Costantini Samuele 332653 - Hani Belal 321158 - CdL Informatica Triennale Unipg

Relazione progetto Uniform Coloring - Introduzione all'Intelligenza Artificiale

Struttura del progetto
Il progetto è stato strutturato orientato alla modularità quindi ogni funzione del sistema è richiamabile ed usabile separatamente dal flusso principale di esecuzione del progetto.

Librerie utilizzate:

1. **Aima**: search.py e utils.py
2. **PIL**: manipolazione immagini
3. **numpy**: calcolo numerico
4. **tensorflow**: machine learning
5. **matplotlib**: visualizzazione dei dati
6. **OpenCv (cv2)**: computer vision
7. **tempfile**: file temporanei

Il progetto è suddiviso principalmente nelle seguenti parti:

1. **Training del modello:**
   - **training-modello.ipynb**: file Jupyter Notebook usato per tenere traccia del training
   - **letter_reconition_model.h5**: modello vero e proprio

2. **Preprocessing immagine e implementazione modello:**
   - AiTextExtractorService.py: classe che implementa tutte le funzioni di preprocessing, di mapping e dell'implementazione vera e propria del modello Formata da:
     - **mapPredictedClassToLetter(classValue):** utilizzata per mappare classe a lettera effettiva
     - **removeColorRange(img_array, start_hex, end_hex)**: rimuove un certo range di colori (non utilizzata)
     - **autoCropLetter(img_array):** sfruttando la Computer Vision di OpenCv ritaglia l'immagine mantenendo le proporzioni della lettera e centrandola in un box quadrato, così da convertirla successivamente in 28x28 per il modello
     - **analyzeImage(imagePath)**: data un'immagine fisica, la trasforma in array di bit e con autoCropLetter ed altre operazioni di preprocessing che approfondiremo dopo, implementa il modello di classificazione per estrapolare la classe di una sola lettera
     - **isolateLettersFromGrid(image_path)**: partendo da un'immagine di una griglia, con OpenCv, riconosce i contorni delle lettere e riesce a restituire un array di lettere con numero di righe e numero di colonne individuate
     - **predictGridLetters(isolatedLetters)**: prendendo l'array di lettere, usando file temporanei, ritorna l'array di classi (lettere estratte) utilizzando analyzeImage
     - **runGridExtraction(image_path)**: riassume tutte le funzioni della classe

3. **Conversione griglia estratta in problema di ricerca Aima**
    - GridProblem.py: classe che definisce tutte gli attributi e le funzioni utili per la creazione di un search problem Aima
        - **initial:** contiene l'initial state ed è una tupla formata da una tupla con la griglia e una tupla con le coordinate
        - **goal_color:** il colore con cui colorare tutta la griglia che può essere un colore della griglia oppure 'a' che prende il colore con più occorrenze, a prescindere dal costo
        - **start_position:** una tupla con le coordinate
        - **color_cost:** impostazione dei costi dei colori nell'ordine g, y, b
        - **rows:** numero righe
        - **cols:** numero colonne
        - **letters:** array raw delle lettere
        - **actions(state)**: definizione delle azioni
        - **result(state, action)**: ritorna il nuovo stato data un azione
        - **goal_test (state)**: riconosce se lo stato è quello goal
        - **path_cost (c, action)**: ritorna il costo in base all'attributo color_cost, assegna un costo ad ogni paint di un colore specifico
        - **h (node)**: definizione di un euristica

4. **Classe principale che contiene l'esecuzione del programma:**
    - UniformColoring.py: La classe main contiene l'istanza delle varie classi descritte prima, l'import delle classi di Aima per la ricerca, simulazioni delle varie ricerche e stampe sull'andamento di ogni singola ricerca.

5. **Directory con tutte le varie immagini ricavate durante i processi**

Costantini Samuele 332653 - Hani Belal 321158 - CdL Informatica Triennale Unipg

Descrizione funzionamento

Per il modello di classificazione abbiamo optato per una CNN (Convolutional Neural Network).

