

# nanoGPT-project

田佳音

June 2024

## 1 Introduction

自从 2017 年 Attention is all you need 论文的发表，Transformer 架构成为继 MLP、CNN 和 RNN 之后的第四个影响深远的神经网络模型。而由人工智能研究公司 OpenAI 开发的一种基于 Transformer 架构的自然语言处理模型，GPT 的发布一鸣惊人，不仅在学术界还在每个人的生活中都有十分深刻的影响。在过去的几十年间，神经网络的研究重点放在具有固定目标的单一任务，比如说图片分类、探测垃圾邮件或兴趣推荐等。但是 GPT 适用于更广泛、更多元得任务，如文本分类、语言翻译、文本摘要、问答系统、文本生成等多种自然语言处理任务。

但是由技术发展历程知道，如果想要模型的表现优秀需要十分庞大的数据、参数。在一般算力的 CPU 上难以执行，而 Andrej Karpathy 构建了一个可以在 CPU 上运行试验的微型 GPT，即 nanoGPT。为了具体理解每个模块在 Transformer 架构中充当怎样的角色，起到怎样的作用，依据上述 nanoGPT 模型由代码自主实现了 GPT 的基本功能。可以根据提供的莎士比亚作品的数据集，生成莎士比亚口吻创作的戏剧或诗篇。通过实际操作，了解到神经网络是如何实际上用代码实现的，以及多头注意机制、迭代次数、前馈层、ResNet、Dropout 函数等在其中的作用。

图 1 是 nanoGPT 的框架。由于只是用于文本生成而非原论文中的机器翻译，所以不需要先将一种语言编码在转换的步骤相较于原论文中的 Transformer 架构去掉了编码器部分和交叉注意力机制的部分。

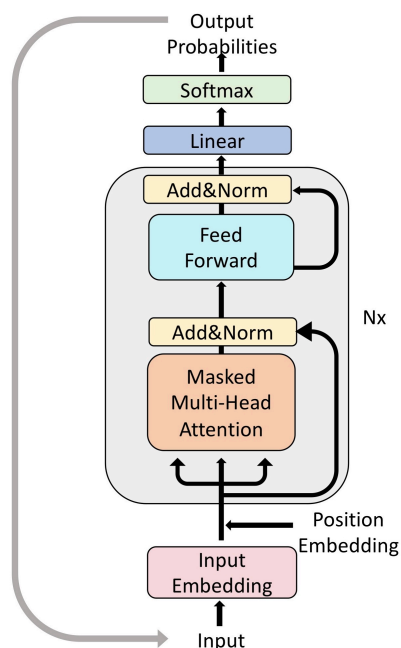


图 1: nanoGPT - model architecture

## 2 Experiment

### 2.1 BigramLanguage Model

在模型中使用二元语言模型并且迭代只有 100 次时，训练后的模型 loss 的值在 4.65 左右。[图 2]

添加了迭代次数至 1000 次后，训练模型，loss 降为 3.70 左右。

添加迭代次数到 10 000 次，训练模型得到 loss 为 2.40 左右 [图 4]

### 2.2 Self Attention Head

#### 2.2.1 Single Head Attention

降低学习率至  $1e-3$ ，改变迭代次数至 5000，增加单个自注意力头，模型训练得 loss 依然在 2.40 左右。[图 5]

