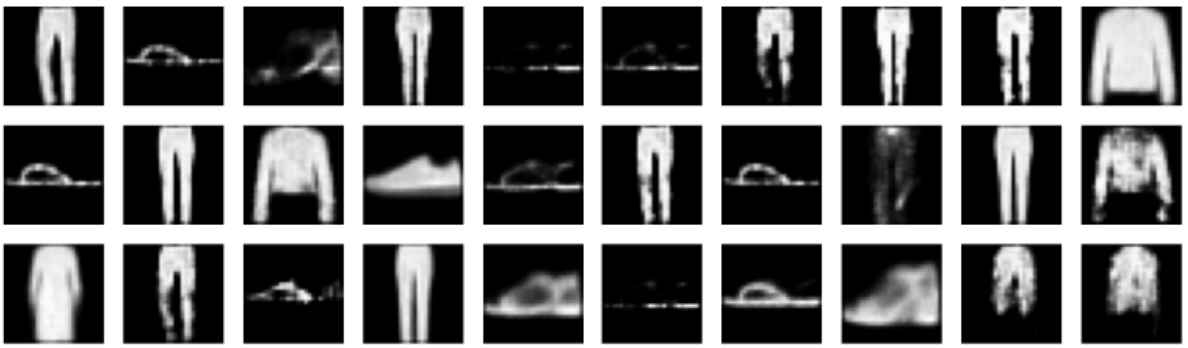
Luiz Felipe Nagatani Rabello

Relatório LAMIA - Módulo 26

Curso Udemy

Na seção 11 do curso Machine Learning, Data Science and Generative AI with Python, são abordados os modelos generativos, a seção se inicia com definição e aplicação de VAEs, mostrados no card 18, os AutoEncoders são redes neurais dedicadas a reduzir a dimensionalidade de uma entrada e reconstruí-la com o máximo de fidelidade possível. já os Variational AutoEncoders são dedicados a geração de amostras, o espaço latente gerado na parte do encoding se assemelha a uma distribuição probabilística, permitindo que o decoder “reconstrua” imagens novas com base na entrada.

```
z = np.random.normal(loc=0, scale=4, size=(250,2))
synth = vae.decoder.predict(z)
plt.figure(figsize= (20,20))
for i in range(256):
    plt.subplot(16,16,i+1)
    plt.axis('off')
    plt.imshow(synth[i].reshape((28,28)), cmap='gray') #expõe as imagens geradas pelo decoding

plt.show()
```



Em seguida é apresentado um conceito novo, os GANs que utiliza o ruído dos vetores latentes para criar uma distribuição probabilística. Os GANs são formados por geradores e discriminadores, onde o gerador criará imagens e o discriminador irá determinar se são reais ou artificiais, a validação acontece quando o discriminador é incapaz de diferenciar a origem da imagem.

```
    gaccSum += gloss[ g_acc ]
    gloss = trainGstep(batch)
    glossSum += gloss["g_loss"]
    gaccSum += gloss["g_acc"]
    cnt += 1

  print("E:{}, Loss G:{:0.4f}, Loss D:{:0.4f}, Acc G:%{:0.2f}, Acc D:%{:0.2f}".format(epoch, gl

  if epoch % 2 == 0:
    plotImages(generator)
```

```
E:7, Loss G:4.0355, Loss D:0.1137, Acc G:%4.43, Acc D:%93.06
E:8, Loss G:4.0839, Loss D:0.1119, Acc G:%4.38, Acc D:%93.13
```

Vídeo Youtube

O vídeo se inicia com a apresentação do conceito de um transformer, uma arquitetura capaz de interpretar as palavras em uma frase independente da ordem e interpretar várias seções da frase simultaneamente.

