Proiect Cercetare – Lab 4

1. Modelarea părții experimentale

1.1. Preprocesarea inputului

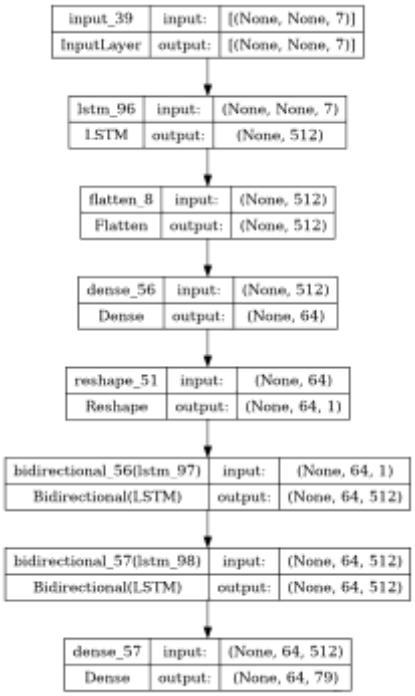
Ideea de bază a proiectului constă în convertirea imagiinlor din setul de date IAM_words din formatul raster intr-un format vectorized. A fost ales algoritmul Potrace pentru rezultatele sale satisfăcătoare oferite într-un timp decent. O implementare proprie se regăsește în cadrul proiectului, în cazul necesității unor modificări ulterioare.

Fie $\mathcal{C}$ multimea spațiului de culori si $\mathcal{G}: \mathcal{M}_{h,w}(\mathcal{C}) \to \mathcal{M}_{h,w}([0,1])$ o funcție de conversie a imaginii în tonuri de gri (grayscale), și $\gamma \in [0,1]$ un factor de threshold. Fie $P = (p_{ij})_{1 \le i \le h, 1 \le j \le w} \in \mathcal{M}_{h,w}(\mathcal{C})$ un input din setul de date. Se construiește imaginea binară $P' = (p'_{ij})_{1 \le i \le h, 1 \le j \le w}$, unde $p'_{ij} = \begin{cases} 0, & \mathcal{G}(P)_{ij} < \gamma, \\ 1, & altfel \end{cases}$ și se aplică algoritmul $Potrace: \mathcal{M}_{h,w}(\{0,1\}) \to (\mathbb{R}^7)^*$, unde $X^* = \{(x_1, \dots, x_k) | k \ge 0, x_i \in X \ \forall i = \overline{1,k}\}$ este multimea listelor ordonate cu elemente din X. Un element $b = (x_1, y_1, x_2, y_2, x_3, y_3, \alpha) \in \mathbb{R}^7$ descrie o curbă Bezier de gradul 3 ce are ca parametri punctele $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = (x_3, y_3)$, și coeficientul de curbură α, al cărei poligon de control este determinat de segmentele P_1P_2, P_1P_3,P_2P_3 și paralela la P_2P_3 care taie P_1P_2 în segmente de proporții $(1 - \alpha)$, α.

De alegerea funcției $\mathcal{G}$ si a factorului γ depinde calitatea outputului oferit de $Potrace$. Blurarea imaginii poate fi de asemenea un factor decisiv pentru mai buna identificare a curbelor.

1.2. Antrenarea modelului

Avantajul preprocesării anterioare este acela că inputul nu mai este limitat de o dimensiune fixă a imaginii, și că există posibilitatea unei procesări secvențiale a acestuia. Astfel, în cadrul unui model de rețea neuronală, putem înlocui tradiționalul CNN cu o celulă LSTM imediat atașată inputului.



Ulterior, două layere de activare reduc dimensiunile feature-urilor la 64, care sunt trecute prin două celule LSTM bidirecționale. Rezultatul este adus la o matrice care are, pe linii, numărul lungimea maximă a unei secvențe care poate fi recunoscută dintr-un input, respectiv numărul maxim de caractere admise, plus încă un „blank character", folosit ulterior de către CTC.

