```python
import os #Working with os
import urllib.request #Downloading online files

#Downloading the dataset from an online url
if not os.path.exists("the-verdict-txt"):
    url = ("https://raw.githubusercontent.com/rasbt/LLMs-from-
scratch/refs/heads/main/ch02/01_main-chapter-code/the-verdict.txt")
    file_path = "the-verdict-txt"
    urllib.request.urlretrieve(url, file_path)

#Opening the file (dataset)
with open("the-verdict-txt", "r", encoding="utf-8") as f:
    raw_text = f.read()

#Looking at text
raw_text
```

```
'I HAD always thought Jack Gisburn rather a cheap genius--though a
good fellow enough--so it was no great surprise to me to hear that, in
the height of his glory, he had dropped his painting, married a rich
widow, and established himself in a villa on the Riviera. (Though I
rather thought it would have been Rome or Florence.)\n\n"The height of
his glory"--that was what the women called it. I can hear Mrs. Gideon
Thwing--his last Chicago sitter--deploring his unaccountable
abdication. "Of course it\'s going to send the value of my
picture \'way up; but I don\'t think of that, Mr. Rickham--the loss to
Arrt is all I think of." The word, on Mrs. Thwing\'s lips, multiplied
its _rs_ as though they were reflected in an endless vista of mirrors.
And it was not only the Mrs. Thwings who mourned. Had not the
exquisite Hermia Croft, at the last Grafton Gallery show, stopped me
before Gisburn\'s "Moon-dancers" to say, with tears in her eyes: "We
shall not look upon its like again"?\n\nWell!--even through the prism
of Hermia\'s tears I felt able to face the fact with equanimity. Poor
Jack Gisburn! The women had made him--it was fitting that they should
mourn him. Among his own sex fewer regrets were heard, and in his own
trade hardly a murmur. Professional jealousy? Perhaps. If it were, the
honour of the craft was vindicated by little Claude Nutley, who, in
all good faith, brought out in the Burlington a very handsome
"obituary" on Jack--one of those showy articles stocked with random
technicalities that I have heard (I won\'t say by whom) compared to
Gisburn\'s painting. And so--his resolve being apparently
irrevocable--the discussion gradually died out, and, as Mrs. Thwing
had predicted, the price of "Gisburns" went up.\n\nIt was not till
three years later that, in the course of a few weeks\' idling on the
Riviera, it suddenly occurred to me to wonder why Gisburn had given up
his painting. On reflection, it really was a tempting problem. To
accuse his wife would have been too easy--his fair sitters had been
denied the solace of saying that Mrs. Gisburn had "dragged him down."
For Mrs. Gisburn--as such--had not existed till nearly a year after
Jack\'s resolve had been taken. It might be that he had married her--
```

since he liked his ease--because he didn\'t want to go on painting; but it would have been hard to prove that he had given up his painting because he had married her.\n\nOf course, if she had not dragged him down, she had equally, as Miss Croft contended, failed to "lift him up"--she had not led him back to the easel. To put the brush into his hand again--what a vocation for a wife! But Mrs. Gisburn appeared to have disdained it--and I felt it might be interesting to find out why.\n\nThe desultory life of the Riviera lends itself to such purely academic speculations; and having, on my way to Monte Carlo, caught a glimpse of Jack\'s balustraded terraces between the pines, I had myself borne thither the next day.\n\nI found the couple at tea beneath their palm-trees; and Mrs. Gisburn\'s welcome was so genial that, in the ensuing weeks, I claimed it frequently. It was not that my hostess was "interesting": on that point I could have given Miss Croft the fullest reassurance. It was just because she was _not_ interesting--if I may be pardoned the bull--that I found her so. For Jack, all his life, had been surrounded by interesting women: they had fostered his art, it had been reared in the hot-house of their adulation. And it was therefore instructive to note what effect the "deadening atmosphere of mediocrity" (I quote Miss Croft) was having on him.\n\nI have mentioned that Mrs. Gisburn was rich; and it was immediately perceptible that her husband was extracting from this circumstance a delicate but substantial satisfaction. It is, as a rule, the people who scorn money who get most out of it; and Jack\'s elegant disdain of his wife\'s big balance enabled him, with an appearance of perfect good-breeding, to transmute it into objects of art and luxury. To the latter, I must add, he remained relatively indifferent; but he was buying Renaissance bronzes and eighteenth-century pictures with a discrimination that bespoke the amplest resources.\n\n"Money\'s only excuse is to put beauty into circulation," was one of the axioms he laid down across the Sevres and silver of an exquisitely appointed luncheon-table, when, on a later day, I had again run over from Monte Carlo; and Mrs. Gisburn, beaming on him, added for my enlightenment: "Jack is so morbidly sensitive to every form of beauty."\n\nPoor Jack! It had always been his fate to have women say such things of him: the fact should be set down in extenuation. What struck me now was that, for the first time, he resented the tone. I had seen him, so often, basking under similar tributes--was it the conjugal note that robbed them of their savour? No--for, oddly enough, it became apparent that he was fond of Mrs. Gisburn--fond enough not to see her absurdity. It was his own absurdity he seemed to be wincing under--his own attitude as an object for garlands and incense.\n\n"My dear, since I\'ve chucked painting people don\'t say that stuff about me--they say it about Victor Grindle," was his only protest, as he rose from the table and strolled out onto the sunlit terrace.\n\nI glanced after him, struck by his last word. Victor Grindle was, in fact, becoming the man of the moment--as Jack himself, one might put it, had been the man of the hour. The younger artist was said to have formed himself at my

friend\'s feet, and I wondered if a tinge of jealousy underlay the latter\'s mysterious abdication. But no--for it was not till after that event that the _rose Dubarry_ drawing-rooms had begun to display their "Grindles."\n\nI turned to Mrs. Gisburn, who had lingered to give a lump of sugar to her spaniel in the dining-room.\n\n"Why _has_ he chucked painting?" I asked abruptly.\n\nShe raised her eyebrows with a hint of good-humoured surprise.\n\n"Oh, he doesn\'t _have_ to now, you know; and I want him to enjoy himself," she said quite simply.\n\nI looked about the spacious white-panelled room, with its _famille-verte_ vases repeating the tones of the pale damask curtains, and its eighteenth-century pastels in delicate faded frames.\n\n"Has he chucked his pictures too? I haven\'t seen a single one in the house."\n\nA slight shade of constraint crossed Mrs. Gisburn\'s open countenance. "It\'s his ridiculous modesty, you know. He says they\'re not fit to have about; he\'s sent them all away except one--my portrait--and that I have to keep upstairs."\n\nHis ridiculous modesty--Jack\'s modesty about his pictures? My curiosity was growing like the bean-stalk. I said persuasively to my hostess: "I must really see your portrait, you know."\n\nShe glanced out almost timorously at the terrace where her husband, lounging in a hooded chair, had lit a cigar and drawn the Russian deerhound\'s head between his knees.\n\n"Well, come while he\'s not looking," she said, with a laugh that tried to hide her nervousness; and I followed her between the marble Emperors of the hall, and up the wide stairs with terra-cotta nymphs poised among flowers at each landing.\n\nIn the dimmest corner of her boudoir, amid a profusion of delicate and distinguished objects, hung one of the familiar oval canvases, in the inevitable garlanded frame. The mere outline of the frame called up all Gisburn\'s past!\n\nMrs. Gisburn drew back the window-curtains, moved aside a _jardiniere_ full of pink azaleas, pushed an arm-chair away, and said: "If you stand here you can just manage to see it. I had it over the mantel-piece, but he wouldn\'t let it stay."\n\nYes--I could just manage to see it--the first portrait of Jack\'s I had ever had to strain my eyes over! Usually they had the place of honour--say the central panel in a pale yellow or _rose Dubarry_ drawing-room, or a monumental easel placed so that it took the light through curtains of old Venetian point. The more modest place became the picture better; yet, as my eyes grew accustomed to the half-light, all the characteristic qualities came out--all the hesitations disguised as audacities, the tricks of prestidigitation by which, with such consummate skill, he managed to divert attention from the real business of the picture to some pretty irrelevance of detail. Mrs. Gisburn, presenting a neutral surface to work on--forming, as it were, so inevitably the background of her own picture--had lent herself in an unusual degree to the display of this false virtuosity. The picture was one of Jack\'s "strongest," as his admirers would have put it--it represented, on his part, a swelling of muscles, a congesting of veins, a balancing, straddling and straining, that reminded one of the circus-clown\'s ironic efforts to lift a feather. It met, in short, at every point the demand of lovely woman

to be painted "strongly" because she was tired of being painted "sweetly"--and yet not to lose an atom of the sweetness.\n\n"It\'s the last he painted, you know," Mrs. Gisburn said with pardonable pride. "The last but one," she corrected herself--"but the other doesn\'t count, because he destroyed it."\n\n"Destroyed it?" I was about to follow up this clue when I heard a footstep and saw Jack himself on the threshold.\n\nAs he stood there, his hands in the pockets of his velveteen coat, the thin brown waves of hair pushed back from his white forehead, his lean sunburnt cheeks furrowed by a smile that lifted the tips of a self-confident moustache, I felt to what a degree he had the same quality as his pictures--the quality of looking cleverer than he was.\n\nHis wife glanced at him deprecatingly, but his eyes travelled past her to the portrait.\n\n"Mr. Rickham wanted to see it," she began, as if excusing herself. He shrugged his shoulders, still smiling.\n\n"Oh, Rickham found me out long ago," he said lightly; then, passing his arm through mine: "Come and see the rest of the house."\n\nHe showed it to me with a kind of naive suburban pride: the bath-rooms, the speaking-tubes, the dress-closets, the trouser-presses--all the complex simplifications of the millionaire\'s domestic economy. And whenever my wonder paid the expected tribute he said, throwing out his chest a little: "Yes, I really don\'t see how people manage to live without that."\n\nWell--it was just the end one might have foreseen for him. Only he was, through it all and in spite of it all--as he had been through, and in spite of, his pictures--so handsome, so charming, so disarming, that one longed to cry out: "Be dissatisfied with your leisure!" as once one had longed to say: "Be dissatisfied with your work!"\n\nBut, with the cry on my lips, my diagnosis suffered an unexpected check.\n\n"This is my own lair," he said, leading me into a dark plain room at the end of the florid vista. It was square and brown and leathery: no "effects"; no bric-a-brac, none of the air of posing for reproduction in a picture weekly--above all, no least sign of ever having been used as a studio.\n\nThe fact brought home to me the absolute finality of Jack\'s break with his old life.\n\n"Don\'t you ever dabble with paint any more?" I asked, still looking about for a trace of such activity.\n\n"Never," he said briefly.\n\n"Or water-colour--or etching?"\n\nHis confident eyes grew dim, and his cheeks paled a little under their handsome sunburn.\n\n"Never think of it, my dear fellow--any more than if I\'d never touched a brush."\n\nAnd his tone told me in a flash that he never thought of anything else.\n\nI moved away, instinctively embarrassed by my unexpected discovery; and as I turned, my eye fell on a small picture above the mantel-piece--the only object breaking the plain oak panelling of the room.\n\n"Oh, by Jove!" I said.\n\nIt was a sketch of a donkey--an old tired donkey, standing in the rain under a wall.\n\n"By Jove--a Stroud!" I cried.\n\nHe was silent; but I felt him close behind me, breathing a little quickly.\n\n"What a wonder! Made with a dozen lines--but on everlasting foundations. You lucky chap, where did you get it?"\n\nHe answered slowly: "Mrs. Stroud gave it to me."\n\n"Ah--I didn\'t know you even knew the Strouds. He

was such an inflexible hermit."\n\n"I didn\'t--till after. . . . She sent for me to paint him when he was dead."\n\n"When he was dead? You?"\n\nI must have let a little too much amazement escape through my surprise, for he answered with a deprecating laugh: "Yes--she\'s an awful simpleton, you know, Mrs. Stroud. Her only idea was to have him done by a fashionable painter--ah, poor Stroud! She thought it the surest way of proclaiming his greatness--of forcing it on a purblind public. And at the moment I was _the_ fashionable painter."\n\n"Ah, poor Stroud--as you say. Was _that_ his history?"\n\n"That was his history. She believed in him, gloried in him--or thought she did. But she couldn\'t bear not to have all the drawing-rooms with her. She couldn\'t bear the fact that, on varnishing days, one could always get near enough to see his pictures. Poor woman! She\'s just a fragment groping for other fragments. Stroud is the only whole I ever knew."\n\n"You ever knew? But you just said--"\n\nGisburn had a curious smile in his eyes.\n\n"Oh, I knew him, and he knew me--only it happened after he was dead."\n\nI dropped my voice instinctively. "When she sent for you?"\n\n"Yes--quite insensible to the irony. She wanted him vindicated--and by me!"\n\nHe laughed again, and threw back his head to look up at the sketch of the donkey. "There were days when I couldn\'t look at that thing--couldn\'t face it. But I forced myself to put it here; and now it\'s cured me--cured me. That\'s the reason why I don\'t dabble any more, my dear Rickham; or rather Stroud himself is the reason."\n\nFor the first time my idle curiosity about my companion turned into a serious desire to understand him better.\n\n"I wish you\'d tell me how it happened," I said.\n\nHe stood looking up at the sketch, and twirling between his fingers a cigarette he had forgotten to light. Suddenly he turned toward me.\n\n"I\'d rather like to tell you--because I\'ve always suspected you of loathing my work."\n\nI made a deprecating gesture, which he negatived with a good-humoured shrug.\n\n"Oh, I didn\'t care a straw when I believed in myself--and now it\'s an added tie between us!"\n\nHe laughed slightly, without bitterness, and pushed one of the deep arm-chairs forward. "There: make yourself comfortable--and here are the cigars you like."\n\nHe placed them at my elbow and continued to wander up and down the room, stopping now and then beneath the picture.\n\n"How it happened? I can tell you in five minutes--and it didn\'t take much longer to happen. . . . I can remember now how surprised and pleased I was when I got Mrs. Stroud\'s note. Of course, deep down, I had always _felt_ there was no one like him--only I had gone with the stream, echoed the usual platitudes about him, till I half got to think he was a failure, one of the kind that are left behind. By Jove, and he _was_ left behind--because he had come to stay! The rest of us had to let ourselves be swept along or go under, but he was high above the current--on everlasting foundations, as you say.\n\n"Well, I went off to the house in my most egregious mood--rather moved, Lord forgive me, at the pathos of poor Stroud\'s career of failure being crowned by the glory of my painting him! Of course I meant to do the picture for nothing--I told Mrs. Stroud so when she began to stammer something

about her poverty. I remember getting off a prodigious phrase about the honour being _mine_--oh, I was princely, my dear Rickham! I was posing to myself like one of my own sitters.\n\n"Then I was taken up and left alone with him. I had sent all my traps in advance, and I had only to set up the easel and get to work. He had been dead only twenty-four hours, and he died suddenly, of heart disease, so that there had been no preliminary work of destruction--his face was clear and untouched. I had met him once or twice, years before, and thought him insignificant and dingy. Now I saw that he was superb.\n\n"I was glad at first, with a merely aesthetic satisfaction: glad to have my hand on such a \'subject.\' Then his strange life-likeness began to affect me queerly--as I blocked the head in I felt as if he were watching me do it. The sensation was followed by the thought: if he _were_ watching me, what would he say to my way of working? My strokes began to go a little wild--I felt nervous and uncertain.\n\n"Once, when I looked up, I seemed to see a smile behind his close grayish beard--as if he had the secret, and were amusing himself by holding it back from me. That exasperated me still more. The secret? Why, I had a secret worth twenty of his! I dashed at the canvas furiously, and tried some of my bravura tricks. But they failed me, they crumbled. I saw that he wasn\'t watching the showy bits--I couldn\'t distract his attention; he just kept his eyes on the hard passages between. Those were the ones I had always shirked, or covered up with some lying paint. And how he saw through my lies!\n\n"I looked up again, and caught sight of that sketch of the donkey hanging on the wall near his bed. His wife told me afterward it was the last thing he had done--just a note taken with a shaking hand, when he was down in Devonshire recovering from a previous heart attack. Just a note! But it tells his whole history. There are years of patient scornful persistence in every line. A man who had swum with the current could never have learned that mighty up-stream stroke. . . .\n\n"I turned back to my work, and went on groping and muddling; then I looked at the donkey again. I saw that, when Stroud laid in the first stroke, he knew just what the end would be. He had possessed his subject, absorbed it, recreated it. When had I done that with any of my things? They hadn\'t been born of me--I had just adopted them. . . .\n\n"Hang it, Rickham, with that face watching me I couldn\'t do another stroke. The plain truth was, I didn\'t know where to put it--_I had never known_. Only, with my sitters and my public, a showy splash of colour covered up the fact--I just threw paint into their faces. . . . Well, paint was the one medium those dead eyes could see through--see straight to the tottering foundations underneath. Don\'t you know how, in talking a foreign language, even fluently, one says half the time not what one wants to but what one can? Well--that was the way I painted; and as he lay there and watched me, the thing they called my \'technique\' collapsed like a house of cards. He didn\'t sneer, you understand, poor Stroud--he just lay there quietly watching, and on his lips, through the gray beard, I seemed to hear the question: \'Are you sure you know where you\'re coming out?\'\n\n"If I could have painted that

```
face, with that question on it, I should have done a great thing. The next greatest thing was to see that I couldn\'t--and that grace was given me. But, oh, at that minute, Rickham, was there anything on earth I wouldn\'t have given to have Stroud alive before me, and to hear him say: \'It\'s not too late--I\'ll show you how\'?\n\n"It _was_ too late--it would have been, even if he\'d been alive. I packed up my traps, and went down and told Mrs. Stroud. Of course I didn\'t tell her _that_--it would have been Greek to her. I simply said I couldn\'t paint him, that I was too moved. She rather liked the idea--she\'s so romantic! It was that that made her give me the donkey. But she was terribly upset at not getting the portrait--she did so want him \'done\' by some one showy! At first I was afraid she wouldn\'t let me off--and at my wits\' end I suggested Grindle. Yes, it was I who started Grindle: I told Mrs. Stroud he was the \'coming\' man, and she told somebody else, and so it got to be true. . . . And he painted Stroud without wincing; and she hung the picture among her husband\'s things. . . ."\n\nHe flung himself down in the arm-chair near mine, laid back his head, and clasping his arms beneath it, looked up at the picture above the chimney-piece.\n\n"I like to fancy that Stroud himself would have given it to me, if he\'d been able to say what he thought that day."\n\nAnd, in answer to a question I put half-mechanically--"Begin again?" he flashed out. "When the one thing that brings me anywhere near him is that I knew enough to leave off?"\n\nHe stood up and laid his hand on my shoulder with a laugh. "Only the irony of it is that I _am_ still painting--since Grindle\'s doing it for me! The Strouds stand alone, and happen once--but there\'s no exterminating our kind of art."'
```

```
#Checking for length of the dataset
len(raw_text)
```

```
20479
```

```
#Tokenization warmup
import re

text = "Hello guys welcome to my youtube channel."
result = re.split(r'(\s)', text)

print(result)
```

```
['Hello', ' ', 'guys', ' ', 'welcome', ' ', 'to', ' ', 'my', ' ', 'youtube', ' ', 'channel.']
```

```
result = re.split(r'([,.]|\s)', text)

print(result)
```

```
['Hello', ' ', 'guys', ' ', 'welcome', ' ', 'to', ' ', 'my', ' ', 'youtube', ' ', 'channel', '.', '']
```

```
#optional step to take out white space characters
result = [item for item in result if item.strip()]
print(result)
```

```
['Hello', 'guys', 'welcome', 'to', 'my', 'youtube', 'channel', '.']
```

```
text = "Hello guys welcome to my youtube channel."
```

```
result = re.split(r'([,.:;?_!"()\']--|\s)', raw_text)
result = [item.strip() for item in result if item.strip()]
print(result)
```

```
['I', 'HAD', 'always', 'thought', 'Jack', 'Gisburn', 'rather', 'a',
'cheap', 'genius--though', 'a', 'good', 'fellow', 'enough--so', 'it',
'was', 'no', 'great', 'surprise', 'to', 'me', 'to', 'hear', 'that,',
'in', 'the', 'height', 'of', 'his', 'glory,', 'he', 'had', 'dropped',
'his', 'painting,', 'married', 'a', 'rich', 'widow,', 'and',
'established', 'himself', 'in', 'a', 'villa', 'on', 'the', 'Riviera.',
'(Though', 'I', 'rather', 'thought', 'it', 'would', 'have', 'been',
'Rome', 'or', 'Florence.)', '"The', 'height', 'of', 'his', 'glory',
'"--', 'that', 'was', 'what', 'the', 'women', 'called', 'it.', 'I',
'can', 'hear', 'Mrs.', 'Gideon', 'Thwing--his', 'last', 'Chicago',
'sitter--deploring', 'his', 'unaccountable', 'abdication.', '"Of',
'course', "it's", 'going', 'to', 'send', 'the', 'value', 'of', 'my',
'picture', "'way", 'up;', 'but', 'I', "don't", 'think', 'of', 'that,',
'Mr.', 'Rickham--the', 'loss', 'to', 'Arrt', 'is', 'all', 'I',
'think', 'of."', 'The', 'word,', 'on', 'Mrs.', "Thwing's", 'lips,',
'multiplied', 'its', '_rs_', 'as', 'though', 'they', 'were',
'reflected', 'in', 'an', 'endless', 'vista', 'of', 'mirrors.', 'And',
'it', 'was', 'not', 'only', 'the', 'Mrs.', 'Thwings', 'who',
'mourned.', 'Had', 'not', 'the', 'exquisite', 'Hermia', 'Croft,',
'at', 'the', 'last', 'Grafton', 'Gallery', 'show,', 'stopped', 'me',
'before', "Gisburn's", '"Moon-dancers"', 'to', 'say,', 'with',
'tears', 'in', 'her', 'eyes:', '"We', 'shall', 'not', 'look', 'upon',
'its', 'like', 'again"?', 'Well', '!--', 'even', 'through', 'the',
'prism', 'of', "Hermia's", 'tears', 'I', 'felt', 'able', 'to', 'face',
'the', 'fact', 'with', 'equanimity.', 'Poor', 'Jack', 'Gisburn!',
'The', 'women', 'had', 'made', 'him--it', 'was', 'fitting', 'that',
'they', 'should', 'mourn', 'him.', 'Among', 'his', 'own', 'sex',
'fewer', 'regrets', 'were', 'heard,', 'and', 'in', 'his', 'own',
'trade', 'hardly', 'a', 'murmur.', 'Professional', 'jealousy?',
'Perhaps.', 'If', 'it', 'were,', 'the', 'honour', 'of', 'the',
'craft', 'was', 'vindicated', 'by', 'little', 'Claude', 'Nutley,',
'who,', 'in', 'all', 'good', 'faith,', 'brought', 'out', 'in', 'the',
'Burlington', 'a', 'very', 'handsome', '"obituary"', 'on', 'Jack--
one', 'of', 'those', 'showy', 'articles', 'stocked', 'with', 'random',
'technicalities', 'that', 'I', 'have', 'heard', '(I', "won't", 'say',
'by', 'whom)', 'compared', 'to', "Gisburn's", 'painting.', 'And',
'so--his', 'resolve', 'being', 'apparently', 'irrevocable--the',
'discussion', 'gradually', 'died', 'out,', 'and,', 'as', 'Mrs.',
```

