Deep Hallucination Classification
Classify images hallucinated by deep generative models
Documentatie

Am început cu un cod e IA folosind keras și un model Sequential. Codul se gaseste in fisierul main1.py .

Documentație pentru primul cod

Acest document descrie abordarea utilizată pentru a antrena un model de învățare automată pentru clasificarea imaginilor. Aceasta abordare se bazează pe utilizarea unei rețele neuronale convoluționale (CNN), ceea ce este o alegere comună pentru problemele de clasificare a imaginilor datorită abilității lor de a capta caracteristici complexe într-un mod ierarhic.

Preprocesarea Datelor

Pentru preprocesarea datelor, am utilizat ImageDataGenerator de la Keras, care este o clasă utilitară pentru generarea loturilor de date tensoriale cu datele imaginilor reale augmentate în timpul execuției. Aceasta transformă imaginile de intrare și le normaliza pentru a fi în intervalul [0, 1]. Datorită acestei tehnici, putem avea o varietate de date de antrenament, ceea ce ajută la evitarea overfitting-ului și îmbunătățește performanța generală a modelului. De asemenea, am folosit flow_from_dataframe pentru a încărca imaginile direct din dataframe, ceea ce facilitează manipularea datelor.

Reprezentarea Caracteristicilor

Modelul nostru este o rețea neurală convoluțională simplă (CNN), care este un tip de rețea neurală adâncă specializată în procesarea datelor cu o structură similară cu grila, cum ar fi o imagine. CNN învață caracteristicile direct din date, evitând necesitatea extragerii manuale a caracteristicilor. În acest model, am utilizat două straturi convoluționale urmate de o operație de max pooling și apoi de un strat complet conectat (Dense). Funcția de activare 'relu' este utilizată în straturile convoluționale și 'softmax' este folosită în stratul de ieșire pentru a returna probabilități.

Detalii de Antrenare

Modelul este antrenat pentru 20 de epoci, folosind o dimensiune de lot de 32. Pentru compilarea modelului, am folosit 'categorical_crossentropy' ca funcție de pierdere, care este adecvată pentru problemele de clasificare

multi-clasă. Ca optimizator, am utilizat SGD cu o rată de învățare de 0.01, un decay de 1e-6, un momentum de 0.9 și nesterov setat pe True. Acești hiperparametri au fost aleși după mai multe runde de experimentare și comparare a performanțelor modelului.

Evaluare și Testare

După antrenare, modelul a fost testat pe setul de date de test. Predictiile sunt realizate prin folosirea metodei predict a modelului, iar clasa cu cea mai mare probabilitate este aleasă ca predicție finală pentru fiecare imagine.

Rezultatele sunt salvate într-un fișier CSV, cu numele imaginii și clasa prezisă pentru fiecare intrare.

Primul cod scris este cel din fisierul main1.py cu rezultatul output1.txt Acuratețea pe datele de antrenare ajunge la 0.9973, dar pe datele de validare este 0.653 , rezulta overfitting. Totusi, pe datele de test, codul obține doar 0.049.

Found 12000 validated image filenames belonging to 96 classes.
Found 1000 validated image filenames belonging to 96 classes.

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy
1/20	3.9982	0.0936	3.0029	0.2430
2/20	2.3052	0.3873	2.0223	0.4530
3/20	1.4954	0.5657	1.5986	0.5440
4/20	0.9868	0.7010	1.4176	0.5910
5/20	0.5985	0.8108	1.5963	0.5880
6/20	0.3537	0.8846	1.7535	0.5880
7/20	0.2342	0.9243	2.1698	0.5650
8/20	0.1419	0.9549	1.8961	0.5960
9/20	0.0873	0.9733	2.1134	0.5810

10/20	0.1059	0.9693	2.1009	0.5890
11/20	0.0593	0.9810	2.3626	0.6090
12/20	0.0740	0.9762	2.3187	0.6110
13/20	0.0526	0.9850	2.4470	0.6010
14/20	0.0468	0.9847	2.2896	0.6170
15/20	0.0467	0.9867	2.5308	0.6110
16/20	0.0148	0.9962	2.3851	0.6400
17/20	0.0240	0.9934	2.6152	0.6120
18/20	0.0257	0.9926	2.7464	0.6150
19/20	0.0140	0.9966	2.4685	0.6450
20/20	0.0128	0.9973	2.3701	0.6530

Found 5000 validated image filenames.
5000/5000 - 14s 3ms/step

In acest moment, lipsa acuratetii pare sa fie datorată modelului, asa ca incercam sa il imbunatatim:

În următorul cod, am făcut următoarele modificări:

1. Am crescut dimensiunea stratului Dense la 512 noduri.
2. Am crescut numărul de epoci la 50.
3. Am folosit optimizatorul Adam cu un learning rate mai mic.
3. Am adăugat straturi suplimentare Conv2D și MaxPooling2D pentru a crește complexitatea modelului.

