Solucionador de Sudoku amb CNNs i processament d'imatge

Oriol Graupera Serra, Josep Bravo Bravo

Abstract— En aquest informe tècnic es mostra el desenvolupament d'un solucionador de Sudoku, on donat una imatge que contingui un Sudoku d'input, et tornarà una imatge idèntica, però amb la solució superposada on abans eren caselles buides.

S'explica tot el processament d'imatge realitzat en les diferents etapes del flux, per tal d'aconseguir els millors resultats en predir amb CNNs i el OCR.

S'explica i s'il·lustra el processament de les imatges a nivell de la imatge global, la imatge retallada amb només el Sudoku, i a escala de les caselles amb un sol dígit.

També s'explica l'obtenció dels models CNN, així com les dades amb les quals treballen i les utilitzades pel seu entrenament, i eines per fer inferència a aquests models.

Finalment, un cop feta la predicció de dígits, calculem i mostrem la confidència de les prediccions, així com el accuracy obtingut comparant amb el Ground Truth, i els passos realitzats per a realitzar la superposició del resultat a la imatge inicial.

1 Introducció

Duent a terme aquest projecte volem aconseguir solucionar un sudoku a partir d'una imatge. El nostre input consistirà en qualsevol fotografia d'un sudoku, la seva procedència pot ser del diari, d'un llibre o inclùs feta a mà. Utilitzant aquesta imatge volem aconseguir una matriu de nombres que reprodueixi el sudoku. Amb aquesta representació del sudoku serem capaços d'obtenir la seva solució utilitzant un algorisme d'Intel·ligència artificial. Finalment obtindrem la imatge inicial amb tots els nombres de la solució inserits.

Quan ens van proposar la idea de desenvolupar un projecte utilitzant la visió per computador, vam pensar a fer la resolució d'un sudoku. Som aficionats a aquest passatemps i ens va semblar molt interessant poder obtenir una imatge d'un sudoku solucionat a partir d'una fotografia del mateix sense solucionar. Analitzant la idea vam observar que podríem aplicar continguts explicats al llarg del curs reforçant així els coneixements i també tindrem la possibilitat d'aprendre moltes coses noves.

2 Estat de l'art

Aquest projecte tracta de dues parts molt diferenciades, la Visió per Computador encarregada de tractament de les imatges, lectura de dígits i tauler, reconstrucció de la imatge, i la part de IA encarregada de resoldre el Sudoku.

Seguidament s'explicaran les diferents tècniques utilitzades actualment per poder desenvolupar la visió per Computador i la Intel·ligència artificial:

2.1 Intel·ligència artificial

Són moltes les diferents IA's capaces de resoldre Sudokus. La gran majoria de tècniques utilitzades per a resoldre Sudokus es basen a fer una cerca exhaustiva de totes les combinacions possibles fins a trobar una que sigui solució.

Fer aquests tipus de cerca és molt viable en aquest problema, ja que la complexitat que ens dóna un tauler 9x9 del Sudoku no és molt elevada i els ordinadors d'avui en dia tenen capacitat d'executar-ho amb facilitat. Dins aquest tipus de tècniques també trobem diferents maneres de fer-ho. Alguns algoritmes utilitzats en aquest tipus de problema són: Backtracking, Branch and Bound... En el nostre projecte utilitzarem Backtracking.

2.2 Visió per Computador

Després de buscar informació sobre quines possibilitats hi ha a l'hora de realitzar aquest projecte, hem trobat diverses solucions que utilitzen tècniques diferents. El projecte es pot dividir en tres parts ben diferenciades i a continuació es comentaran les tècniques que més es fan servir per realitzar-les.

2.2.1 Preprocessament de la imatge

En el preprocessament de la imatge volem passar d'una imatge que conté un Sudoku en qualsevol format, a una imatge que contingui exclusivament el tauler del Sudoku ben retallat, i amb el millor format possible per a fer la lectura d'aquest.

Per a fer aquest preprocessament de la imatge la majoria de projectes relacionats amb la lectura d'un tauler utilitzen les tècniques següents:

- Transformacions geomètriques: En algunes imatges el Sudoku fotografiat estarà rotat o amb una vista que no sigui de planta. Per a solucionar això s'utilitzen diferents tècniques que apliquen transformacions geomètriques com Skew Correction [7] per a orientar bé la imatge, o homografies en cas de tenir imatges molt distorsionades [8].