'Thwing', 'had', 'predicted,', 'the', 'price', 'of', '"Gisburns"',
'went', 'up.', 'It', 'was', 'not', 'till', 'three', 'years', 'later',
'that,', 'in', 'the', 'course', 'of', 'a', 'few', "weeks'", 'idling',
'on', 'the', 'Riviera,', 'it', 'suddenly', 'occurred', 'to', 'me',
'to', 'wonder', 'why', 'Gisburn', 'had', 'given', 'up', 'his',
'painting.', 'On', 'reflection,', 'it', 'really', 'was', 'a',
'tempting', 'problem.', 'To', 'accuse', 'his', 'wife', 'would',
'have', 'been', 'too', 'easy--his', 'fair', 'sitters', 'had', 'been',
'denied', 'the', 'solace', 'of', 'saying', 'that', 'Mrs.', 'Gisburn',
'had', '"dragged', 'him', 'down."', 'For', 'Mrs.', 'Gisburn--as',
'such--had', 'not', 'existed', 'till', 'nearly', 'a', 'year', 'after',
"Jack's", 'resolve', 'had', 'been', 'taken.', 'It', 'might', 'be',
'that', 'he', 'had', 'married', 'her--since', 'he', 'liked', 'his',
'ease--because', 'he', "didn't", 'want', 'to', 'go', 'on',
'painting;', 'but', 'it', 'would', 'have', 'been', 'hard', 'to',
'prove', 'that', 'he', 'had', 'given', 'up', 'his', 'painting',
'because', 'he', 'had', 'married', 'her.', 'Of', 'course,', 'if',
'she', 'had', 'not', 'dragged', 'him', 'down,', 'she', 'had',
'equally,', 'as', 'Miss', 'Croft', 'contended,', 'failed', 'to',
'"lift', 'him', 'up', '"--', 'she', 'had', 'not', 'led', 'him',
'back', 'to', 'the', 'easel.', 'To', 'put', 'the', 'brush', 'into',
'his', 'hand', 'again--what', 'a', 'vocation', 'for', 'a', 'wife!',
'But', 'Mrs.', 'Gisburn', 'appeared', 'to', 'have', 'disdained', 'it--
and', 'I', 'felt', 'it', 'might', 'be', 'interesting', 'to', 'find',
'out', 'why.', 'The', 'desultory', 'life', 'of', 'the', 'Riviera',
'lends', 'itself', 'to', 'such', 'purely', 'academic',
'speculations;', 'and', 'having,', 'on', 'my', 'way', 'to', 'Monte',
'Carlo,', 'caught', 'a', 'glimpse', 'of', "Jack's", 'balustraded',
'terraces', 'between', 'the', 'pines,', 'I', 'had', 'myself', 'borne',
'thither', 'the', 'next', 'day.', 'I', 'found', 'the', 'couple', 'at',
'tea', 'beneath', 'their', 'palm-trees;', 'and', 'Mrs.', "Gisburn's",
'welcome', 'was', 'so', 'genial', 'that,', 'in', 'the', 'ensuing',
'weeks,', 'I', 'claimed', 'it', 'frequently.', 'It', 'was', 'not',
'that', 'my', 'hostess', 'was', '"interesting":', 'on', 'that',
'point', 'I', 'could', 'have', 'given', 'Miss', 'Croft', 'the',
'fullest', 'reassurance.', 'It', 'was', 'just', 'because', 'she',
'was', '_not_', 'interesting--if', 'I', 'may', 'be', 'pardoned',
'the', 'bull--that', 'I', 'found', 'her', 'so.', 'For', 'Jack,',
'all', 'his', 'life,', 'had', 'been', 'surrounded', 'by',
'interesting', 'women:', 'they', 'had', 'fostered', 'his', 'art,',
'it', 'had', 'been', 'reared', 'in', 'the', 'hot-house', 'of',
'their', 'adulation.', 'And', 'it', 'was', 'therefore', 'instructive',
'to', 'note', 'what', 'effect', 'the', '"deadening', 'atmosphere',
'of', 'mediocrity"', '(I', 'quote', 'Miss', 'Croft)', 'was', 'having',
'on', 'him.', 'I', 'have', 'mentioned', 'that', 'Mrs.', 'Gisburn',
'was', 'rich;', 'and', 'it', 'was', 'immediately', 'perceptible',
'that', 'her', 'husband', 'was', 'extracting', 'from', 'this',
'circumstance', 'a', 'delicate', 'but', 'substantial',
'satisfaction.', 'It', 'is,', 'as', 'a', 'rule,', 'the', 'people',

```
'who', 'scorn', 'money', 'who', 'get', 'most', 'out', 'of', 'it;',
'and', "Jack's", 'elegant', 'disdain', 'of', 'his', "wife's", 'big',
'balance', 'enabled', 'him,', 'with', 'an', 'appearance', 'of',
'perfect', 'good-breeding,', 'to', 'transmute', 'it', 'into',
'objects', 'of', 'art', 'and', 'luxury.', 'To', 'the', 'latter,', 'I',
'must', 'add,', 'he', 'remained', 'relatively', 'indifferent;', 'but',
'he', 'was', 'buying', 'Renaissance', 'bronzes', 'and', 'eighteenth-
century', 'pictures', 'with', 'a', 'discrimination', 'that',
'bespoke', 'the', 'amplest', 'resources.', '"Money\'s', 'only',
'excuse', 'is', 'to', 'put', 'beauty', 'into', 'circulation,"', 'was',
'one', 'of', 'the', 'axioms', 'he', 'laid', 'down', 'across', 'the',
'Sevres', 'and', 'silver', 'of', 'an', 'exquisitely', 'appointed',
'luncheon-table,', 'when,', 'on', 'a', 'later', 'day,', 'I', 'had',
'again', 'run', 'over', 'from', 'Monte', 'Carlo;', 'and', 'Mrs.',
'Gisburn,', 'beaming', 'on', 'him,', 'added', 'for', 'my',
'enlightenment:', '"Jack', 'is', 'so', 'morbidly', 'sensitive', 'to',
'every', 'form', 'of', 'beauty."', 'Poor', 'Jack!', 'It', 'had',
'always', 'been', 'his', 'fate', 'to', 'have', 'women', 'say', 'such',
'things', 'of', 'him:', 'the', 'fact', 'should', 'be', 'set', 'down',
'in', 'extenuation.', 'What', 'struck', 'me', 'now', 'was', 'that,',
'for', 'the', 'first', 'time,', 'he', 'resented', 'the', 'tone.', 'I',
'had', 'seen', 'him,', 'so', 'often,', 'basking', 'under', 'similar',
'tributes--was', 'it', 'the', 'conjugal', 'note', 'that', 'robbed',
'them', 'of', 'their', 'savour?', 'No--for,', 'oddly', 'enough,',
'it', 'became', 'apparent', 'that', 'he', 'was', 'fond', 'of', 'Mrs.',
'Gisburn--fond', 'enough', 'not', 'to', 'see', 'her', 'absurdity.',
'It', 'was', 'his', 'own', 'absurdity', 'he', 'seemed', 'to', 'be',
'wincing', 'under--his', 'own', 'attitude', 'as', 'an', 'object',
'for', 'garlands', 'and', 'incense.', '"My', 'dear,', 'since', "I've",
'chucked', 'painting', 'people', "don't", 'say', 'that', 'stuff',
'about', 'me--they', 'say', 'it', 'about', 'Victor', 'Grindle,"',
'was', 'his', 'only', 'protest,', 'as', 'he', 'rose', 'from', 'the',
'table', 'and', 'strolled', 'out', 'onto', 'the', 'sunlit',
'terrace.', 'I', 'glanced', 'after', 'him,', 'struck', 'by', 'his',
'last', 'word.', 'Victor', 'Grindle', 'was,', 'in', 'fact,',
'becoming', 'the', 'man', 'of', 'the', 'moment--as', 'Jack',
'himself,', 'one', 'might', 'put', 'it,', 'had', 'been', 'the', 'man',
'of', 'the', 'hour.', 'The', 'younger', 'artist', 'was', 'said', 'to',
'have', 'formed', 'himself', 'at', 'my', "friend's", 'feet,', 'and',
'I', 'wondered', 'if', 'a', 'tinge', 'of', 'jealousy', 'underlay',
'the', "latter's", 'mysterious', 'abdication.', 'But', 'no--for',
'it', 'was', 'not', 'till', 'after', 'that', 'event', 'that', 'the',
'_rose', 'Dubarry_', 'drawing-rooms', 'had', 'begun', 'to', 'display',
'their', '"Grindles."', 'I', 'turned', 'to', 'Mrs.', 'Gisburn,',
'who', 'had', 'lingered', 'to', 'give', 'a', 'lump', 'of', 'sugar',
'to', 'her', 'spaniel', 'in', 'the', 'dining-room.', '"Why', '_has_',
'he', 'chucked', 'painting?"', 'I', 'asked', 'abruptly.', 'She',
'raised', 'her', 'eyebrows', 'with', 'a', 'hint', 'of', 'good-
humoured', 'surprise.', '"Oh,', 'he', "doesn't", '_have_', 'to',
```

```
'now,', 'you', 'know;', 'and', 'I', 'want', 'him', 'to', 'enjoy',
'himself,"', 'she', 'said', 'quite', 'simply.', 'I', 'looked',
'about', 'the', 'spacious', 'white-panelled', 'room,', 'with', 'its',
'_famille-verte_', 'vases', 'repeating', 'the', 'tones', 'of', 'the',
'pale', 'damask', 'curtains,', 'and', 'its', 'eighteenth-century',
'pastels', 'in', 'delicate', 'faded', 'frames.', '"Has', 'he',
'chucked', 'his', 'pictures', 'too?', 'I', "haven't", 'seen', 'a',
'single', 'one', 'in', 'the', 'house."', 'A', 'slight', 'shade', 'of',
'constraint', 'crossed', 'Mrs.', "Gisburn's", 'open', 'countenance.',
'"It\'s', 'his', 'ridiculous', 'modesty,', 'you', 'know.', 'He',
'says', "they're", 'not', 'fit', 'to', 'have', 'about;', "he's",
'sent', 'them', 'all', 'away', 'except', 'one--my', 'portrait--and',
'that', 'I', 'have', 'to', 'keep', 'upstairs."', 'His', 'ridiculous',
"modesty--Jack's", 'modesty', 'about', 'his', 'pictures?', 'My',
'curiosity', 'was', 'growing', 'like', 'the', 'bean-stalk.', 'I',
'said', 'persuasively', 'to', 'my', 'hostess:', '"I', 'must',
'really', 'see', 'your', 'portrait,', 'you', 'know."', 'She',
'glanced', 'out', 'almost', 'timorously', 'at', 'the', 'terrace',
'where', 'her', 'husband,', 'lounging', 'in', 'a', 'hooded', 'chair,',
'had', 'lit', 'a', 'cigar', 'and', 'drawn', 'the', 'Russian',
"deerhound's", 'head', 'between', 'his', 'knees.', '"Well,', 'come',
'while', "he's", 'not', 'looking,"', 'she', 'said,', 'with', 'a',
'laugh', 'that', 'tried', 'to', 'hide', 'her', 'nervousness;', 'and',
'I', 'followed', 'her', 'between', 'the', 'marble', 'Emperors', 'of',
'the', 'hall,', 'and', 'up', 'the', 'wide', 'stairs', 'with', 'terra-
cotta', 'nymphs', 'poised', 'among', 'flowers', 'at', 'each',
'landing.', 'In', 'the', 'dimmest', 'corner', 'of', 'her', 'boudoir,',
'amid', 'a', 'profusion', 'of', 'delicate', 'and', 'distinguished',
'objects,', 'hung', 'one', 'of', 'the', 'familiar', 'oval',
'canvases,', 'in', 'the', 'inevitable', 'garlanded', 'frame.', 'The',
'mere', 'outline', 'of', 'the', 'frame', 'called', 'up', 'all',
"Gisburn's", 'past!', 'Mrs.', 'Gisburn', 'drew', 'back', 'the',
'window-curtains,', 'moved', 'aside', 'a', '_jardiniere_', 'full',
'of', 'pink', 'azaleas,', 'pushed', 'an', 'arm-chair', 'away,', 'and',
'said:', '"If', 'you', 'stand', 'here', 'you', 'can', 'just',
'manage', 'to', 'see', 'it.', 'I', 'had', 'it', 'over', 'the',
'mantel-piece,', 'but', 'he', "wouldn't", 'let', 'it', 'stay."',
'Yes--I', 'could', 'just', 'manage', 'to', 'see', 'it--the', 'first',
'portrait', 'of', "Jack's", 'I', 'had', 'ever', 'had', 'to', 'strain',
'my', 'eyes', 'over!', 'Usually', 'they', 'had', 'the', 'place', 'of',
'honour--say', 'the', 'central', 'panel', 'in', 'a', 'pale', 'yellow',
'or', '_rose', 'Dubarry_', 'drawing-room,', 'or', 'a', 'monumental',
'easel', 'placed', 'so', 'that', 'it', 'took', 'the', 'light',
'through', 'curtains', 'of', 'old', 'Venetian', 'point.', 'The',
'more', 'modest', 'place', 'became', 'the', 'picture', 'better;',
'yet,', 'as', 'my', 'eyes', 'grew', 'accustomed', 'to', 'the', 'half-
light,', 'all', 'the', 'characteristic', 'qualities', 'came', 'out--
all', 'the', 'hesitations', 'disguised', 'as', 'audacities,', 'the',
'tricks', 'of', 'prestidigitation', 'by', 'which,', 'with', 'such',
```

```
'consummate', 'skill,', 'he', 'managed', 'to', 'divert', 'attention',
'from', 'the', 'real', 'business', 'of', 'the', 'picture', 'to',
'some', 'pretty', 'irrelevance', 'of', 'detail.', 'Mrs.', 'Gisburn,',
'presenting', 'a', 'neutral', 'surface', 'to', 'work', 'on--forming,',
'as', 'it', 'were,', 'so', 'inevitably', 'the', 'background', 'of',
'her', 'own', 'picture--had', 'lent', 'herself', 'in', 'an',
'unusual', 'degree', 'to', 'the', 'display', 'of', 'this', 'false',
'virtuosity.', 'The', 'picture', 'was', 'one', 'of', "Jack's",
'"strongest,"', 'as', 'his', 'admirers', 'would', 'have', 'put', 'it--
it', 'represented,', 'on', 'his', 'part,', 'a', 'swelling', 'of',
'muscles,', 'a', 'congesting', 'of', 'veins,', 'a', 'balancing,',
'straddling', 'and', 'straining,', 'that', 'reminded', 'one', 'of',
'the', "circus-clown's", 'ironic', 'efforts', 'to', 'lift', 'a',
'feather.', 'It', 'met,', 'in', 'short,', 'at', 'every', 'point',
'the', 'demand', 'of', 'lovely', 'woman', 'to', 'be', 'painted',
'"strongly"', 'because', 'she', 'was', 'tired', 'of', 'being',
'painted', '"sweetly', '"--', 'and', 'yet', 'not', 'to', 'lose', 'an',
'atom', 'of', 'the', 'sweetness.', '"It\'s', 'the', 'last', 'he',
'painted,', 'you', 'know,"', 'Mrs.', 'Gisburn', 'said', 'with',
'pardonable', 'pride.', '"The', 'last', 'but', 'one,"', 'she',
'corrected', 'herself--"but', 'the', 'other', "doesn't", 'count,',
'because', 'he', 'destroyed', 'it."', '"Destroyed', 'it?"', 'I',
'was', 'about', 'to', 'follow', 'up', 'this', 'clue', 'when', 'I',
'heard', 'a', 'footstep', 'and', 'saw', 'Jack', 'himself', 'on',
'the', 'threshold.', 'As', 'he', 'stood', 'there,', 'his', 'hands',
'in', 'the', 'pockets', 'of', 'his', 'velveteen', 'coat,', 'the',
'thin', 'brown', 'waves', 'of', 'hair', 'pushed', 'back', 'from',
'his', 'white', 'forehead,', 'his', 'lean', 'sunburnt', 'cheeks',
'furrowed', 'by', 'a', 'smile', 'that', 'lifted', 'the', 'tips', 'of',
'a', 'self-confident', 'moustache,', 'I', 'felt', 'to', 'what', 'a',
'degree', 'he', 'had', 'the', 'same', 'quality', 'as', 'his',
'pictures--the', 'quality', 'of', 'looking', 'cleverer', 'than', 'he',
'was.', 'His', 'wife', 'glanced', 'at', 'him', 'deprecatingly,',
'but', 'his', 'eyes', 'travelled', 'past', 'her', 'to', 'the',
'portrait.', '"Mr.', 'Rickham', 'wanted', 'to', 'see', 'it,"', 'she',
'began,', 'as', 'if', 'excusing', 'herself.', 'He', 'shrugged', 'his',
'shoulders,', 'still', 'smiling.', '"Oh,', 'Rickham', 'found', 'me',
'out', 'long', 'ago,"', 'he', 'said', 'lightly;', 'then,', 'passing',
'his', 'arm', 'through', 'mine:', '"Come', 'and', 'see', 'the',
'rest', 'of', 'the', 'house."', 'He', 'showed', 'it', 'to', 'me',
'with', 'a', 'kind', 'of', 'naive', 'suburban', 'pride:', 'the',
'bath-rooms,', 'the', 'speaking-tubes,', 'the', 'dress-closets,',
'the', 'trouser-presses--all', 'the', 'complex', 'simplifications',
'of', 'the', "millionaire's", 'domestic', 'economy.', 'And',
'whenever', 'my', 'wonder', 'paid', 'the', 'expected', 'tribute',
'he', 'said,', 'throwing', 'out', 'his', 'chest', 'a', 'little:',
'"Yes,', 'I', 'really', "don't", 'see', 'how', 'people', 'manage',
'to', 'live', 'without', 'that."', 'Well--it', 'was', 'just', 'the',
'end', 'one', 'might', 'have', 'foreseen', 'for', 'him.', 'Only',
```

```
'he', 'was,', 'through', 'it', 'all', 'and', 'in', 'spite', 'of',
'it', 'all--as', 'he', 'had', 'been', 'through,', 'and', 'in',
'spite', 'of,', 'his', 'pictures--so', 'handsome,', 'so', 'charming,',
'so', 'disarming,', 'that', 'one', 'longed', 'to', 'cry', 'out:',
'"Be', 'dissatisfied', 'with', 'your', 'leisure!"', 'as', 'once',
'one', 'had', 'longed', 'to', 'say:', '"Be', 'dissatisfied', 'with',
'your', 'work!"', 'But,', 'with', 'the', 'cry', 'on', 'my', 'lips,',
'my', 'diagnosis', 'suffered', 'an', 'unexpected', 'check.', '"This',
'is', 'my', 'own', 'lair,"', 'he', 'said,', 'leading', 'me', 'into',
'a', 'dark', 'plain', 'room', 'at', 'the', 'end', 'of', 'the',
'florid', 'vista.', 'It', 'was', 'square', 'and', 'brown', 'and',
'leathery:', 'no', '"effects";', 'no', 'bric-a-brac,', 'none', 'of',
'the', 'air', 'of', 'posing', 'for', 'reproduction', 'in', 'a',
'picture', 'weekly--above', 'all,', 'no', 'least', 'sign', 'of',
'ever', 'having', 'been', 'used', 'as', 'a', 'studio.', 'The', 'fact',
'brought', 'home', 'to', 'me', 'the', 'absolute', 'finality', 'of',
"Jack's", 'break', 'with', 'his', 'old', 'life.', '"Don\'t', 'you',
'ever', 'dabble', 'with', 'paint', 'any', 'more?"', 'I', 'asked,',
'still', 'looking', 'about', 'for', 'a', 'trace', 'of', 'such',
'activity.', '"Never,"', 'he', 'said', 'briefly.', '"Or', 'water-
colour--or', 'etching?"', 'His', 'confident', 'eyes', 'grew', 'dim,',
'and', 'his', 'cheeks', 'paled', 'a', 'little', 'under', 'their',
'handsome', 'sunburn.', '"Never', 'think', 'of', 'it,', 'my', 'dear',
'fellow--any', 'more', 'than', 'if', "I'd", 'never', 'touched', 'a',
'brush."', 'And', 'his', 'tone', 'told', 'me', 'in', 'a', 'flash',
'that', 'he', 'never', 'thought', 'of', 'anything', 'else.', 'I',
'moved', 'away,', 'instinctively', 'embarrassed', 'by', 'my',
'unexpected', 'discovery;', 'and', 'as', 'I', 'turned,', 'my', 'eye',
'fell', 'on', 'a', 'small', 'picture', 'above', 'the', 'mantel-piece--
the', 'only', 'object', 'breaking', 'the', 'plain', 'oak',
'panelling', 'of', 'the', 'room.', '"Oh,', 'by', 'Jove!"', 'I',
'said.', 'It', 'was', 'a', 'sketch', 'of', 'a', 'donkey--an', 'old',
'tired', 'donkey,', 'standing', 'in', 'the', 'rain', 'under', 'a',
'wall.', '"By', 'Jove--a', 'Stroud!"', 'I', 'cried.', 'He', 'was',
'silent;', 'but', 'I', 'felt', 'him', 'close', 'behind', 'me,',
'breathing', 'a', 'little', 'quickly.', '"What', 'a', 'wonder!',
'Made', 'with', 'a', 'dozen', 'lines--but', 'on', 'everlasting',
'foundations.', 'You', 'lucky', 'chap,', 'where', 'did', 'you', 'get',
'it?"', 'He', 'answered', 'slowly:', '"Mrs.', 'Stroud', 'gave', 'it',
'to', 'me."', '"Ah--I', "didn't", 'know', 'you', 'even', 'knew',
'the', 'Strouds.', 'He', 'was', 'such', 'an', 'inflexible',
'hermit."', '"I', "didn't--till", 'after.', '.', '.', '.', 'She',
'sent', 'for', 'me', 'to', 'paint', 'him', 'when', 'he', 'was',
'dead."', '"When', 'he', 'was', 'dead?', 'You?"', 'I', 'must', 'have',
'let', 'a', 'little', 'too', 'much', 'amazement', 'escape', 'through',
'my', 'surprise,', 'for', 'he', 'answered', 'with', 'a',
'deprecating', 'laugh:', '"Yes--she\'s', 'an', 'awful', 'simpleton,',
'you', 'know,', 'Mrs.', 'Stroud.', 'Her', 'only', 'idea', 'was', 'to',
'have', 'him', 'done', 'by', 'a', 'fashionable', 'painter--ah,',
```

```
'poor', 'Stroud!', 'She', 'thought', 'it', 'the', 'surest', 'way',
'of', 'proclaiming', 'his', 'greatness--of', 'forcing', 'it', 'on',
'a', 'purblind', 'public.', 'And', 'at', 'the', 'moment', 'I', 'was',
'_the_', 'fashionable', 'painter."', '"Ah,', 'poor', 'Stroud--as',
'you', 'say.', 'Was', '_that_', 'his', 'history?"', '"That', 'was',
'his', 'history.', 'She', 'believed', 'in', 'him,', 'gloried', 'in',
'him--or', 'thought', 'she', 'did.', 'But', 'she', "couldn't", 'bear',
'not', 'to', 'have', 'all', 'the', 'drawing-rooms', 'with', 'her.',
'She', "couldn't", 'bear', 'the', 'fact', 'that,', 'on', 'varnishing',
'days,', 'one', 'could', 'always', 'get', 'near', 'enough', 'to',
'see', 'his', 'pictures.', 'Poor', 'woman!', "She's", 'just', 'a',
'fragment', 'groping', 'for', 'other', 'fragments.', 'Stroud', 'is',
'the', 'only', 'whole', 'I', 'ever', 'knew."', '"You', 'ever',
'knew?', 'But', 'you', 'just', 'said--"', 'Gisburn', 'had', 'a',
'curious', 'smile', 'in', 'his', 'eyes.', '"Oh,', 'I', 'knew', 'him,',
'and', 'he', 'knew', 'me--only', 'it', 'happened', 'after', 'he',
'was', 'dead."', 'I', 'dropped', 'my', 'voice', 'instinctively.',
'"When', 'she', 'sent', 'for', 'you?"', '"Yes--quite', 'insensible',
'to', 'the', 'irony.', 'She', 'wanted', 'him', 'vindicated--and',
'by', 'me!"', 'He', 'laughed', 'again,', 'and', 'threw', 'back',
'his', 'head', 'to', 'look', 'up', 'at', 'the', 'sketch', 'of', 'the',
'donkey.', '"There', 'were', 'days', 'when', 'I', "couldn't", 'look',
'at', 'that', "thing--couldn't", 'face', 'it.', 'But', 'I', 'forced',
'myself', 'to', 'put', 'it', 'here;', 'and', 'now', "it's", 'cured',
'me--cured', 'me.', "That's", 'the', 'reason', 'why', 'I', "don't",
'dabble', 'any', 'more,', 'my', 'dear', 'Rickham;', 'or', 'rather',
'Stroud', 'himself', 'is', 'the', 'reason."', 'For', 'the', 'first',
'time', 'my', 'idle', 'curiosity', 'about', 'my', 'companion',
'turned', 'into', 'a', 'serious', 'desire', 'to', 'understand', 'him',
'better.', '"I', 'wish', "you'd", 'tell', 'me', 'how', 'it',
'happened,"', 'I', 'said.', 'He', 'stood', 'looking', 'up', 'at',
'the', 'sketch,', 'and', 'twirling', 'between', 'his', 'fingers', 'a',
'cigarette', 'he', 'had', 'forgotten', 'to', 'light.', 'Suddenly',
'he', 'turned', 'toward', 'me.', '"I\'d', 'rather', 'like', 'to',
'tell', 'you--because', "I've", 'always', 'suspected', 'you', 'of',
'loathing', 'my', 'work."', 'I', 'made', 'a', 'deprecating',
'gesture,', 'which', 'he', 'negatived', 'with', 'a', 'good-humoured',
'shrug.', '"Oh,', 'I', "didn't", 'care', 'a', 'straw', 'when', 'I',
'believed', 'in', 'myself--and', 'now', "it's", 'an', 'added', 'tie',
'between', 'us!"', 'He', 'laughed', 'slightly,', 'without',
'bitterness,', 'and', 'pushed', 'one', 'of', 'the', 'deep', 'arm-
chairs', 'forward.', '"There:', 'make', 'yourself', 'comfortable--
and', 'here', 'are', 'the', 'cigars', 'you', 'like."', 'He', 'placed',
'them', 'at', 'my', 'elbow', 'and', 'continued', 'to', 'wander', 'up',
'and', 'down', 'the', 'room,', 'stopping', 'now', 'and', 'then',
'beneath', 'the', 'picture.', '"How', 'it', 'happened?', 'I', 'can',
'tell', 'you', 'in', 'five', 'minutes--and', 'it', "didn't", 'take',
'much', 'longer', 'to', 'happen.', '.', '.', '.', 'I', 'can',
'remember', 'now', 'how', 'surprised', 'and', 'pleased', 'I', 'was',
```

```
'when', 'I', 'got', 'Mrs.', "Stroud's", 'note.', 'Of', 'course,',
'deep', 'down,', 'I', 'had', 'always', '_felt_', 'there', 'was', 'no',
'one', 'like', 'him--only', 'I', 'had', 'gone', 'with', 'the',
'stream,', 'echoed', 'the', 'usual', 'platitudes', 'about', 'him,',
'till', 'I', 'half', 'got', 'to', 'think', 'he', 'was', 'a',
'failure,', 'one', 'of', 'the', 'kind', 'that', 'are', 'left',
'behind.', 'By', 'Jove,', 'and', 'he', '_was_', 'left', 'behind--
because', 'he', 'had', 'come', 'to', 'stay!', 'The', 'rest', 'of',
'us', 'had', 'to', 'let', 'ourselves', 'be', 'swept', 'along', 'or',
'go', 'under,', 'but', 'he', 'was', 'high', 'above', 'the', 'current--
on', 'everlasting', 'foundations,', 'as', 'you', 'say.', '"Well,',
'I', 'went', 'off', 'to', 'the', 'house', 'in', 'my', 'most',
'egregious', 'mood--rather', 'moved,', 'Lord', 'forgive', 'me,', 'at',
'the', 'pathos', 'of', 'poor', "Stroud's", 'career', 'of', 'failure',
'being', 'crowned', 'by', 'the', 'glory', 'of', 'my', 'painting',
'him!', 'Of', 'course', 'I', 'meant', 'to', 'do', 'the', 'picture',
'for', 'nothing--I', 'told', 'Mrs.', 'Stroud', 'so', 'when', 'she',
'began', 'to', 'stammer', 'something', 'about', 'her', 'poverty.',
'I', 'remember', 'getting', 'off', 'a', 'prodigious', 'phrase',
'about', 'the', 'honour', 'being', '_mine', '_--', 'oh,', 'I', 'was',
'princely,', 'my', 'dear', 'Rickham!', 'I', 'was', 'posing', 'to',
'myself', 'like', 'one', 'of', 'my', 'own', 'sitters.', '"Then', 'I',
'was', 'taken', 'up', 'and', 'left', 'alone', 'with', 'him.', 'I',
'had', 'sent', 'all', 'my', 'traps', 'in', 'advance,', 'and', 'I',
'had', 'only', 'to', 'set', 'up', 'the', 'easel', 'and', 'get', 'to',
'work.', 'He', 'had', 'been', 'dead', 'only', 'twenty-four', 'hours,',
'and', 'he', 'died', 'suddenly,', 'of', 'heart', 'disease,', 'so',
'that', 'there', 'had', 'been', 'no', 'preliminary', 'work', 'of',
'destruction--his', 'face', 'was', 'clear', 'and', 'untouched.', 'I',
'had', 'met', 'him', 'once', 'or', 'twice,', 'years', 'before,',
'and', 'thought', 'him', 'insignificant', 'and', 'dingy.', 'Now', 'I',
'saw', 'that', 'he', 'was', 'superb.', '"I', 'was', 'glad', 'at',
'first,', 'with', 'a', 'merely', 'aesthetic', 'satisfaction:', 'glad',
'to', 'have', 'my', 'hand', 'on', 'such', 'a', "'subject.'", 'Then',
'his', 'strange', 'life-likeness', 'began', 'to', 'affect', 'me',
'queerly--as', 'I', 'blocked', 'the', 'head', 'in', 'I', 'felt', 'as',
'if', 'he', 'were', 'watching', 'me', 'do', 'it.', 'The', 'sensation',
'was', 'followed', 'by', 'the', 'thought:', 'if', 'he', '_were_',
'watching', 'me,', 'what', 'would', 'he', 'say', 'to', 'my', 'way',
'of', 'working?', 'My', 'strokes', 'began', 'to', 'go', 'a', 'little',
'wild--I', 'felt', 'nervous', 'and', 'uncertain.', '"Once,', 'when',
'I', 'looked', 'up,', 'I', 'seemed', 'to', 'see', 'a', 'smile',
'behind', 'his', 'close', 'grayish', 'beard--as', 'if', 'he', 'had',
'the', 'secret,', 'and', 'were', 'amusing', 'himself', 'by',
'holding', 'it', 'back', 'from', 'me.', 'That', 'exasperated', 'me',
'still', 'more.', 'The', 'secret?', 'Why,', 'I', 'had', 'a', 'secret',
'worth', 'twenty', 'of', 'his!', 'I', 'dashed', 'at', 'the', 'canvas',
'furiously,', 'and', 'tried', 'some', 'of', 'my', 'bravura',
'tricks.', 'But', 'they', 'failed', 'me,', 'they', 'crumbled.', 'I',
```

```
'saw', 'that', 'he', "wasn't", 'watching', 'the', 'showy', 'bits--I',
"couldn't", 'distract', 'his', 'attention;', 'he', 'just', 'kept',
'his', 'eyes', 'on', 'the', 'hard', 'passages', 'between.', 'Those',
'were', 'the', 'ones', 'I', 'had', 'always', 'shirked,', 'or',
'covered', 'up', 'with', 'some', 'lying', 'paint.', 'And', 'how',
'he', 'saw', 'through', 'my', 'lies!', '"I', 'looked', 'up', 'again,',
'and', 'caught', 'sight', 'of', 'that', 'sketch', 'of', 'the',
'donkey', 'hanging', 'on', 'the', 'wall', 'near', 'his', 'bed.',
'His', 'wife', 'told', 'me', 'afterward', 'it', 'was', 'the', 'last',
'thing', 'he', 'had', 'done--just', 'a', 'note', 'taken', 'with', 'a',
'shaking', 'hand,', 'when', 'he', 'was', 'down', 'in', 'Devonshire',
'recovering', 'from', 'a', 'previous', 'heart', 'attack.', 'Just',
'a', 'note!', 'But', 'it', 'tells', 'his', 'whole', 'history.',
'There', 'are', 'years', 'of', 'patient', 'scornful', 'persistence',
'in', 'every', 'line.', 'A', 'man', 'who', 'had', 'swum', 'with',
'the', 'current', 'could', 'never', 'have', 'learned', 'that',
'mighty', 'up-stream', 'stroke.', '.', '.', '.', '"I', 'turned',
'back', 'to', 'my', 'work,', 'and', 'went', 'on', 'groping', 'and',
'muddling;', 'then', 'I', 'looked', 'at', 'the', 'donkey', 'again.',
'I', 'saw', 'that,', 'when', 'Stroud', 'laid', 'in', 'the', 'first',
'stroke,', 'he', 'knew', 'just', 'what', 'the', 'end', 'would', 'be.',
'He', 'had', 'possessed', 'his', 'subject,', 'absorbed', 'it,',
'recreated', 'it.', 'When', 'had', 'I', 'done', 'that', 'with', 'any',
'of', 'my', 'things?', 'They', "hadn't", 'been', 'born', 'of', 'me--
I', 'had', 'just', 'adopted', 'them.', '.', '.', '.', '"Hang', 'it,',
'Rickham,', 'with', 'that', 'face', 'watching', 'me', 'I', "couldn't",
'do', 'another', 'stroke.', 'The', 'plain', 'truth', 'was,', 'I',
"didn't", 'know', 'where', 'to', 'put', 'it--_I', 'had', 'never',
'known_.', 'Only,', 'with', 'my', 'sitters', 'and', 'my', 'public,',
'a', 'showy', 'splash', 'of', 'colour', 'covered', 'up', 'the',
'fact--I', 'just', 'threw', 'paint', 'into', 'their', 'faces.', '.',
'.', '.', 'Well,', 'paint', 'was', 'the', 'one', 'medium', 'those',
'dead', 'eyes', 'could', 'see', 'through--see', 'straight', 'to',
'the', 'tottering', 'foundations', 'underneath.', "Don't", 'you',
'know', 'how,', 'in', 'talking', 'a', 'foreign', 'language,', 'even',
'fluently,', 'one', 'says', 'half', 'the', 'time', 'not', 'what',
'one', 'wants', 'to', 'but', 'what', 'one', 'can?', 'Well--that',
'was', 'the', 'way', 'I', 'painted;', 'and', 'as', 'he', 'lay',
'there', 'and', 'watched', 'me,', 'the', 'thing', 'they', 'called',
'my', "'technique'", 'collapsed', 'like', 'a', 'house', 'of',
'cards.', 'He', "didn't", 'sneer,', 'you', 'understand,', 'poor',
'Stroud--he', 'just', 'lay', 'there', 'quietly', 'watching,', 'and',
'on', 'his', 'lips,', 'through', 'the', 'gray', 'beard,', 'I',
'seemed', 'to', 'hear', 'the', 'question:', "'Are", 'you', 'sure',
'you', 'know', 'where', "you're", 'coming', "out?'", '"If', 'I',
'could', 'have', 'painted', 'that', 'face,', 'with', 'that',
'question', 'on', 'it,', 'I', 'should', 'have', 'done', 'a', 'great',
'thing.', 'The', 'next', 'greatest', 'thing', 'was', 'to', 'see',
'that', 'I', "couldn't--and", 'that', 'grace', 'was', 'given', 'me.',
```

```
'But,', 'oh,', 'at', 'that', 'minute,', 'Rickham,', 'was', 'there',
'anything', 'on', 'earth', 'I', "wouldn't", 'have', 'given', 'to',
'have', 'Stroud', 'alive', 'before', 'me,', 'and', 'to', 'hear',
'him', 'say:', "'It's", 'not', 'too', "late--I'll", 'show', 'you',
"how'?", '"It', '_was_', 'too', 'late--it', 'would', 'have', 'been,',
'even', 'if', "he'd", 'been', 'alive.', 'I', 'packed', 'up', 'my',
'traps,', 'and', 'went', 'down', 'and', 'told', 'Mrs.', 'Stroud.',
'Of', 'course', 'I', "didn't", 'tell', 'her', '_that', '_--', 'it',
'would', 'have', 'been', 'Greek', 'to', 'her.', 'I', 'simply', 'said',
'I', "couldn't", 'paint', 'him,', 'that', 'I', 'was', 'too', 'moved.',
'She', 'rather', 'liked', 'the', "idea--she's", 'so', 'romantic!',
'It', 'was', 'that', 'that', 'made', 'her', 'give', 'me', 'the',
'donkey.', 'But', 'she', 'was', 'terribly', 'upset', 'at', 'not',
'getting', 'the', 'portrait--she', 'did', 'so', 'want', 'him',
"'done'", 'by', 'some', 'one', 'showy!', 'At', 'first', 'I', 'was',
'afraid', 'she', "wouldn't", 'let', 'me', 'off--and', 'at', 'my',
"wits'", 'end', 'I', 'suggested', 'Grindle.', 'Yes,', 'it', 'was',
'I', 'who', 'started', 'Grindle:', 'I', 'told', 'Mrs.', 'Stroud',
'he', 'was', 'the', "'coming'", 'man,', 'and', 'she', 'told',
'somebody', 'else,', 'and', 'so', 'it', 'got', 'to', 'be', 'true.',
'.', '.', '.', 'And', 'he', 'painted', 'Stroud', 'without',
'wincing;', 'and', 'she', 'hung', 'the', 'picture', 'among', 'her',
"husband's", 'things.', '.', '.', '."', 'He', 'flung', 'himself',
'down', 'in', 'the', 'arm-chair', 'near', 'mine,', 'laid', 'back',
'his', 'head,', 'and', 'clasping', 'his', 'arms', 'beneath', 'it,',
'looked', 'up', 'at', 'the', 'picture', 'above', 'the', 'chimney-
piece.', '"I', 'like', 'to', 'fancy', 'that', 'Stroud', 'himself',
'would', 'have', 'given', 'it', 'to', 'me,', 'if', "he'd", 'been',
'able', 'to', 'say', 'what', 'he', 'thought', 'that', 'day."', 'And,',
'in', 'answer', 'to', 'a', 'question', 'I', 'put', 'half-
mechanically--"Begin', 'again?"', 'he', 'flashed', 'out.', '"When',
'the', 'one', 'thing', 'that', 'brings', 'me', 'anywhere', 'near',
'him', 'is', 'that', 'I', 'knew', 'enough', 'to', 'leave', 'off?"',
'He', 'stood', 'up', 'and', 'laid', 'his', 'hand', 'on', 'my',
'shoulder', 'with', 'a', 'laugh.', '"Only', 'the', 'irony', 'of',
'it', 'is', 'that', 'I', '_am_', 'still', 'painting--since',
"Grindle's", 'doing', 'it', 'for', 'me!', 'The', 'Strouds', 'stand',
'alone,', 'and', 'happen', 'once--but', "there's", 'no',
'exterminating', 'our', 'kind', 'of', 'art."']
```

```
len(result)
```

```
3646
```

```
#Converting the tokens into token ids
#Looking at all words in dataset
all_words = sorted(set(result))
all_words
```

```
['!--',
 '"--',
 '"Ah,',
 '"Ah--I',
 '"Be',
 '"By',
 '"Come',
 '"Destroyed',
 '"Don\'t',
 '"Gisburns"',
 '"Grindles."',
 '"Hang',
 '"Has',
 '"How',
 '"I',
 '"I\'d',
 '"If',
 '"It',
 '"It\'s',
 '"Jack',
 '"Money\'s',
 '"Moon-dancers"',
 '"Mr.',
 '"Mrs.',
 '"My',
 '"Never',
 '"Never,"',
 '"Of',
 '"Oh,',
 '"Once,',
 '"Only',
 '"Or',
 '"That',
 '"The',
 '"Then',
 '"There',
 '"There:',
 '"This',
 '"We',
 '"Well,',
 '"What',
 '"When',
 '"Why',
 '"Yes,',
 '"Yes--quite',
 '"Yes--she\'s',
 '"You',
 '"deadening',
 '"dragged',
 '"effects";',
```

```
'"interesting":',
'"lift',
'"obituary"',
'"strongest,"',
'"strongly"',
'"sweetly',
"'Are",
"'It's",
"'coming'",
"'done'",
"'subject.'",
"'technique'",
"'way",
'(I',
'(Though',
'.',
'."',
'A',
'Among',
'And',
'And,',
'Arrt',
'As',
'At',
'Burlington',
'But',
'But,',
'By',
'Carlo,',
'Carlo;',
'Chicago',
'Claude',
'Croft',
'Croft)',
'Croft,',
'Devonshire',
"Don't",
'Dubarry_',
'Emperors',
'Florence.)',
'For',
'Gallery',
'Gideon',
'Gisburn',
'Gisburn!',
"Gisburn's",
'Gisburn,',
'Gisburn--as',
'Gisburn--fond',
'Grafton',
```

```
 'Greek',
 'Grindle',
 "Grindle's",
 'Grindle,"',
 'Grindle.',
 'Grindle:',
 'HAD',
 'Had',
 'He',
 'Her',
 'Hermia',
 "Hermia's",
 'His',
 'I',
 "I'd",
 "I've",
 'If',
 'In',
 'It',
 'Jack',
 'Jack!',
 "Jack's",
 'Jack,',
 'Jack--one',
 'Jove!"',
 'Jove,',
 'Jove--a',
 'Just',
 'Lord',
 'Made',
 'Miss',
 'Monte',
 'Mr.',
 'Mrs.',
 'My',
 'No--for,',
 'Now',
 'Nutley,',
 'Of',
 'On',
 'Only',
 'Only,',
 'Perhaps.',
 'Poor',
 'Professional',
 'Renaissance',
 'Rickham',
 'Rickham!',
 'Rickham,',
 'Rickham--the',
```

```
 'Rickham;',
 'Riviera',
 'Riviera,',
 'Riviera.',
 'Rome',
 'Russian',
 'Sevres',
 'She',
 "She's",
 'Stroud',
 'Stroud!',
 'Stroud!"',
 "Stroud's",
 'Stroud--as',
 'Stroud--he',
 'Stroud.',
 'Strouds',
 'Strouds.',
 'Suddenly',
 'That',
 "That's",
 'The',
 'Then',
 'There',
 'They',
 'Those',
 'Thwing',
 "Thwing's",
 'Thwing--his',
 'Thwings',
 'To',
 'Usually',
 'Venetian',
 'Victor',
 'Was',
 'Well',
 'Well,',
 'Well--it',
 'Well--that',
 'What',
 'When',
 'Why,',
 'Yes,',
 'Yes--I',
 'You',
 'You?"',
 '_--',
 '_am_',
 '_famille-verte_',
 '_felt_',
```

```
'_has_',
'_have_',
'_jardiniere_',
'_mine',
'_not_',
'_rose',
'_rs_',
'_that',
'_that_',
'_the_',
'_was_',
'_were_',
'a',
'abdication.',
'able',
'about',
'about;',
'above',
'abruptly.',
'absolute',
'absorbed',
'absurdity',
'absurdity.',
'academic',
'accuse',
'accustomed',
'across',
'activity.',
'add,',
'added',
'admirers',
'adopted',
'adulation.',
'advance,',
'aesthetic',
'affect',
'afraid',
'after',
'after.',
'afterward',
'again',
'again"?',
'again,',
'again--what',
'again.',
'again?"',
'ago,"',
'air',
'alive',
'alive.',
```

```
 'all',
 'all,',
 'all--as',
 'almost',
 'alone',
 'alone,',
 'along',
 'always',
 'amazement',
 'amid',
 'among',
 'amplest',
 'amusing',
 'an',
 'and',
 'and,',
 'another',
 'answer',
 'answered',
 'any',
 'anything',
 'anywhere',
 'apparent',
 'apparently',
 'appearance',
 'appeared',
 'appointed',
 'are',
 'arm',
 'arm-chair',
 'arm-chairs',
 'arms',
 'art',
 'art,',
 'art."',
 'articles',
 'artist',
 'as',
 'aside',
 'asked',
 'asked,',
 'at',
 'atmosphere',
 'atom',
 'attack.',
 'attention',
 'attention;',
 'attitude',
 'audacities,',
 'away',
```

```
'away,',
'awful',
'axioms',
'azaleas,',
'back',
'background',
'balance',
'balancing,',
'balustraded',
'basking',
'bath-rooms,',
'be',
'be.',
'beaming',
'bean-stalk.',
'bear',
'beard,',
'beard--as',
'beauty',
'beauty."',
'became',
'because',
'becoming',
'bed.',
'been',
'been,',
'before',
'before,',
'began',
'began,',
'begun',
'behind',
'behind--because',
'behind.',
'being',
'believed',
'beneath',
'bespoke',
'better.',
'better;',
'between',
'between.',
'big',
'bits--I',
'bitterness,',
'blocked',
'born',
'borne',
'boudoir,',
'bravura',
```

```
 'break',
 'breaking',
 'breathing',
 'bric-a-brac,',
 'briefly.',
 'brings',
 'bronzes',
 'brought',
 'brown',
 'brush',
 'brush."',
 'bull--that',
 'business',
 'but',
 'buying',
 'by',
 'called',
 'came',
 'can',
 'can?',
 'canvas',
 'canvases,',
 'cards.',
 'care',
 'career',
 'caught',
 'central',
 'chair,',
 'chap,',
 'characteristic',
 'charming,',
 'cheap',
 'check.',
 'cheeks',
 'chest',
 'chimney-piece.',
 'chucked',
 'cigar',
 'cigarette',
 'cigars',
 'circulation,"',
 'circumstance',
 "circus-clown's",
 'claimed',
 'clasping',
 'clear',
 'cleverer',
 'close',
 'clue',
 'coat,',
```

```
 'collapsed',
 'colour',
 'come',
 'comfortable--and',
 'coming',
 'companion',
 'compared',
 'complex',
 'confident',
 'congesting',
 'conjugal',
 'constraint',
 'consummate',
 'contended,',
 'continued',
 'corner',
 'corrected',
 'could',
 "couldn't",
 "couldn't--and",
 'count,',
 'countenance.',
 'couple',
 'course',
 'course,',
 'covered',
 'craft',
 'cried.',
 'crossed',
 'crowned',
 'crumbled.',
 'cry',
 'cured',
 'curiosity',
 'curious',
 'current',
 'current--on',
 'curtains',
 'curtains,',
 'dabble',
 'damask',
 'dark',
 'dashed',
 'day,',
 'day.',
 'day."',
 'days',
 'days,',
 'dead',
 'dead."',
```

```
 'dead?',
 'dear',
 'dear,',
 'deep',
 "deerhound's",
 'degree',
 'delicate',
 'demand',
 'denied',
 'deprecating',
 'deprecatingly,',
 'desire',
 'destroyed',
 'destruction--his',
 'desultory',
 'detail.',
 'diagnosis',
 'did',
 'did.',
 "didn't",
 "didn't--till",
 'died',
 'dim,',
 'dimmest',
 'dingy.',
 'dining-room.',
 'disarming,',
 'discovery;',
 'discrimination',
 'discussion',
 'disdain',
 'disdained',
 'disease,',
 'disguised',
 'display',
 'dissatisfied',
 'distinguished',
 'distract',
 'divert',
 'do',
 "doesn't",
 'doing',
 'domestic',
 "don't",
 'done',
 'done--just',
 'donkey',
 'donkey,',
 'donkey--an',
 'donkey.',
```

```
'down',
'down,',
'down."',
'dozen',
'dragged',
'drawing-room,',
'drawing-rooms',
'drawn',
'dress-closets,',
'drew',
'dropped',
'each',
'earth',
'ease--because',
'easel',
'easel.',
'easy--his',
'echoed',
'economy.',
'effect',
'efforts',
'egregious',
'eighteenth-century',
'elbow',
'elegant',
'else,',
'else.',
'embarrassed',
'enabled',
'end',
'endless',
'enjoy',
'enlightenment:',
'enough',
'enough,',
'enough--so',
'ensuing',
'equally,',
'equanimity.',
'escape',
'established',
'etching?"',
'even',
'event',
'ever',
'everlasting',
'every',
'exasperated',
'except',
'excuse',
```

```
'excusing',
'existed',
'expected',
'exquisite',
'exquisitely',
'extenuation.',
'exterminating',
'extracting',
'eye',
'eyebrows',
'eyes',
'eyes.',
'eyes:',
'face',
'face,',
'faces.',
'fact',
'fact,',
'fact--I',
'faded',
'failed',
'failure',
'failure,',
'fair',
'faith,',
'false',
'familiar',
'fancy',
'fashionable',
'fate',
'feather.',
'feet,',
'fell',
'fellow',
'fellow--any',
'felt',
'few',
'fewer',
'finality',
'find',
'fingers',
'first',
'first,',
'fit',
'fitting',
'five',
'flash',
'flashed',
'florid',
```

```
 'flowers',
 'fluently,',
 'flung',
 'follow',
 'followed',
 'fond',
 'footstep',
 'for',
 'forced',
 'forcing',
 'forehead,',
 'foreign',
 'foreseen',
 'forgive',
 'forgotten',
 'form',
 'formed',
 'forward.',
 'fostered',
 'found',
 'foundations',
 'foundations,',
 'foundations.',
 'fragment',
 'fragments.',
 'frame',
 'frame.',
 'frames.',
 'frequently.',
 "friend's",
 'from',
 'full',
 'fullest',
 'furiously,',
 'furrowed',
 'garlanded',
 'garlands',
 'gave',
 'genial',
 'genius--though',
 'gesture,',
 'get',
 'getting',
 'give',
 'given',
 'glad',
 'glanced',
 'glimpse',
 'gloried',
```

```
 'glory',
 'glory,',
 'go',
 'going',
 'gone',
 'good',
 'good-breeding,',
 'good-humoured',
 'got',
 'grace',
 'gradually',
 'gray',
 'grayish',
 'great',
 'greatest',
 'greatness--of',
 'grew',
 'groping',
 'growing',
 'had',
 "hadn't",
 'hair',
 'half',
 'half-light,',
 'half-mechanically--"Begin',
 'hall,',
 'hand',
 'hand,',
 'hands',
 'handsome',
 'handsome,',
 'hanging',
 'happen',
 'happen.',
 'happened',
 'happened,"',
 'happened?',
 'hard',
 'hardly',
 'have',
 "haven't",
 'having',
 'having,',
 'he',
 "he'd",
 "he's",
 'head',
 'head,',
 'hear',
```

```
 'heard',
 'heard,',
 'heart',
 'height',
 'her',
 'her--since',
 'her.',
 'here',
 'here;',
 'hermit."',
 'herself',
 'herself--"but',
 'herself.',
 'hesitations',
 'hide',
 'high',
 'him',
 'him!',
 'him,',
 'him--it',
 'him--only',
 'him--or',
 'him.',
 'him:',
 'himself',
 'himself,',
 'himself,"',
 'hint',
 'his',
 'his!',
 'history.',
 'history?"',
 'holding',
 'home',
 'honour',
 'honour--say',
 'hooded',
 'hostess',
 'hostess:',
 'hot-house',
 'hour.',
 'hours,',
 'house',
 'house."',
 'how',
 "how'?",
 'how,',
 'hung',
 'husband',
```

```
 "husband's",
 'husband,',
 'idea',
 "idea--she's",
 'idle',
 'idling',
 'if',
 'immediately',
 'in',
 'incense.',
 'indifferent;',
 'inevitable',
 'inevitably',
 'inflexible',
 'insensible',
 'insignificant',
 'instinctively',
 'instinctively.',
 'instructive',
 'interesting',
 'interesting--if',
 'into',
 'ironic',
 'irony',
 'irony.',
 'irrelevance',
 'irrevocable--the',
 'is',
 'is,',
 'it',
 "it's",
 'it,',
 'it,"',
 'it--_I',
 'it--and',
 'it--it',
 'it--the',
 'it.',
 'it."',
 'it;',
 'it?"',
 'its',
 'itself',
 'jealousy',
 'jealousy?',
 'just',
 'keep',
 'kept',
 'kind',
```

```
 'knees.',
 'knew',
 'knew."',
 'knew?',
 'know',
 'know,',
 'know,"',
 'know.',
 'know."',
 'know;',
 'known_.',
 'laid',
 'lair,"',
 'landing.',
 'language,',
 'last',
 "late--I'll",
 'late--it',
 'later',
 "latter's",
 'latter,',
 'laugh',
 'laugh.',
 'laugh:',
 'laughed',
 'lay',
 'leading',
 'lean',
 'learned',
 'least',
 'leathery:',
 'leave',
 'led',
 'left',
 'leisure!"',
 'lends',
 'lent',
 'let',
 'lies!',
 'life',
 'life,',
 'life-likeness',
 'life.',
 'lift',
 'lifted',
 'light',
 'light.',
 'lightly;',
 'like',
```

```
 'like."',
 'liked',
 'line.',
 'lines--but',
 'lingered',
 'lips,',
 'lit',
 'little',
 'little:',
 'live',
 'loathing',
 'long',
 'longed',
 'longer',
 'look',
 'looked',
 'looking',
 'looking,"',
 'lose',
 'loss',
 'lounging',
 'lovely',
 'lucky',
 'lump',
 'luncheon-table,',
 'luxury.',
 'lying',
 'made',
 'make',
 'man',
 'man,',
 'manage',
 'managed',
 'mantel-piece,',
 'mantel-piece--the',
 'marble',
 'married',
 'may',
 'me',
 'me!',
 'me!"',
 'me,',
 'me--I',
 'me--cured',
 'me--only',
 'me--they',
 'me.',
 'me."',
 'meant',
```

```
 'mediocrity"',
 'medium',
 'mentioned',
 'mere',
 'merely',
 'met',
 'met,',
 'might',
 'mighty',
 "millionaire's",
 'mine,',
 'mine:',
 'minute,',
 'minutes--and',
 'mirrors.',
 'modest',
 'modesty',
 'modesty,',
 "modesty--Jack's",
 'moment',
 'moment--as',
 'money',
 'monumental',
 'mood--rather',
 'morbidly',
 'more',
 'more,',
 'more.',
 'more?"',
 'most',
 'mourn',
 'mourned.',
 'moustache,',
 'moved',
 'moved,',
 'moved.',
 'much',
 'muddling;',
 'multiplied',
 'murmur.',
 'muscles,',
 'must',
 'my',
 'myself',
 'myself--and',
 'mysterious',
 'naive',
 'near',
 'nearly',
```

```
'negatived',
'nervous',
'nervousness;',
'neutral',
'never',
'next',
'no',
'no--for',
'none',
'not',
'note',
'note!',
'note.',
'nothing--I',
'now',
'now,',
'nymphs',
'oak',
'object',
'objects',
'objects,',
'occurred',
'oddly',
'of',
'of,',
'of."',
'off',
'off--and',
'off?"',
'often,',
'oh,',
'old',
'on',
'on--forming,',
'once',
'once--but',
'one',
'one,"',
'one--my',
'ones',
'only',
'onto',
'open',
'or',
'other',
'our',
'ourselves',
'out',
'out,',
```

```
 'out--all',
 'out.',
 'out:',
 "out?'",
 'outline',
 'oval',
 'over',
 'over!',
 'own',
 ...]
```

```
len(all_words)
```

```
1486
```

```
#Building a vocabulary
vocab = {token:integer for integer,token in enumerate(all_words)}
print(vocab)
```

```
{'!--': 0, '"--': 1, '"Ah,': 2, '"Ah--I': 3, '"Be': 4, '"By': 5,
'"Come': 6, '"Destroyed': 7, '"Don\'t': 8, '"Gisburns"': 9,
'"Grindles."': 10, '"Hang': 11, '"Has': 12, '"How': 13, '"I': 14,
'"I\'d': 15, '"If': 16, '"It': 17, '"It\'s': 18, '"Jack': 19,
'"Money\'s': 20, '"Moon-dancers"': 21, '"Mr.': 22, '"Mrs.': 23, '"My':
24, '"Never': 25, '"Never,"': 26, '"Of': 27, '"Oh,': 28, '"Once,': 29,
'"Only': 30, '"Or': 31, '"That': 32, '"The': 33, '"Then': 34,
'"There': 35, '"There:': 36, '"This': 37, '"We': 38, '"Well,': 39,
'"What': 40, '"When': 41, '"Why': 42, '"Yes,': 43, '"Yes--quite': 44,
'"Yes--she\'s': 45, '"You': 46, '"deadening': 47, '"dragged': 48,
'"effects";': 49, '"interesting":': 50, '"lift': 51, '"obituary"': 52,
'"strongest,"': 53, '"strongly"': 54, '"sweetly': 55, "'Are": 56,
"'It's": 57, "'coming'": 58, "'done'": 59, "'subject.'": 60,
"'technique'": 61, "'way": 62, '(I': 63, '(Though': 64, '.': 65, '."':
66, 'A': 67, 'Among': 68, 'And': 69, 'And,': 70, 'Arrt': 71, 'As': 72,
'At': 73, 'Burlington': 74, 'But': 75, 'But,': 76, 'By': 77, 'Carlo,':
78, 'Carlo;': 79, 'Chicago': 80, 'Claude': 81, 'Croft': 82, 'Croft)':
83, 'Croft,': 84, 'Devonshire': 85, "Don't": 86, 'Dubarry_': 87,
'Emperors': 88, 'Florence.)': 89, 'For': 90, 'Gallery': 91, 'Gideon':
92, 'Gisburn': 93, 'Gisburn!': 94, "Gisburn's": 95, 'Gisburn,': 96,
'Gisburn--as': 97, 'Gisburn--fond': 98, 'Grafton': 99, 'Greek': 100,
'Grindle': 101, "Grindle's": 102, 'Grindle,"': 103, 'Grindle.': 104,
'Grindle:': 105, 'HAD': 106, 'Had': 107, 'He': 108, 'Her': 109,
'Hermia': 110, "Hermia's": 111, 'His': 112, 'I': 113, "I'd": 114,
"I've": 115, 'If': 116, 'In': 117, 'It': 118, 'Jack': 119, 'Jack!':
120, "Jack's": 121, 'Jack,': 122, 'Jack--one': 123, 'Jove!"': 124,
'Jove,': 125, 'Jove--a': 126, 'Just': 127, 'Lord': 128, 'Made': 129,
'Miss': 130, 'Monte': 131, 'Mr.': 132, 'Mrs.': 133, 'My': 134, 'No--
for,': 135, 'Now': 136, 'Nutley,': 137, 'Of': 138, 'On': 139, 'Only':
140, 'Only,': 141, 'Perhaps.': 142, 'Poor': 143, 'Professional': 144,
'Renaissance': 145, 'Rickham': 146, 'Rickham!': 147, 'Rickham,': 148,
```

