati convoluzionali**: Applicano kernel per estrarre caratteristiche.
    - **Strati di pooling**: Ridimensionano gli input mantenendo i valori massimi.
    - Strati densi utilizzati vicino all'output per classificare rappresentazioni trasformate.
- **Flusso dati nelle CNN**: Input ⟶ più livelli ⟶ Output.

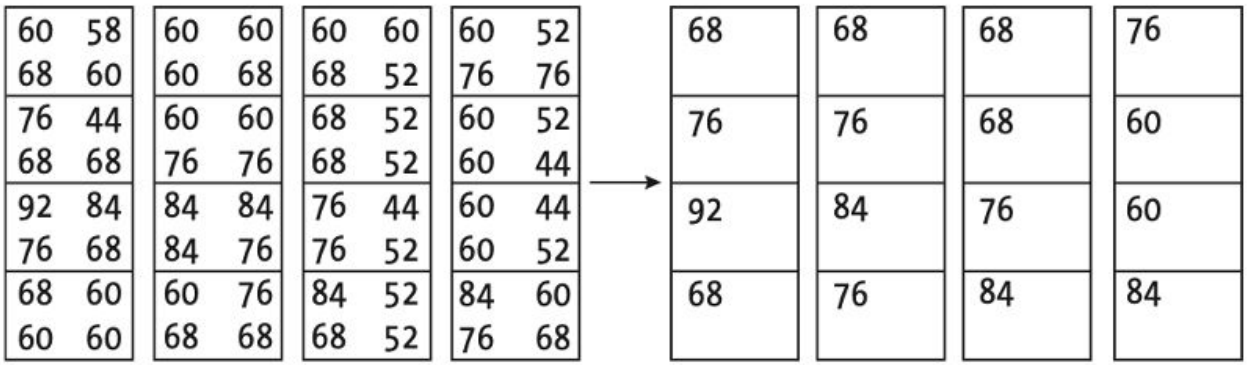
Ruolo dei livelli nelle CNN

Livelli convoluzionali:

- Applicano kernel (i cui valori sono appresi durante l'addestramento).
- Producono più output da un singolo input (Figura 5-4).
- Kernel = pesi dello strato convoluzionale appresi tramite backpropagation.

Livelli di pooling:

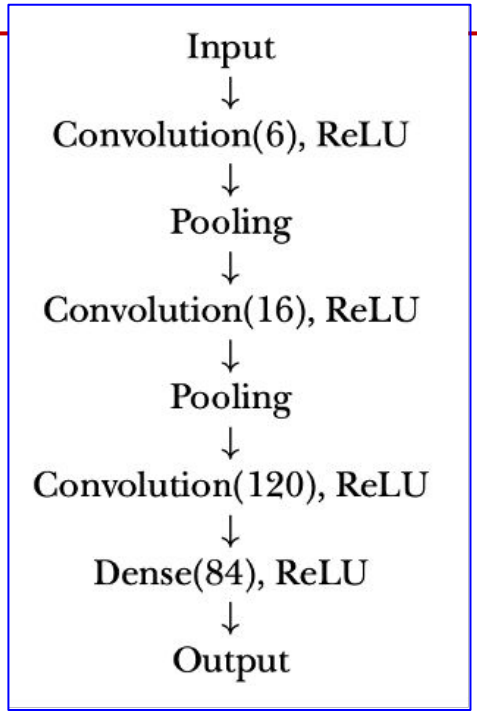
- Non contengono pesi da apprendere.
- Operazione fissa: Mantengono il valore massimo in ogni quadrato 2×2.
- Riduzione spaziale: Es. input 8×8 ⟶ output 4×4 (figura sotto).
- Vantaggi: Riduzione dei parametri e miglioramento dell'efficienza della rete.



Architettura e funzionalità CNN LeNet

Struttura tipica di LeNet:

- 3 strati convoluzionali (6, 16 e 120 filtri).
- 2 strati di pooling.
- 1 strato denso (84 nodi).
- Ogni strato convoluzionale/denso è seguito da uno strato ReLU (attivazione non lineare).