1.3. Clasificarea

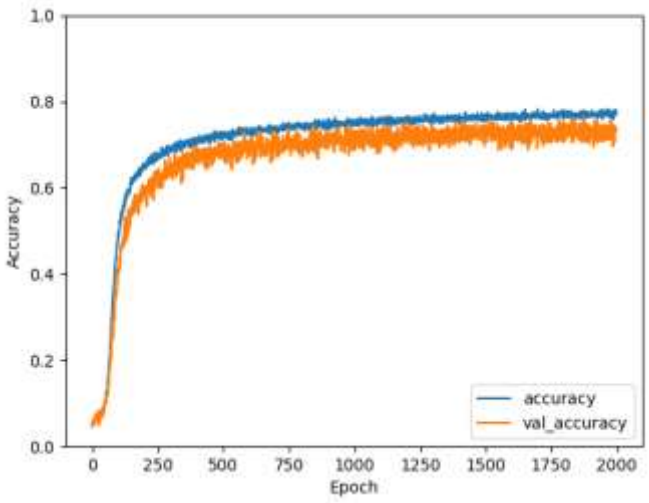
Măsura folosită pentru loss este Connectionist Temporal Classification (CTC), specializată pentru output-ul RNN. Acuratețea modelului se decide pe baza inputurilor pentru care $ctc\_loss = 0$. (care prezic cu exactitate outputul)

2. Studiu de caz

(A se vedea fisierul notebook din repo pentru cod)

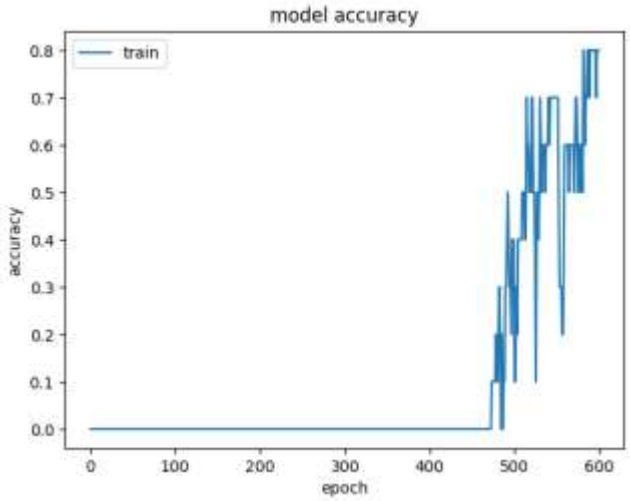
2.1. ANN-based Bezier curve identification

Inițial, este de menționat că s-a încercat tratarea vectorizării ca o problemă de sine stătătoare. În acest sens, am încercat o rețea convoluțională specializată în detectarea anumitor tipuri de curbe în fiecare chunk de 32x32px al unei imagini. Avantajul ideii era aceea că tracing-ul ținea cont și de grosimea liniilor, ceea ce producea outputuri similare cu cele oferite de dispozitivele fizice folosite în online OCR (asemanator [2]), și nu o simplă înfășurătoare geometrică a zonei de interes. Cu toate acestea, desi modelul a fost antrenat cu precizie ridicata, performanțele acestei metode au fost scăzute in realitate, din cauza lipsei unui set de date specializat pe această problemă.



2.2. Vectorizarea Potrace

Reteaua a fost antrenata pe un set mic de 10 date de intrare. Acuratetea tinde sa convearga mai lent decat in cazul modelului CNN-BiLSTM, in schimbul posibilitatii de a propcesa inputuri de lungimi variate.



In ambele cazuri, variatia acuratetei depinde de evolutia lossului, care trebuie sa fie sub un anumic prag pentru a asigura o crestere a ratei de invatare.

3. Related work

Metoda descrisa in [2] folosește o regresie non-liniară pentru a segmenta curbele din imagine, care ar putea fi pus în dificultate în condițiile în care grosimea liniei este mărită. Sunt folosite concepte din fizică și psihologice combinate cu analize statistice pentru recunoașterea caracterelor.

In [6] a fost folosit un model LSTM multidimensional cu clasificator CTC, dar pentru dataset cu scripturi atât latine, cât și arabice, raportând precizie de peste 90%.

TextCaps [11] foloseste FCNN + capsule networks pentru a identifica caracterele din EMNIST cu 95% acuratete. Metoda propusa de autori este utila pentru augmentarea setului de date si nu este destinata recunoasterii, modelul nostru este aplicabil in acest context si are potential de a oferi rezultate promitatoare.