'Rickham--the': 149, 'Rickham;': 150, 'Riviera': 151, 'Riviera,': 152, 'Riviera.': 153, 'Rome': 154, 'Russian': 155, 'Sevres': 156, 'She': 157, "She's": 158, 'Stroud': 159, 'Stroud!': 160, 'Stroud!"': 161, "Stroud's": 162, 'Stroud--as': 163, 'Stroud--he': 164, 'Stroud.': 165, 'Strouds': 166, 'Strouds.': 167, 'Suddenly': 168, 'That': 169, "That's": 170, 'The': 171, 'Then': 172, 'There': 173, 'They': 174, 'Those': 175, 'Thwing': 176, "Thwing's": 177, 'Thwing--his': 178, 'Thwings': 179, 'To': 180, 'Usually': 181, 'Venetian': 182, 'Victor': 183, 'Was': 184, 'Well': 185, 'Well,': 186, 'Well--it': 187, 'Well--that': 188, 'What': 189, 'When': 190, 'Why,': 191, 'Yes,': 192, 'Yes--I': 193, 'You': 194, 'You?"': 195, '_--': 196, '_am_': 197, '_famille-verte_': 198, '_felt_': 199, '_has_': 200, '_have_': 201, '_jardiniere_': 202, '_mine': 203, '_not_': 204, '_rose': 205, '_rs_': 206, '_that': 207, '_that_': 208, '_the_': 209, '_was_': 210, '_were_': 211, 'a': 212, 'abdication.': 213, 'able': 214, 'about': 215, 'about;': 216, 'above': 217, 'abruptly.': 218, 'absolute': 219, 'absorbed': 220, 'absurdity': 221, 'absurdity.': 222, 'academic': 223, 'accuse': 224, 'accustomed': 225, 'across': 226, 'activity.': 227, 'add,': 228, 'added': 229, 'admirers': 230, 'adopted': 231, 'adulation.': 232, 'advance,': 233, 'aesthetic': 234, 'affect': 235, 'afraid': 236, 'after': 237, 'after.': 238, 'afterward': 239, 'again': 240, 'again"?': 241, 'again,': 242, 'again--what': 243, 'again.': 244, 'again?"': 245, 'ago,"': 246, 'air': 247, 'alive': 248, 'alive.': 249, 'all': 250, 'all,': 251, 'all--as': 252, 'almost': 253, 'alone': 254, 'alone,': 255, 'along': 256, 'always': 257, 'amazement': 258, 'amid': 259, 'among': 260, 'amplest': 261, 'amusing': 262, 'an': 263, 'and': 264, 'and,': 265, 'another': 266, 'answer': 267, 'answered': 268, 'any': 269, 'anything': 270, 'anywhere': 271, 'apparent': 272, 'apparently': 273, 'appearance': 274, 'appeared': 275, 'appointed': 276, 'are': 277, 'arm': 278, 'arm-chair': 279, 'arm-chairs': 280, 'arms': 281, 'art': 282, 'art,': 283, 'art."': 284, 'articles': 285, 'artist': 286, 'as': 287, 'aside': 288, 'asked': 289, 'asked,': 290, 'at': 291, 'atmosphere': 292, 'atom': 293, 'attack.': 294, 'attention': 295, 'attention;': 296, 'attitude': 297, 'audacities,': 298, 'away': 299, 'away,': 300, 'awful': 301, 'axioms': 302, 'azaleas,': 303, 'back': 304, 'background': 305, 'balance': 306, 'balancing,': 307, 'balustraded': 308, 'basking': 309, 'bath-rooms,': 310, 'be': 311, 'be.': 312, 'beaming': 313, 'bean-stalk.': 314, 'bear': 315, 'beard,': 316, 'beard--as': 317, 'beauty': 318, 'beauty."': 319, 'became': 320, 'because': 321, 'becoming': 322, 'bed.': 323, 'been': 324, 'been,': 325, 'before': 326, 'before,': 327, 'began': 328, 'began,': 329, 'begun': 330, 'behind': 331, 'behind--because': 332, 'behind.': 333, 'being': 334, 'believed': 335, 'beneath': 336, 'bespoke': 337, 'better.': 338, 'better;': 339, 'between': 340, 'between.': 341, 'big': 342, 'bits--I': 343, 'bitterness,': 344, 'blocked': 345, 'born': 346, 'borne': 347, 'boudoir,': 348, 'bravura': 349, 'break': 350, 'breaking': 351, 'breathing': 352, 'bric-a-brac,': 353, 'briefly.': 354, 'brings': 355, 'bronzes': 356, 'brought': 357, 'brown': 358, 'brush': 359, 'brush."':

360, 'bull--that': 361, 'business': 362, 'but': 363, 'buying': 364, 'by': 365, 'called': 366, 'came': 367, 'can': 368, 'can?': 369, 'canvas': 370, 'canvases,': 371, 'cards.': 372, 'care': 373, 'career': 374, 'caught': 375, 'central': 376, 'chair,': 377, 'chap,': 378, 'characteristic': 379, 'charming,': 380, 'cheap': 381, 'check.': 382, 'cheeks': 383, 'chest': 384, 'chimney-piece.': 385, 'chucked': 386, 'cigar': 387, 'cigarette': 388, 'cigars': 389, 'circulation,"': 390, 'circumstance': 391, "circus-clown's": 392, 'claimed': 393, 'clasping': 394, 'clear': 395, 'cleverer': 396, 'close': 397, 'clue': 398, 'coat,': 399, 'collapsed': 400, 'colour': 401, 'come': 402, 'comfortable--and': 403, 'coming': 404, 'companion': 405, 'compared': 406, 'complex': 407, 'confident': 408, 'congesting': 409, 'conjugal': 410, 'constraint': 411, 'consummate': 412, 'contended,': 413, 'continued': 414, 'corner': 415, 'corrected': 416, 'could': 417, "couldn't": 418, "couldn't--and": 419, 'count,': 420, 'countenance.': 421, 'couple': 422, 'course': 423, 'course,': 424, 'covered': 425, 'craft': 426, 'cried.': 427, 'crossed': 428, 'crowned': 429, 'crumbled.': 430, 'cry': 431, 'cured': 432, 'curiosity': 433, 'curious': 434, 'current': 435, 'current--on': 436, 'curtains': 437, 'curtains,': 438, 'dabble': 439, 'damask': 440, 'dark': 441, 'dashed': 442, 'day,': 443, 'day.': 444, 'day."': 445, 'days': 446, 'days,': 447, 'dead': 448, 'dead."': 449, 'dead?': 450, 'dear': 451, 'dear,': 452, 'deep': 453, "deerhound's": 454, 'degree': 455, 'delicate': 456, 'demand': 457, 'denied': 458, 'deprecating': 459, 'deprecatingly,': 460, 'desire': 461, 'destroyed': 462, 'destruction--his': 463, 'desultory': 464, 'detail.': 465, 'diagnosis': 466, 'did': 467, 'did.': 468, "didn't": 469, "didn't--till": 470, 'died': 471, 'dim,': 472, 'dimmest': 473, 'dingy.': 474, 'dining-room.': 475, 'disarming,': 476, 'discovery;': 477, 'discrimination': 478, 'discussion': 479, 'disdain': 480, 'disdained': 481, 'disease,': 482, 'disguised': 483, 'display': 484, 'dissatisfied': 485, 'distinguished': 486, 'distract': 487, 'divert': 488, 'do': 489, "doesn't": 490, 'doing': 491, 'domestic': 492, "don't": 493, 'done': 494, 'done--just': 495, 'donkey': 496, 'donkey,': 497, 'donkey--an': 498, 'donkey.': 499, 'down': 500, 'down,': 501, 'down."': 502, 'dozen': 503, 'dragged': 504, 'drawing-room,': 505, 'drawing-rooms': 506, 'drawn': 507, 'dress-closets,': 508, 'drew': 509, 'dropped': 510, 'each': 511, 'earth': 512, 'ease--because': 513, 'easel': 514, 'easel.': 515, 'easy--his': 516, 'echoed': 517, 'economy.': 518, 'effect': 519, 'efforts': 520, 'egregious': 521, 'eighteenth-century': 522, 'elbow': 523, 'elegant': 524, 'else,': 525, 'else.': 526, 'embarrassed': 527, 'enabled': 528, 'end': 529, 'endless': 530, 'enjoy': 531, 'enlightenment:': 532, 'enough': 533, 'enough,': 534, 'enough--so': 535, 'ensuing': 536, 'equally,': 537, 'equanimity.': 538, 'escape': 539, 'established': 540, 'etching?"': 541, 'even': 542, 'event': 543, 'ever': 544, 'everlasting': 545, 'every': 546, 'exasperated': 547, 'except': 548, 'excuse': 549, 'excusing': 550, 'existed': 551, 'expected': 552, 'exquisite': 553, 'exquisitely': 554, 'extenuation.': 555, 'exterminating': 556, 'extracting': 557, 'eye': 558, 'eyebrows': 559,

'eyes': 560, 'eyes.': 561, 'eyes:': 562, 'face': 563, 'face,': 564,
'faces.': 565, 'fact': 566, 'fact,': 567, 'fact--I': 568, 'faded':
569, 'failed': 570, 'failure': 571, 'failure,': 572, 'fair': 573,
'faith,': 574, 'false': 575, 'familiar': 576, 'fancy': 577,
'fashionable': 578, 'fate': 579, 'feather.': 580, 'feet,': 581,
'fell': 582, 'fellow': 583, 'fellow--any': 584, 'felt': 585, 'few':
586, 'fewer': 587, 'finality': 588, 'find': 589, 'fingers': 590,
'first': 591, 'first,': 592, 'fit': 593, 'fitting': 594, 'five': 595,
'flash': 596, 'flashed': 597, 'florid': 598, 'flowers': 599,
'fluently,': 600, 'flung': 601, 'follow': 602, 'followed': 603,
'fond': 604, 'footstep': 605, 'for': 606, 'forced': 607, 'forcing':
608, 'forehead,': 609, 'foreign': 610, 'foreseen': 611, 'forgive':
612, 'forgotten': 613, 'form': 614, 'formed': 615, 'forward.': 616,
'fostered': 617, 'found': 618, 'foundations': 619, 'foundations,':
620, 'foundations.': 621, 'fragment': 622, 'fragments.': 623, 'frame':
624, 'frame.': 625, 'frames.': 626, 'frequently.': 627, "friend's":
628, 'from': 629, 'full': 630, 'fullest': 631, 'furiously,': 632,
'furrowed': 633, 'garlanded': 634, 'garlands': 635, 'gave': 636,
'genial': 637, 'genius--though': 638, 'gesture,': 639, 'get': 640,
'getting': 641, 'give': 642, 'given': 643, 'glad': 644, 'glanced':
645, 'glimpse': 646, 'gloried': 647, 'glory': 648, 'glory,': 649,
'go': 650, 'going': 651, 'gone': 652, 'good': 653, 'good-breeding,':
654, 'good-humoured': 655, 'got': 656, 'grace': 657, 'gradually': 658,
'gray': 659, 'grayish': 660, 'great': 661, 'greatest': 662,
'greatness--of': 663, 'grew': 664, 'groping': 665, 'growing': 666,
'had': 667, "hadn't": 668, 'hair': 669, 'half': 670, 'half-light,':
671, 'half-mechanically--"Begin': 672, 'hall,': 673, 'hand': 674,
'hand,': 675, 'hands': 676, 'handsome': 677, 'handsome,': 678,
'hanging': 679, 'happen': 680, 'happen.': 681, 'happened': 682,
'happened,"': 683, 'happened?': 684, 'hard': 685, 'hardly': 686,
'have': 687, "haven't": 688, 'having': 689, 'having,': 690, 'he': 691,
"he'd": 692, "he's": 693, 'head': 694, 'head,': 695, 'hear': 696,
'heard': 697, 'heard,': 698, 'heart': 699, 'height': 700, 'her': 701,
'her--since': 702, 'her.': 703, 'here': 704, 'here;': 705, 'hermit."':
706, 'herself': 707, 'herself--"but': 708, 'herself.': 709,
'hesitations': 710, 'hide': 711, 'high': 712, 'him': 713, 'him!': 714,
'him,': 715, 'him--it': 716, 'him--only': 717, 'him--or': 718, 'him.':
719, 'him:': 720, 'himself': 721, 'himself,': 722, 'himself,"': 723,
'hint': 724, 'his': 725, 'his!': 726, 'history.': 727, 'history?"':
728, 'holding': 729, 'home': 730, 'honour': 731, 'honour--say': 732,
'hooded': 733, 'hostess': 734, 'hostess:': 735, 'hot-house': 736,
'hour.': 737, 'hours,': 738, 'house': 739, 'house."': 740, 'how': 741,
"how'?": 742, 'how,': 743, 'hung': 744, 'husband': 745, "husband's":
746, 'husband,': 747, 'idea': 748, "idea--she's": 749, 'idle': 750,
'idling': 751, 'if': 752, 'immediately': 753, 'in': 754, 'incense.':
755, 'indifferent;': 756, 'inevitable': 757, 'inevitably': 758,
'inflexible': 759, 'insensible': 760, 'insignificant': 761,
'instinctively': 762, 'instinctively.': 763, 'instructive': 764,
'interesting': 765, 'interesting--if': 766, 'into': 767, 'ironic':

768, 'irony': 769, 'irony.': 770, 'irrelevance': 771, 'irrevocable--the': 772, 'is': 773, 'is,': 774, 'it': 775, "it's": 776, 'it,': 777, 'it,"': 778, 'it--_I': 779, 'it--and': 780, 'it--it': 781, 'it--the': 782, 'it.': 783, 'it."': 784, 'it;': 785, 'it?"': 786, 'its': 787, 'itself': 788, 'jealousy': 789, 'jealousy?': 790, 'just': 791, 'keep': 792, 'kept': 793, 'kind': 794, 'knees.': 795, 'knew': 796, 'knew."': 797, 'knew?': 798, 'know': 799, 'know,': 800, 'know,"': 801, 'know.': 802, 'know."': 803, 'know;': 804, 'known_.': 805, 'laid': 806, 'lair,"': 807, 'landing.': 808, 'language,': 809, 'last': 810, "late--I'll": 811, 'late--it': 812, 'later': 813, "latter's": 814, 'latter,': 815, 'laugh': 816, 'laugh.': 817, 'laugh:': 818, 'laughed': 819, 'lay': 820, 'leading': 821, 'lean': 822, 'learned': 823, 'least': 824, 'leathery:': 825, 'leave': 826, 'led': 827, 'left': 828, 'leisure!"': 829, 'lends': 830, 'lent': 831, 'let': 832, 'lies!': 833, 'life': 834, 'life,': 835, 'life-likeness': 836, 'life.': 837, 'lift': 838, 'lifted': 839, 'light': 840, 'light.': 841, 'lightly;': 842, 'like': 843, 'like."': 844, 'liked': 845, 'line.': 846, 'lines--but': 847, 'lingered': 848, 'lips,': 849, 'lit': 850, 'little': 851, 'little:': 852, 'live': 853, 'loathing': 854, 'long': 855, 'longed': 856, 'longer': 857, 'look': 858, 'looked': 859, 'looking': 860, 'looking,"': 861, 'lose': 862, 'loss': 863, 'lounging': 864, 'lovely': 865, 'lucky': 866, 'lump': 867, 'luncheon-table,': 868, 'luxury.': 869, 'lying': 870, 'made': 871, 'make': 872, 'man': 873, 'man,': 874, 'manage': 875, 'managed': 876, 'mantel-piece,': 877, 'mantel-piece--the': 878, 'marble': 879, 'married': 880, 'may': 881, 'me': 882, 'me!': 883, 'me!"': 884, 'me,': 885, 'me--I': 886, 'me--cured': 887, 'me--only': 888, 'me--they': 889, 'me.': 890, 'me."': 891, 'meant': 892, 'mediocrity"': 893, 'medium': 894, 'mentioned': 895, 'mere': 896, 'merely': 897, 'met': 898, 'met,': 899, 'might': 900, 'mighty': 901, "millionaire's": 902, 'mine,': 903, 'mine:': 904, 'minute,': 905, 'minutes--and': 906, 'mirrors.': 907, 'modest': 908, 'modesty': 909, 'modesty,': 910, "modesty--Jack's": 911, 'moment': 912, 'moment--as': 913, 'money': 914, 'monumental': 915, 'mood--rather': 916, 'morbidly': 917, 'more': 918, 'more,': 919, 'more.': 920, 'more?"': 921, 'most': 922, 'mourn': 923, 'mourned.': 924, 'moustache,': 925, 'moved': 926, 'moved,': 927, 'moved.': 928, 'much': 929, 'muddling;': 930, 'multiplied': 931, 'murmur.': 932, 'muscles,': 933, 'must': 934, 'my': 935, 'myself': 936, 'myself--and': 937, 'mysterious': 938, 'naive': 939, 'near': 940, 'nearly': 941, 'negatived': 942, 'nervous': 943, 'nervousness;': 944, 'neutral': 945, 'never': 946, 'next': 947, 'no': 948, 'no--for': 949, 'none': 950, 'not': 951, 'note': 952, 'note!': 953, 'note.': 954, 'nothing--I': 955, 'now': 956, 'now,': 957, 'nymphs': 958, 'oak': 959, 'object': 960, 'objects': 961, 'objects,': 962, 'occurred': 963, 'oddly': 964, 'of': 965, 'of,': 966, 'of."': 967, 'off': 968, 'off--and': 969, 'off?"': 970, 'often,': 971, 'oh,': 972, 'old': 973, 'on': 974, 'on--forming,': 975, 'once': 976, 'once--but': 977, 'one': 978, 'one,"': 979, 'one--my': 980, 'ones': 981, 'only': 982, 'onto': 983, 'open': 984, 'or': 985, 'other': 986, 'our': 987, 'ourselves': 988, 'out': 989, 'out,': 990, 'out--all': 991,

'out.': 992, 'out:': 993, "out?'": 994, 'outline': 995, 'oval': 996, 'over': 997, 'over!': 998, 'own': 999, 'packed': 1000, 'paid': 1001, 'paint': 1002, 'paint.': 1003, 'painted': 1004, 'painted,': 1005, 'painted;': 1006, 'painter--ah,': 1007, 'painter."': 1008, 'painting': 1009, 'painting,': 1010, 'painting--since': 1011, 'painting.': 1012, 'painting;': 1013, 'painting?"': 1014, 'pale': 1015, 'paled': 1016, 'palm-trees;': 1017, 'panel': 1018, 'panelling': 1019, 'pardonable': 1020, 'pardoned': 1021, 'part,': 1022, 'passages': 1023, 'passing': 1024, 'past': 1025, 'past!': 1026, 'pastels': 1027, 'pathos': 1028, 'patient': 1029, 'people': 1030, 'perceptible': 1031, 'perfect': 1032, 'persistence': 1033, 'persuasively': 1034, 'phrase': 1035, 'picture': 1036, 'picture--had': 1037, 'picture.': 1038, 'pictures': 1039, 'pictures--so': 1040, 'pictures--the': 1041, 'pictures.': 1042, 'pictures?': 1043, 'pines,': 1044, 'pink': 1045, 'place': 1046, 'placed': 1047, 'plain': 1048, 'platitudes': 1049, 'pleased': 1050, 'pockets': 1051, 'point': 1052, 'point.': 1053, 'poised': 1054, 'poor': 1055, 'portrait': 1056, 'portrait,': 1057, 'portrait--and': 1058, 'portrait--she': 1059, 'portrait.': 1060, 'posing': 1061, 'possessed': 1062, 'poverty.': 1063, 'predicted,': 1064, 'preliminary': 1065, 'presenting': 1066, 'prestidigitation': 1067, 'pretty': 1068, 'previous': 1069, 'price': 1070, 'pride.': 1071, 'pride:': 1072, 'princely,': 1073, 'prism': 1074, 'problem.': 1075, 'proclaiming': 1076, 'prodigious': 1077, 'profusion': 1078, 'protest,': 1079, 'prove': 1080, 'public,': 1081, 'public.': 1082, 'purblind': 1083, 'purely': 1084, 'pushed': 1085, 'put': 1086, 'qualities': 1087, 'quality': 1088, 'queerly--as': 1089, 'question': 1090, 'question:': 1091, 'quickly.': 1092, 'quietly': 1093, 'quite': 1094, 'quote': 1095, 'rain': 1096, 'raised': 1097, 'random': 1098, 'rather': 1099, 'real': 1100, 'really': 1101, 'reared': 1102, 'reason': 1103, 'reason."': 1104, 'reassurance.': 1105, 'recovering': 1106, 'recreated': 1107, 'reflected': 1108, 'reflection,': 1109, 'regrets': 1110, 'relatively': 1111, 'remained': 1112, 'remember': 1113, 'reminded': 1114, 'repeating': 1115, 'represented,': 1116, 'reproduction': 1117, 'resented': 1118, 'resolve': 1119, 'resources.': 1120, 'rest': 1121, 'rich': 1122, 'rich;': 1123, 'ridiculous': 1124, 'robbed': 1125, 'romantic!': 1126, 'room': 1127, 'room,': 1128, 'room.': 1129, 'rose': 1130, 'rule,': 1131, 'run': 1132, 'said': 1133, 'said,': 1134, 'said--"': 1135, 'said.': 1136, 'said:': 1137, 'same': 1138, 'satisfaction.': 1139, 'satisfaction:': 1140, 'savour?': 1141, 'saw': 1142, 'say': 1143, 'say,': 1144, 'say.': 1145, 'say:': 1146, 'saying': 1147, 'says': 1148, 'scorn': 1149, 'scornful': 1150, 'secret': 1151, 'secret,': 1152, 'secret?': 1153, 'see': 1154, 'seemed': 1155, 'seen': 1156, 'self-confident': 1157, 'send': 1158, 'sensation': 1159, 'sensitive': 1160, 'sent': 1161, 'serious': 1162, 'set': 1163, 'sex': 1164, 'shade': 1165, 'shaking': 1166, 'shall': 1167, 'she': 1168, 'shirked,': 1169, 'short,': 1170, 'should': 1171, 'shoulder': 1172, 'shoulders,': 1173, 'show': 1174, 'show,': 1175, 'showed': 1176, 'showy': 1177, 'showy!': 1178, 'shrug.': 1179, 'shrugged': 1180, 'sight': 1181, 'sign': 1182, 'silent;': 1183,