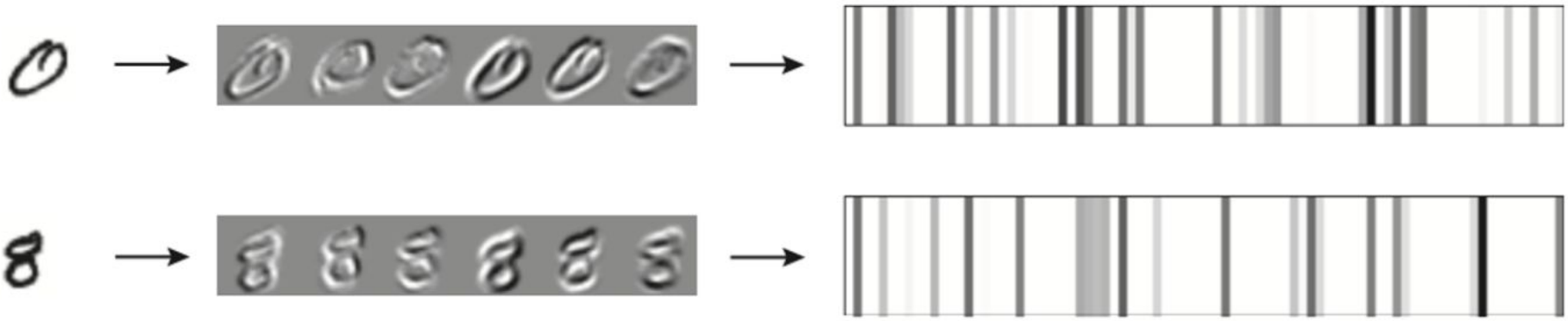


Kernel e filtri:

- Filtri = Raccolta di kernel convoluzionali.
- Esempio:
  - Primo strato: 6 filtri, 1 kernel per filtro → Totale: 6 kernel.
  - Secondo strato: 16 filtri, 6 kernel per filtro → Totale: 96 kernel.
  - Terzo strato: 120 filtri, 16 kernel per filtro → Totale: 1.920 kernel.
- Totale kernel da apprendere: 2.022.

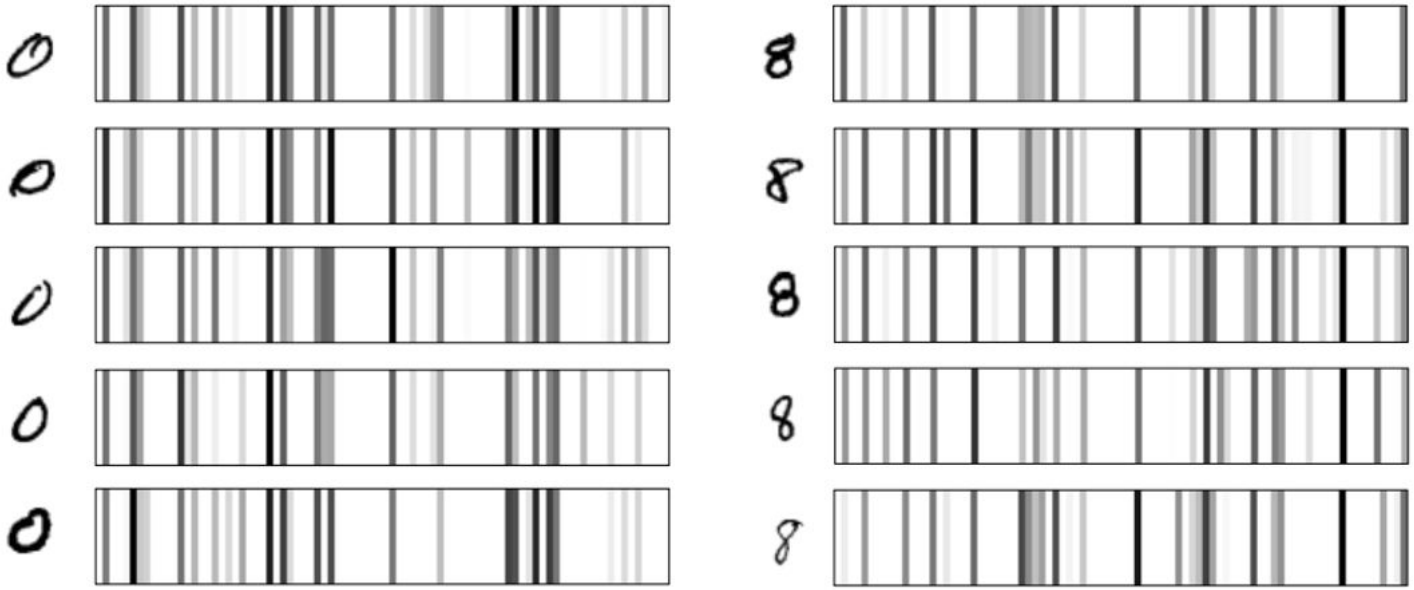
Obiettivo della CNN: Estrarre una rappresentazione che evidenzi strutture fondamentali utili alla classificazione.

