Andrei Mihnea-Stefan

Deit – Facebook's facial recognition

Introducere

Facebook tine updatat un AI pentru recunoasterea facial chiar pe github pe gratis: https://github.com/facebookresearch/deit . Ceea ce ne intereseaza pe noi este arhitectura neuronala, care se gaseste la urmatorul link: https://github.com/facebookresearch/deit/blob/main/models_v2.py . Creierasul in sine este definit printr-o serie de clase pana la linia 270, in momentul in care scriu acest README. Dupa aceasta linie, chestiile care incep cu *@register_model* sunt practice exemple de initializa re a creierului. Aceste exemple arata ce parametrii Facebook considera a fi cei mai buni pentru acest AI, care este clasa *vit_models* aici.

Pentru mai multe detalii, puteti citi articolul care este in curs de publicare de oamenii de la Facebook si de la universitatea Sorbonne: https://arxiv.org/pdf/2012.12877.pdf .

Cod

Numele codului este *DeiT_facebookV2.ipynb* sau *DeiT_facebookV2.py* in prezent.

Pentru a aplica, pe noi de fapt nu ne intereseaza decat sa copiem totul de la linkul https://github.com/facebookresearch/deit/blob/main/models_v2.py pana la linia 270, si sa importam imaginile si sa facem antrenarea exact asa cum o faceam pentru resnet. Practic nu trebuie decat sa copiem si sa lipim parti din cod de la Facebook cu ce aveam scris la resnet (va rog sa cititi documentatia pentru resnetpentru mai multe detalii).

Asadar, prima parte din cod este exact aceeasi cu cea de la resnet si face grupuri de *batches* din imaginile fetelor detectate de MTCNN (care au 160 de pixeli pe 160 de pixeli, asa cum am mentionat in documentatia MTCNN).

Urmatoarea parte este exact copy paste de la link-ul https://github.com/facebookresearch/deit/blob/main/models_v2.py pana la linia 270.

Dupa aceea, va puteti uita la exemplele pe care Facebook le are pentru parametrii care trebuie dati acestui AI (clasa *vit_models* in acest caz). Eu am luat exemplul cel mai mic ca sa pot rula repede pe laptop sa vad daca merge:

```
#The below number is 160 because the faces extracted by MTCNN are 160x160.
img_size=160
#The number of names that the model will see (population of romania over 14).
num_classes=data_sizes["train"]
#Variables inside the network. Please check examples that start at line 271 here:
#https://github.com/facebookresearch/deit/blob/main/models_v2.py
embed_dim=192
depth=12
num_heads=3
mlp_ratio=4

model_fit=vit_models(num_classes=num_classes,img_size=img_size,patch_size=batches,
                     embed_dim=embed_dim,depth=depth,num_heads=num_heads,mlp_ratio=mlp_ratio,qkv_bias=True,
                    norm_layer=partial(nn.LayerNorm,eps=1e**(-6)),
                    block_layers=Layer_scale_init_Block).to(device)
```

Singurul lucru diferit in cazul nostrum este ca *img_size* este 160 din moment ce fetele detectate de MTCNN sunt de 160 de pixeli pe 160 de pixeli. Facebook-ul are imagini de fete de rezolutie 224x224.

Dupa aceea, ne uitam la articolul lor pentru a vedea ce optimizator si learning rate au folosit (pagina 16 de la link-ul https://arxiv.org/pdf/2012.12877.pdf):

Methods	ViT-B [15]	DeiT-B
Epochs	300	300
Batch size	4096	1024
Optimizer	AdamW	AdamW
learning rate	0.003	$0.0005 \times \frac{\text{batchsize}}{512}$
Learning rate decay	cosine	cosine
Weight decay	0.3	0.05
Warmup epochs	3.4	5
Label smoothing ε	✗	0.1
Dropout	0.1	✗
Stoch. Depth	✗	0.1
Repeated Aug	✗	✓
Gradient Clip.	✓	✗
Rand Augment	✗	9/0.5
Mixup prob.	✗	0.8
Cutmix prob.	✗	1.0
Erasing prob.	✗	0.25

Implemetarea coloanei din stanga este foarte simpla pentru ca deja sunt create functii in pytorch pentru totul:

```
num_epochs=25
lr=3*10**(-3)
weight_decay=0.01
criterion=nn.CrossEntropyLoss()
#Facebook does AdamW. Please see paper.
#optimizer=optim.SGD(model_fit.parameters(),lr=10**(-2),momentum=0.9)
optimizer=torch.optim.AdamW(model_fit.parameters(),lr=lr, betas=(0.9, 0.999), eps=1e-08, weight_decay=weight_decay)
#lr_decay=lr_scheduler.StepLR(optimizer,step_size=7,gamma=0.1)
lr_decay=torch.optim.lr_scheduler.CosineAnnealingLR(optimizer=optimizer, T_max=num_epochs)
```

Dupa aceea antrenarea si evaluarea se fac exact la fel ca la resnet, fara sa schimbam nici o litera din cod. Clar cei de la Facebook stiu aceasta retea extrem de bine. Va rog sa tineti cont de folosirea optimizatorului si learning rate-ului mentionate de ei in articol. Prima data nu am vazut asta, si am pus ce aveam la resnet: optimizator SGD. Acesta nu scade loss-ul, dar folosind ce mentioneaza Facebook-ul in articol, loss-ul scade superb. Trebuie sa rulati mai multe reteaua si va trebuie placi grafice pentru o retea mai mare (cea de aici este cea mai mica mentionata de Facebook). De asemena, trebuie sa folositi *batches* mult mai mare: Facebook foloseste 4096 si 2048, nu 32 cat am eu aici.

Ultima parte a codului a fost adaugata acum doar pentru a vedea vizual cum scade loss-ul in antrenare:

```
plt.figure(figsize=(10,5))
plt.title("Training losses")
plt.plot(train_losses,label="train losses")
plt.xlabel("iterations")
plt.ylabel("loss")
plt.legend()
plt.show()
```