- Eliminació de soroll: Es possible que les imatges tinguin soroll, en aquest cas s'haurà d'eliminar per a poder binaritzar la imatge. Per a tractar el soroll es poden implementar diferents tipus de filtres com: *Median Filter*, *Mean Filter*, *Gaussian Filter*... [6]

- Binarització: Per poder processar de manera més senzilla la imatge es transforma la matge en color a una imatge en blanc i negre. En aquesta tècnica és molt important trobar un bon threshold que independentment de la qualitat de la imatge i les ombres, aconsegueixi binaritzar la imatge sense perdre informació.

- Vores: Per a poder trobar únicament el sudoku s'utilitzen diferents tècniques per aconseguir les vores del tauler, com per exemple Canny, Sobel... [3].

- Morfologia Binària: En aquest apartat tenim un seguit d'operacions que ens ajudaran a tractar la imatge binària per

ressaltar la informació que volem llegir [4]. Algunes de les operacions que es poden utilitzar són les següents:

- Erosion and dilation
- Opening and closing
- Thinning and Thickenning
- Skeleton

2.2.2 Lectura del tauler

La lectura del tauler consisteix a obtenir els números i les caselles a partir de la imatge processada. Hi ha diferents maneres d'obtenir els números dels taulers:

- OCR (Optical Character Recognition): El OCR serà l'encarregat d'obtenir un text a partir de la imatge processada. La imatge es divideix en les caselles que té el sudoku, i s'utilitza el OCR per detectar si en la casella corresponent hi ha un dígit o no, en cas que si n'hi hagi un, haurà d'identificar quin és. Aquesta implementació és més senzilla, ja que hi ha llibreries a python com pytesseract que faciliten molt la seva implementació.[2][1]

- CNN (Convolutionan Neural Network): Per a realitzar el reconeixement de dígits, és comú utilitzar una xarxa neuronal entrenada amb datasets com per exemple MNIST. Encara que la creació d'un model pot ser és complexa, els resultats que s'obtenen són millors que els obtinguts mitjançant un OCR.[5]

2.2.3 Reconstrucció de la imatge

L'última tasca que s'ha de fer és reconstruir la imatge amb la solució trobada. Per poder-ho fer és necessari tenir el sudoku resolt amb la Intel·ligència artificial. Quan es tenen tots els nombres s'han d'identificar els que formen part de la solució per poder-los incloure a la imatge. Per poder aconseguir la imatge final, s'inverteix tot el preprocessament que s'ha realitzat amb la solució del sudoku inclosa, d'aquesta manera s'obté la imatge inicial amb les insercions dels nombres.

3 PROPOSTA

Desenvoluparem aquest sistema per a sudokus estàndard de 9x9, però provarem sudokus de tot tipus per poder aconseguir un sistema robust i que de imatges amb Sudokus mal fetes, aconsegueixi obtenir el Sudoku i la seva respectiva solució.

Sobre les diferents imatges d'entrada, és a dir, les diferents imatges que continguin Sudokus en formats diferents, les farem de manera manual, ja que aconseguir aquestes imatges és una tasca simple i no gaire extensa, per tant, desenvoluparem els nostres datasets amb les diferents imatges de sudokus.

Utilitzarem dos datasets per entrenar dos CNN diferents:

- MNIST dataset - Imatges de nombres escrits a mà
- Digital digits dataset - Imatges de nombres d'ordinador utilitzant diferents Fonts de lletra

Cada dataset està format pels diferents dígits, i conté entre 600-1000 imatges de cada dígit.

El dataset l'hem dividit en un 80% train, un 20% test, i d'aquest 20% de test hem agafat un 20% per a la validation.

Els dos models CNN han sigut creats amb el mateix nombre i tipus de capes, així com el mateix learning rate, i input d'una imatge 32x32x1 píxels.

Les altres dades que necessitarem estaran relacionades amb informació interna pròpia de cada solució. Aquestes dades seran els diferents dígits que formen un Sudoku així com la posició a la qual es troben (respecte al tauler). També són importants i necessàries les dades relacionades amb la informació associada al tauler, és a dir, haurem d'obtenir una estructura de dades que representi computacionalment la imatge capturada.