Convolutional networks *are used to process data organized in a grid-like structure, such as images and videos. The* ***convolution operation*** *is crucial because it helps identify interesting parts of an image, such as object edges.*

These networks possess the following properties:

1. Parameter Sharing: This involves using the same parameter for multiple units within the same network.

2. Sparse Interactions: Each input influences only a limited number of outputs, and each output is affected by a limited number of inputs. Typically, each input affects all outputs, and all outputs are influenced by all inputs.

3. Equivariant Representations: The convolution operation is translation-invariant, a consequence of parameter sharing.

Fonte: [Deep Learning and EMNIST: How to use a Convolutional Neural Network for image recognition | by Marco Speciale | Medium]

Dopo aver importato e normalizzato il dataset “**emnist-letters.mat**”, abbiamo usato Keras per il training della CNN con la seguente definizione:

```
model = keras.Sequential([
    keras.Input(shape=(28, 28, 1)),  # Input layer con immagini 28x28 in scala di grigi
    layers.Conv2D(32, (3,3), activation='relu'), # Prima convoluzione
    layers.MaxPooling2D((2,2)), # Max pooling, riduzione della dimensione
    layers.Conv2D(64, (3,3), activation='relu'), # Seconda convoluzione
    layers.MaxPooling2D((2,2)),# Max pooling, ulteriore riduzione della dimensione
    layers.Conv2D(128, (3,3), activation='relu'),# Terza convoluzione
    layers.Flatten(),# Appiattimento dei dati per il fully connected layer
    layers.Dense(128, activation='relu'),# Fully connected layer
    layers.Dense(26, activation='softmax')  # 26 classi, una per lettera
])
```

Model summary:

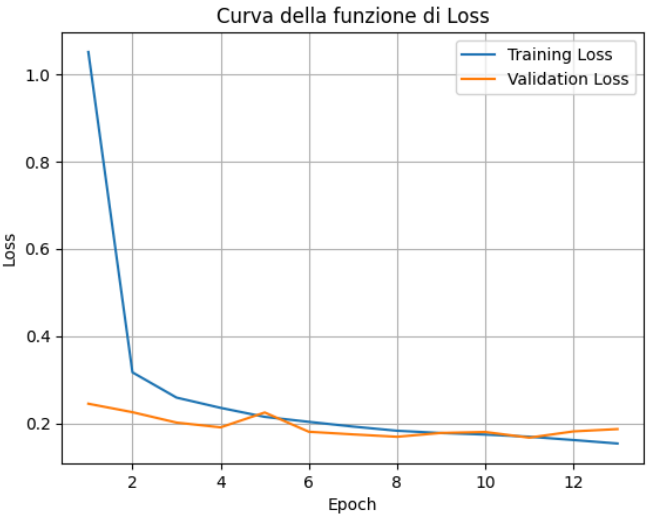
```
Dimensioni train_images: (124800, 28, 28, 1)
Dimensioni train_labels: (124800, 1)
Dimensioni test_images: (20800, 28, 28, 1)
Dimensioni test_labels: (20800, 1)

Model: "sequential"
```

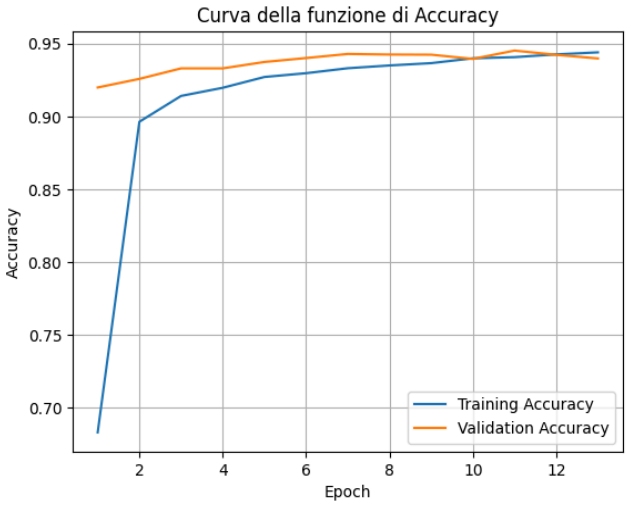
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73,856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 128)	147,584
dense_1 (Dense)	(None, 26)	3,354

```
Total params: 243,610 (951.60 KB)

Trainable params: 243,610 (951.60 KB)

Non-trainable params: 0 (0.00 B)
```