Rezultatul obtinut: 0.05

În continuare rezultatele se mențin la fel de scăzute pe datele de test.

Am adăugat regularizare L2, utilizând argumentul kernel_regularizer la construcția straturilor modelului.

from keras.regularizers import l2

model.add(Conv2D(32, (3, 3), activation='relu', padding='same', kernel_regularizer=l2(0.01), input_shape=(img_width, img_height, 3)))
# ...
model.add(Dense(512, activation='relu', kernel_regularizer=l2(0.01)))

Am obtinut codul din main2.py cu output-ul in fisierul output2.py

Rezultatul obtinut: 0.05

Deoarece aceste modificări nu au dat rezultatele așteptate, am schimbat abordarea. Am renuntat la folosirea modulului "keras" și am folosit direct “torch”. Am scris codul main3.py cu rezultatele in fisierul output3.txt

Rezultatul pe datele de test a fost: 0.614 - semnificativ mai bun.

Training started...
Epoch: 0
Validation accuracy is: 46.9%
Epoch: 1
Validation accuracy is: 58.1%
Epoch: 2
Validation accuracy is: 60.5%
Epoch: 3
Validation accuracy is: 62.4%
Epoch: 4
Validation accuracy is: 63.1%
Epoch: 5
Validation accuracy is: 63.8%
Epoch: 6
Validation accuracy is: 62.8%
Epoch: 7
Validation accuracy is: 61.7%
Epoch: 8
Validation accuracy is: 61.3%
Epoch: 9
Validation accuracy is: 64.1%

Din acuratetea generata partial, se observa o creștere semnificativă în primele 3 epoci, apoi o creștere semnificativă, pana la epoca 4-5, unde urmeaza o stagnarea a valorilor.

Documentație pentru acest cod:

Această documentație descrie abordarea de învățare automată utilizată pentru a antrena un model de clasificare a imaginilor bazat pe o rețea neuronală convoluțională (CNN) folosind PyTorch, o bibliotecă populară de învățare automată profundă.

Preprocesarea Datelor

În primul rând, datele sunt preprocesate prin normalizare și transformare în tensori. Transformarea normalizează datele de intrare prin scăderea mediilor (0.5, 0.5, 0.5) și divizarea prin deviațiile standard (0.5, 0.5, 0.5) pe fiecare canal de culoare. Această normalizare ajută la accelerarea procesului de antrenament, menținând valoarea pixelilor între -1 și 1. În plus, este definit un set de date personalizat (clasa CustomDataset), care permite citirea și prelucrarea datelor de intrare.

Reprezentarea Caracteristicilor

Modelul nostru este o rețea neuronală convoluțională (CNN), care este un tip de model de învățare automată adâncă eficient pentru problemele de procesare a imaginilor. În acest model, sunt utilizate două straturi convoluționale, fiecare urmat de o funcție ReLU și un strat de max-pooling. Aceste straturi extrag caractere din imagine și reduc dimensiunea datelor, respectiv. La final, datele sunt aplatizate și trec prin două straturi liniare (dense), cu o funcție ReLU între ele. Stratul final este un strat liniar cu 96 de neuroni, care corespunde celor 96 de clase din problema noastră.

Detalii de Antrenare

Modelul este antrenat folosind funcția de pierdere CrossEntropyLoss și optimizerul Adam cu o rată de învățare de 0.001. Acestea au fost alese după experimentare și s-au dovedit a oferi rezultate bune pentru această problemă. Modelul este antrenat pentru 10 epoci, fiecare epocă fiind un ciclu complet prin setul de date de antrenament.

Evaluare și Testare

După fiecare epocă, modelul este validat pe setul de date de validare, calculând acuratețea pe acest set. În final, modelul este utilizat pentru a

prezice clasele pe setul de date de test. Rezultatele sunt salvate într-un fișier CSV.

Îmbunătățiri viitoare

Există mai multe direcții pentru a îmbunătăți acest model în viitor. Un prim pas ar fi să se facă mai multe experimente cu diferite arhitecturi de rețea și parametri de antrenament. De asemenea, tehnici mai avansate de preprocesare a datelor și augmentare a datelor pot ajuta la îmbunătățirea performanței. În cele din urmă, pot fi utilizate tehnici de regularizare, precum dropout sau weight decay, pentru a preveni supraantrenarea modelului.