El flux de procés del nostre sistema constarà de les següents parts:

- Binarització bàsica de la imatge completa
- Obtenció dels contorns de la imatge, trobar el contorna amb àrea màxima, i recollir els 4 punts que el defineixen
- Aplicar una homografia (transformació geomètrica - warp) utilitzant els 4 punts anteriors per obtenir una vista zenital del Sudoku
- Binarització completa per a maximitzar l'eliminació de redundància, i obtenir les diferents 81 caselles binaritzades.
- Processament de cada una de les caselles per a eliminar sorolls, i contorns no desitjats, així com ressaltar el dígit per a maximitzar la confidència dels models en predir el dígit.
- Predicció de les diferents caselles amb els dos CNNs diferents i l'OCR.
- Solucionar el Sudoku (IA)
- Sobreposar la solució al sudoku original

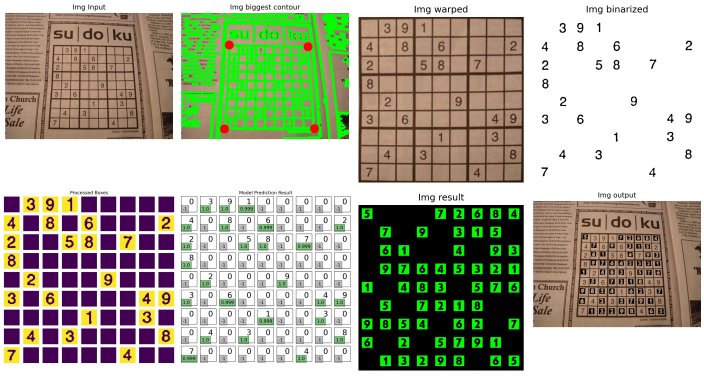

Fig. 1: Flux del projecte

Per poder mesurar el rendiment del sistema ho dividirem en dues parts, el accuracy obtingut en la predicció dels diferents dígits del Sudoku, i la confidència mitjana que té cada un dels models provats a l'hora de predir els diferents dígits.

Per poder calcular el accuracy obtingut s'haurà de comparar si el dígit que hem introduït és igual al dígit que hi hauria d'anar (Ground Truth), i dividir tots els dígits correctament detectats entre el total de dígits que hi ha a predir (sense tenir en compte forats buits).

La confidència mitjana de cada model l'obtindrem en el moment de fer la predicció de cada nombre, sumarem la confidència màxima que tingui del nombre a predir i dividirem pel nombre de les diferents prediccions fetes.

4 EXPERIMENTS, RESULTATS I ANÀLISI

Podem dividir el nostre projecte en 3 grans blocs diferents que tenen els seus experiments resultats i anàlisis propis:

4.1 Obtenció de la imatge únicament amb el Sudoku

Per a obtenir tots els contorns de l'imatge d'input primer caldra binaritzar l'imatge. El proces d'aquesta primera binarització es molt simple, simplement apliquem un Blur Gaussia per eliminar soroll, i seguidament apliquem un threshold adaptatiu gaussia que sencarregara de binaritzar l'imatge, independentment de la iluminació d'aquesta. Per ultim utilitzant la funció de connected Components eliminarem illes de pixels molt petites.

Un cop binaritzada la imatge procedim a extreure tots els contorns d'aquesta utilitzant altra vegada OpenCV.

Per últim hem creat una funció per detectar l'àrea més gran dels diferents contorns, i extreure els 4 punts que la conformen, de manera que ja tindrem els 4 punts per a fer l'homografia de la imatge input, i posar-la en vista zenital, en una imatge amb relació d'aspecte 1:1.

Els resultats que obtenim són molt bons tots, inclús si la imatge presenta ombres, o angles molt aguts. Tot i així no és 100% robust, presentant falles en imatges on els vores del Sudoku no són notoris, o en fulls quadriculats, on no acabem de detectar sempre bé el contorn que forma el Sudoku.

4.2 Processament d'imatge

Un cop ja tenim la imatge amb únicament el Sudoku en un quadrat, cal aplicar processament d'imatge per a obtenir les 81 caselles de dígits diferents que conformen el Sudoku.

El primer que hem fet és processar la imatge del Sudoku, binaritzar-la i treure el màxim de redundància possible.

La binarització en aquest cas és una mica més complexa. Primer de tot fem el mateix procés d'aplicar un Blur i threshold adaptatiu gaussià.