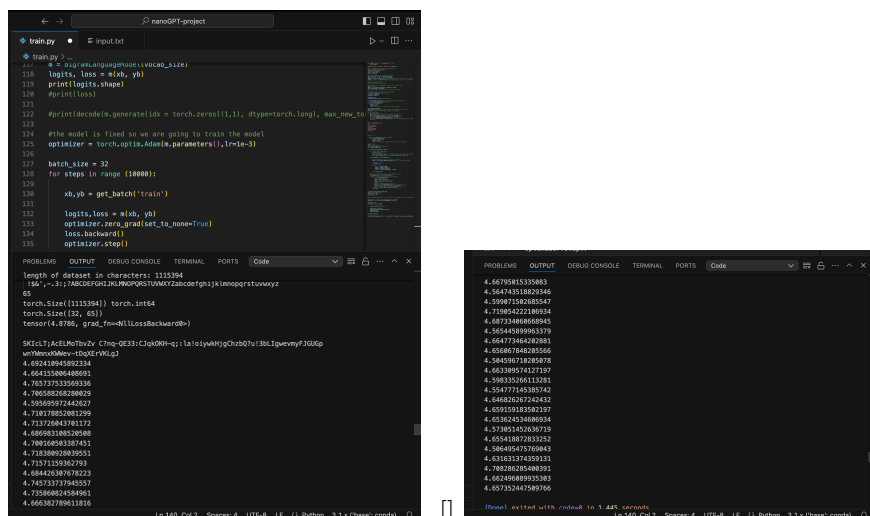


图 2: BigramLanguage Model (100 iterations)

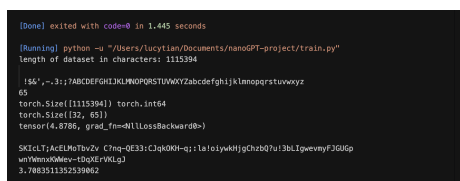


图 3: BigramLanguage Model (1000 iterations)

## 2.2.2 Multi-Head Attention

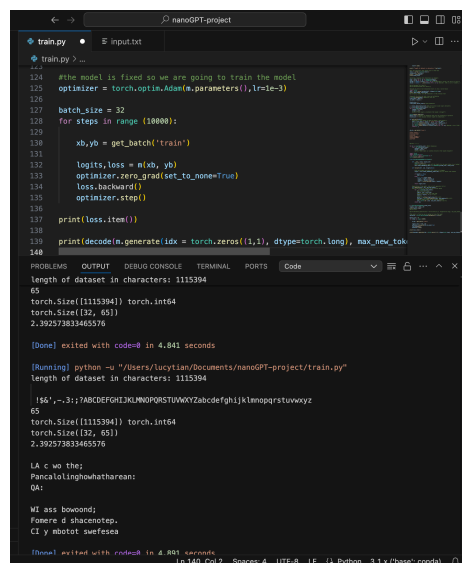
并行地建立多个头，改为多头注意力机制，模型训练得 loss 降为 2.28，变化其实并不大。[图 6]

## 2.3 Position-wise Feed-Forward Network

添加位置前馈层，训练模型得 loss 降为 2.24。

## 2.4 Residual Network

添加残差层，训练模型得 loss 降为 2.10 左右。稍微出现了过度拟合的现象。[图 8]



```
train.py • E input.txt
#the model is fixed so we are going to train the model
optimizer = torch.optim.Adam(model.parameters()), lr=1e-3

batch_size = 32
for steps in range(10000):
    xb, yb = get_batch('train')
    logits, loss = model(xb, yb)
    optimizer.zero_grad(set_to_none=True)
    loss.backward()
    optimizer.step()
    print(loss.item())

print(model.generate_idx = torch.zeros(1, 1), dtype=torch.long, max_new_tok=
length of dataset in characters: 1115394
65
torch.Size([1115394]) torch.int64
torch.Size([32, 65])
2.39257383465576

[Done] exited with code=0 in 4.841 seconds

[Running] python -u "/Users/loeytan/Documents/nanoGPT-project/train.py"
length of dataset in characters: 1115394
150',-3;7ABCEFGHIJKLMNOPQRSTUVWXYZabcedefghijklmnopqrstuvwxyz
65
torch.Size([1115394]) torch.int64
torch.Size([32, 65])
2.39257383465576

LA c an the;
Pmclal.lngmwhatharean;
QM;
MI xss bowand;
Fomere d shacenotep;
CI y mbotot swefesca

[Done] exited with code=0 in 4.841 seconds
Ln 140, Col 2 Spaces: 4 UTF-8 LF Python 3.12 (base:conda)
```

图 4: BigramLanguage Model (10 000 iterations)

## 2.5 Layer Norm

结合规范层，训练模型，loss 降为 2.09。[图 9]