'silver': 1184, 'similar': 1185, 'simpleton,': 1186, 'simplifications': 1187, 'simply': 1188, 'simply.': 1189, 'since': 1190, 'single': 1191, 'sitter--deploring': 1192, 'sitters': 1193, 'sitters.': 1194, 'sketch': 1195, 'sketch,': 1196, 'skill,': 1197, 'slight': 1198, 'slightly,': 1199, 'slowly:': 1200, 'small': 1201, 'smile': 1202, 'smiling.': 1203, 'sneer,': 1204, 'so': 1205, 'so--his': 1206, 'so.': 1207, 'solace': 1208, 'some': 1209, 'somebody': 1210, 'something': 1211, 'spacious': 1212, 'spaniel': 1213, 'speaking-tubes,': 1214, 'speculations;': 1215, 'spite': 1216, 'splash': 1217, 'square': 1218, 'stairs': 1219, 'stammer': 1220, 'stand': 1221, 'standing': 1222, 'started': 1223, 'stay!': 1224, 'stay."': 1225, 'still': 1226, 'stocked': 1227, 'stood': 1228, 'stopped': 1229, 'stopping': 1230, 'straddling': 1231, 'straight': 1232, 'strain': 1233, 'straining,': 1234, 'strange': 1235, 'straw': 1236, 'stream,': 1237, 'stroke,': 1238, 'stroke.': 1239, 'strokes': 1240, 'strolled': 1241, 'struck': 1242, 'studio.': 1243, 'stuff': 1244, 'subject,': 1245, 'substantial': 1246, 'suburban': 1247, 'such': 1248, 'such--had': 1249, 'suddenly': 1250, 'suddenly,': 1251, 'suffered': 1252, 'sugar': 1253, 'suggested': 1254, 'sunburn.': 1255, 'sunburnt': 1256, 'sunlit': 1257, 'superb.': 1258, 'sure': 1259, 'surest': 1260, 'surface': 1261, 'surprise': 1262, 'surprise,': 1263, 'surprise.': 1264, 'surprised': 1265, 'surrounded': 1266, 'suspected': 1267, 'sweetness.': 1268, 'swelling': 1269, 'swept': 1270, 'swum': 1271, 'table': 1272, 'take': 1273, 'taken': 1274, 'taken.': 1275, 'talking': 1276, 'tea': 1277, 'tears': 1278, 'technicalities': 1279, 'tell': 1280, 'tells': 1281, 'tempting': 1282, 'terra-cotta': 1283, 'terrace': 1284, 'terrace.': 1285, 'terraces': 1286, 'terribly': 1287, 'than': 1288, 'that': 1289, 'that,': 1290, 'that."': 1291, 'the': 1292, 'their': 1293, 'them': 1294, 'them.': 1295, 'then': 1296, 'then,': 1297, 'there': 1298, "there's": 1299, 'there,': 1300, 'therefore': 1301, 'they': 1302, "they're": 1303, 'thin': 1304, 'thing': 1305, "thing--couldn't": 1306, 'thing.': 1307, 'things': 1308, 'things.': 1309, 'things?': 1310, 'think': 1311, 'this': 1312, 'thither': 1313, 'those': 1314, 'though': 1315, 'thought': 1316, 'thought:': 1317, 'three': 1318, 'threshold.': 1319, 'threw': 1320, 'through': 1321, 'through,': 1322, 'through--see': 1323, 'throwing': 1324, 'tie': 1325, 'till': 1326, 'time': 1327, 'time,': 1328, 'timorously': 1329, 'tinge': 1330, 'tips': 1331, 'tired': 1332, 'to': 1333, 'told': 1334, 'tone': 1335, 'tone.': 1336, 'tones': 1337, 'too': 1338, 'too?': 1339, 'took': 1340, 'tottering': 1341, 'touched': 1342, 'toward': 1343, 'trace': 1344, 'trade': 1345, 'transmute': 1346, 'traps': 1347, 'traps,': 1348, 'travelled': 1349, 'tribute': 1350, 'tributes--was': 1351, 'tricks': 1352, 'tricks.': 1353, 'tried': 1354, 'trouser-presses--all': 1355, 'true.': 1356, 'truth': 1357, 'turned': 1358, 'turned,': 1359, 'twenty': 1360, 'twenty-four': 1361, 'twice,': 1362, 'twirling': 1363, 'unaccountable': 1364, 'uncertain.': 1365, 'under': 1366, 'under,': 1367, 'under--his': 1368, 'underlay': 1369, 'underneath.': 1370, 'understand': 1371, 'understand,': 1372, 'unexpected': 1373, 'untouched.': 1374, 'unusual': 1375, 'up': 1376,

```
'up,': 1377, 'up-stream': 1378, 'up.': 1379, 'up;': 1380, 'upon':
1381, 'upset': 1382, 'upstairs."': 1383, 'us': 1384, 'us!"': 1385,
'used': 1386, 'usual': 1387, 'value': 1388, 'varnishing': 1389,
'vases': 1390, 'veins,': 1391, 'velveteen': 1392, 'very': 1393,
'villa': 1394, 'vindicated': 1395, 'vindicated--and': 1396,
'virtuosity.': 1397, 'vista': 1398, 'vista.': 1399, 'vocation': 1400,
'voice': 1401, 'wall': 1402, 'wall.': 1403, 'wander': 1404, 'want':
1405, 'wanted': 1406, 'wants': 1407, 'was': 1408, 'was,': 1409,
'was.': 1410, "wasn't": 1411, 'watched': 1412, 'watching': 1413,
'watching,': 1414, 'water-colour--or': 1415, 'waves': 1416, 'way':
1417, 'weekly--above': 1418, "weeks'": 1419, 'weeks,': 1420,
'welcome': 1421, 'went': 1422, 'were': 1423, 'were,': 1424, 'what':
1425, 'when': 1426, 'when,': 1427, 'whenever': 1428, 'where': 1429,
'which': 1430, 'which,': 1431, 'while': 1432, 'white': 1433, 'white-
panelled': 1434, 'who': 1435, 'who,': 1436, 'whole': 1437, 'whom)':
1438, 'why': 1439, 'why.': 1440, 'wide': 1441, 'widow,': 1442, 'wife':
1443, 'wife!': 1444, "wife's": 1445, 'wild--I': 1446, 'wincing': 1447,
'wincing;': 1448, 'window-curtains,': 1449, 'wish': 1450, 'with':
1451, 'without': 1452, "wits'": 1453, 'woman': 1454, 'woman!': 1455,
'women': 1456, 'women:': 1457, "won't": 1458, 'wonder': 1459,
'wonder!': 1460, 'wondered': 1461, 'word,': 1462, 'word.': 1463,
'work': 1464, 'work!"': 1465, 'work,': 1466, 'work.': 1467, 'work."':
1468, 'working?': 1469, 'worth': 1470, 'would': 1471, "wouldn't":
1472, 'year': 1473, 'years': 1474, 'yellow': 1475, 'yet': 1476,
'yet,': 1477, 'you': 1478, "you'd": 1479, "you're": 1480, 'you--
because': 1481, 'you?"': 1482, 'younger': 1483, 'your': 1484,
'yourself': 1485}

import re

class SimpleTokenizerV1:
    """A simple tokenizer that splits text and converts tokens to
integer IDs."""
    def __init__(self, vocab):
        self.str_to_int = vocab
        self.int_to_str = {i: s for s, i in vocab.items()}

    def encode(self, text):
        """Converts a string of text into a list of integer IDs."""
        # Split text by punctuation, '--', or whitespace, keeping
delimiters
        preprocessed = re.split(r'([,.:;?_!"()\']|--|\s)', text)

        # Remove empty strings and strip whitespace from each token
        # Using a list comprehension preserves the order and
duplicates
        preprocessed = [item.strip() for item in preprocessed if
item.strip()]

        # Convert tokens to integers, ignoring any tokens not in the
```

```
vocabulary
        ids = [self.str_to_int[s] for s in preprocessed if s in
self.str_to_int]
        return ids

    def decode(self, ids):
        """Converts a list of integer IDs back into a string of
text."""
        # Convert integers back to string tokens and join them with
spaces
        text = " ".join([self.int_to_str[i] for i in ids])

        # Clean up spacing around punctuation for a more natural
output
        text = re.sub(r'\s+([,.:;?_!"()\'])', r'\1', text)
        return text

vocab["Jack"]
```

```
119
```

```
int_to_str = {i: s for s, i in vocab.items()}

int_to_str[119]
```

```
'Jack'
```

```
tokenizer = SimpleTokenizerV1(vocab)

text = """It's the last he painted, you know," Mrs. Gisburn said with
pardonable pride,"""

ids = tokenizer.encode(text)
print(ids)
```

```
[118, 1292, 810, 691, 1004, 1478, 799, 65, 93, 1133, 1451, 1020]
```

```
tokenizer.decode(ids)
```

```
'It the last he painted you know. Gisburn said with pardonable'
```

```
text = "Hello, do you like tea, is this-- a test?"

tokenizer.encode(text)
```

```
[489, 1478, 843, 1277, 773, 1312, 212]
```

```
#Adding new tokens into vocab
all_wordss = sorted(list(set(result)))
all_words.extend(["<|endoftext|>", "<|unk|>"])

vocab = {token:integer for integer,token in enumerate(all_words)}
```

```
#verify if new added tokens are in
len(vocab.items())
```

```
1488
```

```
for i, item in enumerate(list(vocab.items())[-5:]):
    print(item)
```

```
('younger', 1483)
('your', 1484)
('yourself', 1485)
('<|endoftext|>', 1486)
('<|unk|>', 1487)
```

```
#New improved tokenizer to deal with unknown tokens
import re

class SimpleTokenizerV2:
    """A simple tokenizer that splits text and converts tokens to
integer IDs."""
    def __init__(self, vocab):
        self.str_to_int = vocab
        self.int_to_str = {i: s for s, i in vocab.items()}

    def encode(self, text):
        """Converts a string of text into a list of integer IDs."""
        # Split text by punctuation, '--', or whitespace, keeping
delimiters
        preprocessed = re.split(r'([,.:;?_!"()\']|--|\s)', text)

        # Remove empty strings and strip whitespace from each token
        preprocessed = [item.strip() for item in preprocessed if
item.strip()]

        #Dealing with unknown tokens
        unk_token_id = self.str_to_int["<|unk|>"]
        ids = [self.str_to_int.get(s, unk_token_id) for s in
preprocessed]
        return ids

    def decode(self, ids):
        """Converts a list of integer IDs back into a string of
text."""
        # Convert integers back to string tokens and join them with
spaces
        text = " ".join([self.int_to_str[i] for i in ids])

        # Clean up spacing around punctuation for a more natural
output
        text = re.sub(r'\s+([,.:;?_!"()\'])', r'\1', text)
        return text
```

```
text = "Hello this is japan"
tokenizer = SimpleTokenizerV2(vocab)

tokenizer.encode(text)
```

```
[1487, 1312, 773, 1487]
```

```
tokenizer.decode(tokenizer.encode(text))
```

```
'<|unk|> this is <|unk|>'
```

```
#Byte Pair Encoding

import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")

tokenizer.encode("Hello World!")
```

```
[15496, 2159, 0]
```

```
tokenizer.decode(tokenizer.encode("Hello World!"))
```

```
'Hello World!'
```

```
text = "Hello world <|endoftext|> i eat food"

tokenizer.encode(text)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[31], line 3
      1 text = "Hello world <|endoftext|> i eat food"
----> 3 tokenizer.encode(text)

File ~\OneDrive\Desktop\LLMs-from-scratch-main\.venv\Lib\site-packages\tiktoken\core.py:121, in Encoding.encode(self, text, allowed_special, disallowed_special)
    119         disallowed_special = frozenset(disallowed_special)
    120     if match := _special_token_regex(disallowed_special).search(text):
--> 121         raise_disallowed_special_token(match.group())
    123 try:
    124     return self._core_bpe.encode(text, allowed_special)

File ~\OneDrive\Desktop\LLMs-from-scratch-main\.venv\Lib\site-packages\tiktoken\core.py:432, in raise_disallowed_special_token(token)
    431 def raise_disallowed_special_token(token: str) -> NoReturn:
--> 432     raise ValueError(
```

```
    433         f"Encountered text corresponding to disallowed special
token {token!r}.\n"
    434         "If you want this text to be encoded as a special
token, "
    435         f"pass it to `allowed_special`, e.g.
`allowed_special={{{token!r}, ...}}`.\n"
    436         f"If you want this text to be encoded as normal text,
disable the check for this token "
    437         f"by passing
`disallowed_special=(enc.special_tokens_set - {{{token!r}}})`.\n"
    438         "To disable this check for all special tokens, pass
`disallowed_special=()`.\n"
    439     )

ValueError: Encountered text corresponding to disallowed special token
'<|endoftext|>'.
If you want this text to be encoded as a special token, pass it to
`allowed_special`, e.g. `allowed_special={'<|endoftext|>', ...}`.
If you want this text to be encoded as normal text, disable the check
for this token by passing `disallowed_special=(enc.special_tokens_set
- {'<|endoftext|>'})`.
To disable this check for all special tokens, pass
`disallowed_special=()`.
```

```
#EOD causes error unless specified otherwise
text = "Hello world <|endoftext|> i eat food"

tokenizer.encode(text, allowed_special={"<|endoftext|>"})
```

```
[15496, 995, 220, 50256, 1312, 4483, 2057]
```

```
#Data sampling with a sliding window

with open("the-verdict-txt", "r", encoding="utf=8") as f:
    raw_text = f.read()

enc_text = tokenizer.encode(raw_text)
print(len(enc_text))
```

```
5145
```

```
enc_text
```

```
[40,
 367,
 2885,
 1464,
 1807,
 3619,
 402,
```

```
271,
10899,
2138,
257,
7026,
15632,
438,
2016,
257,
922,
5891,
1576,
438,
568,
340,
373,
645,
1049,
5975,
284,
502,
284,
3285,
326,
11,
287,
262,
6001,
286,
465,
13476,
11,
339,
550,
5710,
465,
12036,
11,
6405,
257,
5527,
27075,
11,
290,
4920,
2241,
287,
257,
4489,
```

```
64,
319,
262,
34686,
41976,
13,
357,
10915,
314,
2138,
1807,
340,
561,
423,
587,
10598,
393,
28537,
2014,
198,
198,
1,
464,
6001,
286,
465,
13476,
1,
438,
5562,
373,
644,
262,
1466,
1444,
340,
13,
314,
460,
3285,
9074,
13,
46606,
536,
5469,
438,
14363,
938,
4842,
```

```
1650,
353,
438,
2934,
489,
3255,
465,
48422,
540,
450,
67,
3299,
13,
366,
5189,
1781,
340,
338,
1016,
284,
3758,
262,
1988,
286,
616,
4286,
705,
1014,
510,
26,
475,
314,
836,
470,
892,
286,
326,
11,
1770,
13,
8759,
2763,
438,
1169,
2994,
284,
943,
17034,
318,
```

```
477,
314,
892,
286,
526,
383,
1573,
11,
319,
9074,
13,
536,
5469,
338,
11914,
11,
33096,
663,
4808,
3808,
62,
355,
996,
484,
547,
12548,
287,
281,
13079,
410,
12523,
286,
22353,
13,
843,
340,
373,
407,
691,
262,
9074,
13,
536,
48819,
508,
25722,
276,
13,
11161,
```

```
407,
262,
40123,
18113,
544,
9325,
701,
11,
379,
262,
938,
402,
1617,
261,
12917,
905,
11,
5025,
502,
878,
402,
271,
10899,
338,
366,
31640,
12,
67,
20811,
1,
284,
910,
11,
351,
10953,
287,
607,
2951,
25,
366,
1135,
2236,
407,
804,
2402,
663,
588,
757,
13984,
```

```
198,
198,
5779,
28112,
10197,
832,
262,
46475,
286,
18113,
544,
338,
10953,
314,
2936,
1498,
284,
1986,
262,
1109,
351,
1602,
11227,
414,
13,
23676,
3619,
402,
271,
10899,
0,
383,
1466,
550,
925,
683,
438,
270,
373,
15830,
326,
484,
815,
25722,
683,
13,
9754,
465,
898,
```

```
1714,
7380,
30090,
547,
2982,
11,
290,
287,
465,
898,
3292,
8941,
257,
4636,
28582,
13,
18612,
35394,
30,
8673,
13,
1002,
340,
547,
11,
262,
15393,
286,
262,
5977,
373,
29178,
3474,
416,
1310,
40559,
11959,
1636,
11,
508,
11,
287,
477,
922,
4562,
11,
3181,
503,
287,
```

```
262,
37090,
257,
845,
22665,
366,
672,
270,
2838,
1,
319,
3619,
438,
505,
286,
883,
905,
88,
6685,
42070,
351,
4738,
6276,
871,
326,
314,
423,
2982,
357,
40,
1839,
470,
910,
416,
4150,
8,
3688,
284,
402,
271,
10899,
338,
12036,
13,
843,
523,
438,
14363,
10568,
```

```
852,
5729,
11331,
18893,
540,
438,
1169,
5114,
11835,
3724,
503,
11,
290,
11,
355,
9074,
13,
536,
5469,
550,
11001,
11,
262,
2756,
286,
366,
38,
271,
10899,
82,
1,
1816,
510,
13,
198,
198,
1026,
373,
407,
10597,
1115,
812,
1568,
326,
11,
287,
262,
1781,
286,
```

```
257,
1178,
2745,
6,
4686,
1359,
319,
262,
34686,
41976,
11,
340,
6451,
5091,
284,
502,
284,
4240,
1521,
402,
271,
10899,
550,
1813,
510,
465,
12036,
13,
1550,
14580,
11,
340,
1107,
373,
257,
29850,
1917,
13,
1675,
24456,
465,
3656,
561,
423,
587,
1165,
2562,
438,
14363,
3148,
```

```
1650,
1010,
550,
587,
6699,
262,
1540,
558,
286,
2282,
326,
9074,
13,
402,
271,
10899,
550,
366,
7109,
14655,
683,
866,
526,
1114,
9074,
13,
402,
271,
10899,
438,
292,
884,
438,
18108,
407,
11196,
10597,
3016,
257,
614,
706,
3619,
338,
10568,
550,
587,
2077,
13,
632,
```

```
1244,
307,
326,
339,
550,
6405,
607,
438,
20777,
339,
8288,
465,
10152,
438,
13893,
339,
1422,
470,
765,
284,
467,
319,
12036,
26,
475,
340,
561,
423,
587,
1327,
284,
5879,
326,
339,
550,
1813,
510,
465,
12036,
780,
339,
550,
6405,
607,
13,
198,
198,
5189,
1781,
```

```
11,
611,
673,
550,
407,
17901,
683,
866,
11,
673,
550,
8603,
11,
355,
4544,
9325,
701,
42397,
11,
4054,
284,
366,
26282,
683,
510,
1,
438,
7091,
550,
407,
2957,
683,
736,
284,
262,
1396,
417,
13,
1675,
1234,
262,
14093,
656,
465,
1021,
757,
438,
10919,
257,
```

```
410,
5040,
329,
257,
3656,
0,
887,
9074,
13,
402,
271,
10899,
4120,
284,
423,
595,
67,
1328,
340,
438,
392,
314,
2936,
340,
1244,
307,
3499,
284,
1064,
503,
1521,
13,
198,
198,
464,
748,
586,
652,
1204,
286,
262,
34686,
41976,
37733,
2346,
284,
884,
14177,
8233,
```

```
1020,
5768,
26,
290,
1719,
11,
319,
616,
835,
284,
22489,
40089,
11,
4978,
257,
19350,
286,
3619,
338,
3652,
436,
81,
5286,
8812,
2114,
1022,
262,
279,
1127,
11,
314,
550,
3589,
28068,
294,
1555,
262,
1306,
1110,
13,
198,
198,
40,
1043,
262,
3155,
379,
8887,
11061,
```

```
511,
18057,
12,
83,
6037,
26,
290,
9074,
13,
402,
271,
10899,
338,
7062,
373,
523,
2429,
498,
326,
11,
287,
262,
29543,
2745,
11,
314,
4752,
340,
6777,
13,
632,
373,
407,
326,
616,
2583,
408,
373,
366,
47914,
1298,
319,
326,
966,
314,
714,
423,
1813,
4544,
```

```
9325,
701,
262,
40830,
12719,
3874,
13,
632,
373,
655,
780,
673,
373,
4808,
1662,
62,
3499,
438,
361,
314,
743,
307,
41746,
12004,
262,
6473,
438,
5562,
314,
1043,
607,
523,
13,
1114,
3619,
11,
477,
465,
1204,
11,
550,
587,
11191,
416,
3499,
1466,
25,
484,
550,
```

```
26546,
1068,
465,
1242,
11,
340,
550,
587,
302,
1144,
287,
262,
3024,
12,
4803,
286,
511,
512,
1741,
13,
843,
340,
373,
4361,
5048,
425,
284,
3465,
644,
1245,
262,
366,
25124,
3101,
8137,
286,
16957,
1696,
414,
1,
357,
40,
9577,
4544,
9325,
701,
8,
373,
1719,
```

```
319,
683,
13,
198,
198,
40,
423,
4750,
326,
9074,
13,
402,
271,
10899,
373,
5527,
26,
290,
340,
373,
3393,
34953,
856,
326,
607,
5229,
373,
37895,
422,
428,
25179,
257,
19217,
475,
8904,
14676,
13,
632,
318,
11,
355,
257,
3896,
11,
262,
661,
508,
40987,
1637,
```

```
508,
651,
749,
503,
286,
340,
26,
290,
3619,
338,
19992,
31564,
286,
465,
3656,
338,
1263,
5236,
9343,
683,
11,
351,
281,
5585,
286,
2818,
922,
12,
49705,
11,
284,
21595,
1133,
340,
656,
5563,
286,
1242,
290,
13064,
13,
1675,
262,
6846,
11,
314,
1276,
751,
11,
```

```
 339,
 6150,
 5365,
 31655,
 26,
 475,
 339,
 373,
 7067,
 29396,
 18443,
 12271,
 ...]
```

```
enc_sample = enc_text[50:]
```

```
#We will assume as 4 window but generally its very large like 1024 for
gpt-2.0
```

```
len(enc_sample)
```

```
5095
```

```
context_size = 4
```

```
x = enc_sample[:context_size]
y = enc_sample[1:context_size+1]
```

```
print(f" x: {x}")
print(f" y:      {y}")
```

```
 x: [290, 4920, 2241, 287]
 y:      [4920, 2241, 287, 257]
```

```
for i in range(1, context_size+1):
    context = enc_sample[:i]
    desired = enc_sample[i]

    print(context, "---->", desired)
```

```
[290] ----> 4920
[290, 4920] ----> 2241
[290, 4920, 2241] ----> 287
[290, 4920, 2241, 287] ----> 257
```

```
for i in range(1, context_size+1):
    context = enc_sample[:i]
    desired = enc_sample[i]

    print(tokenizer.decode(context), "---->",
tokenizer.decode([desired]))
```

```
 and ---->  established
 and established ---->  himself
 and established himself ---->  in
 and established himself in ---->  a

#PyTorch Usage

import torch

from torch.utils.data import Dataset, DataLoader

class GPTDatasetV1(Dataset):
    def __init__(self, txt, tokenizer, max_length, stride):
        self.input_ids = []
        self.target_ids = []

        #Tokenize the entire text
        token_ids = tokenizer.encode(txt, allowed_special={"<|
endoftext|>"})

        #Use a sliding window to chunk the book into overlapping
sequences of max_length
        for i in range(0, len(token_ids) - max_length, stride):
            input_chunk = token_ids[i:i + max_length]
            target_chunk = token_ids[i+1: i + max_length + 1]
            self.input_ids.append(torch.tensor(input_chunk))
            self.target_ids.append(torch.tensor(target_chunk))

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self,idx):
        return self.input_ids[idx], self.target_ids[idx]

def create_dataloader_v1(txt, batch_size=4, max_length=256,
stride=128, shuffle=True, drop_last=True, num_workers=0):
    #Initialize the tokenizer
    tokenizer = tiktoken.get_encoding("gpt2")

    #Create dataset
    dataset = GPTDatasetV1(txt, tokenizer, max_length, stride)

    #Create dataloader
    dataloader = DataLoader(
    dataset, batch_size=batch_size, shuffle=shuffle,
drop_last=drop_last, num_workers=num_workers)

    return dataloader

with open("the-verdict-txt", "r", encoding="utf=8") as f:
    raw_text = f.read()
```

```
dataloader = create_dataloader_v1( raw_text, batch_size=1,
max_length=4, stride=1, shuffle=False)
data_iter = iter(dataloader)
first_batch = next(data_iter)
print(first_batch)
```

```
[tensor([[  40,  367, 2885, 1464]]), tensor([[ 367, 2885, 1464,
1807]])]
```

```
second_batch = next(data_iter)
print(second_batch)
```

```
[tensor([[ 367, 2885, 1464, 1807]]), tensor([[2885, 1464, 1807,
3619]])]
```

```
#this stride=1 can lead to overfitting so lets do with stride= 4
instead
dataloader = create_dataloader_v1( raw_text, batch_size=1,
max_length=4, stride=4, shuffle=False)
data_iter = iter(dataloader)
first_batch = next(data_iter)
print(first_batch)
```

```
[tensor([[  40,  367, 2885, 1464]]), tensor([[ 367, 2885, 1464,
1807]])]
```

```
second_batch = next(data_iter)
print(second_batch)
```

```
[tensor([[1807, 3619,  402,  271]]), tensor([[ 3619,   402,   271,
10899]])]
```

```
#with different batch size
dataloader = create_dataloader_v1(raw_text, batch_size=4,
max_length=4, stride=4, shuffle=False)
data_iter = iter(dataloader)
inputs, targets = next(data_iter)
print("Inputs:", inputs)
print("Targets:", targets)
```

```
Inputs: tensor([[   40,   367,  2885,  1464],
        [ 1807,  3619,   402,   271],
        [10899,  2138,   257,  7026],
        [15632,   438,  2016,   257]])
Targets: tensor([[  367,  2885,  1464,  1807],
        [ 3619,   402,   271, 10899],
        [ 2138,   257,  7026, 15632],
        [  438,  2016,   257,   922]])
```

```
#Creating token embeddings
input_ids = torch.tensor([ 2,   3,    5,    1])
```

```
tokenizer.n_vocab
```

```
50257
```

```
vocab_size = 6
output_dim = 3
torch.manual_seed(123) #Random seed
embedding_layer = torch.nn.Embedding(vocab_size, output_dim)

#Getting weight parameters
print(embedding_layer.weight)
```

```
Parameter containing:
tensor([[ 0.3374, -0.1778, -0.1690],
        [ 0.9178,  1.5810,  1.3010],
        [ 1.2753, -0.2010, -0.1606],
        [-0.4015,  0.9666, -1.1481],
        [-1.1589,  0.3255, -0.6315],
        [-2.8400, -0.7849, -1.4096]], requires_grad=True)
```

```
#This is the 3rd index row in the previous output
embedding_layer(torch.tensor([3]))
```

```
tensor([[-0.4015,  0.9666, -1.1481]], grad_fn=<EmbeddingBackward0>)
```

```
#This is the 2nd index row in the previous output
embedding_layer(torch.tensor([2]))
```

```
tensor([[ 1.2753, -0.2010, -0.1606]], grad_fn=<EmbeddingBackward0>)
```

```
embedding_layer(input_ids)
```

```
tensor([[ 1.2753, -0.2010, -0.1606],
        [-0.4015,  0.9666, -1.1481],
        [-2.8400, -0.7849, -1.4096],
        [ 0.9178,  1.5810,  1.3010]], grad_fn=<EmbeddingBackward0>)
```

```
vocab_size = 6
output_dim = 3
torch.manual_seed(123) #Random seed
embedding_layer = torch.nn.Embedding(tokenizer.n_vocab, output_dim)

print(embedding_layer.weight)
```

```
Parameter containing:
tensor([[ 0.3374, -0.1778, -0.3035],
        [-0.5880,  0.3486,  0.6603],
        [-0.2196, -0.3792,  0.7671],
        ...,
        [-0.5931,  1.0895, -0.6854],
        [ 0.7447,  0.5803, -0.4246],
        [-0.3130,  0.7558, -1.2656]], requires_grad=True)
```

```
#Encoding word positions
vocab_size = 50257
output_dim = 256

token_embedding_layer = torch.nn.Embedding(vocab_size, output_dim)

max_length = 4
dataloader = create_dataloader_v1(
    raw_text, batch_size=8, max_length=max_length,
    stride=max_length, shuffle=False
)
data_iter = iter(dataloader)
inputs, targets = next(data_iter)

print("Token IDs:\n", inputs)
print("\nInputs shape:\n", inputs.shape)
```

```
Token IDs:
 tensor([[   40,   367,  2885,  1464],
        [ 1807,  3619,   402,   271],
        [10899,  2138,   257,  7026],
        [15632,   438,  2016,   257],
        [  922,  5891,  1576,   438],
        [  568,   340,   373,   645],
        [ 1049,  5975,   284,   502],
        [  284,  3285,   326,    11]])

Inputs shape:
 torch.Size([8, 4])
```

```
#Each token id now gets even more dimensions (256 here)
token_embeddings = token_embedding_layer(inputs)
token_embeddings.shape
```

```
torch.Size([8, 4, 256])

context_length = max_length
pos_embedding_layer = torch.nn.Embedding(context_length, output_dim)

torch.arange(max_length)

tensor([0, 1, 2, 3])

pos_embedding_layer.weight

Parameter containing:
tensor([[-1.3798,  1.3476, -0.3612,  ..., -0.7712,  0.1523, -0.5973],
        [ 0.3611, -0.5228, -0.2888,  ...,  0.8571,  0.2221,  0.1976],
        [ 1.2194,  0.8234,  0.2277,  ...,  2.5752, -1.7081, -0.5515],
        [-0.5765, -1.6450, -1.3456,  ...,  0.7075,  0.0123, -1.2205]],
       requires_grad=True)
```

```
pos_embedding_layer(torch.arange(max_length))
```

```
tensor([[-1.3798,  1.3476, -0.3612,  ..., -0.7712,  0.1523, -0.5973],
        [ 0.3611, -0.5228, -0.2888,  ...,  0.8571,  0.2221,  0.1976],
        [ 1.2194,  0.8234,  0.2277,  ...,  2.5752, -1.7081, -0.5515],
        [-0.5765, -1.6450, -1.3456,  ...,  0.7075,  0.0123, -1.2205]],
       grad_fn=<EmbeddingBackward0>)
```

```
pos_embeddings = pos_embedding_layer(torch.arange(max_length))
print(pos_embeddings.shape)
```

```
torch.Size([4, 256])
```

```
token_embeddings.shape
```

```
torch.Size([8, 4, 256])
```

```
token_embeddings[0] + pos_embeddings
```

```
tensor([[-0.9206,  2.1135, -2.0881,  ..., -2.4569, -0.8164, -1.1871],
        [ 0.3567,  1.1969,  0.0850,  ...,  0.3700,  0.8277,  0.8440],
        [ 1.1740, -0.0410, -1.7390,  ...,  3.6197, -4.8771,  1.4966],
        [-0.9454, -1.5052, -1.5424,  ...,  0.6199,  1.6592, -1.4935]],
       grad_fn=<AddBackward0>)
```

```
input_embeddings = token_embeddings + pos_embeddings
print(input_embeddings.shape)
```

```
torch.Size([8, 4, 256])
```

```
#A simple self-attention mechanism without trainable weights
import torch

inputs = torch.tensor(
    [[0.43, 0.15, 0.89], # Your (x^1)
     [0.55, 0.87, 0.66], # journey (x^2)
     [0.57, 0.85, 0.64], # starts (x^3)
     [0.22, 0.58, 0.33], # with (x^4)
     [0.77, 0.25, 0.10], # one (x^5)
     [0.05, 0.80, 0.55]] # step (x^6)
)

input_query = inputs[1] # Assuming journey as the word
input_query
```

```
tensor([0.5500, 0.8700, 0.6600])
```

```
input_1 = inputs[0]
input_1
```

```
tensor([0.4300, 0.1500, 0.8900])
```

```
0.55*0.43 + 0.87*0.15 + 0.66*0.89 #Manual way
```

```
0.9544
```

```
torch.dot(input_query, input_1) #Pytorch commands
```

```
tensor(0.9544)
```

```
for idx,ele in enumerate(inputs[0]):
    print(inputs[0][idx])
```

```
tensor(0.4300)
tensor(0.1500)
tensor(0.8900)
```

```
res = 0
for idx,ele in enumerate(inputs[0]):
    res += inputs[0][idx] * input_query[idx]
res
```

```
tensor(0.9544)
```

```
res = 0
for idx,ele in enumerate(inputs[1]):
    res += inputs[1][idx] * input_query[idx]
res
```

```
tensor(1.4950)
```

```
i = 3
res = torch.dot(inputs[i], input_query)
res
```

```
tensor(0.8434)
```

```
inputs.shape[0]
```

```
6
```

```
torch.empty(inputs.shape[0])
```

```
tensor([0., 0., 0., 0., 0., 0.])
```

```
#automating for all rows calculating the attention scores
query = inputs[1] #2nd input token is the query
attn_scores_2 = torch.empty(inputs.shape[0])
for i, x_i in enumerate(inputs):
    attn_scores_2[i] = torch.dot(x_i, input_query)
print(attn_scores_2)
```

```
tensor([0.9544, 1.4950, 1.4754, 0.8434, 0.7070, 1.0865])
```

```
#Normalizing them
attn_weights_2_tmp = attn_scores_2 / attn_scores_2.sum()
attn_weights_2_tmp
```

```
tensor([0.1455, 0.2278, 0.2249, 0.1285, 0.1077, 0.1656])
```

```
attn_weights_2_tmp.sum()
```

```
tensor(1.0000)
```

```
def softmax_naive(x):
    return torch.exp(x) / torch.exp(x).sum(dim=0)

softmax_naive(attn_scores_2)
```

```
tensor([0.1385, 0.2379, 0.2333, 0.1240, 0.1082, 0.1581])
```

```
#This is the generally used way instead of making manual function
attn_weights_2 = torch.softmax(attn_scores_2, dim=0)

for i,x_i in enumerate(inputs):
    print(i, inputs[i])
```

```
0 tensor([0.4300, 0.1500, 0.8900])
1 tensor([0.5500, 0.8700, 0.6600])
2 tensor([0.5700, 0.8500, 0.6400])
3 tensor([0.2200, 0.5800, 0.3300])
4 tensor([0.7700, 0.2500, 0.1000])
5 tensor([0.0500, 0.8000, 0.5500])
```

```
query = inputs[1] #2nd input taken as the query
context_vec_2 = torch.zeros(query.shape)
for i,x_i in enumerate(inputs):
    print(f"{attn_weights_2[i]} ---> {inputs[i]}")
```

```
0.13854756951332092 ---> tensor([0.4300, 0.1500, 0.8900])
0.2378913015127182 ---> tensor([0.5500, 0.8700, 0.6600])
0.23327402770519257 ---> tensor([0.5700, 0.8500, 0.6400])
0.12399158626794815 ---> tensor([0.2200, 0.5800, 0.3300])
0.10818186402320862 ---> tensor([0.7700, 0.2500, 0.1000])
0.15811361372470856 ---> tensor([0.0500, 0.8000, 0.5500])
```

```
query = inputs[1] #2nd input taken as the query
context_vec_2 = torch.zeros(query.shape)
for i,x_i in enumerate(inputs):
    context_vec_2 += attn_weights_2[i]*inputs[i]
print (context_vec_2)
```

```
tensor([0.4419, 0.6515, 0.5683])
```

```
inputs = torch.tensor(
    [[0.43, 0.15, 0.89], # Your (x^1)
     [0.55, 0.87, 0.66], # journey (x^2)
     [0.57, 0.85, 0.64], # starts (x^3)
     [0.22, 0.58, 0.33], # with (x^4)
     [0.77, 0.25, 0.10], # one (x^5)
     [0.05, 0.80, 0.55]] # step (x^6)
)

#Continuation of previous section but unlike focusing on one query it
will be generalized
attn_scores = torch.empty(6, 6)
for i, x_i in enumerate(inputs):
    for j, x_j in enumerate(inputs):
        attn_scores[i,j] = torch.dot(x_i, x_j)
print(attn_scores)
```

```
tensor([[0.9995, 0.9544, 0.9422, 0.4753, 0.4576, 0.6310],
        [0.9544, 1.4950, 1.4754, 0.8434, 0.7070, 1.0865],
        [0.9422, 1.4754, 1.4570, 0.8296, 0.7154, 1.0605],
        [0.4753, 0.8434, 0.8296, 0.4937, 0.3474, 0.6565],
        [0.4576, 0.7070, 0.7154, 0.3474, 0.6654, 0.2935],
        [0.6310, 1.0865, 1.0605, 0.6565, 0.2935, 0.9450]])
```

```
#Alternative way using matrix multiplication
attn_scores = inputs @ inputs.T
attn_scores
```

```
tensor([[0.9995, 0.9544, 0.9422, 0.4753, 0.4576, 0.6310],
        [0.9544, 1.4950, 1.4754, 0.8434, 0.7070, 1.0865],
        [0.9422, 1.4754, 1.4570, 0.8296, 0.7154, 1.0605],
```

```
        [0.4753, 0.8434, 0.8296, 0.4937, 0.3474, 0.6565],
        [0.4576, 0.7070, 0.7154, 0.3474, 0.6654, 0.2935],
        [0.6310, 1.0865, 1.0605, 0.6565, 0.2935, 0.9450]])

#Calculating attention weights from attention scores by normalizing
them
attn_weights = torch.softmax(attn_scores, dim=1)
attn_weights

tensor([[0.2098, 0.2006, 0.1981, 0.1242, 0.1220, 0.1452],
        [0.1385, 0.2379, 0.2333, 0.1240, 0.1082, 0.1581],
        [0.1390, 0.2369, 0.2326, 0.1242, 0.1108, 0.1565],
        [0.1435, 0.2074, 0.2046, 0.1462, 0.1263, 0.1720],
        [0.1526, 0.1958, 0.1975, 0.1367, 0.1879, 0.1295],
        [0.1385, 0.2184, 0.2128, 0.1420, 0.0988, 0.1896]])

attn_weights.sum(dim=1)

tensor([1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 1.0000])

#Now we can see all we have done till now in a very small reduced code
version
attn_scores = inputs @ inputs.T
attn_weights = torch.softmax(attn_scores, dim=1)
attn_weights

tensor([[0.2098, 0.2006, 0.1981, 0.1242, 0.1220, 0.1452],
        [0.1385, 0.2379, 0.2333, 0.1240, 0.1082, 0.1581],
        [0.1390, 0.2369, 0.2326, 0.1242, 0.1108, 0.1565],
        [0.1435, 0.2074, 0.2046, 0.1462, 0.1263, 0.1720],
        [0.1526, 0.1958, 0.1975, 0.1367, 0.1879, 0.1295],
        [0.1385, 0.2184, 0.2128, 0.1420, 0.0988, 0.1896]])

all_context_vecs = attn_weights @ inputs
all_context_vecs

tensor([[0.4421, 0.5931, 0.5790],
        [0.4419, 0.6515, 0.5683],
        [0.4431, 0.6496, 0.5671],
        [0.4304, 0.6298, 0.5510],
        [0.4671, 0.5910, 0.5266],
        [0.4177, 0.6503, 0.5645]])

#so now the final code with all work done from start to end
attn_scores = inputs @ inputs.T
attn_weights = torch.softmax(attn_scores, dim=1)
all_context_vecs = attn_weights @ inputs
all_context_vecs

tensor([[0.4421, 0.5931, 0.5790],
        [0.4419, 0.6515, 0.5683],
```

```
        [0.4431, 0.6496, 0.5671],
        [0.4304, 0.6298, 0.5510],
        [0.4671, 0.5910, 0.5266],
        [0.4177, 0.6503, 0.5645]])

#Training Self atetention with trainable weights

inputs

tensor([[0.4300, 0.1500, 0.8900],
        [0.5500, 0.8700, 0.6600],
        [0.5700, 0.8500, 0.6400],
        [0.2200, 0.5800, 0.3300],
        [0.7700, 0.2500, 0.1000],
        [0.0500, 0.8000, 0.5500]])

x_2 = inputs[1]
d_in = inputs.shape[1]
d_out = 2

x_2

tensor([0.5500, 0.8700, 0.6600])

d_in

3

torch.manual_seed(123)

W_query = torch.nn.Parameter(torch.rand(d_in, d_out))
W_query #requires training unfortunately which will come in next parts

Parameter containing:
tensor([[0.2961, 0.5166],
        [0.2517, 0.6886],
        [0.0740, 0.8665]], requires_grad=True)

W_key = torch.nn.Parameter(torch.rand(d_in, d_out))
W_value = torch.nn.Parameter(torch.rand(d_in, d_out))

query_2 = x_2 @ W_query

query_2

tensor([0.4306, 1.4551], grad_fn=<SqueezeBackward4>)

keys  = inputs @ W_key
value = inputs @ W_value

keys.shape
value.shape
```

```
torch.Size([6, 2])
```

```
keys
```

```
tensor([[0.3669, 0.7646],
        [0.4433, 1.1419],
        [0.4361, 1.1156],
        [0.2408, 0.6706],
        [0.1827, 0.3292],
        [0.3275, 0.9642]], grad_fn=<MmBackward0>)
```

```
value
```

```
tensor([[0.1855, 0.8812],
        [0.3951, 1.0037],
        [0.3879, 0.9831],
        [0.2393, 0.5493],
        [0.1492, 0.3346],
        [0.3221, 0.7863]], grad_fn=<MmBackward0>)
```

```
keys_2 = keys[1]
attn_score_22 = torch.dot(query_2, keys_2)
attn_score_22
```

```
tensor(1.8524, grad_fn=<DotBackward0>)
```

```
attn_score_2 = query_2 @ keys.T
attn_score_2
```

```
tensor([1.2705, 1.8524, 1.8111, 1.0795, 0.5577, 1.5440],
       grad_fn=<SqueezeBackward4>)
```

```
d_k = keys.shape[1]
attn_weight_2 = torch.softmax(attn_score_2 / d_k**0.5, dim=-1)
attn_weight_2
```

```
tensor([0.1500, 0.2264, 0.2199, 0.1311, 0.0906, 0.1820],
       grad_fn=<SoftmaxBackward0>)
```

```
torch.sum(attn_weight_2)
```

```
tensor(1., grad_fn=<SumBackward0>)
```

```
context_vec_2 = attn_weight_2 @ value
```

```
context_vec_2
```

```
tensor([0.3061, 0.8210], grad_fn=<SqueezeBackward4>)
```

```
#Implementing a compact self attention class
```

```
import torch.nn as nn
```

```
class SelfAttention_v1(nn.Module):

    def __init__(self, d_in, d_out):
        super().__init__()
        self.W_query = torch.nn.Parameter(torch.rand(d_in, d_out))
        self.W_key = torch.nn.Parameter(torch.rand(d_in, d_out))
        self.W_value = torch.nn.Parameter(torch.rand(d_in, d_out))

    def forward(self, x):
        queries = inputs @ W_query
        keys = inputs @ W_key
        values = inputs @ W_value
        attn_scores = queries @ keys.T
        attn_weights = torch.softmax(attn_scores/ d_k**0.5, dim=-1)
        context_vec = attn_weights @ values
        return context_vec

torch.manual_seed(123)
sa_v1 = SelfAttention_v1(d_in, d_out)

sa_v1(inputs)
```

```
tensor([[0.2996, 0.8053],
        [0.3061, 0.8210],
        [0.3058, 0.8203],
        [0.2948, 0.7939],
        [0.2927, 0.7891],
        [0.2990, 0.8040]], grad_fn=<MmBackward0>)
```

```
#Implementing a compact self attention class version 2
#Gives better weight initialization

import torch.nn as nn

class SelfAttention_v2(nn.Module):

    def __init__(self, d_in, d_out, qkv_bias=False):
        super().__init__()
        self.W_query = torch.nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = torch.nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = torch.nn.Linear(d_in, d_out, bias=qkv_bias)

    def forward(self, x):
        queries = self.W_query(inputs)
        keys = self.W_key(inputs)
        values = self.W_value(inputs)
        attn_scores = queries @ keys.T
        attn_weights = torch.softmax(attn_scores/ d_k**0.5, dim=-1)
        context_vec = attn_weights @ values
        return context_vec
```

```
torch.manual_seed(123)
sa_v2 = SelfAttention_v2(d_in, d_out)

sa_v2(inputs) #Different output because different way of weight
initialization
```

```
tensor([[-0.5337, -0.1051],
        [-0.5323, -0.1080],
        [-0.5323, -0.1079],
        [-0.5297, -0.1076],
        [-0.5311, -0.1066],
        [-0.5299, -0.1081]], grad_fn=<MmBackward0>)
```

```
#Hiding future words with causal attention
#Modification to self attention mechanism to avoid future words
#Applying a causal attention mask
#Your journey starts with one step

queries = sa_v2.W_query(inputs)
keys = sa_v2.W_key(inputs)
values = sa_v2.W_value(inputs)
attn_scores = queries @ keys.T
attn_weights = torch.softmax(attn_scores/ d_k**0.5, dim=-1)

attn_weights
```

```
tensor([[0.1717, 0.1762, 0.1761, 0.1555, 0.1627, 0.1579],
        [0.1636, 0.1749, 0.1746, 0.1612, 0.1605, 0.1652],
        [0.1637, 0.1749, 0.1746, 0.1611, 0.1606, 0.1651],
        [0.1636, 0.1704, 0.1702, 0.1652, 0.1632, 0.1674],
        [0.1667, 0.1722, 0.1721, 0.1618, 0.1633, 0.1639],
        [0.1624, 0.1709, 0.1706, 0.1654, 0.1625, 0.1682]],
       grad_fn=<SoftmaxBackward0>)
```

```
context_length = attn_scores.shape[0]
mask_simple = torch.tril(torch.ones(context_length, context_length))
print(mask_simple)
```

```
tensor([[1., 0., 0., 0., 0., 0.],
        [1., 1., 0., 0., 0., 0.],
        [1., 1., 1., 0., 0., 0.],
        [1., 1., 1., 1., 0., 0.],
        [1., 1., 1., 1., 1., 0.],
        [1., 1., 1., 1., 1., 1.]])
```

```
masked_simple = attn_weights * mask_simple
masked_simple
```

```
tensor([[0.1717, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.1636, 0.1749, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.1637, 0.1749, 0.1746, 0.0000, 0.0000, 0.0000],
```

```
        [0.1636, 0.1704, 0.1702, 0.1652, 0.0000, 0.0000],
        [0.1667, 0.1722, 0.1721, 0.1618, 0.1633, 0.0000],
        [0.1624, 0.1709, 0.1706, 0.1654, 0.1625, 0.1682]],
       grad_fn=<MulBackward0>)
```

```
row_sums = masked_simple.sum(dim=-1, keepdim=True)
masked_simple_norm = masked_simple / row_sums
print(masked_simple_norm)
```

```
tensor([[1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.4833, 0.5167, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.3190, 0.3408, 0.3402, 0.0000, 0.0000, 0.0000],
        [0.2445, 0.2545, 0.2542, 0.2468, 0.0000, 0.0000],
        [0.1994, 0.2060, 0.2058, 0.1935, 0.1953, 0.0000],
        [0.1624, 0.1709, 0.1706, 0.1654, 0.1625, 0.1682]],
       grad_fn=<DivBackward0>)
```

```
#Another way to normalize
mask = torch.triu(torch.ones(context_length, context_length),
diagonal=1)
masked = attn_scores.masked_fill(mask.bool(), -torch.inf)
print(masked)
```

```
tensor([[0.3111,   -inf,   -inf,   -inf,   -inf,   -inf],
        [0.1655, 0.2602,   -inf,   -inf,   -inf,   -inf],
        [0.1667, 0.2602, 0.2577,   -inf,   -inf,   -inf],
        [0.0510, 0.1080, 0.1064, 0.0643,   -inf,   -inf],
        [0.1415, 0.1875, 0.1863, 0.0987, 0.1121,   -inf],
        [0.0476, 0.1192, 0.1171, 0.0731, 0.0477, 0.0966]],
       grad_fn=<MaskedFillBackward0>)
```

```
torch.exp(torch.tensor(-999999999))
```

```
tensor(0.)
```

```
torch.exp(torch.tensor(float("-inf")))
```

```
tensor(0.)
```

```
attn_weights = torch.softmax(masked / d_k**0.5, dim=-1)
print(attn_weights)
```

```
tensor([[1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.4833, 0.5167, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.3190, 0.3408, 0.3402, 0.0000, 0.0000, 0.0000],
        [0.2445, 0.2545, 0.2542, 0.2468, 0.0000, 0.0000],
        [0.1994, 0.2060, 0.2058, 0.1935, 0.1953, 0.0000],
        [0.1624, 0.1709, 0.1706, 0.1654, 0.1625, 0.1682]],
       grad_fn=<SoftmaxBackward0>)
```

```
#Masking additional attention weights with dropout
#Used to remove overfitting so model relies rest on certain positions
```

```
torch.manual_seed(123)
layer = torch.nn.Dropout(0.5) #0.5 means drops 50% of positions

example = torch.ones(6,6)
example
```

```
tensor([[1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1.]])
```

```
layer(example)
```

```
tensor([[2., 2., 0., 2., 2., 0.],
        [0., 0., 0., 2., 0., 2.],
        [2., 2., 2., 2., 0., 2.],
        [0., 2., 2., 0., 0., 2.],
        [0., 2., 0., 2., 0., 2.],
        [0., 2., 2., 2., 2., 0.]])
```

```
dropout_rate = 0.5
1 / (1-dropout_rate)
```

```
2.0
```

```
layer(attn_weights)
```

```
tensor([[2.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000],
        [0.0000, 0.6816, 0.6804, 0.0000, 0.0000, 0.0000],
        [0.0000, 0.0000, 0.5085, 0.4936, 0.0000, 0.0000],
        [0.0000, 0.0000, 0.0000, 0.0000, 0.3906, 0.0000],
        [0.3249, 0.3418, 0.0000, 0.3308, 0.3249, 0.3363]],
       grad_fn=<MulBackward0>)
```

```
#Implementing a compact causal self attention class

batch = torch.stack((inputs,inputs), dim=0)
batch.shape
```

```
torch.Size([2, 6, 3])
```

```
import torch
import torch.nn as nn

class CausalAttention(nn.Module):

    def __init__(self, d_in, d_out, context_length, dropout,
qkv_bias=False):
```

```
        super().__init__()
        self.d_out = d_out
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.dropout = nn.Dropout(dropout)
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
diagonal=1)
        )

    def forward(self, x):
        b, num_tokens, d_in = x.shape

        queries = self.W_query(x)
        keys = self.W_key(x)
        values = self.W_value(x)

        # Correctly compute attention scores via matrix multiplication
        attn_scores = torch.matmul(queries, keys.transpose(-2, -1))

        attn_scores.masked_fill_(
            self.mask[:num_tokens, :num_tokens].bool(), -torch.inf)

        attn_weights = torch.softmax(attn_scores / self.d_out**0.5,
dim=-1)
        attn_weights = self.dropout(attn_weights)

        context_vec = torch.matmul(attn_weights, values)

        return context_vec

batch.shape
```

```
torch.Size([2, 6, 3])
```

```
batch.shape[1]
```

```
6
```

```
torch.manual_seed(789)
context_length = batch.shape[1]
dropout = 0.0
ca = CausalAttention(d_in, d_out, context_length, dropout)
ca(batch)
```

```
tensor([[[-0.0872,  0.0286],
         [-0.0991,  0.0501],
         [-0.0999,  0.0633],
         [-0.0983,  0.0489],
```

```
         [-0.0514,  0.1098],
         [-0.0754,  0.0693]],

        [[-0.0872,  0.0286],
         [-0.0991,  0.0501],
         [-0.0999,  0.0633],
         [-0.0983,  0.0489],
         [-0.0514,  0.1098],
         [-0.0754,  0.0693]]], grad_fn=<UnsafeViewBackward0>)
```

```
batch
```

```
tensor([[[0.4300, 0.1500, 0.8900],
         [0.5500, 0.8700, 0.6600],
         [0.5700, 0.8500, 0.6400],
         [0.2200, 0.5800, 0.3300],
         [0.7700, 0.2500, 0.1000],
         [0.0500, 0.8000, 0.5500]],

        [[0.4300, 0.1500, 0.8900],
         [0.5500, 0.8700, 0.6600],
         [0.5700, 0.8500, 0.6400],
         [0.2200, 0.5800, 0.3300],
         [0.7700, 0.2500, 0.1000],
         [0.0500, 0.8000, 0.5500]]])
```

```
#Extending single head attention to multi head attention

#stacking multiple single head attention layers

class MultiHeadAttentionWrapper(nn.Module):
    def __init__(self, d_in, d_out, context_length, dropout,
num_heads=2, qkv_bias=False):
        super().__init__()
        self.heads = nn.ModuleList({
            CausalAttention(d_in, d_out, context_length, dropout,
qkv_bias) for _ in range(num_heads)
        })

    def forward(self, x):
        return torch.cat([head(x) for head in self.heads], dim=-1)

torch.manual_seed(123)
context_length = batch.shape[1]
d_in, d_out = 3,2
mha = MultiHeadAttentionWrapper(d_in, d_out, context_length,
dropout=0.0, num_heads=2)
mha(batch)
```

```
tensor([[[-0.4519,  0.2216,  0.4772,  0.1063],
         [-0.5874,  0.0058,  0.5891,  0.3257],
```

```
         [-0.6300, -0.0632,  0.6202,  0.3860],
         [-0.5675, -0.0843,  0.5478,  0.3589],
         [-0.5526, -0.0981,  0.5321,  0.3428],
         [-0.5299, -0.1081,  0.5077,  0.3493]],

        [[-0.4519,  0.2216,  0.4772,  0.1063],
         [-0.5874,  0.0058,  0.5891,  0.3257],
         [-0.6300, -0.0632,  0.6202,  0.3860],
         [-0.5675, -0.0843,  0.5478,  0.3589],
         [-0.5526, -0.0981,  0.5321,  0.3428],
         [-0.5299, -0.1081,  0.5077,  0.3493]]],
grad_fn=<CatBackward0>)

#Implementing multi head attention with weight splits

class MultiHeadAttention(nn.Module):
    def __init__(self, d_in, d_out, context_length, dropout,
num_heads, qkv_bias=False):
        super().__init__()
        assert( d_out % num_heads == 0 ), \
            "d_out must be divisible by num_heads"
        self.d_out = d_out
        self.num_heads = num_heads
        self.head_dim = d_out // num_heads
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.out_proj = nn.Linear(d_out, d_out)
        self.dropout = nn.Dropout(dropout)
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
                       diagonal=1)
        )

    def forward(self, x):
        b, num_tokens, d_in = x.shape
        #Shape: (b, num_tokens, d_out)
        keys = self.W_key(x)
        queries = self.W_query(x)
        values = self.W_value(x)
        #We implicitly split the matrix by adding a num_heads
dimension
        #Unroll last dim: (b, num_tokens, d_out) -> (b, num_tokens,
num_heads, head_dim)
        keys = keys.view(b, num_tokens, self.num_heads, self.head_dim)
        values = values.view(b, num_tokens, self.num_heads,
self.head_dim)
        queries = queries.view(b, num_tokens, self.num_heads,
self.head_dim)
```

```
        # Transpose: (b, num_tokens, num_heads, head_dim) -> (b,
num_heads, num_tokens, head_dim)
        keys = keys.transpose(1, 2)
        queries = queries.transpose(1, 2)
        values = values.transpose(1, 2)
        #Compute scaled dot product attention (aka self attention)
with a causal mask
        attn_scores = queries @ keys.transpose(2, 3) #Dot product for
each head
        #Original mask truncated to the number of tokens and converted
to boolean
        mask_bool = self.mask.bool()[:num_tokens, :num_tokens]
        #Use the mask to fill attention scores
        attn_scores.masked_fill_(mask_bool, -torch.inf)
        attn_weights = torch.softmax(attn_scores / keys.shape[-
1]**0.5, dim=-1)
        attn_weights = self.dropout(attn_weights)
        #Shape: (b, num_tokens, num_heads, head_dim)
        context_vec = (attn_weights @ values).transpose(1, 2)
        #Combine heads, where self.d_out = elf.num_heads *
self.head_dim
        context_vec = context_vec.contiguous().view(b, num_tokens,
self.d_out)
        context_vec = self.out_proj(context_vec) #Optional Projection
        return context_vec

torch.manual_seed(123)
batch_size, context_length, d_in = batch.shape
d_out = 4
mha = MultiHeadAttention(d_in, d_out, context_length, 0.0,
num_heads=2)
context_vecs = mha(batch)
print(context_vecs)
print("context_vecs.shape:", context_vecs.shape)
```

```
tensor([[[ 0.1184,  0.3120, -0.0847, -0.5774],
         [ 0.0178,  0.3221, -0.0763, -0.4225],
         [-0.0147,  0.3259, -0.0734, -0.3721],
         [-0.0116,  0.3138, -0.0708, -0.3624],
         [-0.0117,  0.2973, -0.0698, -0.3543],
         [-0.0132,  0.2990, -0.0689, -0.3490]],

        [[ 0.1184,  0.3120, -0.0847, -0.5774],
         [ 0.0178,  0.3221, -0.0763, -0.4225],
         [-0.0147,  0.3259, -0.0734, -0.3721],
         [-0.0116,  0.3138, -0.0708, -0.3624],
         [-0.0117,  0.2973, -0.0698, -0.3543],
         [-0.0132,  0.2990, -0.0689, -0.3490]]],
grad_fn=<ViewBackward0>)
context_vecs.shape: torch.Size([2, 6, 4])
```

```python
#Implementing a GPT Model from scratch to generate text
#Coding an LLM architecture

GPT_CONFIG_124M = {
    "vocab_size": 50257, #Vocabulary size
    "context_length": 1024, #Context length
    "emb_dim": 768, #Embedding Dimension
    "n_heads": 12, #Number of attention heads
    "n_layers": 12, #Number of layers
    "drop_rate": 0.1, #Dropout rate
    "qkv_bias": False #Query-Key-Value bias
}

import torch
import torch.nn as nn

class DummyGPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"],
cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

        #Use a placeholder for Transformer Block
        self.trf_blocks = nn.Sequential(
            *[DummyTransformerBlock(cfg) for _ in
range(cfg["n_layers"])])

        #Use a placeholder for LayerNorm
        self.final_norm = DummyLayerNorm(cfg["emb_dim"])
        self.out_head = nn.Linear(
            cfg["emb_dim"], cfg["vocab_size"], bias=False
        )

    def forward(self, in_idx):
        batch_size, seq_len = in_idx.shape
        tok_embeds = self.tok_emb(in_idx)
        pos_embeds = self.pos_emb(torch.arange(seq_len,
device=in_idx.device))
        x = tok_embeds + pos_embeds
        x = self.drop_emb(x)
        x = self.trf_blocks(x)
        x = self.final_norm(x)
        logits = self.out_head(x)
        return logits

class DummyTransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
```

```
        #A simple placeholder

    def forward(self, x):
        #This block does nothing and just returns its input
        return x

class DummyLayerNorm(nn.Module):
    def __init__(self, normalized_shape):
        super().__init__()
        # This is just a placeholder, so __init__ can be simple

    def forward(self, x):
        # It doesn't perform any operation, just returns the input
        return x

import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")

batch = []

txt1 = "Every effort moves you"
txt2 = "Every day holds a"

batch.append(torch.tensor(tokenizer.encode(txt1)))
batch.append(torch.tensor(tokenizer.encode(txt2)))
batch = torch.stack(batch, dim=0)
print(batch)

tensor([[6109, 3626, 6100,  345],
        [6109, 1110, 6622,  257]])

torch.manual_seed(123)
model = DummyGPTModel(cfg=GPT_CONFIG_124M)
logits = model(batch)
print("Output Shape:", logits.shape)
print(logits)

Output Shape: torch.Size([2, 4, 50257])
tensor([[[-1.2034,  0.3201, -0.7130,  ..., -1.5548, -0.2390, -0.4667],
         [-0.1192,  0.4539, -0.4432,  ...,  0.2392,  1.3469,  1.2430],
         [ 0.5307,  1.6720, -0.4695,  ...,  1.1966,  0.0111,  0.5835],
         [ 0.0139,  1.6754, -0.3388,  ...,  1.1586, -0.0435, -
1.0400]],

        [[-1.0908,  0.1798, -0.9484,  ..., -1.6047,  0.2439, -0.4530],
         [-0.7860,  0.5581, -0.0610,  ...,  0.4835, -0.0077,  1.6621],
         [ 0.3567,  1.2698, -0.6398,  ..., -0.0162, -0.1296,  0.3717],
         [-0.2407, -0.7349, -0.5102,  ...,  2.0057, -0.3694,
0.1814]]],
       grad_fn=<UnsafeViewBackward0>)
```

```
#Normalizing Activations with layer normalization

torch.manual_seed(123)

batch_sample = torch.rand(2, 5)
batch_sample

tensor([[0.2961, 0.5166, 0.2517, 0.6886, 0.0740],
        [0.8665, 0.1366, 0.1025, 0.1841, 0.7264]])

layer = nn.Sequential(nn.Linear(5, 6), nn.ReLU())
out = layer(batch_sample)
out

tensor([[0.0000, 0.0000, 0.4091, 0.6587, 0.3914, 0.0000],
        [0.0000, 0.0000, 0.1902, 0.3182, 0.6486, 0.0000]],
       grad_fn=<ReluBackward0>)

out.mean()

tensor(0.2180, grad_fn=<MeanBackward0>)

mean = out.mean(dim=-1, keepdim= True) #Calculates for each row
mean

tensor([[0.2432],
        [0.1928]], grad_fn=<MeanBackward1>)

var = out.var(dim=-1, keepdim= True)
var

tensor([[0.0799],
        [0.0670]], grad_fn=<VarBackward0>)

(out - mean).mean(dim=-1)

tensor([-1.9868e-08, -2.4835e-09], grad_fn=<MeanBackward1>)

torch.set_printoptions(sci_mode=False)

(out - mean).mean(dim=-1)

tensor([    -0.0000,     -0.0000], grad_fn=<MeanBackward1>)

normed = ((out - mean) / torch. sqrt(var))
normed.var(dim=-1, keepdim= True)

tensor([[1.0000],
        [1.0000]], grad_fn=<VarBackward0>)

class LayerNorm(nn.Module):
    def __init__(self, emb_dim):
        super().__init__()
```

```
        self.eps = 1e-5
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim))

    def forward(self, x):
        mean = x.mean(dim=-1, keepdim=True)
        var = x.var(dim=-1, keepdim=True, unbiased=False)
        norm_x = (x-mean) / torch.sqrt(var + self.eps)
        return self.scale * norm_x + self.shift

torch.manual_seed(123)
ln = LayerNorm(6)
outputs_normed = ln(out)

outputs_normed
```

```
tensor([[-0.9423, -0.9423,  0.6428,  1.6100,  0.5742, -0.9423],
        [-0.8160, -0.8160, -0.0111,  0.5306,  1.9286, -0.8160]],
       grad_fn=<AddBackward0>)
```

```
outputs_normed.mean(dim=-1, keepdim=True)
```