Evoluzione dei dati attraverso LeNet



- Un modello LeNet addestrato su cifre MNIST manipola due immagini di input
- Output degli strati convoluzionali:
  - Primo strato: 6 immagini di output (centrali) ⟶ ogni kernel evidenzia caratteristiche diverse.
  - L'output finale dello strato convoluzionale è un vettore di caratteristiche.

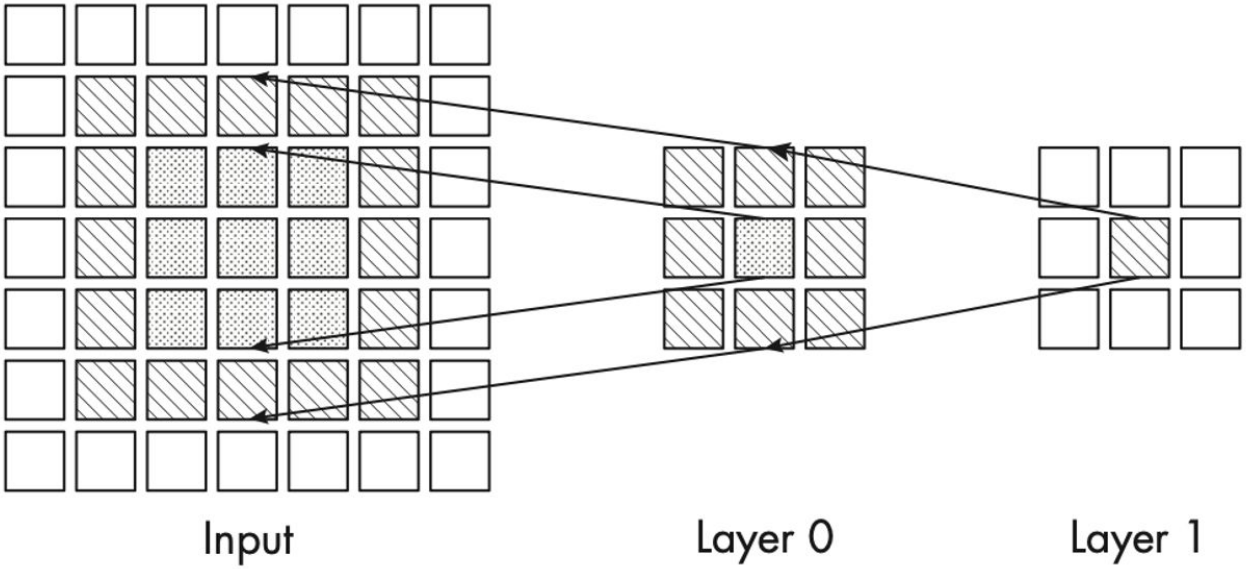
Evoluzione dei dati attraverso LeNet (2)



» Strato denso finale:

- Converte il vettore in 84 numeri (mappati come barre).
- I vettori per input simili (es. "0" e "8") condividono somiglianze strutturali.