Un cop tenim la imatge binaritzada volem eliminar el màxim possible la redundància, és a dir, tot el que no siguin els dígits. Per a fer això fem diferents, primer de tot eliminem els vores que envolten el Sudoku en cas que n'hi hagi. Per a fer això hem creat una funció que detecta la quantitat de píxels en columnes i files i en cas de superar un llindar, s'elimina aquella part.

Després hem creat una funció per eliminar els vores que separen les diferents caselles, detectant els components connectats, detectant així les illes de píxels grans com aquests vores, i tornem a binaritzar la imatge eliminant-los.

Finalment eliminem les petites illes de píxels que queden.

Un cop tenim aquesta imatge ja binaritzada, dividim la imatge en 81 imatges, que són les respectives caselles. En aquest punt encara tenim soroll a les caselles. També ens trobem que els dígits poden no estar centrats, i que no tinguin una forma prou entenedora per a predir-los correctament.

Per últim quedarà fer un processament d'imatge complex per a cada casella, obtenint així una casella binària amb únicament un dígit clarament remarcat.

El flux per processar cada casella és el següent:

- Primer de tot eliminem illes de píxels petites (soroll que pugui quedar)
- Cridem la nostra funció que detecta els vores de la imatge per eliminar-los
- Un cop tenim únicament el dígit en la imatge, trobem la bounding box d'aquest dígit, i retallem la imatge per tenir exclusivament el dígit sense "padding".
- Finalment centrem i afegim un padding al dígit de manera que sigui el màxim semblant a l'input que està acostumat la nostra xarxa neuronal.

Alguns dels resultats obtinguts són els següents:

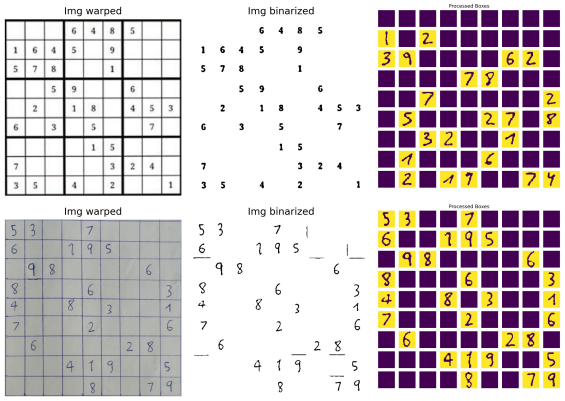

Fig. 2: Exemple resultat del processament d'imatge

El processament de la imatge del Sudoku, i de les caselles, ha sigut un procés amb moltíssims experiments diferents, aplicant tota classe de tècniques per intentar obtenir el millor resultat possible. Algunes tècniques que també hem provat són: Skelenitzation dels dígits i dibuixar-los a partir de l'esquelet, treballar únicament amb la imatge que conté el sudoku i no amb caselles i buscar els dígits trobant la seva bounding box...

4.3 Entrenament del model i predicció dels dígits

En el nostre projecte hem fet servir dos CNNs i el Tesseract OCR. Per entrenar les dues CNNs hem utilitzat les dades ja comentades a l'apartat de Proposta. En la construcció de les dues Xarxes Neuronals hem provat diferents experiments, ja sigui canviar el learning rate, tipus d'input, processament del dataset, nombre i tipus de les diferents capes...

Per avaluar com de bé funcionava el nostre model utilitzàvem les imatges de test per obtenir un Accuracy.

A part de fer inferència amb el propi dataset, així com amb imatges nostres, també vam utilitzar un fitxer Python que et crea

una UI per dibuixar dígits amb el ratolí de l'ordinador, i així fer inferència al model de manera ràpida i interactiva:

Fig. 3: Exemple UI per fer inferència al model

Sobre l'OCR, utilitzarem el Tesseract, amb la configuració de només dígits.

Un cop ja tenim tots els models preparats, només falta donar-li la casella d'input, i fer el predict d'aquesta.

Hem optat per fer 3 prediccions per cada dígit, la dels dos models diferents i la de l'OCR, i quedant-se amb el dígit que tingui la confidència més elevada.

En el cas que la confidència sigui menys del 50% és que es tracta d'una casella on no hi ha cap dígit, és a dir, una casella buida, que marcarem amb un 0. En aquesta funció de predicció aprofitarem per obtenir els resultats de confidència de cada model.