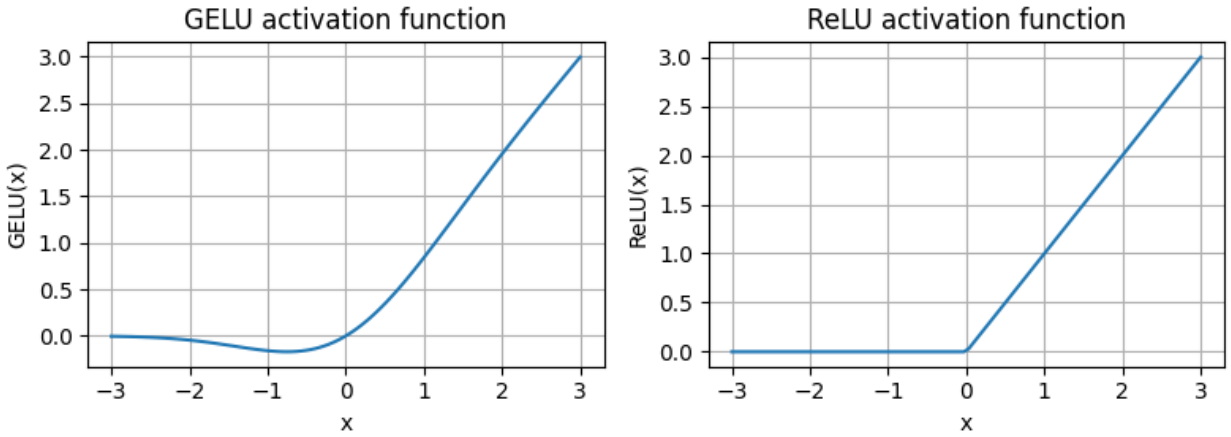
```
tensor([[    -0.0000],
        [    -0.0000]], grad_fn=<MeanBackward1>)
```

```
outputs_normed.var(dim=-1, keepdim=True) #It will come 1 if the
dimensions are very large unlike our example
```

```
tensor([[1.1998],
        [1.1998]], grad_fn=<VarBackward0>)
```

```
#Implementing a feedforward network with GELU activations

class GELU(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        return 0.5 * x * (1 + torch.tanh(
                torch.sqrt(torch.tensor(2.0 / torch.pi)) *
                (x + 0.044715 * torch.pow(x, 3))
        ))

import matplotlib.pyplot as plt

gelu, relu = GELU(), nn.ReLU()

#Some sample data
x = torch.linspace(-3, 3, 100)
y_gelu, y_relu = gelu(x), relu(x)

plt.figure(figsize=(8,3))
```

```
for i, (y,label) in enumerate(zip([y_gelu, y_relu], ["GELU", "ReLU"]),
1):
    plt.subplot(1, 2, i)
    plt.plot(x, y)
    plt.title(f"{label} activation function")
    plt.xlabel("x")
    plt.ylabel(f"{label}(x)")
    plt.grid(True)

plt.tight_layout()
plt.show()
```




```
class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(cfg["emb_dim"], 4 * cfg["emb_dim"]),
            GELU(),
            nn.Linear(4 * cfg["emb_dim"], cfg["emb_dim"]),
        )

    def forward(self, x):
        return self.layers(x)

ffn = FeedForward(GPT_CONFIG_124M)

x = torch.rand(2, 3, 768)
ffn(x).shape
```

```
torch.Size([2, 3, 768])
```

```
ffn.layers
```

```
Sequential(
  (0): Linear(in_features=768, out_features=3072, bias=True)
  (1): GELU()
```

```
  (2): Linear(in_features=3072, out_features=768, bias=True)
)

ffn.layers[0]

Linear(in_features=768, out_features=3072, bias=True)

ffn.layers[0].weight

Parameter containing:
tensor([[-0.0147,  0.0012, -0.0179,  ...,  0.0328,  0.0257, -0.0029],
        [-0.0199, -0.0141, -0.0143,  ...,  0.0034, -0.0034,  0.0187],
        [ 0.0140, -0.0182, -0.0064,  ...,  0.0277, -0.0007,  0.0069],
        ...,
        [-0.0214,  0.0018,  0.0189,  ...,  0.0037, -0.0348, -0.0303],
        [ 0.0082, -0.0077,  0.0067,  ..., -0.0244, -0.0194, -0.0247],
        [-0.0186, -0.0195,  0.0010,  ...,  0.0210, -0.0041,  0.0300]],
       requires_grad=True)

#Adding Shortcut Connections

from torch.nn import GELU

class ExampleDeepNeuralNetworks(nn.Module):
    def __init__(self, layer_sizes, use_shortcut):
        super().__init__()
        self.use_shortcut = use_shortcut
        self.layers = nn.ModuleList([
            nn.Sequential(nn.Linear(layer_sizes[0], layer_sizes[1]),
GELU()),
            nn.Sequential(nn.Linear(layer_sizes[1], layer_sizes[2]),
GELU()),
            nn.Sequential(nn.Linear(layer_sizes[2], layer_sizes[3]),
GELU()),
            nn.Sequential(nn.Linear(layer_sizes[3], layer_sizes[4]),
GELU()),
            nn.Sequential(nn.Linear(layer_sizes[4], layer_sizes[5]),
GELU())
        ])

    def forward(self, x):
        for layer in self.layers:
            #Compute the output of the current layer
            layer_output = layer(x)
            #Check if shortcut can be applied
            if self.use_shortcut and x.shape == layer_output.shape:
                x = x + layer_output
            else:
                x = layer_output
        return x
```

```
def print_gradients(model, x):
    #Forward pass
    output = model(x)
    target = torch.tensor([[0.]])

    #Calculate loss based on how close the target
    # and output are
    loss = nn.MSELoss()
    loss = loss(output, target)

    #Backward pass to calculate the gradients
    loss.backward()

    for name, param in model.named_parameters():
        if 'weight' in name:
            #Print the mean absolute gradient of the weights
            print(f"{name} has gradient mean of
{param.grad.abs().mean().item()}")

torch.manual_seed(123)
layer_sizes = [1, 20, 20, 20, 20, 1]
sample_input = torch.tensor([[2.0]])
model_without_shortcut = ExampleDeepNeuralNetworks(
    layer_sizes, use_shortcut=True
)
print_gradients(model_without_shortcut, sample_input)
```

```
layers.0.0.weight has gradient mean of 0.019939452409744263
layers.1.0.weight has gradient mean of 0.0038494232576340437
layers.2.0.weight has gradient mean of 0.004893491510301828
layers.3.0.weight has gradient mean of 0.004734581336379051
layers.4.0.weight has gradient mean of 0.1044139489531517
```

```
#Connecting attention and linear layers in a transformer block

from previous_chapters import MultiHeadAttention

class TransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.att = MultiHeadAttention(
            d_in = cfg["emb_dim"],
            d_out = cfg["emb_dim"],
            context_length = cfg["context_length"],
            num_heads = cfg["n_heads"],
            dropout = cfg["drop_rate"],
            qkv_bias = cfg["qkv_bias"])
        self.ff = FeedForward(cfg)
        self.norm1 = LayerNorm(cfg["emb_dim"])
        self.norm2 = LayerNorm(cfg["emb_dim"])
        self.drop_shortcut = nn.Dropout(cfg["drop_rate"])
```

```
    def forward(self, x):
        #Shortcut connection for attention block
        shortcut = x
        x = self.norm1(x)
        x = self.att(x) #Shape [batch_size, num_tokens, emb_size]
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back

        #Shortcut connection for feed forward block
        shortcut = x
        x = self.norm2(x)
        x = self.ff(x)
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back

        return x

torch.manual_seed(123)

x = torch.rand(2, 4, 768)
block = TransformerBlock(GPT_CONFIG_124M)
output = block(x)

x.shape
```

```
torch.Size([2, 4, 768])
```

```
output.shape
```

```
torch.Size([2, 4, 768])
```

```
#Coding the GPT Model

import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")

batch = []

txt1 = "Every effort moves you"
txt2 = "Every day holds a"

batch.append(torch.tensor(tokenizer.encode(txt1)))
batch.append(torch.tensor(tokenizer.encode(txt2)))
batch = torch.stack(batch, dim=0)
print(batch)
```

```
tensor([[6109, 3626, 6100,  345],
        [6109, 1110, 6622,  257]])
```

```
class GPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"],
cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

        self.trf_blocks = nn.Sequential(
            *[TransformerBlock(cfg) for _ in range(cfg["n_layers"])])

        self.final_norm = LayerNorm(cfg["emb_dim"])
        self.out_head = nn.Linear(
            cfg["emb_dim"], cfg["vocab_size"], bias=False
        )

    def forward(self, in_idx):
        batch_size, seq_len = in_idx.shape
        tok_embeds = self.tok_emb(in_idx)
        pos_embeds = self.pos_emb(torch.arange(seq_len,
device=in_idx.device))
        x = tok_embeds + pos_embeds
        x = self.drop_emb(x)
        x = self.trf_blocks(x)
        x = self.final_norm(x)
        logits = self.out_head(x)
        return logits

torch.manual_seed(123)

model = GPTModel(GPT_CONFIG_124M)
out = model(batch)

model.tok_emb.weight.shape

torch.Size([50257, 768])

model.out_head.weight.shape

torch.Size([50257, 768])

batch.shape

torch.Size([2, 4])

out.shape

torch.Size([2, 4, 50257])

batch.numel()

8
```

```
total_params = sum(p.numel() for p in model.parameters())
total_params
```

```
163009536
```

```
print(f"{total_params - model.out_head.weight.numel():,}")
```

```
124,412,160
```

```
#Generating the text
```

```
start_context = "Hello, I am"

encoded = tokenizer.encode(start_context)
print("encoded", encoded)
```

```
encoded [15496, 11, 314, 716]
```

```
encoded_tensor = torch.tensor(encoded).unsqueeze(0)
print("encoded_tensor.shape:", encoded_tensor.shape)
```

```
encoded_tensor.shape: torch.Size([1, 4])
```

```
def generate_text_simple(model, idx, max_new_tokens, context_size):
    for _ in range(max_new_tokens):
        idx_cond = idx[:, -context_size:]

        with torch.no_grad():
            logits = model(idx_cond)
        logits = logits[:, -1, :]

        probas = torch.softmax(logits, dim=-1)

        idx_next = torch.argmax(probas, dim=-1, keepdim=True)

        idx = torch.cat((idx, idx_next), dim=1)

    return idx


torch.argmax(torch.tensor([14, 1, -1, 1, 15]))
```

```
tensor(4)
```

```
start_context = "Hello, I am"

encoded = tokenizer.encode(start_context)
print("encoded", encoded)
```

```
encoded [15496, 11, 314, 716]
```

```
encoded_tensor = torch.tensor(encoded).unsqueeze(0)
print("encoded_tensor.shape:", encoded_tensor.shape)
```

```
encoded_tensor.shape: torch.Size([1, 4])

encoded_tensor

tensor([[15496,    11,   314,   716]])

out = generate_text_simple(
    model=model,
    idx=encoded_tensor,
    max_new_tokens=6,
    context_size=GPT_CONFIG_124M["context_length"]
)

out

tensor([[15496,    11,   314,   716, 27018, 24086, 47843, 30961,
38891, 34320]])

tokenizer.decode(out) #Doesnt work because it is tensor not python
list

---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[72], line 1
----> 1 tokenizer.decode(out) #Doesnt work because it is tensor not
python list

File ~\OneDrive\Desktop\LLMs-from-scratch-main\.venv\Lib\site-
packages\tiktoken\core.py:284, in Encoding.decode(self, tokens,
errors)
    272 def decode(self, tokens: Sequence[int], errors: str =
"replace") -> str:
    273     """Decodes a list of tokens into a string.
    274
    275     WARNING: the default behaviour of this function is lossy,
since decoded bytes are not
   (...)    282     ```
    283     """
--> 284     return self._core_bpe.decode_bytes(tokens).decode("utf-8",
errors=errors)

TypeError: argument 'tokens': 'Tensor' object cannot be converted to
'Sequence'

(out.squeeze(0)).tolist()

[15496, 11, 314, 716, 27018, 24086, 47843, 30961, 38891, 34320]

tokenizer.decode((out.squeeze(0)).tolist())
```

```
'Hello, I am Featureiman Byeswick palpMust'
```

```
#Pretraining on unlabeled data
#Evaluating generative text models

import torch

device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Using device: {device}")

Using device: cuda

#Previous chapters imported
# CHAPTER 2
import tiktoken
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

class GPTDatasetV1(Dataset):
    def __init__(self, txt, tokenizer, max_length, stride):
        self.input_ids = []
        self.target_ids = []

        #Tokenize the entire text
        token_ids = tokenizer.encode(txt, allowed_special={"<|
endoftext|>"})

        #Use a sliding window to chunk the book into overlapping
sequences of max_length
        for i in range(0, len(token_ids) - max_length, stride):
            input_chunk = token_ids[i:i + max_length]
            target_chunk = token_ids[i+1: i + max_length + 1]
            self.input_ids.append(torch.tensor(input_chunk))
            self.target_ids.append(torch.tensor(target_chunk))

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self,idx):
        return self.input_ids[idx], self.target_ids[idx]

def create_dataloader_v1(txt, batch_size=4, max_length=256,
stride=128, shuffle=True, drop_last=True, num_workers=0):
    #Initialize the tokenizer
    tokenizer = tiktoken.get_encoding("gpt2")

    #Create dataset
    dataset = GPTDatasetV1(txt, tokenizer, max_length, stride)

    #Create dataloader
    dataloader = DataLoader(
    dataset, batch_size=batch_size, shuffle=shuffle,
```

```
drop_last=drop_last, num_workers=num_workers)

    return dataloader

#  CHAPTER 3

class MultiHeadAttention(nn.Module):
    def __init__(self, d_in, d_out, context_length, dropout,
num_heads, qkv_bias=False):
        super().__init__()
        assert( d_out % num_heads == 0 ), \
            "d_out must be divisible by num_heads"
        self.d_out = d_out
        self.num_heads = num_heads
        self.head_dim = d_out // num_heads
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.out_proj = nn.Linear(d_out, d_out)
        self.dropout = nn.Dropout(dropout)
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
                       diagonal=1)
        )

    def forward(self, x):
        b, num_tokens, d_in = x.shape
        #Shape: (b, num_tokens, d_out)
        keys = self.W_key(x)
        queries = self.W_query(x)
        values = self.W_value(x)
        #We implicitly split the matrix by adding a num_heads
dimension
        #Unroll last dim: (b, num_tokens, d_out) -> (b, num_tokens,
num_heads, head_dim)
        keys = keys.view(b, num_tokens, self.num_heads, self.head_dim)
        values = values.view(b, num_tokens, self.num_heads,
self.head_dim)
        queries = queries.view(b, num_tokens, self.num_heads,
self.head_dim)
        # Transpose: (b, num_tokens, num_heads, head_dim) -> (b,
num_heads, num_tokens, head_dim)
        keys = keys.transpose(1, 2)
        queries = queries.transpose(1, 2)
        values = values.transpose(1, 2)
        #Compute scaled dot product attention (aka self attention)
with a causal mask
        attn_scores = queries @ keys.transpose(2, 3) #Dot product for
each head
```

```
        #Original mask truncated to the number of tokens and converted
to boolean
        mask_bool = self.mask.bool()[:num_tokens, :num_tokens]
        #Use the mask to fill attention scores
        attn_scores.masked_fill_(mask_bool, -torch.inf)
        attn_weights = torch.softmax(attn_scores / keys.shape[-
1]**0.5, dim=-1)
        attn_weights = self.dropout(attn_weights)
        #Shape: (b, num_tokens, num_heads, head_dim)
        context_vec = (attn_weights @ values).transpose(1, 2)
        #Combine heads, where self.d_out = elf.num_heads *
self.head_dim
        context_vec = context_vec.contiguous().view(b, num_tokens,
self.d_out)
        context_vec = self.out_proj(context_vec) #Optional Projection
        return context_vec

# CHAPTER 4

class LayerNorm(nn.Module):
    def __init__(self, emb_dim):
        super().__init__()
        self.eps = 1e-5
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim))

    def forward(self, x):
        mean = x.mean(dim=-1, keepdim=True)
        var = x.var(dim=-1, keepdim=True, unbiased=False)
        norm_x = (x-mean) / torch.sqrt(var + self.eps)
        return self.scale * norm_x + self.shift

class GELU(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        return 0.5 * x * (1 + torch.tanh(
                torch.sqrt(torch.tensor(2.0 / torch.pi)) *
                (x + 0.044715 * torch.pow(x, 3))
        ))

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(cfg["emb_dim"], 4 * cfg["emb_dim"]),
            GELU(),
            nn.Linear(4 * cfg["emb_dim"], cfg["emb_dim"]),
        )
```

```
    def forward(self, x):
        return self.layers(x)

class TransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.att = MultiHeadAttention(
            d_in = cfg["emb_dim"],
            d_out = cfg["emb_dim"],
            context_length = cfg["context_length"],
            num_heads = cfg["n_heads"],
            dropout = cfg["drop_rate"],
            qkv_bias = cfg["qkv_bias"])
        self.ff = FeedForward(cfg)
        self.norm1 = LayerNorm(cfg["emb_dim"])
        self.norm2 = LayerNorm(cfg["emb_dim"])
        self.drop_shortcut = nn.Dropout(cfg["drop_rate"])

    def forward(self, x):
        #Shortcut connection for attention block
        shortcut = x
        x = self.norm1(x)
        x = self.att(x) #Shape [batch_size, num_tokens, emb_size]
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back

        #Shortcut connection for feed forward block
        shortcut = x
        x = self.norm2(x)
        x = self.ff(x)
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back
        return x

class GPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"],
cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

        self.trf_blocks = nn.Sequential(
            *[TransformerBlock(cfg) for _ in range(cfg["n_layers"])])

        self.final_norm = LayerNorm(cfg["emb_dim"])
        self.out_head = nn.Linear(
            cfg["emb_dim"], cfg["vocab_size"], bias=False
        )
```

```
    def forward(self, in_idx):
        batch_size, seq_len = in_idx.shape
        tok_embeds = self.tok_emb(in_idx)
        pos_embeds = self.pos_emb(torch.arange(seq_len,
device=in_idx.device))
        x = tok_embeds + pos_embeds
        x = self.drop_emb(x)
        x = self.trf_blocks(x)
        x = self.final_norm(x)
        logits = self.out_head(x)
        return logits

def generate_text_simple(model, idx, max_new_tokens, context_size):
     # Automatically move the input tensor to the model's device
(e.g., "cuda")
    device = next(model.parameters()).device
    idx = idx.to(device)

    for _ in range(max_new_tokens):
        idx_cond = idx[:, -context_size:]

        with torch.no_grad():
            logits = model(idx_cond)
        logits = logits[:, -1, :]

        probas = torch.softmax(logits, dim=-1)

        idx_next = torch.argmax(probas, dim=-1, keepdim=True)

        idx = torch.cat((idx, idx_next), dim=1)
    return idx

#Using GPT to generate text

from importlib.metadata import version

pkgs = {"matplotlib",
        "numpy",
        "tiktoken",
        "torch",
        "tensorflow" #For OpenAI's Pretrained weights
       }
for p in pkgs:
    print(f"{p} version: {version(p)}")
```

```
numpy version: 1.26.4
tensorflow version: 2.16.1
matplotlib version: 3.8.4
```

```
torch version: 2.5.1+cu121
tiktoken version: 0.11.0
```

```
GPT_CONFIG_124M = {
    "vocab_size": 50257, #Vocabulary size
    "context_length": 256, #Context length
    "emb_dim": 768, #Embedding Dimension
    "n_heads": 12, #Number of attention heads
    "n_layers": 12, #Number of layers
    "drop_rate": 0.1, #Dropout rate
    "qkv_bias": False #Query-Key-Value bias
}
```

```
import torch

torch.manual_seed(123)
model = GPTModel(GPT_CONFIG_124M)
model.eval() #Disabling dropout so no random dropping during inference
```

```
GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(256, 768)
  (drop_emb): Dropout(p=0.1, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
```

```
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
```

```
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
```

```
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
```

```
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
```

```
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)

device = "cuda"

model.to(device)
```

```
GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(256, 768)
  (drop_emb): Dropout(p=0.1, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (2): TransformerBlock(
```

```
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
```

```
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
```

```
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
```

```
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
```

```
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)

import torch

torch.manual_seed(123)
model = GPTModel(GPT_CONFIG_124M)
model.eval(); #Disabling dropout so no random dropping during
inference

import tiktoken

def text_to_token_ids(text, tokenizer):
    encoded = tokenizer.encode(text, allowed_special = {"<|endoftext|
>"})
    encoded_tensor = torch.tensor(encoded).unsqueeze(0) #Add bacth
dimension
    return encoded_tensor

start_context = "Every effort moves you"
tokenizer = tiktoken.get_encoding("gpt2")

token_ids = text_to_token_ids(start_context, tokenizer)
token_ids

tensor([[6109, 3626, 6100,  345]])

def token_ids_to_text(token_ids, tokenizer):
    flat = token_ids.squeeze(0) # remove batch dimension
    return tokenizer.decode(flat.tolist())
```

```
token_ids_to_text(token_ids, tokenizer)
```

```
'Every effort moves you'
```

```
token_ids.squeeze(0).shape
```

```
torch.Size([4])
```

```
token_ids = generate_text_simple(
    model=model,
    idx=text_to_token_ids(start_context, tokenizer),
    max_new_tokens=10,
    context_size=GPT_CONFIG_124M["context_length"]
)
```

```
token_ids.squeeze(0).shape
```

```
torch.Size([14])
```

```
token_ids_to_text(token_ids, tokenizer)
```

```
'Every effort moves you rentingetic wasnم refres RexMeCHicular stren'
```

```
#Calculating the text generation loss: cross-entropy and perplexity
```

```
inputs = torch.tensor([[16833, 3626, 6100], #every effort moves
                       [40, 1107, 588]])     #I really like
```

```
targets = torch.tensor([[3626, 6100, 345],  #effort moves you
                        [1107, 588, 11311]]) #really like chocolate
```

```
with torch.no_grad():
    logits = model(inputs)
```

```
logits.shape
```

```
torch.Size([2, 3, 50257])
```

```
probas = torch.softmax(logits, dim=-1)
probas.shape
```

```
torch.Size([2, 3, 50257])
```

```
probas
```

```
tensor([[[1.8849e-05, 1.5172e-05, 1.1687e-05,  ..., 2.2409e-05,
          6.9776e-06, 1.8776e-05],
         [9.1569e-06, 1.0062e-05, 7.8786e-06,  ..., 2.9090e-05,
          6.0103e-06, 1.3571e-05],
         [2.9877e-05, 8.8507e-06, 1.5741e-05,  ..., 3.5456e-05,
          1.4094e-05, 1.3526e-05]],
```

```
        [[1.2561e-05, 2.0538e-05, 1.4332e-05,  ..., 1.0389e-05,
          3.4784e-05, 1.4239e-05],
         [7.2731e-06, 1.7864e-05, 1.0565e-05,  ..., 2.1206e-05,
          1.1390e-05, 1.5559e-05],
         [2.9496e-05, 3.3605e-05, 4.1029e-05,  ..., 6.5249e-06,
          5.8203e-05, 1.3698e-05]]])

probas.sum()

tensor(6.0000)

token_ids = torch.argmax(probas, dim=-1, keepdim=True)
print("Token IDs:\n", token_ids)

Token IDs:
 tensor([[[16657],
         [  339],
         [42826]],

        [[49906],
         [29669],
         [41751]]])

print(f"Targets batch 1: {token_ids_to_text(targets[0], tokenizer)}")
print(f"Outputs batch 1: {token_ids_to_text(token_ids[0].flatten(),
tokenizer)}")

Targets batch 1:  effort moves you
Outputs batch 1:  Armed heNetflix

text_idx = 0
target_probas_1 = probas[text_idx, [0,1,2], targets[text_idx]]
print("Text 1:", target_probas_1)

Text 1: tensor([7.4540e-05, 3.1061e-05, 1.1563e-05])

targets[text_idx]

tensor([3626, 6100,  345])

text_idx = 1
target_probas_2 = probas[text_idx, [0,1,2], targets[text_idx]]
print("Text 2:", target_probas_2)

Text 2: tensor([1.0337e-05, 5.6776e-05, 4.7559e-06])

targets[text_idx]

tensor([ 1107,   588, 11311])

#Compute logarithm of all token probabilities
log_probas = torch.log(torch.cat((target_probas_1, target_probas_2)))
print(log_probas)
```

```
tensor([ -9.5042, -10.3796, -11.3677, -11.4798,  -9.7764, -12.2561])
```

```
torch.mean(log_probas)
```

```
tensor(-10.7940)
```

```
-1*torch.mean(log_probas) #Cross-Entropy loss , our aim is to brin it as close to 0 as we can during the training phase
```

```
tensor(10.7940)
```

```
logits.shape
```

```
torch.Size([2, 3, 50257])
```

```
logits_flat = logits.flatten(0, 1)
logits_flat.shape
```

```
torch.Size([6, 50257])
```

```
targets.shape
```

```
torch.Size([2, 3])
```

```
targets_flat = targets.flatten()
targets_flat.shape
```

```
torch.Size([6])
```

```
torch.nn.functional.cross_entropy(logits_flat, targets_flat)
```

```
tensor(10.7940)
```

```
#Calculating the training and validation set losses
```

```
import os
import urllib.request

file_path = "the-verdict-txt"
url =
"https://raw.githubusercontent.com/rasbt/LLMs-from-scratch/refs/heads/main/ch02/01_main-chapter-code/the-verdict.txt"

if not os.path.exists(file_path):
    with urllib.request.urlopen(url) as response:
        text_data = response.read().decode('utf-8')
    with open(file_path, "w", encoding="utf-8") as file:
        file.write(text_data)
else:
    with open(file_path, "r", encoding="utf-8") as file:
        text_data = file.read()

text_data[:99]
```

```
'I HAD always thought Jack Gisburn rather a cheap genius--though a good fellow enough--so it was no '
```

```
total_characters = len(text_data)
total_tokens = len(tokenizer.encode(text_data))

print("Characters:", total_characters)
print("Tokens:", total_tokens)
```

```
Characters: 20479
Tokens: 5145
```

```
tokenizer.encode(text_data)
```

```
[40,
 367,
 2885,
 1464,
 1807,
 3619,
 402,
 271,
 10899,
 2138,
 257,
 7026,
 15632,
 438,
 2016,
 257,
 922,
 5891,
 1576,
 438,
 568,
 340,
 373,
 645,
 1049,
 5975,
 284,
 502,
 284,
 3285,
 326,
 11,
 287,
 262,
 6001,
 286,
```

```
465,
13476,
11,
339,
550,
5710,
465,
12036,
11,
6405,
257,
5527,
27075,
11,
290,
4920,
2241,
287,
257,
4489,
64,
319,
262,
34686,
41976,
13,
357,
10915,
314,
2138,
1807,
340,
561,
423,
587,
10598,
393,
28537,
2014,
198,
198,
1,
464,
6001,
286,
465,
13476,
1,
438,
```

```
5562,
373,
644,
262,
1466,
1444,
340,
13,
314,
460,
3285,
9074,
13,
46606,
536,
5469,
438,
14363,
938,
4842,
1650,
353,
438,
2934,
489,
3255,
465,
48422,
540,
450,
67,
3299,
13,
366,
5189,
1781,
340,
338,
1016,
284,
3758,
262,
1988,
286,
616,
4286,
705,
1014,
510,
```

```
26,
475,
314,
836,
470,
892,
286,
326,
11,
1770,
13,
8759,
2763,
438,
1169,
2994,
284,
943,
17034,
318,
477,
314,
892,
286,
526,
383,
1573,
11,
319,
9074,
13,
536,
5469,
338,
11914,
11,
33096,
663,
4808,
3808,
62,
355,
996,
484,
547,
12548,
287,
281,
13079,
```

```
410,
12523,
286,
22353,
13,
843,
340,
373,
407,
691,
262,
9074,
13,
536,
48819,
508,
25722,
276,
13,
11161,
407,
262,
40123,
18113,
544,
9325,
701,
11,
379,
262,
938,
402,
1617,
261,
12917,
905,
11,
5025,
502,
878,
402,
271,
10899,
338,
366,
31640,
12,
67,
20811,
```

```
1,
284,
910,
11,
351,
10953,
287,
607,
2951,
25,
366,
1135,
2236,
407,
804,
2402,
663,
588,
757,
13984,
198,
198,
5779,
28112,
10197,
832,
262,
46475,
286,
18113,
544,
338,
10953,
314,
2936,
1498,
284,
1986,
262,
1109,
351,
1602,
11227,
414,
13,
23676,
3619,
402,
271,
```

```
10899,
0,
383,
1466,
550,
925,
683,
438,
270,
373,
15830,
326,
484,
815,
25722,
683,
13,
9754,
465,
898,
1714,
7380,
30090,
547,
2982,
11,
290,
287,
465,
898,
3292,
8941,
257,
4636,
28582,
13,
18612,
35394,
30,
8673,
13,
1002,
340,
547,
11,
262,
15393,
286,
262,
```

```
5977,
373,
29178,
3474,
416,
1310,
40559,
11959,
1636,
11,
508,
11,
287,
477,
922,
4562,
11,
3181,
503,
287,
262,
37090,
257,
845,
22665,
366,
672,
270,
2838,
1,
319,
3619,
438,
505,
286,
883,
905,
88,
6685,
42070,
351,
4738,
6276,
871,
326,
314,
423,
2982,
357,
```

```
40,
1839,
470,
910,
416,
4150,
8,
3688,
284,
402,
271,
10899,
338,
12036,
13,
843,
523,
438,
14363,
10568,
852,
5729,
11331,
18893,
540,
438,
1169,
5114,
11835,
3724,
503,
11,
290,
11,
355,
9074,
13,
536,
5469,
550,
11001,
11,
262,
2756,
286,
366,
38,
271,
10899,
```

```
82,
1,
1816,
510,
13,
198,
198,
1026,
373,
407,
10597,
1115,
812,
1568,
326,
11,
287,
262,
1781,
286,
257,
1178,
2745,
6,
4686,
1359,
319,
262,
34686,
41976,
11,
340,
6451,
5091,
284,
502,
284,
4240,
1521,
402,
271,
10899,
550,
1813,
510,
465,
12036,
13,
1550,
```

```
14580,
11,
340,
1107,
373,
257,
29850,
1917,
13,
1675,
24456,
465,
3656,
561,
423,
587,
1165,
2562,
438,
14363,
3148,
1650,
1010,
550,
587,
6699,
262,
1540,
558,
286,
2282,
326,
9074,
13,
402,
271,
10899,
550,
366,
7109,
14655,
683,
866,
526,
1114,
9074,
13,
402,
271,
10899,
```

```
438,
292,
884,
438,
18108,
407,
11196,
10597,
3016,
257,
614,
706,
3619,
338,
10568,
550,
587,
2077,
13,
632,
1244,
307,
326,
339,
550,
6405,
607,
438,
20777,
339,
8288,
465,
10152,
438,
13893,
339,
1422,
470,
765,
284,
467,
319,
12036,
26,
475,
340,
561,
423,
587,
```

```
1327,
284,
5879,
326,
339,
550,
1813,
510,
465,
12036,
780,
339,
550,
6405,
607,
13,
198,
198,
5189,
1781,
11,
611,
673,
550,
407,
17901,
683,
866,
11,
673,
550,
8603,
11,
355,
4544,
9325,
701,
42397,
11,
4054,
284,
366,
26282,
683,
510,
1,
438,
7091,
550,
```

```
407,
2957,
683,
736,
284,
262,
1396,
417,
13,
1675,
1234,
262,
14093,
656,
465,
1021,
757,
438,
10919,
257,
410,
5040,
329,
257,
3656,
0,
887,
9074,
13,
402,
271,
10899,
4120,
284,
423,
595,
67,
1328,
340,
438,
392,
314,
2936,
340,
1244,
307,
3499,
284,
1064,
```

```
503,
1521,
13,
198,
198,
464,
748,
586,
652,
1204,
286,
262,
34686,
41976,
37733,
2346,
284,
884,
14177,
8233,
1020,
5768,
26,
290,
1719,
11,
319,
616,
835,
284,
22489,
40089,
11,
4978,
257,
19350,
286,
3619,
338,
3652,
436,
81,
5286,
8812,
2114,
1022,
262,
279,
1127,
```

```
11,
314,
550,
3589,
28068,
294,
1555,
262,
1306,
1110,
13,
198,
198,
40,
1043,
262,
3155,
379,
8887,
11061,
511,
18057,
12,
83,
6037,
26,
290,
9074,
13,
402,
271,
10899,
338,
7062,
373,
523,
2429,
498,
326,
11,
287,
262,
29543,
2745,
11,
314,
4752,
340,
6777,
```

```
13,
632,
373,
407,
326,
616,
2583,
408,
373,
366,
47914,
1298,
319,
326,
966,
314,
714,
423,
1813,
4544,
9325,
701,
262,
40830,
12719,
3874,
13,
632,
373,
655,
780,
673,
373,
4808,
1662,
62,
3499,
438,
361,
314,
743,
307,
41746,
12004,
262,
6473,
438,
5562,
314,
```

```
1043,
607,
523,
13,
1114,
3619,
11,
477,
465,
1204,
11,
550,
587,
11191,
416,
3499,
1466,
25,
484,
550,
26546,
1068,
465,
1242,
11,
340,
550,
587,
302,
1144,
287,
262,
3024,
12,
4803,
286,
511,
512,
1741,
13,
843,
340,
373,
4361,
5048,
425,
284,
3465,
644,
```

```
1245,
262,
366,
25124,
3101,
8137,
286,
16957,
1696,
414,
1,
357,
40,
9577,
4544,
9325,
701,
8,
373,
1719,
319,
683,
13,
198,
198,
40,
423,
4750,
326,
9074,
13,
402,
271,
10899,
373,
5527,
26,
290,
340,
373,
3393,
34953,
856,
326,
607,
5229,
373,
37895,
422,
```

```
428,
25179,
257,
19217,
475,
8904,
14676,
13,
632,
318,
11,
355,
257,
3896,
11,
262,
661,
508,
40987,
1637,
508,
651,
749,
503,
286,
340,
26,
290,
3619,
338,
19992,
31564,
286,
465,
3656,
338,
1263,
5236,
9343,
683,
11,
351,
281,
5585,
286,
2818,
922,
12,
49705,
```

```
 11,
 284,
 21595,
 1133,
 340,
 656,
 5563,
 286,
 1242,
 290,
 13064,
 13,
 1675,
 262,
 6846,
 11,
 314,
 1276,
 751,
 11,
 339,
 6150,
 5365,
 31655,
 26,
 475,
 339,
 373,
 7067,
 29396,
 18443,
 12271,
 ...]

#Train-Validation ratio
train_ratio = 0.9
split_idx = int(train_ratio * len(text_data))
train_data = text_data[:split_idx]
val_data = text_data[split_idx:]

torch.manual_seed(123)

train_loader = create_dataloader_v1(
    train_data,
    batch_size=2,
    max_length=GPT_CONFIG_124M["context_length"],
    stride=GPT_CONFIG_124M["context_length"],
    drop_last=True,
    shuffle=True,
    num_workers=0
```

```
)

val_loader = create_dataloader_v1(
    val_data,
    batch_size=2,
    max_length=GPT_CONFIG_124M["context_length"],
    stride=GPT_CONFIG_124M["context_length"],
    drop_last=False,
    shuffle=False,
    num_workers=0
)

print("Train loader:")
for x,y in train_loader:
    print(x.shape,y.shape)
```

```
Train loader:
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
torch.Size([2, 256]) torch.Size([2, 256])
```

```
print("Val loader:")
for x,y in val_loader:
    print(x.shape,y.shape)
```

```
Val loader:
torch.Size([2, 256]) torch.Size([2, 256])
```

```
train_tokens = 0
for input_batch, target_batch in train_loader:
    train_tokens += input_batch.numel()

val_tokens = 0
for input_batch, target_batch in val_loader:
    val_tokens += input_batch.numel()

print("Training Tokens:", train_tokens)
print("Validation Tokens:", val_tokens)
print("All Tokens:", train_tokens + val_tokens)
```

```
Training Tokens: 4608
Validation Tokens: 512
All Tokens: 5120
```

```
device = "cuda"
```

```
def calc_loss_batch(input_batch, target_batch, model, device):
    input_batch, target_batch = input_batch.to(device),
target_batch.to(device)
    logits = model(input_batch)
    loss = torch.nn.functional.cross_entropy(logits.flatten(0, 1),
target_batch.flatten())
    return loss

def calc_loss_loader(data_loader, model, device, num_batches=None):
    total_loss = 0
    if len(data_loader) == 0:
        return float("nan")
    elif num_batches is None:
        num_batches = len(data_loader)
    else:
        #Reduce the number of batches to match the total number of
batches in the data loader
        #If num_batches exceeds the number of batches in data_loader
        num_batches = min(num_batches, len(data_loader))
    for i, (input_batch, target_batch) in enumerate(data_loader):
        if i < num_batches:
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            total_loss += loss.item()
        else:
            break
    return total_loss / num_batches

model.to(device)
```

```
GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(256, 768)
  (drop_emb): Dropout(p=0.1, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
```

```
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
```

```
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
```

```
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
```

```
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
```

```
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
```

```
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)

torch.manual_seed(123)

with torch.no_grad(): #Disable gradient tracking for efficiency
because we are not training, yet
    train_loss = calc_loss_loader(train_loader, model, device)
    val_loss = calc_loss_loader(val_loader, model, device)

print("Training loss:", train_loss)
print("Validation loss:", val_loss)

Training loss: 10.987583690219456
Validation loss: 10.98110580444336

#Perplexity
torch.exp(torch.tensor([0.5]))

tensor([1.6487])

tokenizer.n_vocab

50257

#Training an LLM

def train_model_simple(model, train_loader, val_loader, optimizer,
device, num_epochs,
                       eval_freq, eval_iter, start_context,
tokenizer):
    #Initialize lists to track losses and tokens seen
    train_losses, val_losses, track_tokens_seen = [], [], []
    tokens_seen, global_step = 0, -1

    #Main training loop
    for epoch in range(num_epochs):
        model.train() #Set model to training mode
        for input_batch, target_batch in train_loader:
            optimizer.zero_grad() #Reset loss gradients from previous
batch iteration
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            loss.backward() #Calculate loss gradients
            optimizer.step() #Update model weights using loss
```