## 2.6 Dropout

添加 dropout 层，阻止某些节点进行通信。dropout=0.2，即超过五分之一的中间计算都去除。迭代次数改为 2000 次。训练模型后得到 loss 降至 1.85 左右。[图 10]

迭代次数改为 5000 次。训练后模型 loss 降至 1.68 左右。具有过度拟合的现象。[图 11]

## 3 Results

基本上完成了根据输入语料生成文本的任务，但是仔细看生成的语言单词拼写有错有对，并且没有内容逻辑可言。但是通过层层搭建，直观地理解了 GPT 的原理，每一层存在的作用，数据间的交流。

```
[Done] exited with code=0 in 7.285 seconds

[Running] python -u "/Users/Lucylian/Documents/nanoGPT-project/train.py"
length of dataset in characters: 1115394

16',-;3;7ABCEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
05
torch.Size([1115394]) torch.int64
step 0: train loss 4.2842, val loss 4.2808
step 500: train loss 2.7802, val loss 2.7266
step 1000: train loss 2.5121, val loss 2.5298
step 1500: train loss 2.4792, val loss 2.4606
step 2000: train loss 2.4392, val loss 2.4632
step 2500: train loss 2.4258, val loss 2.4487
step 3000: train loss 2.4157, val loss 2.4462
step 3500: train loss 2.3971, val loss 2.4311
step 4000: train loss 2.4647, val loss 2.4652
step 4500: train loss 2.3968, val loss 2.4972

SAN:
f'ccocod bel ary, ayst kn:
AII, CQJName
Aloou Sileon, sde, ay oy ghasthe theth ind, hofe arts vlas!

ses, Aran ati
Ans, romloret de frs.

S; ant! wos fucow aithisrou theppoud ord,
Why law with yowand?

lto aundy whge thoust adre-
ti
Maour ther he, vesce fri prade adithath arees th t'sut I foroubelale ter fifer balay th the, lant.
de.

More inut siw de alk my heshacke out? deword ile ireavest.
mes os, hofur.

Sing, we fr outh sew fed sand lilll bat.
Hcoughe he, te your, adr xhagrat.

[Done] exited with code=0 in 7.474 seconds
```

图 5: Single Head Attention

## 4 补充

代码的源文件 train.py 在压缩包中, 也包含了数据集 input.txt。压缩包出问题可以在 GitHub 上找到。笔记本下载了 Pytorch 可以在本地 python 环境下跑, 训练时间的也很短。可以改变 input 的内容, 不过数据集需要尽量大一点。

下载并解压文件后, cd 到当前文件目录下, 即 nanoGPT-project 下。然后只接输入 python train.py 便可以看到 input 数据集集中的所有字符数, 字符类型, 类型数, 5000 次迭代下 loss 的变化, 以及最终生成 500 字的文本。可以更改 input 内容再次训练模型, 看生成文本。

GitHub address : <https://github.com/baazinga/nanoGPT-project>

```

torch.Size([1115394]) torch.int64
step 0: train loss 4.2227, val loss 4.2226
step 500: train loss 2.6587, val loss 2.6728
step 1000: train loss 2.4975, val loss 2.5057
step 1500: train loss 2.4279, val loss 2.4337
step 2000: train loss 2.3698, val loss 2.3832
step 2500: train loss 2.3399, val loss 2.3549
step 3000: train loss 2.3129, val loss 2.3336
step 3500: train loss 2.2987, val loss 2.3168
step 4000: train loss 2.2877, val loss 2.2859
step 4500: train loss 2.2726, val loss 2.2849

KORNG
you'd rad winnan:

Soo hittis covis is.

BOF lign the tus darge.

Win bregun cou tind spoon meand for.

GANCORNE Srir, al maxittem fot in,
Shour so the urg
And youurture, fakesit shibre?

Mouit wof the hany wilighereequith to nou thou s, it ap steranebe you on htkak be lelliliver?
STY:
Guterd bu thy are het, tom nithat.

FANG ENRL:
Acu Espouill of what hiemmagr.

BOUCKR:
Whe I dyen, mey froh hott
A mallice pry'll low's wdt:
Youh,
Rferat blougar,
A sin wislar,
Mr thit you tlaverdis

[Done] exited with code=0 in 14.953 seconds