Evoluzione dei dati attraverso LeNet (3)



- Campo recettivo:
  - Gli strati convoluzionali più profondi "vedono" porzioni più grandi dell'input:
    - Strato 1: Dipende da regioni 3×3 dell'input.
    - Strato 2: Dipende da regioni 5×5 (come mostrato in figura 5-8).
  - Effetto complessivo: Creazione di una mappa globale basata su dettagli locali.

ESEMPIO

Esperimento: versione in scala di grigi di CIFAR-10 (img reali)

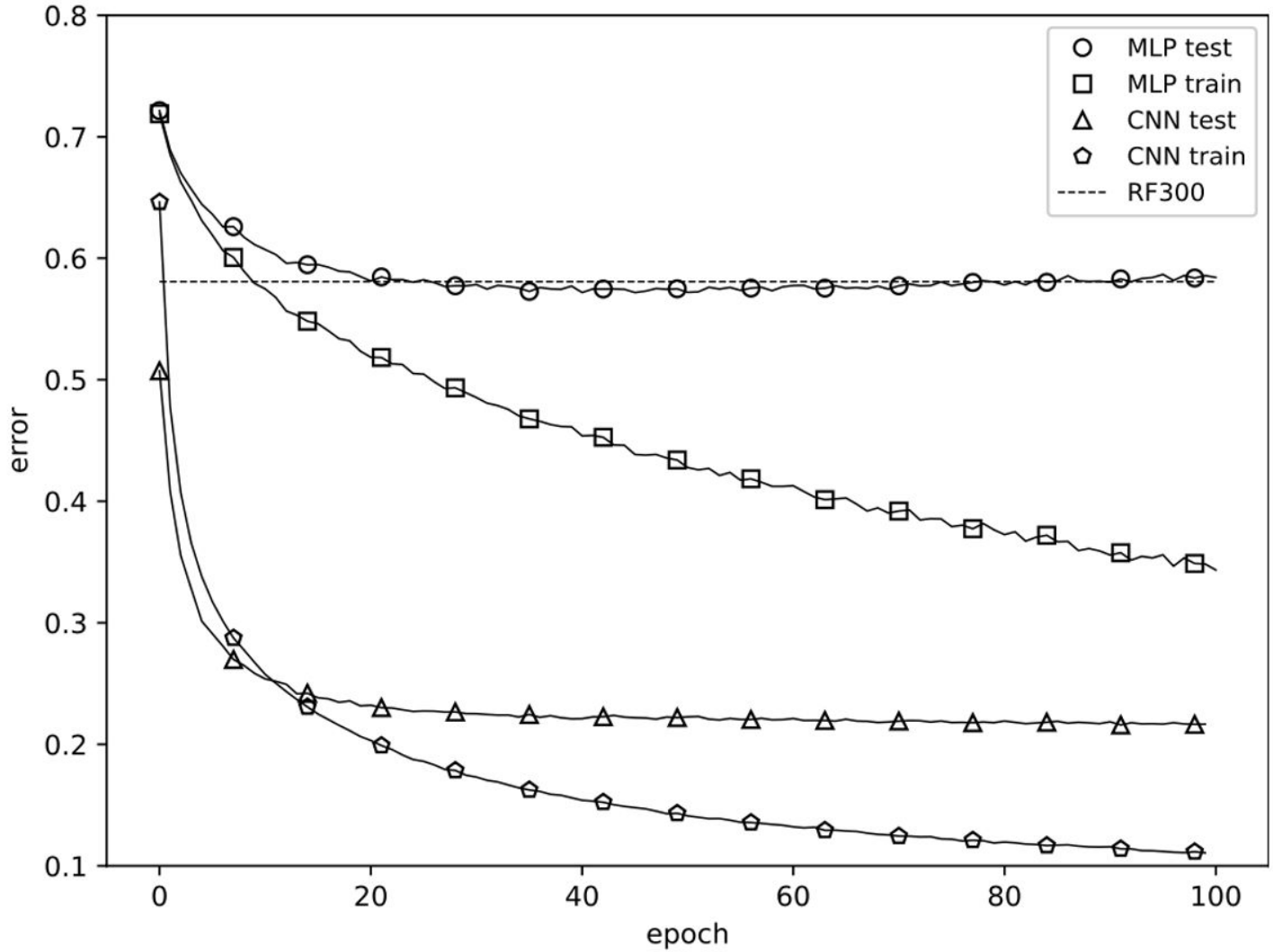
- Modelli Testati:
  - Foresta Casuale: 300 alberi.
  - Rete Neurale Tradizionale:
    - 2 livelli nascosti (512 nodi, 100 nodi).
  - Rete Neurale Convoluzionale (CNN):
    - 4 strati convoluzionali, 2 di pooling, 1 strato denso (472 nodi).
- Set di Dati:
  - Training: 50.000 immagini (10 classi, 5.000/class).
  - Test: 10.000 immagini (10 classi, 1.000/class).
  - Input:
    - Foresta Casuale & Rete Tradizionale: Immagini 32×32 → Vettore di 1.024 numeri.
    - CNN: Immagini bidimensionali.