In [3.] s-a folosit o structura CNN-RNN 50-98% rata de identificare corecta a cuvintelor pe setul de date, folosind CTC ca metrica de evaluare. Autorii discuta si influenta ratei train-test in determinarea preciziei finale. De asemenea, sunt abordate metode de segmentare a textului cu OpenCV, care in contextul lucrarii curente sunt echivalentul vectorizarii.

Lucrarea [13] aplica GRU pe IAM cu 95% acuratete. Functia de loss folosita este cross-entropy. Modelul descris are si aplicatii generative.

4. Github

https://github.com/stefnmUBB/ProiectCercetare

master · ProiectCercetare / licenta.ipynb

stefrenUBB added notebook

Preview | Code | Blame | 5 lines (1 loc) · 176 KB | Code 55% faster with GitHub Copilot | Raw

In [1]:
```
import PIL
import tensorflow as tf
```

Defining datasets

We will use a custom preprocessed version of IAM dataset, where each image has been converted to a vectorized list of paths using the Potrace algorithm.

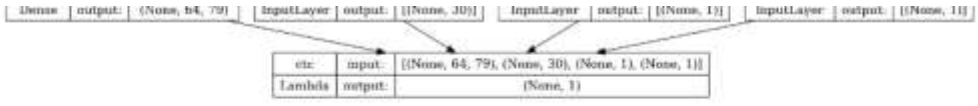
Input structure

In [2]:
```
from os import path

minibec_path = "/kaggle/input/minibec"
iam_words_path = "/kaggle/input/iamdatasets/IAM_Words/iam_words"
```

In [3]:
```
def read_words():
    def get_id_and_word(line_data:list):
        return (line_data[0], "".join(line_data[8:]), line_data[1]=="ok")
```

Code | Blame | 50 lines (36 loc) · 1.43 KB | Code 55% faster with GitHub Copilot | Raw

```
import tensorflow as tf
from keras import datasets, layers, models, activations
import matplotlib.pyplot as plt

from datasets import load_bezier_dataset, split_train_test

inputs, outputs = load_bezier_dataset()

train_i, train_o, test_i, test_o = split_train_test(inputs, outputs)

print(train_i.shape)
print(train_o.shape)

print(test_i.shape)
print(test_o.shape)

model = models.Sequential()
model.add(layers.Conv2D(32, (5, 5), activation=activations.gelu, input_shape=(64, 64, 1)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(96, (3, 3), activation=activations.exponential))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation=activations.exponential))
model.add(layers.Flatten())
model.add(layers.Dense(480, activation=activations.leaky_relu))
model.add(layers.Dense(104))
```



In [13]:
```
model.compile(loss={'ctc': lambda y_true, y_pred: y_pred}, optimizer = 'sgd', metrics=["accuracy"])

history = model.fit(dataset,
                    batch_size=5,
                    epochs=600,
                    verbose=2,
                    #callbacks=callbacks_list
                   )
```

```
Epoch 1/600
10/10 - 14s - loss: 54.9385 - accuracy: 0.0000e+00 - 14s/epoch - 1s/step
Epoch 2/600
10/10 - 4s - loss: 24.1944 - accuracy: 0.0000e+00 - 4s/epoch - 415ms/step
Epoch 3/600
10/10 - 4s - loss: 22.7811 - accuracy: 0.0000e+00 - 4s/epoch - 417ms/step
Epoch 4/600
10/10 - 4s - loss: 21.9421 - accuracy: 0.0000e+00 - 4s/epoch - 398ms/step
Epoch 5/600
10/10 - 4s - loss: 21.3396 - accuracy: 0.0000e+00 - 4s/epoch - 405ms/step
Epoch 6/600
```


master
Commits on Nov 24, 2023
added beztrain
stefnmUB8 committed 17 minutes ago
ac63ced
Commits on Nov 23, 2023
added notebook
stefnmUB8 committed 11 hours ago
8589a80
Commits on Nov 22, 2023
added preprocessing code
stefnmUB8 committed 2 days ago
e4a034c