```

图 6: Multi-Head Attention

```

[Running] python -u "python3 src/documents/mem2p3-project/train.py"
length of dataset in characters: 1115394
[1, ..., 3] (AMC8ZF0GJLHNM9P8T5WAT5Acw6p8j3k1emagp5tweayw
0)
torch.Size([1115394]) torch.int64
step 0: train loss 4.2227, val loss 4.2226
step 500: train loss 2.5986, val loss 2.5872
step 1000: train loss 2.4822, val loss 2.4842
step 1500: train loss 2.3965, val loss 2.3939
step 2000: train loss 2.3279, val loss 2.3451
step 2500: train loss 2.2801, val loss 2.3109
step 3000: train loss 2.2601, val loss 2.2912
step 3500: train loss 2.2406, val loss 2.2798
step 4000: train loss 2.2402, val loss 2.2482
step 4500: train loss 2.2258, val loss 2.2483

Vuel heris hut?

WU:
Eto help grem.
Thess lise?
It's so wery gaster compaf,
Blawevite wofis.
The is lile if thut get the naty fot lue it you to kiffis I twethit may hore dore, is or ter nou's it wldt for adawide lilly.
So this cet of hse weme to.
Boretilik liseed are And wemes on it to cant poly this.

Sord timewels,
Inverse thard dliettey:
Wome gowt and lictet to Nefine
Bartrot eper uou?
Thut?

To Pkissier feed your the, to it to pook.
Pry fot whalls prying sacromtis

[Done] exited with code=0 in 42.159 seconds

```

图 7: Feed-Forward Network

```

[Running] python -u "python3 src/documents/mem2p3-project/train.py"
length of dataset in characters: 1115394
[1, ..., 3] (AMC8ZF0GJLHNM9P8T5WAT5Acw6p8j3k1emagp5tweayw
0)
torch.Size([1115394]) torch.int64
step 0: train loss 4.2227, val loss 4.2226
step 500: train loss 2.3874, val loss 2.3869
step 1000: train loss 2.3096, val loss 2.3098
step 1500: train loss 2.1973, val loss 2.2008
step 2000: train loss 2.1403, val loss 2.1626
step 2500: train loss 2.1047, val loss 2.1105
step 3000: train loss 2.0852, val loss 2.1025
step 3500: train loss 2.0582, val loss 2.1183
step 4000: train loss 2.0528, val loss 2.1062
step 4500: train loss 2.0618, val loss 2.1026
step 4999: train loss 2.0623, val loss 2.1008

You dound.
Willbay as grow Bay ay for san dautistide this your upe? it heort bowing the wotises sadiske welfare but to so for and you folated cry.

WU:
Let's glawfare the firtat.
It isent not use in the pries to in hore, boodregep, a hot of wery, deat of and dard dweat, we come soured.
Aun Jorkeaf to the low thare feli fortat? But the post Gellio pay in the ate but at hase connecticlosest? but claudier, your thy gwe.
The first dardline fere fere pte.
The fere not wate or any.

[Done] exited with code=0 in 37.188 seconds