Andamento training modello:

```
Epoch 1/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 57s 28ms/step - accuracy: 0.6833 - loss: 1.0504 - val_accuracy: 0.9198 - val_loss: 0.2452
Epoch 2/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 70s 36ms/step - accuracy: 0.8963 - loss: 0.3170 - val_accuracy: 0.9257 - val_loss: 0.2258
Epoch 3/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 70s 36ms/step - accuracy: 0.9140 - loss: 0.2592 - val_accuracy: 0.9329 - val_loss: 0.2020
Epoch 4/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 78s 40ms/step - accuracy: 0.9196 - loss: 0.2357 - val_accuracy: 0.9373 - val_loss: 0.1911
Epoch 5/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 66s 34ms/step - accuracy: 0.9270 - loss: 0.2151 - val_accuracy: 0.9261 - val_loss: 0.2251
Epoch 6/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 74s 38ms/step - accuracy: 0.9296 - loss: 0.2037 - val_accuracy: 0.9400 - val_loss: 0.1810
Epoch 7/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 82s 42ms/step - accuracy: 0.9330 - loss: 0.1929 - val_accuracy: 0.9428 - val_loss: 0.1750
Epoch 8/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 84s 43ms/step - accuracy: 0.9349 - loss: 0.1833 - val_accuracy: 0.9424 - val_loss: 0.1695
Epoch 9/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 77s 39ms/step - accuracy: 0.9365 - loss: 0.1781 - val_accuracy: 0.9423 - val_loss: 0.1783
Epoch 10/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 78s 40ms/step - accuracy: 0.9398 - loss: 0.1746 - val_accuracy: 0.9394 - val_loss: 0.1805
Epoch 11/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 75s 38ms/step - accuracy: 0.9406 - loss: 0.1696 - val_accuracy: 0.9451 - val_loss: 0.1675
Epoch 12/15
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 74s 38ms/step - accuracy: 0.9425 - loss: 0.1621 - val_accuracy: 0.9422 - val_loss: 0.1818
Epoch 13/15
...
1950/1950 ━━━━━━━━━━━━━━━━━━━━ 67s 34ms/step - accuracy: 0.9439 - loss: 0.1541 - val_accuracy: 0.9397 - val_loss: 0.1871
650/650 - 3s - 4ms/step - accuracy: 0.9397 - loss: 0.1871

Accuratezza sul test set: 0.9396634697914124
```



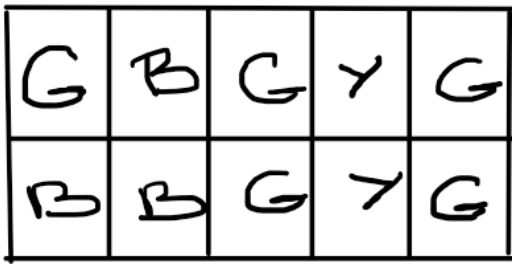


Possiamo notare l'assenza di overfitting o underfitting da questo deduciamo che il modello sia stabile.
Per aumentare le dimensioni del dataset abbiamo anche adottato delle tecniche di data augmentation.