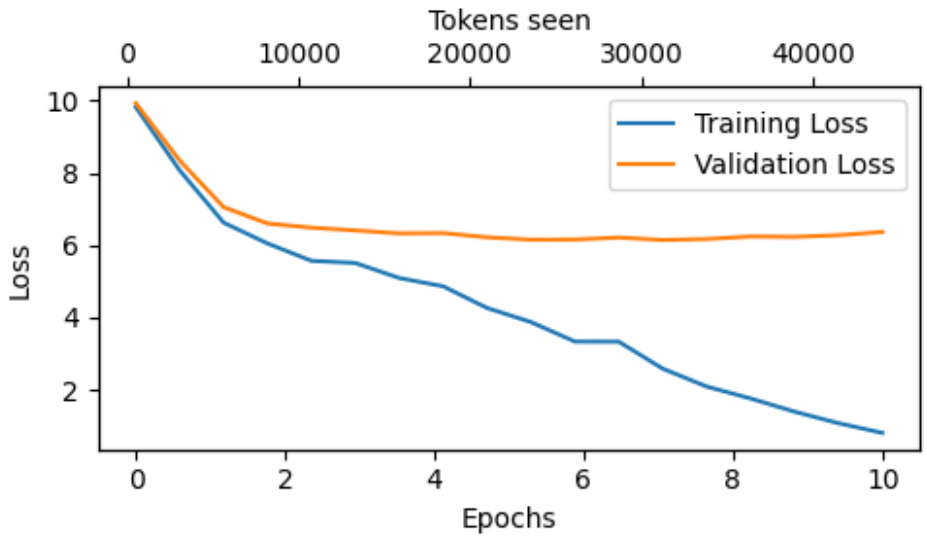
```
gradients
            tokens_seen += input_batch.numel()
            global_step += 1

            #Optional evaluation step
            if global_step % eval_freq == 0:
                train_loss, val_loss = evaluate_model(
                    model, train_loader, val_loader, device,
eval_iter)
                train_losses.append(train_loss)
                val_losses.append(val_loss)
                track_tokens_seen.append(tokens_seen)
                print(f"Ep {epoch+1} (Step {global_step:06d}): "
                      f"Train loss {train_loss:.3f}, Val loss
{val_loss:.3f}")

        #Print a sample text after each epoch
        generate_and_print_sample(
                model, tokenizer, device, start_context
        )

    return train_losses, val_losses, track_tokens_seen

def evaluate_model(model, train_loader, val_loader, device,
eval_iter):
    model.eval()
    with torch.no_grad():
        train_loss = calc_loss_loader(train_loader, model, device,
num_batches=eval_iter)
        val_loss = calc_loss_loader(val_loader, model, device,
num_batches=eval_iter)
    model.train()
    return train_loss, val_loss

def generate_and_print_sample(model, tokenizer, device,
start_context):
    model.eval()
    context_size = model.pos_emb.weight.shape[0]
    encoded = text_to_token_ids(start_context, tokenizer).to(device)
    with torch.no_grad():
        token_ids = generate_text_simple(
            model=model, idx=encoded,
            max_new_tokens=50, context_size=context_size
        )
    decoded_text = token_ids_to_text(token_ids, tokenizer)
    print(decoded_text.replace("\n", " ")) #Compact Print Format
    model.train()

torch.manual_seed(123)
model = GPTModel(GPT_CONFIG_124M)
```

```
model.to(device)
optimizer = torch.optim.AdamW(model.parameters(), lr=0.0004,
weight_decay=0.1)

num_epochs = 10
train_losses, val_losses, tokens_seen = train_model_simple(
    model, train_loader, val_loader, optimizer, device,
    num_epochs=num_epochs, eval_freq=5, eval_iter=5,
    start_context="Every effort moves you", tokenizer=tokenizer
)
```

```
Ep 1 (Step 000000): Train loss 9.819, Val loss 9.926
Ep 1 (Step 000005): Train loss 8.070, Val loss 8.341
Every effort moves you,,,,,,,,,,,,.

Ep 2 (Step 000010): Train loss 6.624, Val loss 7.051
Ep 2 (Step 000015): Train loss 6.047, Val loss 6.599
Every effort moves you, and,, and,, and,,,, and,.

Ep 3 (Step 000020): Train loss 5.567, Val loss 6.483
Ep 3 (Step 000025): Train loss 5.507, Val loss 6.408
Every effort moves you, and, and of the of the of the, and, and. G.
Gis, and, and, and, and, and, and, and, and, and, and, and, and,
and, and,
Ep 4 (Step 000030): Train loss 5.090, Val loss 6.324
Ep 4 (Step 000035): Train loss 4.862, Val loss 6.334
Every effort moves you.  "I had been the picture-- the picture.
"I was a the of the of the of the of the picture"I had been the
of
Ep 5 (Step 000040): Train loss 4.262, Val loss 6.217
Every effort moves you know the "I had been--I to me--as of the
donkey.     "Oh, in the man of the picture--as Jack himself at the
donkey--and it's the donkey--and it's it's
Ep 6 (Step 000045): Train loss 3.872, Val loss 6.151
Ep 6 (Step 000050): Train loss 3.334, Val loss 6.155
Every effort moves you know the "Oh, and.  "Oh, and in a little: "--I
looked up, and in a little.       "Oh, and he was, and down the room,
and I
Ep 7 (Step 000055): Train loss 3.329, Val loss 6.210
Ep 7 (Step 000060): Train loss 2.583, Val loss 6.143
Every effort moves you know the picture.  I glanced after him, and I
was. I had been his pictures's an the fact, and I felt. I was his
pictures--I had not the picture. I was, and down the room, I was
Ep 8 (Step 000065): Train loss 2.089, Val loss 6.168
Ep 8 (Step 000070): Train loss 1.759, Val loss 6.241
Every effort moves you?"  "Yes--I glanced after him, and uncertain.
"I looked up, with the fact, the cigars you like."  He placed them at
my elbow and as he said, and down the room, when I
Ep 9 (Step 000075): Train loss 1.394, Val loss 6.229
Ep 9 (Step 000080): Train loss 1.074, Val loss 6.274
```

```
Every effort moves you know," was one of the picture for nothing--I
told Mrs.  "Once, I was, in fact, and to see a smile behind his close
grayish beard--as if he had the donkey. "There were days when I
Ep 10 (Step 000085): Train loss 0.806, Val loss 6.364
Every effort moves you?"  "Yes--quite insensible to the irony. She
wanted him vindicated--and by me!"  He laughed again, and threw back
his head to look up at the sketch of the donkey. "There were days when
I
```

```
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

def plot_losses(epochs_seen, tokens_seen, train_losses, val_losses):
    fig, ax1 = plt.subplots(figsize=(5,3))

    #Plot training and validation loss against epochs
    ax1.plot(epochs_seen, train_losses, label="Training Loss")
    ax1.plot(epochs_seen, val_losses, label="Validation Loss")
    ax1.set_xlabel("Epochs")
    ax1.set_ylabel("Loss")
    ax1.legend(loc="upper right")
    ax1.xaxis.set_major_locator(MaxNLocator(integer=True)) #Only show
integer labels on x-axis

    #Create a second x-axis for tokens seen
    ax2 = ax1.twiny() #Create a second x-axis that shares the same y-
axis
    ax2.plot(tokens_seen, train_losses, alpha=0) #Invisible plo for
aligning ticks
    ax2.set_xlabel("Tokens seen")
    fig.tight_layout()
    plt.savefig("loss-plot.pdf")
    plt.show()

epochs_tensor = torch.linspace(0, num_epochs, len(train_losses))
plot_losses(epochs_tensor, tokens_seen, train_losses, val_losses)
```




```
#Decoding strategies to control randomness

model.to("cuda")

GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(256, 768)
  (drop_emb): Dropout(p=0.1, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
```

```
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
```

```
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
```

```
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
```

```
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
```

```
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=False)
        (W_key): Linear(in_features=768, out_features=768, bias=False)
        (W_value): Linear(in_features=768, out_features=768,
bias=False)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)
```

```
model.eval()

tokenizer = tiktoken.get_encoding("gpt2")

token_ids = generate_text_simple(
    model=model,
    idx=text_to_token_ids("Every effort moves you", tokenizer),
    max_new_tokens=25,
    context_size=GPT_CONFIG_124M["context_length"]
)

print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves you?"

"Yes--quite insensible to the irony. She wanted him vindicated--and by
me!"
```

```
#Temperature Scaling

vocab = {
    "closer": 0,
    "every": 1,
    "effort": 2,
    "forward": 3,
    "inches": 4,
    "moves": 5,
    "pizza": 6,
    "toward": 7,
    "you": 8,
}
inverse_vocab = {v:k for k, v in vocab.items()}
inverse_vocab
```

```
{0: 'closer',
 1: 'every',
 2: 'effort',
 3: 'forward',
 4: 'inches',
 5: 'moves',
 6: 'pizza',
 7: 'toward',
 8: 'you'}
```

```
next_token_logits = torch.tensor(
    [4.51, 0.89, -1.90, 6.75, 1.63, -1.62, -1.89, 6.28, 1.79]
)
```

```
probas = torch.softmax(next_token_logits, dim=0)
probas
```

```
tensor([6.0907e-02, 1.6313e-03, 1.0019e-04, 5.7212e-01, 3.4190e-03,
1.3257e-04,
        1.0120e-04, 3.5758e-01, 4.0122e-03])
```

```
next_token_id = torch.argmax(probas).item()
next_token_id
```

```
3
```

```
inverse_vocab[next_token_id]
```

```
'forward'
```

```
torch.manual_seed(123)
next_token_id = torch.multinomial(probas, num_samples=1).item()
print(inverse_vocab[next_token_id])
```

```
toward
```

```
next_token_id = torch.multinomial(probas, num_samples=1).item()
print(inverse_vocab[next_token_id])
```

```
forward
```