O intuito do vídeo é criar um modelo GPT treinando-o com um dataset composto por obras de William Shakespeare, uma diferença entre a prática deste card para o card 22 que se tratava de NLP, é que a tokenização era executada por palavra, neste exemplo ela é feita por caractere.

```
stoi = { ch:i for i,ch in enumerate(chars) }
itos = { i:ch for i,ch in enumerate(chars) }
encode = lambda s: [stoi[c] for c in s]
decode = lambda l: ''.join([itos[i] for i in l]) # tokenização básica

print(encode("hii there"))
print(decode(encode("hii there")))
print(decode([46])) #a tokenização é realizada por letra aqui, diferente do card 22
```

```
[46, 47, 47, 1, 58, 46, 43, 56, 43]
hii there
h
```

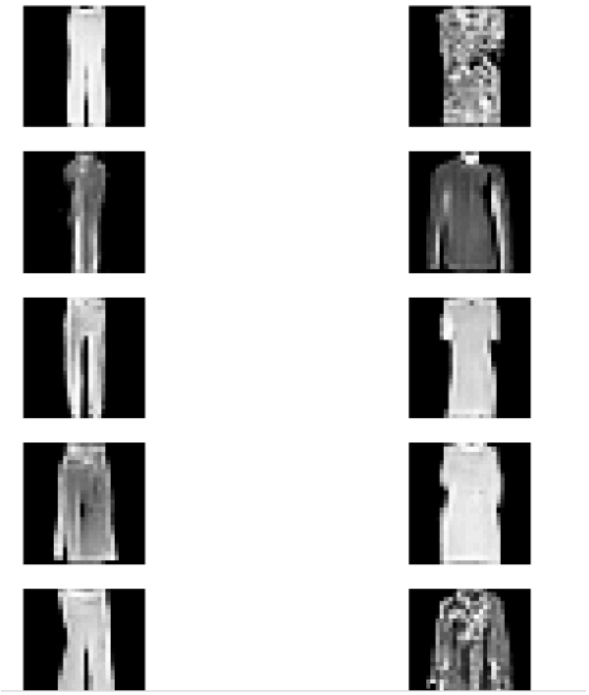
Este método se provou ineficiente pois mesmo após 30000 epochs, o resultado apresentou estruturas similares a palavras, porém nenhuma palavra em si.

```
print(decode(m.generate(idx = torch.zeros((1, 1), dtype=torch.long), max_new_tokens=500)[0].tolist()))
```

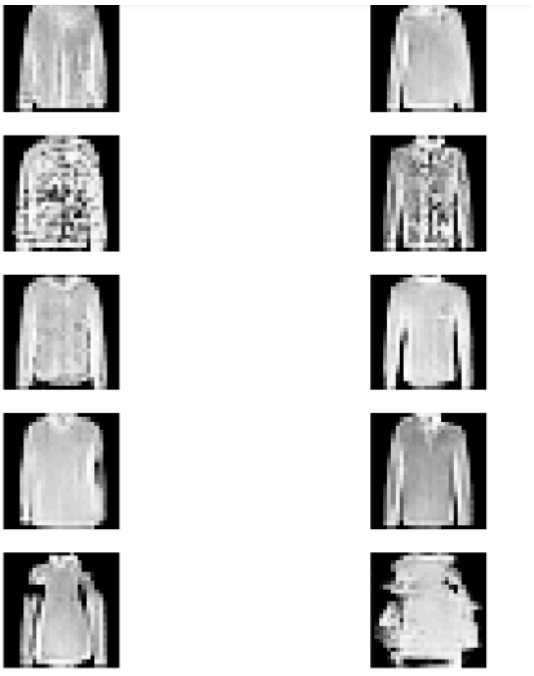
```
ABus

Wh wico, wingibey s, vanday
K:
He'ly I,-gheils acechel ins arf asoull wo me n scrdgf kippry be s dsidn bage gnot m thavithitathis d ghelo be ase bl medourne, arit tepr wh re heryour fe?
Hee llancthe hind alch f chutithowis; gboust f we uly k?
LELorg,
T:

A tyof, wathild the and hiscome win, ckis y came paun t iwe w, ond:
TCHAUSAUplyo l!
HERCKIERWin om mu!
Rortafo aisang aiso aceardgsiert I:
Thowhanou t sug Burssoutha lldn y MO ad ge,
Wheayou sthid OR:
Wh pen.
CUSe the he r heilllovent upo
```

Experimentação

Para fins de experimentação nos GANs, foi adicionado um bloco de código para criar imagens aleatórias com base em ruído e separar 10 imagens que registrassem acima de 70% no discriminador e 70% abaixo, e em seguida expô-las para mostrar a diferença entre elas e entender melhor como o discriminador determina o que é real e falso.



Real



Falso