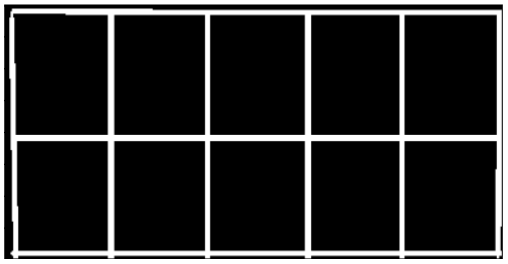
CLASSIFICAZIONE:

Il processo di classificazione consiste nell'affidare una classe ad ogni lettera estratta dall'immagine.Come input viene passata una foto di una griglia, che rappresenterà lo stato iniziale del problema, sull'immagine viene svolto un lavoro di preprocessing, con delle funzioni di computer vision, per rendere l'immagine, oltre che nel formato 28x28 scala grigi, più simile alle immagini del dataset con cui abbiamo allenato il modello, così da ridurre l'errore.

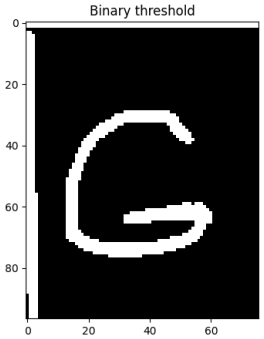
Fasi preprocessing dell'immagine

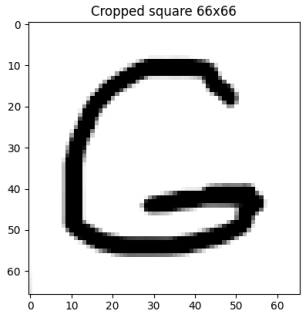


Griglia originale



Individuazione bordi





Ritaglio della lettera

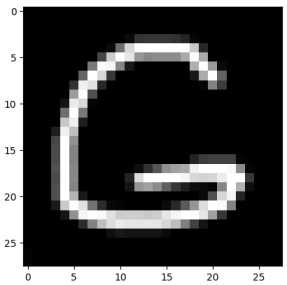


Immagine pronta per il modello di classificazione

Descrizione problema di ricerca

Il dominio del problema è colorare tutte le celle colorate dello stesso colore;
In partenza le celle della nostra griglia saranno colorate con 3 colori diversi, e l'obiettivo è far sì che la testina (che parte da una cella iniziale) si sposti nelle celle in cui va cambiato il colore.

VINCOLI:
v1=la testina può muoversi solamente al interno della griglia e non al di fuori di essa
v2=l'agente può compiere un solo passo alla volta
v3=la testina si può spostare su tutte le celle confinanti;
v4=Quindi in determinate celle possiamo fare solamente determinati movimenti;
v5=Si colora la cella solamente se non presenta il colore con cui si deve colorare la griglia.

DESCRIZIONE DEL PROBLEMA:

STATE SPACE (S): Rappresenta tutti i possibili stati che la griglia può assumere.
possiamo rappresentarlo con la tupla: (C(x,y))
C: configurazione della griglia(una di tutte le possibili configurazioni possibili)
(x,y): coordinate della posizione della testina

INITIAL:
Rappresenta lo stato iniziale, quando la testina di trova nella posizione C(0,0)

GOAL TEST: si raggiunge il GOAL TEST quando tutte le celle (eccetto la prima) sono state colorate dello stesso colore.

ACTIONS:
Up: Sposta la testina di una cella verso l'alto, se possibile.
Down: Sposta la testina di una cella verso il basso, se possibile.
Left: Sposta la testina di una cella a sinistra, se possibile.
Right: Sposta la testina di una cella a destra, se possibile.
Paint: Colora la cella corrente con un colore specificato/sceglie il colore

RESULT: la funzione result restituisce un nuovo stato dopo che è stata eseguita una certa azione, tra cui:
Spostare la posizione della testina
Cambiare il colore
Fino a che non verrà eseguita una di queste azioni, non ci sarà nessun cambiamento di stato

PATH COST:
Questa funzione serve a calcolare il costo necessario per arrivare ad un certo stato partendo dal initial state; il calcolo del costo è la somma di tutti i costi necessari ad arrivare allo stato in questione.