ESEMPIO

Esperimento: versione in scala di grigi di CIFAR-10 (img reali)

- Addestramento:
  - Ogni modello addestrato 10 volte, risultati mediati per compensare la casualità.
  - Reti Neurali → Addestramento per 100 epoche.
    - Batch size: 200 (250 passi per epoca).
    - Aggiornamenti totali: 25.000.
- Monitoraggio dell'Errore:
  - Errori raccolti su set di training e test al termine di ogni epoca.
- Grafico Finale:
  - Confronta l'andamento dell'errore tra i tre modelli durante l'addestramento.
  - Rivela insight sulle prestazioni, complessità e comportamento dei modelli nel tempo.

Esperimento: versione in scala di grigi di CIFAR-10 (img reali)



ESEMPIO

Esperimento: versione in scala di grigi di CIFAR-10 (img reali)

Obiettivo del Grafico:

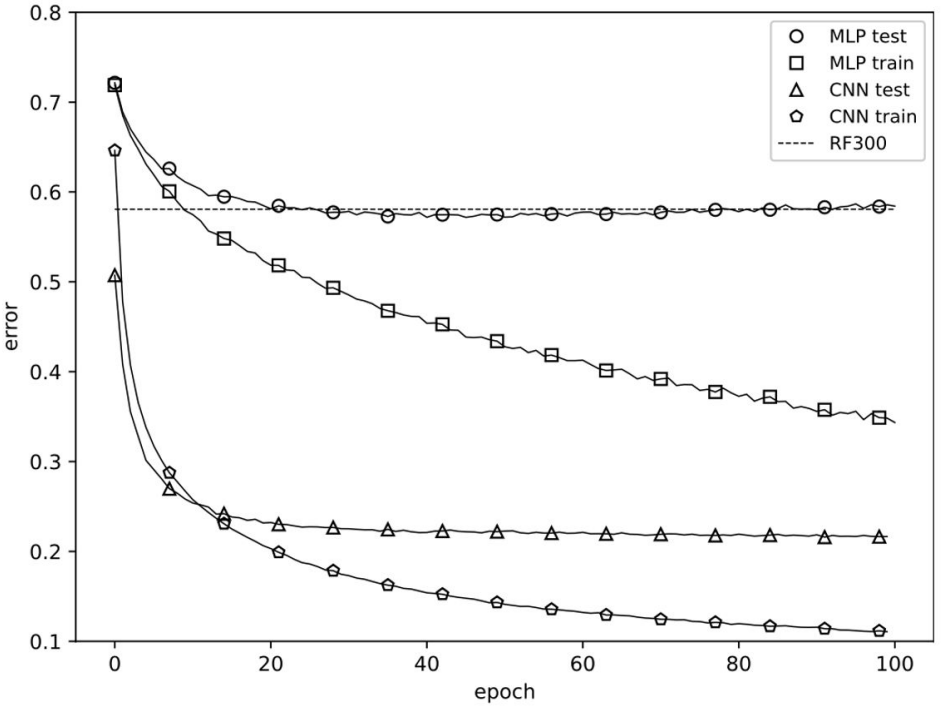
» Analizzare come cambiano le prestazioni dei modelli durante il training.