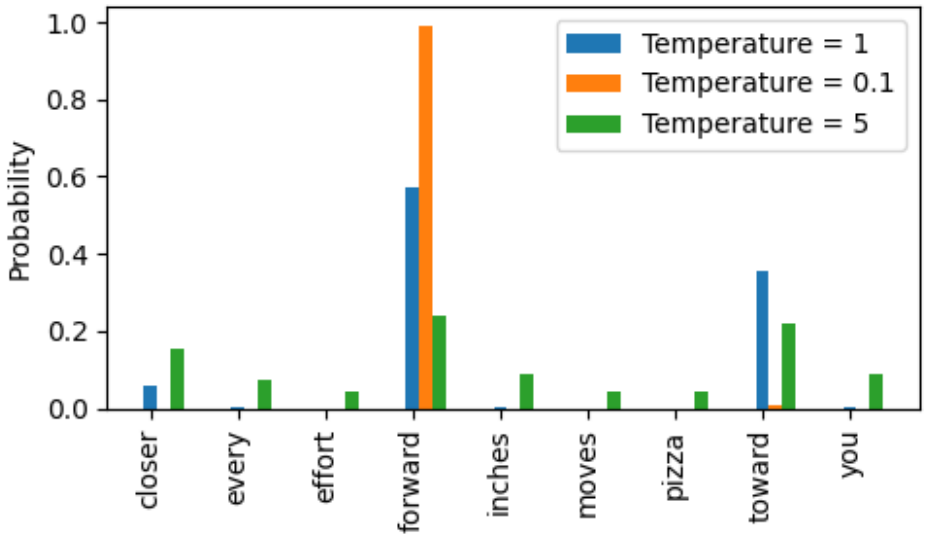
```
next_token_id = torch.multinomial(probas, num_samples=1).item()
print(inverse_vocab[next_token_id])
```

```
toward
```

```
def print_sampled_tokens(probas):
    torch.manual_seed(123)
    sample = [torch.multinomial(probas, num_samples=1).item() for i in
range(1_000)]
    sampled_ids = torch.bincount(torch.tensor(sample))
    for i,freq in enumerate(sampled_ids):
            print(f"{freq} X {inverse_vocab[i]}")

print_sampled_tokens(probas)
```

```
71 X closer
2 X every
0 X effort
544 X forward
2 X inches
1 X moves
0 X pizza
376 X toward
4 X you
```

```
def softmax_with_temperature(logits, temperature):
    scaled_logits = logits / temperature
    return torch.softmax(scaled_logits, dim=0)

temperatures = [1, 0.1, 5]

#Calculate scaled probabilites
scaled_probas = [softmax_with_temperature(next_token_logits, T) for T
in temperatures]

scaled_probas[0]

tensor([6.0907e-02, 1.6313e-03, 1.0019e-04, 5.7212e-01, 3.4190e-03,
1.3257e-04,
        1.0120e-04, 3.5758e-01, 4.0122e-03])

probas

tensor([6.0907e-02, 1.6313e-03, 1.0019e-04, 5.7212e-01, 3.4190e-03,
1.3257e-04,
        1.0120e-04, 3.5758e-01, 4.0122e-03])

scaled_probas[1]

tensor([1.8530e-10, 3.5189e-26, 2.6890e-38, 9.9099e-01, 5.7569e-23,
4.4220e-37,
        2.9718e-38, 9.0133e-03, 2.8514e-22])

scaled_probas[2]

tensor([0.1546, 0.0750, 0.0429, 0.2421, 0.0869, 0.0454, 0.0430,
0.2203, 0.0898])

#Plotting
x = torch.arange(len(vocab))
bar_width = 0.15
fig, ax = plt.subplots(figsize=(5,3))
for i, T in enumerate(temperatures):
    rects = ax.bar( x + i * bar_width, scaled_probas[i], bar_width,
label=f'Temperature = {T}')
ax.set_ylabel("Probability")
ax.set_xticks(x)
ax.set_xticklabels(vocab.keys(), rotation=90)
ax.legend()
plt.tight_layout()
plt.savefig("temperature-plot.pdf")
plt.show()
```




```
#Top-K sampling

next_token_logits = torch.tensor(
    [4.51, 0.89, -1.90, 6.75, 1.63, -1.62, -1.89, 6.28, 1.79]
)

top_k = 3
top_logits, top_pos = torch.topk(next_token_logits, top_k)
print(top_logits, top_pos)
```

```
tensor([6.7500, 6.2800, 4.5100]) tensor([3, 7, 0])
```

```
new_logits = torch.where(
    condition = next_token_logits < top_logits[-1],
    input = torch.tensor(float("-inf")),
    other = next_token_logits
)

new_logits
```

```
tensor([4.5100,   -inf,   -inf, 6.7500,   -inf,   -inf,   -inf,
6.2800,   -inf])
```

```
torch.softmax(new_logits, dim=0)
```

```
tensor([0.0615, 0.0000, 0.0000, 0.5775, 0.0000, 0.0000, 0.0000,
0.3610, 0.0000])
```

```
#Modifying the text generation function

tokenizer.decode([50256])
```

```
'<|endoftext|>'
```

```
def generate(model, idx, max_new_tokens, context_size,
temperature=1.0, top_k=None, eos_id=None):

    # Get the device the model's parameters are on
    device = next(model.parameters()).device
    # Ensure the initial input is on the same device as the model
    idx = idx.to(device)

    for _ in range(max_new_tokens):
        # Crop idx to the last context_size tokens
        idx_cond = idx[:, -context_size:]

        # Get the predictions from the model
        with torch.no_grad():
            logits = model(idx_cond)

        # Focus only on the logits for the last time step
        logits = logits[:, -1, :]

        # Apply top-k filtering if specified
        if top_k is not None:
            top_logits, _ = torch.topk(logits, top_k)
            # Set all logits not in the top-k to negative infinity
            logits[logits < top_logits[:, [-1]]] = float('-inf')

        # Apply temperature scaling and sample the next token
        if temperature > 0.0:
            # Apply temperature scaling
            scaled_logits = logits / temperature
            probs = torch.softmax(scaled_logits, dim=-1)

            # Sample from the distribution
            idx_next = torch.multinomial(probs, num_samples=1)
        else:
            # Greedily select the most likely token (if temperature is
0)
            idx_next = torch.argmax(logits, dim=-1, keepdim=True)

        # Stop if the end-of-sequence token is generated
        if eos_id is not None and idx_next == eos_id:
            break

        # Append the sampled token to the running sequence
        idx = torch.cat((idx, idx_next), dim=1)

    return idx

gpt = GPTModel(GPT_CONFIG_124M)

device = "cuda" if torch.cuda.is_available() else "cpu"
```

```
gpt.to(device)

# The tokenizer is used on the CPU
tokenizer = tiktoken.get_encoding("gpt2")

# Generate text
torch.manual_seed(123) # for reproducibility
token_ids = generate(
    model=gpt,
    # Move the input tensor to the same device as the model
    idx=text_to_token_ids("Every effort moves you",
tokenizer).to(device),
    max_new_tokens=15,
    context_size=GPT_CONFIG_124M["context_length"],
    top_k=25,
    temperature=1.4
)

print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves you causing Sed Pall household.] pops320
Jacksonvillegger £ discouraged synergy StarcraftCrossallic
```

```
token_ids = generate(
    model=model,
    idx=text_to_token_ids("Every effort moves you", tokenizer),
    max_new_tokens=15,
    context_size=GPT_CONFIG_124M["context_length"],
    top_k=25,
    #temperature=1.4
)
print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves you?"

I felt to Mrs. . . .

Well--
```

```
token_ids = generate(
    model=model,
    idx=text_to_token_ids("Every effort moves you", tokenizer),
    max_new_tokens=15,
    context_size=GPT_CONFIG_124M["context_length"],
    #top_k=25,
    temperature=1.4
)
print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves youaldi benchmark phrase as chest of tea Seas
nominations couldfcskirts renters then Sears
```

```
token_ids = generate(
    model=model,
    idx=text_to_token_ids("Every effort moves you", tokenizer),
    max_new_tokens=15,
    context_size=GPT_CONFIG_124M["context_length"],
    top_k=25,
    temperature=1.4
)
print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves you like to through my bravrowed by a smile that
lifted the frame last
```

```
#Loading and saving model weights with PyTorch
```

```
torch.save(model.state_dict(), "model.pth")
```

```
from previous_chapters import GPTModel

GPT_CONFIG_124M = {
    "vocab_size": 50257, #Vocabulary size
    "context_length": 256, #Context length
    "emb_dim": 768, #Embedding Dimension
    "n_heads": 12, #Number of attention heads
    "n_layers": 12, #Number of layers
    "drop_rate": 0.1, #Dropout rate
    "qkv_bias": False #Query-Key-Value bias
}

model = GPTModel(GPT_CONFIG_124M)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.load_state_dict(torch.load("model.pth", map_location=device))
```

```
C:\Users\tanis\AppData\Local\Temp\ipykernel_16876\761034928.py:2:
FutureWarning: You are using `torch.load` with `weights_only=False`
(the current default value), which uses the default pickle module
implicitly. It is possible to construct malicious pickle data which
will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
```

```
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  model.load_state_dict(torch.load("model.pth", map_location=device))

<All keys matched successfully>

#Loading pretrained weights from OpenAI

print("Tensorflow version:", version("tensorflow"))
print("tqdm version:", version("tqdm"))

Tensorflow version: 2.16.1
tqdm version: 4.66.4

from gpt_download import download_and_load_gpt2

settings, params = download_and_load_gpt2(model_size="124M",
models_dir="gpt2")

Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\checkpoint) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
checkpoint
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
checkpoint) and backup URL (https://f001.backblazeb2.com/file/LLMs-
from-scratch/gpt2\124M\checkpoint).
Check your internet connection or the file availability.
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\encoder.json) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
encoder.json
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
encoder.json) and backup URL (https://f001.backblazeb2.com/file/LLMs-
from-scratch/gpt2\124M\encoder.json).
Check your internet connection or the file availability.
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\hparams.json) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
hparams.json
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
hparams.json) and backup URL (https://f001.backblazeb2.com/file/LLMs-
from-scratch/gpt2\124M\hparams.json).
Check your internet connection or the file availability.
```

```
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\model.ckpt.data-00000-of-00001) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
model.ckpt.data-00000-of-00001
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
model.ckpt.data-00000-of-00001) and backup URL
(https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
model.ckpt.data-00000-of-00001).
Check your internet connection or the file availability.
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
File already exists and is up-to-date: gpt2\124M\model.ckpt.index
File already exists and is up-to-date: gpt2\124M\model.ckpt.meta
File already exists and is up-to-date: gpt2\124M\vocab.bpe
```

```
print("Settings:", settings)
```

```
Settings: {'n_vocab': 50257, 'n_ctx': 1024, 'n_embd': 768, 'n_head':
12, 'n_layer': 12}
```

```
print("Parameter dictionary keys:", params.keys())
```

```
Parameter dictionary keys: dict_keys(['blocks', 'b', 'g', 'wpe',
'wte'])
```

```
#Define model configurations in a dictionary for compactness

GPT_CONFIG_124M = {
    "vocab_size": 50257, #Vocabulary size
    "context_length": 1024, #Context length
    "emb_dim": 768, #Embedding Dimension
    "n_heads": 12, #Number of attention heads
    "n_layers": 12, #Number of layers
    "drop_rate": 0.1, #Dropout rate
    "qkv_bias": False #Query-Key-Value bias
}

model_configs = {
    "gpt2-small (124M)": {"emb_dim": 768, "n_layers": 12, "n_heads":
12},
    "gpt2-medium (355M)": {"emb_dim": 1024, "n_layers": 24, "n_heads":
16},
    "gpt2-large (774M)": {"emb_dim": 1280, "n_layers": 36, "n_heads":
20},
    "gpt2-x1 (1558M)": {"emb_dim": 1600, "n_layers": 48, "n_heads":
25},
}
```

```
#Copy the base configuration and update with specific model settings
model_name = "gpt2-small (124M)" #Example model name
NEW_CONFIG = GPT_CONFIG_124M.copy()
NEW_CONFIG.update(model_configs[model_name])
NEW_CONFIG.update({"context_length": 1024, "qkv_bias": True})

gpt = GPTModel(NEW_CONFIG)
gpt.eval();

def assign(left,right):
    if left.shape != right.shape:
        raise ValueError(f"Shape mismatch. Left: {left.shape}, Right:
{right.shape}")
    return torch.nn.Parameter(torch.tensor(right))

device = "cuda"

import numpy as np

def load_weights_into_gpt(gpt, params):
    gpt.pos_emb.weight = assign(gpt.pos_emb.weight, params['wpe'])
    gpt.tok_emb.weight = assign(gpt.tok_emb.weight, params['wte'])

    for b in range(len(params["blocks"])):
        q_w, k_w, v_w = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["w"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.weight = assign(
            gpt.trf_blocks[b].att.W_query.weight, q_w.T)
        gpt.trf_blocks[b].att.W_key.weight = assign(
            gpt.trf_blocks[b].att.W_key.weight, k_w.T)
        gpt.trf_blocks[b].att.W_value.weight = assign(
            gpt.trf_blocks[b].att.W_value.weight, v_w.T)

        q_b, k_b, v_b = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["b"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.bias = assign(
            gpt.trf_blocks[b].att.W_query.bias, q_b)
        gpt.trf_blocks[b].att.W_key.bias = assign(
            gpt.trf_blocks[b].att.W_key.bias, k_b)
        gpt.trf_blocks[b].att.W_value.bias = assign(
            gpt.trf_blocks[b].att.W_value.bias, v_b)

        gpt.trf_blocks[b].att.out_proj.weight = assign(
            gpt.trf_blocks[b].att.out_proj.weight,
            params["blocks"][b]["attn"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].att.out_proj.bias = assign(
            gpt.trf_blocks[b].att.out_proj.bias,
            params["blocks"][b]["attn"]["c_proj"]["b"])

        gpt.trf_blocks[b].ff.layers[0].weight = assign(
```

```
            gpt.trf_blocks[b].ff.layers[0].weight,
            params["blocks"][b]["mlp"]["c_fc"]["w"].T)
        gpt.trf_blocks[b].ff.layers[0].bias = assign(
            gpt.trf_blocks[b].ff.layers[0].bias,
            params["blocks"][b]["mlp"]["c_fc"]["b"])
        gpt.trf_blocks[b].ff.layers[2].weight = assign(
            gpt.trf_blocks[b].ff.layers[2].weight,
            params["blocks"][b]["mlp"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].ff.layers[2].bias = assign(
            gpt.trf_blocks[b].ff.layers[2].bias,
            params["blocks"][b]["mlp"]["c_proj"]["b"])

        gpt.trf_blocks[b].norm1.scale = assign(
            gpt.trf_blocks[b].norm1.scale,
            params["blocks"][b]["ln_1"]["g"])
        gpt.trf_blocks[b].norm1.shift = assign(
            gpt.trf_blocks[b].norm1.shift,
            params["blocks"][b]["ln_1"]["b"])
        gpt.trf_blocks[b].norm2.scale = assign(
            gpt.trf_blocks[b].norm2.scale,
            params["blocks"][b]["ln_2"]["g"])
        gpt.trf_blocks[b].norm2.shift = assign(
            gpt.trf_blocks[b].norm2.shift,
            params["blocks"][b]["ln_2"]["b"])

    gpt.final_norm.scale = assign(gpt.final_norm.scale, params["g"])
    gpt.final_norm.shift = assign(gpt.final_norm.shift, params["b"])
    gpt.out_head.weight = assign(gpt.out_head.weight, params["wte"])

load_weights_into_gpt(gpt, params)
gpt.to(device)
```

```
GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(1024, 768)
  (drop_emb): Dropout(p=0.1, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
```

```
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
```

```
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (5): TransformerBlock(
```

```
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
```

```
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
```

```
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
```

```
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.1, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)

token_ids = generate(
    model=gpt,
    idx=text_to_token_ids("Every effort moves you", tokenizer),
    max_new_tokens=15,
    context_size=GPT_CONFIG_124M["context_length"],
    top_k=25,
    temperature=1.4
)
print("Output text:\n", token_ids_to_text(token_ids, tokenizer))
```

```
Output text:
 Every effort moves you through every step with an inexorabilible
force. It has no effect
```

```
#Fine-tuning for classification

import torch

device = "cuda" if torch.cuda.is_available() else "cpu"
device

'cuda'

#Previous chapters modules

# CHAPTER 2
import tiktoken
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

class GPTDatasetV1(Dataset):
    def __init__(self, txt, tokenizer, max_length, stride):
        self.input_ids = []
        self.target_ids = []

        #Tokenize the entire text
        token_ids = tokenizer.encode(txt, allowed_special={"<|
endoftext|>"})

        #Use a sliding window to chunk the book into overlapping
sequences of max_length
        for i in range(0, len(token_ids) - max_length, stride):
            input_chunk = token_ids[i:i + max_length]
            target_chunk = token_ids[i+1: i + max_length + 1]
            self.input_ids.append(torch.tensor(input_chunk))
            self.target_ids.append(torch.tensor(target_chunk))

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self,idx):
        return self.input_ids[idx], self.target_ids[idx]

def create_dataloader_v1(txt, batch_size=4, max_length=256,
stride=128, shuffle=True, drop_last=True, num_workers=0):
    #Initialize the tokenizer
    tokenizer = tiktoken.get_encoding("gpt2")

    #Create dataset
    dataset = GPTDatasetV1(txt, tokenizer, max_length, stride)

    #Create dataloader
    dataloader = DataLoader(
    dataset, batch_size=batch_size, shuffle=shuffle,
```

```
drop_last=drop_last, num_workers=num_workers)

    return dataloader

#  CHAPTER 3

class MultiHeadAttention(nn.Module):
    def __init__(self, d_in, d_out, context_length, dropout,
num_heads, qkv_bias=False):
        super().__init__()
        assert( d_out % num_heads == 0 ), \
            "d_out must be divisible by num_heads"
        self.d_out = d_out
        self.num_heads = num_heads
        self.head_dim = d_out // num_heads
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.out_proj = nn.Linear(d_out, d_out)
        self.dropout = nn.Dropout(dropout)
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
                       diagonal=1)
        )

    def forward(self, x):
        b, num_tokens, d_in = x.shape
        #Shape: (b, num_tokens, d_out)
        keys = self.W_key(x)
        queries = self.W_query(x)
        values = self.W_value(x)
        #We implicitly split the matrix by adding a num_heads
dimension
        #Unroll last dim: (b, num_tokens, d_out) -> (b, num_tokens,
num_heads, head_dim)
        keys = keys.view(b, num_tokens, self.num_heads, self.head_dim)
        values = values.view(b, num_tokens, self.num_heads,
self.head_dim)
        queries = queries.view(b, num_tokens, self.num_heads,
self.head_dim)
        # Transpose: (b, num_tokens, num_heads, head_dim) -> (b,
num_heads, num_tokens, head_dim)
        keys = keys.transpose(1, 2)
        queries = queries.transpose(1, 2)
        values = values.transpose(1, 2)
        #Compute scaled dot product attention (aka self attention)
with a causal mask
        attn_scores = queries @ keys.transpose(2, 3) #Dot product for
each head
```

```
        #Original mask truncated to the number of tokens and converted
to boolean
        mask_bool = self.mask.bool()[:num_tokens, :num_tokens]
        #Use the mask to fill attention scores
        attn_scores.masked_fill_(mask_bool, -torch.inf)
        attn_weights = torch.softmax(attn_scores / keys.shape[-
1]**0.5, dim=-1)
        attn_weights = self.dropout(attn_weights)
        #Shape: (b, num_tokens, num_heads, head_dim)
        context_vec = (attn_weights @ values).transpose(1, 2)
        #Combine heads, where self.d_out = elf.num_heads *
self.head_dim
        context_vec = context_vec.contiguous().view(b, num_tokens,
self.d_out)
        context_vec = self.out_proj(context_vec) #Optional Projection
        return context_vec

# CHAPTER 4

class LayerNorm(nn.Module):
    def __init__(self, emb_dim):
        super().__init__()
        self.eps = 1e-5
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim))

    def forward(self, x):
        mean = x.mean(dim=-1, keepdim=True)
        var = x.var(dim=-1, keepdim=True, unbiased=False)
        norm_x = (x-mean) / torch.sqrt(var + self.eps)
        return self.scale * norm_x + self.shift

class GELU(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        return 0.5 * x * (1 + torch.tanh(
                torch.sqrt(torch.tensor(2.0 / torch.pi)) *
                (x + 0.044715 * torch.pow(x, 3))
        ))

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(cfg["emb_dim"], 4 * cfg["emb_dim"]),
            GELU(),
            nn.Linear(4 * cfg["emb_dim"], cfg["emb_dim"]),
        )
```

```
    def forward(self, x):
        return self.layers(x)

class TransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.att = MultiHeadAttention(
            d_in = cfg["emb_dim"],
            d_out = cfg["emb_dim"],
            context_length = cfg["context_length"],
            num_heads = cfg["n_heads"],
            dropout = cfg["drop_rate"],
            qkv_bias = cfg["qkv_bias"])
        self.ff = FeedForward(cfg)
        self.norm1 = LayerNorm(cfg["emb_dim"])
        self.norm2 = LayerNorm(cfg["emb_dim"])
        self.drop_shortcut = nn.Dropout(cfg["drop_rate"])

    def forward(self, x):
        #Shortcut connection for attention block
        shortcut = x
        x = self.norm1(x)
        x = self.att(x) #Shape [batch_size, num_tokens, emb_size]
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back

        #Shortcut connection for feed forward block
        shortcut = x
        x = self.norm2(x)
        x = self.ff(x)
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back
        return x

class GPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"],
cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

        self.trf_blocks = nn.Sequential(
            *[TransformerBlock(cfg) for _ in range(cfg["n_layers"])])

        self.final_norm = LayerNorm(cfg["emb_dim"])
        self.out_head = nn.Linear(
            cfg["emb_dim"], cfg["vocab_size"], bias=False
        )
```

```
    def forward(self, in_idx):
        batch_size, seq_len = in_idx.shape
        tok_embeds = self.tok_emb(in_idx)
        pos_embeds = self.pos_emb(torch.arange(seq_len,
device=in_idx.device))
        x = tok_embeds + pos_embeds
        x = self.drop_emb(x)
        x = self.trf_blocks(x)
        x = self.final_norm(x)
        logits = self.out_head(x)
        return logits

def generate_text_simple(model, idx, max_new_tokens, context_size):
    for _ in range(max_new_tokens):
        idx_cond = idx[:, -context_size:]

        with torch.no_grad():
            logits = model(idx_cond)
        logits = logits[:, -1, :]

        probas = torch.softmax(logits, dim=-1)

        idx_next = torch.argmax(probas, dim=-1, keepdim=True)

        idx = torch.cat((idx, idx_next), dim=1)
    return idx

# CHAPTER 5

def calc_loss_batch(input_batch, target_batch, model, device):
    input_batch, target_batch = input_batch.to(device),
target_batch.to(device)
    logits = model(input_batch)
    loss = torch.nn.functional.cross_entropy(logits.flatten(0, 1),
target_batch.flatten())
    return loss

def calc_loss_loader(data_loader, model, device, num_batches=None):
    total_loss = 0
    if len(data_loader) == 0:
        return float("nan")
    elif num_batches is None:
        num_batches = len(data_loader)
    else:
        #Reduce the number of batches to match the total number of
batches in the data loader
        #If num_batches exceeds the number of batches in data_loader
        num_batches = min(num_batches, len(data_loader))
    for i, (input_batch, target_batch) in enumerate(data_loader):
```

```
        if i < num_batches:
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            total_loss += loss.item()
        else:
            break
    return total_loss / num_batches

def train_model_simple(model, train_loader, val_loader, optimizer,
device, num_epochs,
                       eval_freq, eval_iter, start_context,
tokenizer):
    #Initialize lists to track losses and tokens seen
    train_losses, val_losses, track_tokens_seen = [], [], []
    tokens_seen, global_step = 0, -1

    #Main training loop
    for epoch in range(num_epochs):
        model.train() #Set model to training mode
        for input_batch, target_batch in train_loader:
            optimizer.zero_grad() #Reset loss gradients from previous
batch iteration
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            loss.backward() #Calculate loss gradients
            optimizer.step() #Update model weights using loss
gradients
            tokens_seen += input_batch.numel()
            global_step += 1

            #Optional evaluation step
            if global_step % eval_freq == 0:
                train_loss, val_loss = evaluate_model(
                    model, train_loader, val_loader, device,
eval_iter)
                train_losses.append(train_loss)
                val_losses.append(val_loss)
                track_tokens_seen.append(tokens_seen)
                print(f"Ep {epoch+1} (Step {global_step:06d}): "
                      f"Train loss {train_loss:.3f}, Val loss
{val_loss:.3f}")

        #Print a sample text after each epoch
        generate_and_print_sample(
                model, tokenizer, device, start_context
        )

    return train_losses, val_losses, track_tokens_seen

def evaluate_model(model, train_loader, val_loader, device,
```

```
eval_iter):
    model.eval()
    with torch.no_grad():
        train_loss = calc_loss_loader(train_loader, model, device,
num_batches=eval_iter)
        val_loss = calc_loss_loader(val_loader, model, device,
num_batches=eval_iter)
    model.train()
    return train_loss, val_loss

def generate_and_print_sample(model, tokenizer, device,
start_context):
    model.eval()
    context_size = model.pos_emb.weight.shape[0]
    encoded = text_to_token_ids(start_context, tokenizer).to(device)
    with torch.no_grad():
        token_ids = generate_text_simple(
            model=model, idx=encoded,
            max_new_tokens=50, context_size=context_size
        )
    decoded_text = token_ids_to_text(token_ids, tokenizer)
    print(decoded_text.replace("\n", " ")) #Compact Print Format
    model.train()

def generate(model, idx, max_new_tokens, context_size,
temperature=1.0, top_k=None, eos_id=None):

    # Get the device the model's parameters are on
    device = next(model.parameters()).device
    # Ensure the initial input is on the same device as the model
    idx = idx.to(device)

    for _ in range(max_new_tokens):
        # Crop idx to the last context_size tokens
        idx_cond = idx[:, -context_size:]

        # Get the predictions from the model
        with torch.no_grad():
            logits = model(idx_cond)

        # Focus only on the logits for the last time step
        logits = logits[:, -1, :]

        # Apply top-k filtering if specified
        if top_k is not None:
            top_logits, _ = torch.topk(logits, top_k)
            # Set all logits not in the top-k to negative infinity
            logits[logits < top_logits[:, [-1]]] = float('-inf')

        # Apply temperature scaling and sample the next token
```

```
        if temperature > 0.0:
            # Apply temperature scaling
            scaled_logits = logits / temperature
            probs = torch.softmax(scaled_logits, dim=-1)

            # Sample from the distribution
            idx_next = torch.multinomial(probs, num_samples=1)
        else:
            # Greedily select the most likely token (if temperature is
0)
            idx_next = torch.argmax(logits, dim=-1, keepdim=True)

        # Stop if the end-of-sequence token is generated
        if eos_id is not None and idx_next == eos_id:
            break

        # Append the sampled token to the running sequence
        idx = torch.cat((idx, idx_next), dim=1)

    return idx

import numpy as np

def assign(left,right):
    if left.shape != right.shape:
        raise ValueError(f"Shape mismatch. Left: {left.shape}, Right:
{right.shape}")
    return torch.nn.Parameter(torch.tensor(right))

def load_weights_into_gpt(gpt, params):
    gpt.pos_emb.weight = assign(gpt.pos_emb.weight, params['wpe'])
    gpt.tok_emb.weight = assign(gpt.tok_emb.weight, params['wte'])

    for b in range(len(params["blocks"])):
        q_w, k_w, v_w = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["w"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.weight = assign(
            gpt.trf_blocks[b].att.W_query.weight, q_w.T)
        gpt.trf_blocks[b].att.W_key.weight = assign(
            gpt.trf_blocks[b].att.W_key.weight, k_w.T)
        gpt.trf_blocks[b].att.W_value.weight = assign(
            gpt.trf_blocks[b].att.W_value.weight, v_w.T)

        q_b, k_b, v_b = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["b"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.bias = assign(
            gpt.trf_blocks[b].att.W_query.bias, q_b)
        gpt.trf_blocks[b].att.W_key.bias = assign(
            gpt.trf_blocks[b].att.W_key.bias, k_b)
        gpt.trf_blocks[b].att.W_value.bias = assign(
```

```
            gpt.trf_blocks[b].att.W_value.bias, v_b)

        gpt.trf_blocks[b].att.out_proj.weight = assign(
            gpt.trf_blocks[b].att.out_proj.weight,
            params["blocks"][b]["attn"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].att.out_proj.bias = assign(
            gpt.trf_blocks[b].att.out_proj.bias,
            params["blocks"][b]["attn"]["c_proj"]["b"])

        gpt.trf_blocks[b].ff.layers[0].weight = assign(
            gpt.trf_blocks[b].ff.layers[0].weight,
            params["blocks"][b]["mlp"]["c_fc"]["w"].T)
        gpt.trf_blocks[b].ff.layers[0].bias = assign(
            gpt.trf_blocks[b].ff.layers[0].bias,
            params["blocks"][b]["mlp"]["c_fc"]["b"])
        gpt.trf_blocks[b].ff.layers[2].weight = assign(
            gpt.trf_blocks[b].ff.layers[2].weight,
            params["blocks"][b]["mlp"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].ff.layers[2].bias = assign(
            gpt.trf_blocks[b].ff.layers[2].bias,
            params["blocks"][b]["mlp"]["c_proj"]["b"])

        gpt.trf_blocks[b].norm1.scale = assign(
            gpt.trf_blocks[b].norm1.scale,
            params["blocks"][b]["ln_1"]["g"])
        gpt.trf_blocks[b].norm1.shift = assign(
            gpt.trf_blocks[b].norm1.shift,
            params["blocks"][b]["ln_1"]["b"])
        gpt.trf_blocks[b].norm2.scale = assign(
            gpt.trf_blocks[b].norm2.scale,
            params["blocks"][b]["ln_2"]["g"])
        gpt.trf_blocks[b].norm2.shift = assign(
            gpt.trf_blocks[b].norm2.shift,
            params["blocks"][b]["ln_2"]["b"])

    gpt.final_norm.scale = assign(gpt.final_norm.scale, params["g"])
    gpt.final_norm.shift = assign(gpt.final_norm.shift, params["b"])
    gpt.out_head.weight = assign(gpt.out_head.weight, params["wte"])

#Preparing the dataset

import urllib.request
import zipfile
import os
from pathlib import Path

url =
"https://archive.ics.uci.edu/static/public/228/sms+spam+collection.zip
"
zip_path = "sms_spam_collection.zip"
```

```
extracted_path = "sms_spam_collection"
data_file_path = Path(extracted_path) / "SMSSpamCollection.tsv"

def download_and_unzip_spam_data(url, zip_path, extracted_path,
data_file_path):
    if data_file_path.exists():
        print(f"{data_file_path} already exists. Skipping download and
extraction")
        return

    #Downloading the file
    with urllib.request.urlopen(url) as response:
        with open(zip_path, "wb") as out_file:
            out_file.write(response.read())

    #Unzipping the file
    with zipfile.ZipFile(zip_path, "r") as zip_ref:
        zip_ref.extractall(extracted_path)

    #Add .tsv file extension
    original_file_path = Path(extracted_path) / "SMSSpamCollection"
    os.rename(original_file_path, data_file_path)
    print(f"File downloaded and saved as {data_file_path}")

try:
    download_and_unzip_spam_data(url, zip_path, extracted_path,
data_file_path)
except (urllib.error.HTTPError, urllib.error.URLError, TimeoutError)
as e:
    print(f"Primary URL Failed: {e}. Trying backup URL")
    url = "https://f001.backblazeb2.com/file/LLMs-from-scratch/sms
%2Bspam%2Bcollection.zip"
    download_and_unzip_spam_data(url, zip_path, extracted_path,
data_file_path)
```

```
File downloaded and saved as sms_spam_collection\SMSSpamCollection.tsv
```

```
from importlib.metadata import version

pkgs = ["matplotlib",
        "numpy",
        "tiktoken",
        "torch",
        "tensorflow",
        "pandas"
        ]

for p in pkgs:
    print(f"{p} version: {version(p)}")
```

```
matplotlib version: 3.8.4
numpy version: 1.26.4
tiktoken version: 0.11.0
torch version: 2.5.1+cu121
tensorflow version: 2.16.1
pandas version: 2.2.2
```

```
import pandas as pd

df = pd.read_csv(data_file_path, sep="\t", header=None,
names=["Label", "Text"])

df.head()
```

```
  Label                                               Text
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