Un exemple de resultat obtingut en la predicció és el següent:

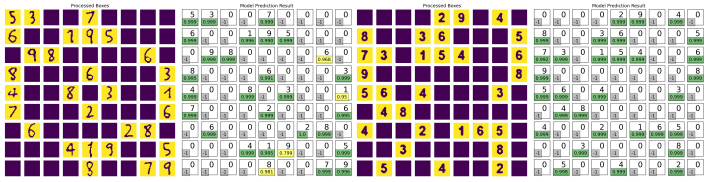

Fig. 4: Exemple prediccions del model

Els resultats de confidència mitjana i Accuracy obtinguda en l'exemple de Sudoku dibuixat a mà són els següents:

```
The Mean confidence obtained with
the different models when predicting the
digits in the sudoku is:

Model1 - Computer Digits: 0.98864
Model2 - Mnist - Handwritten Digits: 0.96359
Tesseract OCR: 0.41866

Accuracy obtained (Numbers well predicted
/ numbers to predict): 1.0
```

En els resultats anteriors podem veure com el Tesseract OCR ens dóna resultats pitjor a l'aleatorietat, de manera que no s'utilitzarà mai. En el cas de Sudokus digitals sol donar millors resultats, arribant fins a 70% de confidència mitjana. Tot i així és un resultat molt dolent com per fer-se servir. Per altra banda els models ens donen resultats molt bons pròxims al 100% de confidència. Pel que fa a l'accuracy, en Sudokus digitals sempre obtenim un accuracy del 100%, detectant amb molta confidència tots els dígits que conformen el Sudoku, i fent la predicció d'aquests. En Sudokus fets a mà solem obtenir també el màxim de accuracy dibuixant nombres que s'assemblin als digitals, on hi ha més diferencies entre nombres semblants. En casos de Sudokus amb nombres molt confosos on costa diferenciar nombres semblants, aconseguir el màxim d'accuracy, és a dir, predir tots els dígits del Sudoku bé, sol ser poc comú. Tot i així aconseguim confidències mitjanes del voltant del 95%.

Finalment, un cop hem realitzat la predicció dels dígits, aplicarem la IA que resol el Sudoku. En cas que no hi hagi solució retornarem que no s'ha trobat solució. Si hi ha solució, obtindrem els nombres que la conformen. Seguidament crearem una imatge que contingui visualment els dígits d'aquesta solució, i sobreposarem aquesta solució a la imatge original, fent la transformació geomètrica inversa que fèiem inicialment.

5 Conclusions

Durant el transcurs d'aquest projecte hem pogut aplicar moltes coses que hem après durant el transcurs de l'assignatura, i que hem vist i aplicat en les diferents pràctiques i reptes que ens plantejava l'assignatura, i ho hem pogut portar a molt més nivell, i aplicar-ho en un problema real i complex. Alguns exemples serien les diferents binaritzacions de les imatges i tractament d'aquestes, així com aplicar transformacions geomètriques...

Gràcies a aquest projecte també hem tingut la motivació d'aprendre coses noves pel nostre compte, fent cursos per internet i provant nosaltres mateixos. Un clar exemple d'això és la utilització de Xarxes Neuronals i el Tesseract OCR. Hem pogut aprendre sobre Keras, un framework d'alt nivell escrit en Python, que ens ha permès crear dos models classificadors d'imatges, per a poder classificar els dígits.

I finalment, graciés a aquest projecte hem pogut apropant-se més al món de la Visió per Computador, aplicant diferents camps d'aquesta per a poder un resoldre amb una aplicació real i molt interessant.

Referències

[1] A comprehensive guide to ocr with tesseract, opencv and python. https://nanonets.com/blog/ocr-with-tesseract/.

[2] Opencv sudou solver and ocr. https://www.pyimagesearch.com/2020/08/10/opencv-sudoku-solver-and-ocr/.

[3] Solving sudoku puzzles. https://jponttuset.cat/solving-sudokus-like-a-pro-1/.

[4] Sudoku and cell extraction. https://becominghuman.ai/sudoku-and-cell-extraction-sudokuai-opencv-38b

[5] Sudoku image solver. https://github.com/fzy1995/SudokuImageSolver.

[6] Sudoku solver using computer vision and deep learning. https://aakashjhawar.medium.com/sudoku-solver-using-opencv-and-dl-part-1-490f0

[7] Text skew correction with opencv and python. https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/.

[8] Using opencv to solve a sudoku. https://golsteyn.com/writing/sudoku.