Assi del Grafico:

» **Asse x**: Epoche (passaggi completi attraverso il set di training).

» **Asse y**: Errore (frazione di predizioni errate).

- 0,1 (10%) ⟶ Migliore
- 0,8 (80%) ⟶ Peggiore



Legenda:

» **MLP** (Rete Neurale Tradizionale):

- Cerchi (Test) e Quadrati (Training).

» **CNN**:

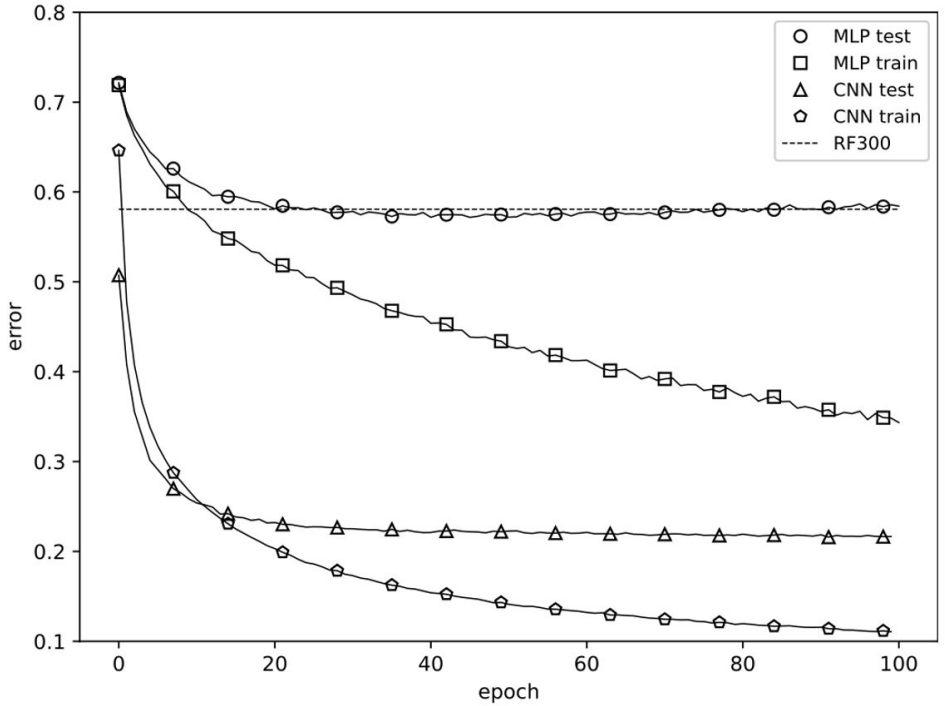
- Triangoli (Test) e Pentagoni (Training).

» **RF300** (Foresta Casuale):

- Linea tratteggiata fissa (Errore test: 58%).

Esperimento: CIFAR-10
Risultati della CNN

- **Errore sul Set di Training:**
  - Diminuisce fino all'**11%** entro 100 epoche.
- **Errore sul Set di Test:**
  - Si stabilizza a **23%** (accuratezza 77%).
  - **Molto superiore** rispetto a MLP e Foresta Casuale.