```

图 8: Residual Network

```
(pytorch) (base) lucytian@Lucys-MacBook-Air nanoGPT-project % python train.py
length of dataset in characters: 1115394

[58',-3;7ABCEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
65
torch.Size([1115394]) torch.int64
step 0: train loss 4.2383, val loss 4.3897
step 500: train loss 2.3987, val loss 2.3995
step 1000: train loss 2.2635, val loss 2.2652
step 1500: train loss 2.1655, val loss 2.1806
step 2000: train loss 2.1297, val loss 2.1668
step 2500: train loss 2.0793, val loss 2.1383
step 3000: train loss 2.0495, val loss 2.1232
step 3500: train loss 2.0437, val loss 2.1050
step 4000: train loss 2.0181, val loss 2.0932
step 4500: train loss 1.9980, val loss 2.0946
step 4999: train loss 1.9980, val loss 2.0946

shall, and men this at lan by in hone hold dove:
Awell. This satute a dume
they
Thou that my is si:
Micherey on seat
Wich so for and you foldie I tentyel? Grodand enver' withfore the fick
Hadill!
The duke? O's? Thin him.

SINGSice, Theel-fatresp but as thou dun of bet of and
Sthe doak'd you manited with Me. YORJ; Our to but looughit:
My have with thy heast,
shat didd pay vldest dreads to lood shough him?

KING VINCENTIUS:
Thien are ay gran
The that nod loves may hear you
The joinder God helved
(pytorch) (base) lucytian@Lucys-MacBook-Air nanoGPT-project %
```

图 9: Layer Norm

```
nanoGPT-project --zsh --100x35
Last login: Sun Jun 30 11:13:00 on ttys004
(base) lucytian@Lucys-MacBook-Air ~ % cd Documents
(base) lucytian@Lucys-MacBook-Air Documents % cd nanoGPT-project
(base) lucytian@Lucys-MacBook-Air nanoGPT-project % python train.py
length of dataset in characters: 1115394

[58',-3;7ABCEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
65
torch.Size([1115394]) torch.int64
step 0: train loss 4.5071, val loss 4.5132
step 500: train loss 2.1137, val loss 2.1621
step 1000: train loss 1.8096, val loss 2.0882
step 1500: train loss 1.7599, val loss 1.8574
step 1999: train loss 1.7599, val loss 1.8574

Will dukanten-dwith unchious say praard
Strave arive, air. 'I will be bet' one tall!
You cannot will our tiseitf.
There is to curves, and you of to frome the doe,
And clont bottles anglion; ou now: not yet
tll
be bettle-mort; where will clonting and no flain!

PULINA:
Be stage by to tunge.

LEONTER:
I have bette hands upon will you till greet.;
Ye my etill wolve re this villoud said is ingain ne of of braze
That upon's pabssin stand with his slal us: the have.

HERMID:
no my lord one are few
(base) lucytian@Lucys-MacBook-Air nanoGPT-project %
```

图 10: Dropout (2000 iterations)

```
nanoGPT-project --zsh --128x47
Last login: Sun Jun 30 11:17:12 on ttys004
(base) lucytian@Lucys-MacBook-Air ~ % cd Documents
(base) lucytian@Lucys-MacBook-Air Documents % cd nanoGPT-project
(base) lucytian@Lucys-MacBook-Air nanoGPT-project % python train.py
length of dataset in characters: 1115394

[58',-3;7ABCEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
65
torch.Size([1115394]) torch.int64
step 0: train loss 4.5254, val loss 4.5249
step 500: train loss 2.2375, val loss 2.1387
step 1000: train loss 1.8739, val loss 1.8751
step 1500: train loss 1.7086, val loss 1.8803
step 2000: train loss 1.6072, val loss 1.8458
step 2500: train loss 1.4316, val loss 1.7945
step 3000: train loss 1.3881, val loss 1.7382
step 3500: train loss 1.5049, val loss 1.7382
step 4000: train loss 1.5027, val loss 1.7514
step 4500: train loss 1.4998, val loss 1.6879
step 4999: train loss 1.4998, val loss 1.6879

GATER:
If that we are this in Pabssid can person
Lord Nio I Juliet-wast?

GLOUCESTER:
He, father, it is, brook! the one that?

CONTOLAND:
Sir, may! he that tendit!—

BENVOLO:
I be notice the fair be tooth spears to hear do
wink-doid natied, after speak no non none's
this heicety; who wanting outlins of ne profitin.

CAPULET:
When are?

RUCKINDAM:
Alas much we sites, as I'll relie, see great for my
scuricid the chies may upon be: the duns
high alikment in alik'd to pleasey:
Let be th
(base) lucytian@Lucys-MacBook-Air nanoGPT-project %
```

图 11: Dropout (5000 iterations)