STEP COST:
OGNI AZIONE HA UN COSTO PRESTABILITO.
1)Muoversi da una casella al altra costa esattamente 1
2)Colorare le celle ha diversi costi:
a)Colorarla di Blu costa 1
b)Colorarla di Giallo costa 2
c)Colorarla di Verde costa 3

Ogni combinazione di colori della griglia è uno stato, ogni movimento o azione della testina modifica lo stato.

Nello stato iniziale avremo la testina posizionata nella posizione iniziale, con la griglia colorata di colori diversi.

Bisogna raggiungere lo stato finale, ovvero colorare la griglia di colori diversi.
Avremo la modifica dello stato nel momento in cui viene spostata la testina oppure viene cambiato il colore.

La nostra griglia verrà colorata in base al colore più presente;
Una volta scelto il colore con cui colorare la griglia, avremo a disposizione 4 algoritmi di ricerca per confrontarli e stabilire quale sia il più conveniente da usare.

DOPO CHE SI TROVA IL DOMINIO C'È L'IMPLEMENTAZIONE DEI 4 ALGORITMI DI RICERCA:

1)UCS: funziona esplorando i nodi in ordine di costo cumulativo crescente, di modo che che quando raggiunge una soluzione, il percorso trovato sia effettivamente
quello con il costo totale più basso possibile.
QUINDI VIENE UTILIZZATO PER TROVARE UN PERCORSO A COSTO MINIMO IN UNO SPAZIO DI STATI.

2)DFS: E' chiamata ricerca in profondità perché segue la traiettoria iniziale e continua ad esplorarla. E' molto utile per esplorare alberi di grande dimensioni;

Sicuramente ha dei costi più elevati rispetto ad UCS, per questo è consigliato usarlo quando non si deve tener conto del percorso meno costoso.

3)A*:è uno dei più efficienti per la ricerca informata di percorsi ottimali all'interno di uno spazio di stati. Combina la ricerca in Ampiezza con l'euristica.
l'euristica utilizzata calcola il costo di colorazione delle celle non colorate correttamente e la distanza massima dalla testina alla cella più lontana.
Questa euristica è progettata per stimare il costo totale necessario per raggiungere lo stato obiettivo
g(n): il costo reale dal punto di partenza allo stato
h(n): una stima euristica del costo dal nodo

f(n) = g(n) + h(n)
f(n)=g(n)+h(n): la somma dei due, che determina la priorità del nodo nella coda.

4)GREEDY BEST:Il Greedy Best-First Search è una strategia di ricerca informata che sceglie di espandere, a ogni passo, lo stato che sembra più vicino all'obiettivo secondo una funzione euristica h(n).
L'algoritmo non prende in considerazione il costo del percorso già seguito, ma valuta solo la "distanza stimata" dall'obiettivo secondo l'euristica.
La differenza con A* è che non usa g(n), ma solamente h(n).

OSSERVAZIONI: nel nostro caso UCS e A* sono i migliori algoritmi di ricerca a pari merito. DFS e GREEDY FIRST SEARCH impiegherebbero un costo maggiore per colorare la griglia.

ALGORITMO	LUNGHEZZA	COSTO
UCS	12	20
A*	12	20
GREEDY	20	28
DFS	22	30

E INVECE QUANTO SONO VELOCI I NOSTRI ALGORITMI?

ALGORITMO:	TEMPO DI ESECUZIONE:
GREEDY	0.003357 secondi
A*	0.005709 secondi
UCS	0.009089 secondi
DFS	0.011154 secondi

IN QUESTO CASO E' IL GREEDY BEST FIRST AD ESSERE IL MIGLIORE

PERCHE?
Non dovendo valutare il costo reale (COME FA A*) impiega meno tempo ad essere eseguito