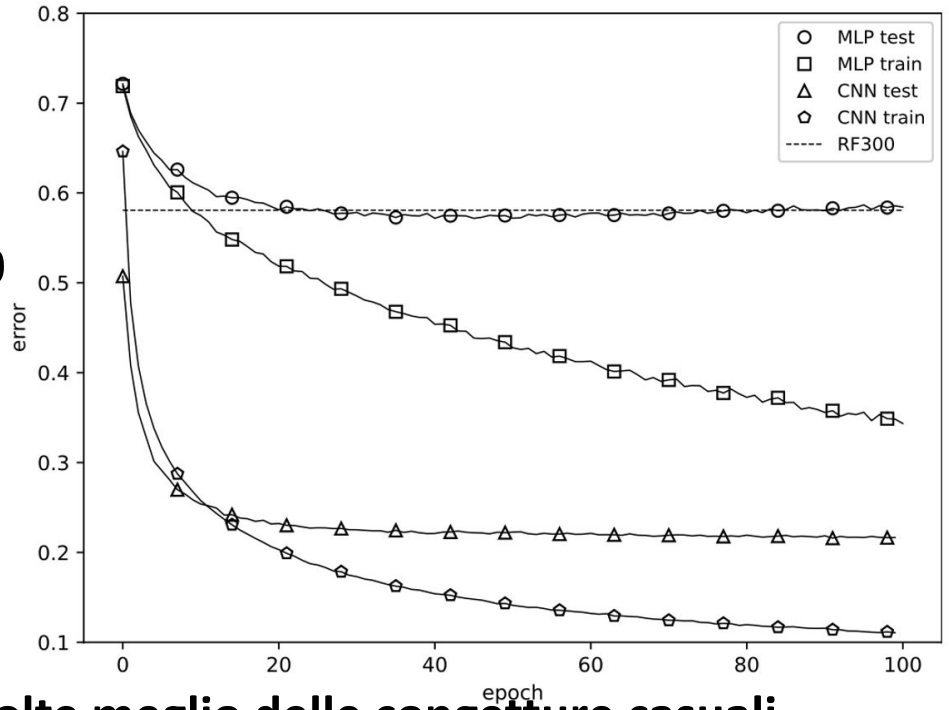


- **Motivo del Successo:**
  - La CNN impara a rappresentare parti rilevanti dell'immagine.
  - Nuove rappresentazioni migliorano la classificazione finale.

ESEMPIO

Esperimento: CIFAR-10 Confronto Modelli:

- **Foresta Casuale:** Errore fisso 58% ⟶ Limiti dell'apprendimento classico.
- **MLP:** Migliore performance a 40 epoche, ma vulnerabile all'overfitting.
- **CNN:**
  - **Errore Test:** 23% ⟶ Chiaro vantaggio su dati visivi complessi.
  - Accuratezza: 77%, quasi **8 volte meglio delle congetture casuali (10%)**.
- **Impatto del Paradigma CNN:**
  - Cambiamento di paradigma nel riconoscimento visivo.
  - Esempio reale: Identificazione di aeroplani satellitari con precisione migliorata.

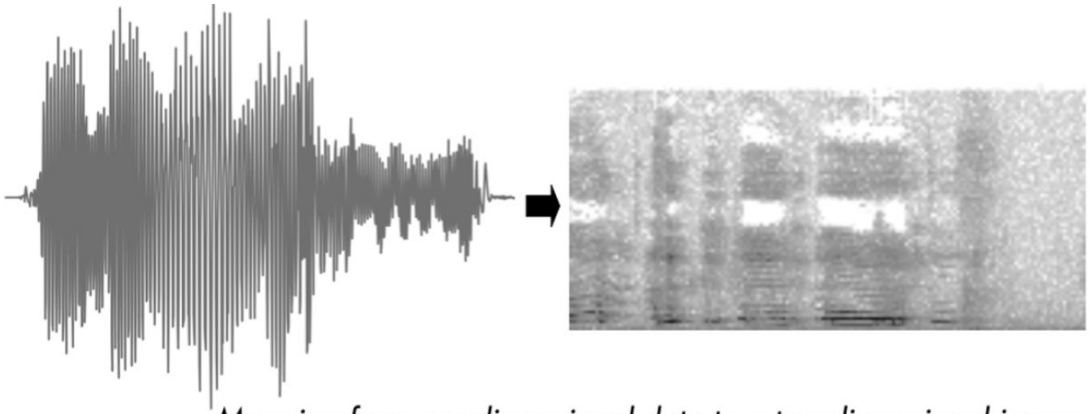


Lo Zoo delle Architetture CNN

- **Evoluzione delle CNN**
  - Un decennio di sviluppo ha prodotto architetture con oltre 100 strati:
    - **ResNet, DenseNet, Inception, MobileNet, U-Net**, ecc.
- Architetture Specializzate
  - U-Net:
    - **Segmentazione semantica**: Etichetta ogni pixel di un'immagine con una classe.
    - Esempio: Identificazione precisa di un oggetto (come un cane) per pixel.
  - **YOLO e Faster R-CNN**:
    - Producono **riquadri di delimitazione** per identificare e localizzare oggetti.
- Diversi Formati di Input
  - Non solo immagini!
    - Audio (spettrogrammi): Trasformare segnali audio 1D in rappresentazioni 2D.
    - Qualsiasi dato con **struttura 2D** è adatto per una CNN.

Spettrogrammi e CNN 2D
Trasformare il Segnale Audio

- L'audio unidimensionale può essere rappresentato come uno spettrogramma 2D:
  - Asse x: Tempo
  - Asse y: Frequenze (basse → alte)
  - Intensità: Colore/pixel



Mapping from one-dimensional data to a two-dimensional image

- Vantaggi dello spettrogramma:
  - Contiene ricche informazioni strutturali (es. pianto di un bambino).
  - Permette alle CNN di apprendere modelli complessi, migliorando le previsioni rispetto al solo segnale audio 1D.
- Qualsiasi **trasformazione** che estrae struttura dai dati e li adatta a una CNN è valida.

Creare una CNN e l'Avvento di AutoML

Sfide nella progettazione delle CNN, decisioni iniziali fondamentali:

- Architettura: Quanti strati e in quale ordine?
- Parametri: Kernel 3×3 o 5×5? Dimensione del minibatch?
- Training: Numero di epoche ottimale?
- Un tempo, richiedeva esperienza, intuizione e scienza, processo complesso e altamente specialistico.

AutoML: automatizzare il Machine Learning

- Strumenti AutoML (es. Azure Machine Learning, SageMaker Autopilot):
- Automatizzano la selezione del modello e l'ottimizzazione degli iperparametri.
- Applicabili sia alle reti neurali sia ai modelli classici.
- Obiettivo: Identificare il miglior modello con minima esperienza utente.

Il dibattito: **AutoML vs. progettazione manuale**

- AutoML offre comodità ma ha limiti:
  - I professionisti del deep learning possono superare AutoML in scenari complessi.
  - Analogie storiche: Programmatori assembly vs. linguaggi compilati.
- Preferenze personali: Alcuni scelgono ancora la creazione manuale dei modelli.

Toolkit Open Source e Differenziazione Automatica

- Passaggi chiave per l'**addestramento di Reti Neurali**: **backpropagation e gradient dicendent**.
  - La **differenziazione** è essenziale per calcolare i derivati richiesti.
  - Toolkit trasformano la rete in un **grafo computazionale** per gestire queste operazioni.
- **Differenziazione automatica: approcci**
  - **Avanti**:
    - Più semplice da implementare (es. numeri duali, introdotti nel 1873).
    - Non ideale per reti neurali.
  - **Indietro**:
    - Ottimale per reti neurali.
    - Permette di gestire modelli complessi con maggiore efficienza.
- **Conclusione**
  - Toolkit open source e differenziazione automatica sono pilastri dello sviluppo delle moderne CNN e del deep learning.

Riassumendo, sulle CNN:

- Prosperare sulla struttura dei loro input, che è l'esatto opposto dei classici modelli di apprendimento automatico
- Apprendere nuove rappresentazioni dei loro input suddividendoli in parti e gruppi di parti
- Usa molti tipi diversi di strati combinati in vari modi
- Può classificare gli input, localizzare gli input o assegnare un'etichetta di classe a ogni pixel nei loro input
- Sono ancora addestrati tramite retropropagazione e discesa del gradiente, proprio come le reti neurali tradizionali
- Ha guidato la creazione di potenti toolkit open source che hanno democratizzato il deep learning

Le CNN seguono la tradizione dei classici modelli di apprendimento meccanico: prendono un input e gli assegnano, in qualche modo, un'etichetta di classe. La rete funziona come una funzione matematica, accettando un input e producendo un output.