```
df
```

```
     Label                                               Text
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
...    ...                                               ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568   ham              Will ü b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So...any other s...
5570   ham  The guy did some bitching but I acted like i'd...
5571   ham                         Rofl. Its true to its name

[5572 rows x 2 columns]
```

```
df["Label"].value_counts()
```

```
Label
ham     4825
spam     747
Name: count, dtype: int64
```

```
num_spam = df[df["Label"] == "spam"].shape[0]

df[df["Label"] == "ham"].sample(num_spam, random_state=123).shape
```

```
(747, 2)
```

```
def create_balanced_dataset(df):
    #Count the instances of "spam"
    num_spam = df[df["Label"] == "spam"].shape[0]
    #Randomly sample "ham" instances to match the number of "spam"
instances
    ham_subset = df[df["Label"] == "ham"].sample(num_spam,
random_state=123)
    #Combine ham "subset" with "spam"
    balanced_df = pd.concat([ham_subset, df[df["Label"] == "spam"]])
    return balanced_df

balanced_df = create_balanced_dataset(df)
print(balanced_df["Label"].value_counts())
```

```
Label
ham     747
spam    747
Name: count, dtype: int64
```

```
balanced_df.head()
```

	Label	Text
4307	ham	Awww dat is sweet! We can think of something t...
4138	ham	Just got to <#>
4831	ham	The word "Checkmate" in chess comes from the P...
4461	ham	This is wishing you a great day. Moji told me ...
5440	ham	Thank you. do you generally date the brothas?

```
#ham -> 0
#spam -> 1

map_dict = {"ham":0, "spam":1}
balanced_df["Label"] = balanced_df["Label"].map(map_dict)

balanced_df.head()
```

	Label	Text
4307	0	Awww dat is sweet! We can think of something t...
4138	0	Just got to <#>
4831	0	The word "Checkmate" in chess comes from the P...
4461	0	This is wishing you a great day. Moji told me ...
5440	0	Thank you. do you generally date the brothas?

```
balanced_df.tail()
```

	Label	Text
5537	1	Want explicit SEX in 30 secs? Ring 02073162414...
5540	1	ASKED 3MOBILE IF 0870 CHATLINES INCLU IN FREE ...
5547	1	Had your contract mobile 11 Mnths? Latest Moto...
5566	1	REMINDER FROM O2: To get 2.50 pounds free call...
5567	1	This is the 2nd time we have tried 2 contact u...

```
print(balanced_df["Label"].value_counts())
```

```
Label
0    747
1    747
Name: count, dtype: int64
```

```
def random_split(df, train_frac, validation_frac):
    #Shuffle the entire dataframe
    df = df.sample(frac=1, random_state=123).reset_index(drop=True)
    #Calculate the split indices
    train_end = int(len(df) * train_frac)
    validation_end = train_end + int(len(df) * validation_frac)
    #100, train_end 1 to 70, val_end 70 to 90, 90 to 100
    #Split the dataframe
    train_df = df[:train_end]
    validation_df = df[train_end:validation_end]
    test_df = df[validation_end:]
    return train_df, validation_df, test_df

train_df , validation_df, test_df = random_split(balanced_df, 0.7,
0.1)
#Test size is implied to be 0.2
train_df.to_csv("train.csv", index=None)
validation_df.to_csv("validation.csv", index=None)
test_df.to_csv("test.csv", index=None)
```

```
train_df
```

	Label	Text
0	0	Dude how do you like the buff wind.
1	0	Tessy..pls do me a favor. Pls convey my birthd...
2	1	Reminder: You have not downloaded the content ...
3	1	Got what it takes 2 take part in the WRC Rally...
4	1	Shop till u Drop, IS IT YOU, either 10K, 5K, £...
...	...	...
1040	1	4mths half price Orange line rental & latest c...
1041	1	Thanks for the Vote. Now sing along with the s...
1042	1	IMPORTANT INFORMATION 4 ORANGE USER 0796XXXXXX...
1043	1	Urgent! call 09066612661 from landline. Your c...
1044	0	His frens go then he in lor. Not alone wif my ...

```
[1045 rows x 2 columns]
```

```
validation_df
```

	Label	Text
1045	1	Mila, age23, blonde, new in UK. I look sex wit...
1046	1	Hungry gay guys feeling hungry and up 4 it, no...
1047	0	Ugh. Gotta drive back to sd from la. My butt i...
1048	0	Please leave this topic..sorry for telling that..

```
1049      1  We tried to contact you re our offer of New Vi...
...     ...                                                ...
1189      0  Hey gorgeous man. My work mobile number is. Ha...
1190      0                          Just sleeping..and surfing
1191      0           I'm in solihull, | do you want anything?
1192      0                       Jay told me already, will do
1193      1  U can WIN £100 of Music Gift Vouchers every we...

[149 rows x 2 columns]
```

```
test_df
```

```
      Label                                               Text
1194      1                     85233 FREE>Ringtone!Reply REAL
1195      1  Ur cash-balance is currently 500 pounds - to m...
1196      1  Thanks for your ringtone order, reference numb...
1197      0                We live in the next  &lt;#&gt; mins
1198      1  1st wk FREE! Gr8 tones str8 2 u each wk. Txt N...
...     ...                                                ...
1489      1  FREE2DAY sexy St George's Day pic of Jordan!Tx...
1490      1  Urgent! Please call 09066612661 from your land...
1491      1  For your chance to WIN a FREE Bluetooth Headse...
1492      1  * FREE* POLYPHONIC RINGTONE Text SUPER to 8713...
1493      1  Your unique user ID is 1172. For removal send ...

[300 rows x 2 columns]
```

```
#Creating data loaders

import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")
print(tokenizer.decode([50256]))
```

```
<|endoftext|>
```

```
from torch.utils.data import Dataset

class SpamDataset(Dataset):
    def __init__(self, csv_file, tokenizer, max_length=None,
pad_token_id=50256):
        self.data = pd.read_csv(csv_file)

        #Pre=tokenize texts
        self.encoded_texts = [
            tokenizer.encode(text) for text in self.data["Text"]
        ]

        if max_length is None:
            self.max_length = self._longest_encoded_length()
        else:
```

```
            self.max_length = max_length
            #Truncate sequences if they are longer than max_length
            self.encoded_texts = [
                encoded_text[:self.max_length]
                for encoded_text in self.encoded_texts
            ]

        #Pad sequences to the longest sequence
        self.encoded_texts = [
            encoded_text + [pad_token_id] * (self.max_length -
len(encoded_text))
            for encoded_text in self.encoded_texts
        ]

    def __getitem__(self, index):
        encoded = self.encoded_texts[index]
        label = self.data.iloc[index]["Label"]
        return (
            torch.tensor(encoded, dtype=torch.long),
            torch.tensor(label, dtype=torch.long)
        )

    def __len__(self):
        return len(self.data)

    def _longest_encoded_length(self):
        max_length = 0
        for encoded_text in self.encoded_texts:
            encoded_length = len(encoded_text)
            if encoded_length > max_length:
                max_length = encoded_length
        return max_length
    #A more pythonic version to implement this method is the following
which is also used in the next chapter
    #return max(len(encoded_text) for encoded_text in
self.encoded_texts)

train_dataset = SpamDataset(
    csv_file="train.csv",
    max_length=None,
    tokenizer=tokenizer
)

print(train_dataset.max_length)
```

```
120
```

```
val_dataset = SpamDataset(
    csv_file="validation.csv",
    max_length=None,
```

```
    tokenizer=tokenizer
)

test_dataset = SpamDataset(
    csv_file="test.csv",
    max_length=None,
    tokenizer=tokenizer
)

print(f" The max length for validation dataset is
{val_dataset.max_length}")
print(f" The max length for test dataset is
{test_dataset.max_length}")
```

```
 The max length for validation dataset is 71
 The max length for test dataset is 92
```

```
from torch.utils.data import DataLoader

num_workers = 0
batch_size = 8

torch.manual_seed(123)

train_loader = DataLoader(
    dataset=train_dataset,
    batch_size=batch_size,
    shuffle=True,
    num_workers=num_workers,
    drop_last=True,
)

val_loader = DataLoader(
    dataset=val_dataset,
    batch_size=batch_size,
    shuffle=True,
    num_workers=num_workers,
    drop_last=False,
)

test_loader = DataLoader(
    dataset=test_dataset,
    batch_size=batch_size,
    shuffle=True,
    num_workers=num_workers,
    drop_last=False,
)

print("Train loader:")
for input_batch, target_batch in train_loader:
    pass
```

```
print("Input batch dimensions:", input_batch.shape)
print("Label batch dimensions:", target_batch.shape)
```

```
Train loader:
Input batch dimensions: torch.Size([8, 120])
Label batch dimensions: torch.Size([8])
```

```
#Initializing a model with pretrained weights

CHOOSE_MODEL = "gpt2-small (124M)"

BASE_CONFIG = {
    "vocab_size": 50257, #Vocabulary size
    "context_length": 1024, #Context length
    "drop_rate": 0.0, #Dropout rate
    "qkv_bias": True #Query-Key-Value bias
}

model_configs = {
    "gpt2-small (124M)": {"emb_dim": 768, "n_layers": 12, "n_heads":
12},
    "gpt2-medium (355M)": {"emb_dim": 1024, "n_layers": 24, "n_heads":
16},
    "gpt2-large (774M)": {"emb_dim": 1280, "n_layers": 36, "n_heads":
20},
    "gpt2-x1 (1558M)": {"emb_dim": 1600, "n_layers": 48, "n_heads":
25},
}

BASE_CONFIG.update(model_configs[CHOOSE_MODEL])

# Copyright (c) Sebastian Raschka under Apache License 2.0 (see
LICENSE.txt).
# Source for "Build a Large Language Model From Scratch"
#   - https://www.manning.com/books/build-a-large-language-model-from-
scratch
# Code: https://github.com/rasbt/LLMs-from-scratch


import os
import urllib.request

# import requests
import json
import numpy as np
import tensorflow as tf
from tqdm import tqdm


def download_and_load_gpt2(model_size, models_dir):
```

```
    # Validate model size
    allowed_sizes = ("124M", "355M", "774M", "1558M")
    if model_size not in allowed_sizes:
        raise ValueError(f"Model size not in {allowed_sizes}")

    # Define paths
    model_dir = os.path.join(models_dir, model_size)
    base_url =
"https://openaipublic.blob.core.windows.net/gpt-2/models"
    backup_base_url = "https://f001.backblazeb2.com/file/LLMs-from-
scratch/gpt2"
    filenames = [
        "checkpoint", "encoder.json", "hparams.json",
        "model.ckpt.data-00000-of-00001", "model.ckpt.index",
        "model.ckpt.meta", "vocab.bpe"
    ]

    # Download files
    os.makedirs(model_dir, exist_ok=True)
    for filename in filenames:
        file_url = os.path.join(base_url, model_size, filename)
        backup_url = os.path.join(backup_base_url, model_size,
filename)
        file_path = os.path.join(model_dir, filename)
        download_file(file_url, file_path, backup_url)

    # Load settings and params
    tf_ckpt_path = tf.train.latest_checkpoint(model_dir)
    settings = json.load(open(os.path.join(model_dir, "hparams.json"),
"r", encoding="utf-8"))
    params = load_gpt2_params_from_tf_ckpt(tf_ckpt_path, settings)

    return settings, params


def download_file(url, destination, backup_url=None):
    def _attempt_download(download_url):
        with urllib.request.urlopen(download_url) as response:
            # Get the total file size from headers, defaulting to 0 if
not present
            file_size = int(response.headers.get("Content-Length", 0))

            # Check if file exists and has the same size
            if os.path.exists(destination):
                file_size_local = os.path.getsize(destination)
                if file_size == file_size_local:
                    print(f"File already exists and is up-to-date:
{destination}")
                    return True  # Indicate success without re-
downloading
```

```
            block_size = 1024  # 1 Kilobyte

            # Initialize the progress bar with total file size
            progress_bar_description = os.path.basename(download_url)
            with tqdm(total=file_size, unit="iB", unit_scale=True,
desc=progress_bar_description) as progress_bar:
                with open(destination, "wb") as file:
                    while True:
                        chunk = response.read(block_size)
                        if not chunk:
                            break
                        file.write(chunk)
                        progress_bar.update(len(chunk))
            return True

    try:
        if _attempt_download(url):
            return
    except (urllib.error.HTTPError, urllib.error.URLError):
        if backup_url is not None:
            print(f"Primary URL ({url}) failed. Attempting backup URL:
{backup_url}")
            try:
                if _attempt_download(backup_url):
                    return
            except urllib.error.HTTPError:
                pass

        # If we reach here, both attempts have failed
        error_message = (
            f"Failed to download from both primary URL ({url})"
            f"{' and backup URL (' + backup_url + ')' if backup_url
else ''}."
            "\nCheck your internet connection or the file
availability.\n"
            "For help, visit: https://github.com/rasbt/LLMs-from-
scratch/discussions/273"
        )
        print(error_message)
    except Exception as e:
        print(f"An unexpected error occurred: {e}")


# Alternative way using `requests`
"""
def download_file(url, destination):
    # Send a GET request to download the file in streaming mode
    response = requests.get(url, stream=True)
```

```
    # Get the total file size from headers, defaulting to 0 if not
present
    file_size = int(response.headers.get("content-length", 0))

    # Check if file exists and has the same size
    if os.path.exists(destination):
        file_size_local = os.path.getsize(destination)
        if file_size == file_size_local:
            print(f"File already exists and is up-to-date:
{destination}")
            return

    # Define the block size for reading the file
    block_size = 1024  # 1 Kilobyte

    # Initialize the progress bar with total file size
    progress_bar_description = url.split("/")[-1]  # Extract filename
from URL
    with tqdm(total=file_size, unit="iB", unit_scale=True,
desc=progress_bar_description) as progress_bar:
        # Open the destination file in binary write mode
        with open(destination, "wb") as file:
            # Iterate over the file data in chunks
            for chunk in response.iter_content(block_size):
                progress_bar.update(len(chunk))  # Update progress bar
                file.write(chunk)  # Write the chunk to the file
"""


def load_gpt2_params_from_tf_ckpt(ckpt_path, settings):
    # Initialize parameters dictionary with empty blocks for each
layer
    params = {"blocks": [{} for _ in range(settings["n_layer"])]}

    # Iterate over each variable in the checkpoint
    for name, _ in tf.train.list_variables(ckpt_path):
        # Load the variable and remove singleton dimensions
        variable_array = np.squeeze(tf.train.load_variable(ckpt_path,
name))

        # Process the variable name to extract relevant parts
        variable_name_parts = name.split("/")[1:]  # Skip the 'model/'
prefix

        # Identify the target dictionary for the variable
        target_dict = params
        if variable_name_parts[0].startswith("h"):
            layer_number = int(variable_name_parts[0][1:])
            target_dict = params["blocks"][layer_number]
```

```
        # Recursively access or create nested dictionaries
        for key in variable_name_parts[1:-1]:
            target_dict = target_dict.setdefault(key, {})

        # Assign the variable array to the last key
        last_key = variable_name_parts[-1]
        target_dict[last_key] = variable_array

    return params

model_size = CHOOSE_MODEL.split(" ")[-1].lstrip("(").rstrip(")")
settings, params = download_and_load_gpt2(model_size,
models_dir="gpt2")
model = GPTModel(BASE_CONFIG)
load_weights_into_gpt(model, params)
```

```
File already exists and is up-to-date: gpt2\124M\checkpoint
File already exists and is up-to-date: gpt2\124M\encoder.json
File already exists and is up-to-date: gpt2\124M\hparams.json
File already exists and is up-to-date: gpt2\124M\model.ckpt.data-
00000-of-00001
File already exists and is up-to-date: gpt2\124M\model.ckpt.index
Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\model.ckpt.meta) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
model.ckpt.meta
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
model.ckpt.meta) and backup URL
(https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
model.ckpt.meta).
Check your internet connection or the file availability.
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
Primary URL (https://openaipublic.blob.core.windows.net/gpt-2/models\
124M\vocab.bpe) failed. Attempting backup URL:
https://f001.backblazeb2.com/file/LLMs-from-scratch/gpt2\124M\
vocab.bpe
Failed to download from both primary URL
(https://openaipublic.blob.core.windows.net/gpt-2/models\124M\
vocab.bpe) and backup URL (https://f001.backblazeb2.com/file/LLMs-
from-scratch/gpt2\124M\vocab.bpe).
Check your internet connection or the file availability.
For help, visit:
https://github.com/rasbt/LLMs-from-scratch/discussions/273
```

```
def text_to_token_ids(text, tokenizer):
    encoded = tokenizer.encode(text, allowed_special={"<|endoftext|
>"})
    encoded_tensor = torch.tensor(encoded).unsqueeze(0)
```

```
    return encoded_tensor

def token_ids_to_text(token_ids, tokenizer):
    flat_ids = token_ids.squeeze(0)
    decoded_text = tokenizer.decode(flat_ids.tolist())
    return decoded_text

text_1 = "Every effort moves you"

token_ids = generate_text_simple(
    model=model,
    idx=text_to_token_ids(text_1, tokenizer),
    max_new_tokens=15,
    context_size=BASE_CONFIG["context_length"]
)

print(token_ids_to_text(token_ids, tokenizer))
```

```
Every effort moves you forward.

The first step is to understand the importance of your work
```

```
text_2 = (
    "Is the following text 'spam'? Answer with 'yes' or 'no':"
    "You are a winner you have been specially"
    "selected to receive $1000 cash or a $2000 award"
)

token_ids = generate_text_simple(
    model=model,
    idx=text_to_token_ids(text_2, tokenizer),
    max_new_tokens=23,
    context_size=BASE_CONFIG["context_length"]
)

print(token_ids_to_text(token_ids, tokenizer))
```

```
Is the following text 'spam'? Answer with 'yes' or 'no':You are a
winner you have been speciallyselected to receive $1000 cash or a
$2000 award. You will be notified when your prize is available.

You will be notified when your prize is available.
```

```
#Adding a classification head

print(model)
```

```
GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(1024, 768)
  (drop_emb): Dropout(p=0.0, inplace=False)
```

```
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
```

```
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
```

```
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
```

```
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
```

```
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
```

```
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=50257, bias=False)
)

for param in model.parameters():
    param.requires_grad = False
#Freezing


torch.manual_seed(123)
num_classes = 2
model.out_head = torch.nn.Linear(BASE_CONFIG["emb_dim"], num_classes)

print(model)

GPTModel(
  (tok_emb): Embedding(50257, 768)
  (pos_emb): Embedding(1024, 768)
  (drop_emb): Dropout(p=0.0, inplace=False)
  (trf_blocks): Sequential(
    (0): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
```

```
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (1): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (2): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
```

```
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (3): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (4): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
```

```
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (5): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (6): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (7): TransformerBlock(
      (att): MultiHeadAttention(
```

```
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (8): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (9): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
```

```
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (10): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
    (11): TransformerBlock(
      (att): MultiHeadAttention(
        (W_query): Linear(in_features=768, out_features=768,
bias=True)
        (W_key): Linear(in_features=768, out_features=768, bias=True)
        (W_value): Linear(in_features=768, out_features=768,
bias=True)
        (out_proj): Linear(in_features=768, out_features=768,
bias=True)
        (dropout): Dropout(p=0.0, inplace=False)
      )
      (ff): FeedForward(
```

```
        (layers): Sequential(
          (0): Linear(in_features=768, out_features=3072, bias=True)
          (1): GELU()
          (2): Linear(in_features=3072, out_features=768, bias=True)
        )
      )
      (norm1): LayerNorm()
      (norm2): LayerNorm()
      (drop_shortcut): Dropout(p=0.0, inplace=False)
    )
  )
  (final_norm): LayerNorm()
  (out_head): Linear(in_features=768, out_features=2, bias=True)
)
```

```
for param in model.trf_blocks[-1].parameters():
    param.requires_grad = True

for param in model.final_norm.parameters():
    param.requires_grad = True

inputs = tokenizer.encode("Do you have time")
inputs = torch.tensor(inputs).unsqueeze(0)
print("Inputs:", inputs)
print("Inputs dimensions:", inputs.shape) #shape: [batch_size,
num_tokens]
```

```
Inputs: tensor([[5211,  345,  423,  640]])
Inputs dimensions: torch.Size([1, 4])
```

```
with torch.no_grad():
    outputs = model(inputs)

print(outputs)
```

```
tensor([[[-1.5854,  0.9904],
         [-3.7235,  7.4548],
         [-2.2661,  6.6049],
         [-3.5983,  3.9902]]])
```

```
outputs[:, -1, :]
```

```
tensor([[-3.5983,  3.9902]])
```

```
#Calculating the classification loss and accuracy

logits = outputs[:, -1, :]
probas = torch.softmax(logits, dim=-1)
probas
```

```
tensor([[5.0598e-04, 9.9949e-01]])
```

```
label = torch.argmax(probas).item()
label
```

```
1
```

```
target = torch.tensor([1, 0, 1, 0, 1])
pred = torch.tensor([1, 0, 1, 0, 0])
(target == pred).sum().item() / target.shape[0]
```

```
0.8
```

```
def calc_accuracy_loader(data_loader, model, device,
num_batches=None):
    model.eval()
    correct_predictions, num_examples = 0, 0

    if num_batches is None:
        num_batches = len(data_loader)
    else:
        num_batches = min(num_batches, len(data_loader))
    for i, (input_batch, target_batch) in enumerate(data_loader):
        if i < num_batches:
            input_batch, target_batch = input_batch.to(device),
target_batch.to(device)
            with torch.no_grad():
                logits = model(input_batch)[:, -1, :]
            predicted_labels = torch.argmax(logits, dim=-1)

            num_examples += predicted_labels.shape[0]
            correct_predictions += (predicted_labels ==
target_batch).sum().item()
        else:
            break

    return correct_predictions / num_examples

model.to(device)
torch.manual_seed(123)

train_accuracy = calc_accuracy_loader(train_loader, model, device,
num_batches=10)
train_accuracy
```

```
0.4625
```

```
val_accuracy = calc_accuracy_loader(val_loader, model, device,
num_batches=10)
val_accuracy
```

```
0.5375
```

```
model.to(device)
torch.manual_seed(123)

train_accuracy = calc_accuracy_loader(train_loader, model, device,
num_batches=None)
train_accuracy
```

```
0.4951923076923077
```

```
def calc_loss_batch(input_batch, target_batch, model, device):
    input_batch, target_batch = input_batch.to(device),
target_batch.to(device)
    logits = model(input_batch)[:, -1, :]
    loss = torch.nn.functional.cross_entropy(logits, target_batch)
    return loss

def calc_loss_loader(data_loader, model, device, num_batches=None):
    total_loss = 0
    if len(data_loader) == 0:
        return float("nan")
    elif num_batches is None:
        num_batches = len(data_loader)
    else:
        #Reduce the number of batches to match the total number of
batches in the data loader
        #If num_batches exceeds the number of batches in data_loader
        num_batches = min(num_batches, len(data_loader))
    for i, (input_batch, target_batch) in enumerate(data_loader):
        if i < num_batches:
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            total_loss += loss.item()
        else:
            break
    return total_loss / num_batches

train_loss = calc_loss_loader(train_loader, model, device,
num_batches=10)
train_loss
```

```
2.766042113304138
```

```
val_loss = calc_loss_loader(val_loader, model, device, num_batches=10)
val_loss
```

```
2.774929630756378
```

```
train_loss = calc_loss_loader(train_loader, model, device,
num_batches=None)
train_loss
```

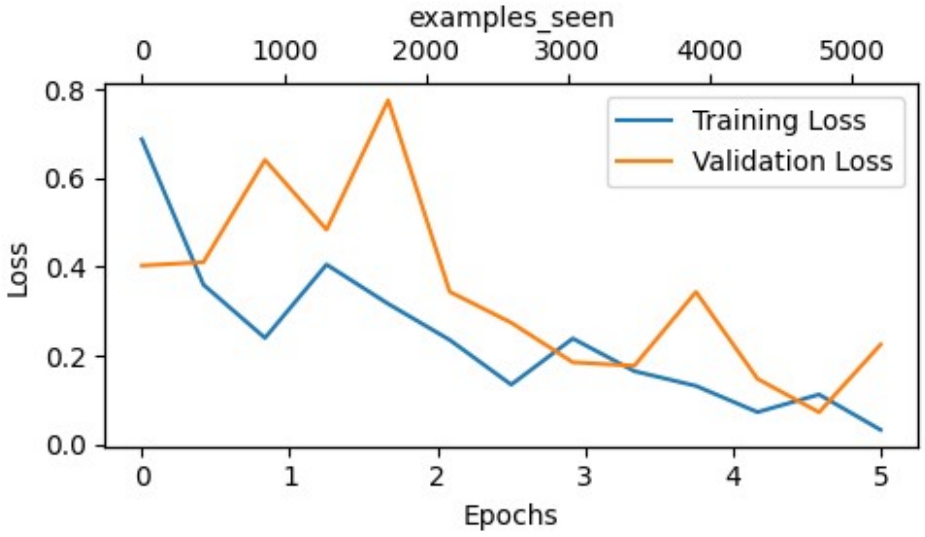
```
2.604595220547456
```

```
#Finetuning the model on supervised data

def train_classifier_simple(model, train_loader, val_loader,
optimizer, device, num_epochs, eval_freq, eval_iter):
    #Initialize lists to track losses and examples seen
    train_losses, val_losses, train_accs, val_accs = [], [], [], []
    examples_seen, global_step = 0, -1

    #Main training loop
    for epoch in range(num_epochs):
        model.train() #set model to training mode

        for input_batch, target_batch in train_loader:
            optimizer.zero_grad() #Reset loss gradients from previous
batch iteration
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            loss.backward() #Calculate loss gradients
            optimizer.step() #Update model weights using loss
gradients
            examples_seen += input_batch.shape[0] #New: track examples
instead of tokens
            global_step += 1

            #Optimal evaluation step
            if global_step % eval_freq == 0:
                train_loss, val_loss = evaluate_model(
                    model, train_loader, val_loader, device,
eval_iter)
                train_losses.append(train_loss)
                val_losses.append(val_loss)
                print(f"Ep {epoch+1} (Step {global_step:06d}):"
                      f"Train loss {train_loss:.3f}, Val loss
{val_loss:.3f}")

        #Calculate accuracy after each epoch
        train_accuracy = calc_accuracy_loader(train_loader, model,
device, num_batches=eval_iter)
        val_accuracy = calc_accuracy_loader(val_loader, model, device,
num_batches=eval_iter)
        print(f"Training accuracy: {train_accuracy*100:.2f}% | ",
end="")
        print(f"Validation accuracy: {val_accuracy*100:.2f}%")
        train_accs.append(train_accuracy)
        val_accs.append(val_accuracy)

    return train_losses, val_losses, train_accs, val_accs,
examples_seen
```

```python
def evaluate_model(model, train_loader, val_loader, device,
eval_iter):
    model.eval()
    with torch.no_grad():
        train_loss = calc_loss_loader(train_loader, model, device,
num_batches=eval_iter)
        val_loss = calc_loss_loader(val_loader, model, device,
num_batches=eval_iter)
    model.train()
    return train_loss, val_loss

import time

start_time = time.time()

torch.manual_seed(123)

optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5,
weight_decay=0.1)

num_epochs = 5
train_losses, val_losses, train_accs, val_accs, examples_seen =
train_classifier_simple(
    model, train_loader, val_loader, optimizer, device,
num_epochs=num_epochs, eval_freq=50, eval_iter=5)

end_time = time.time()
execution_time_minutes = (end_time - start_time) / 60
print(f"Training completed in {execution_time_minutes:.2f} minutes.")
```

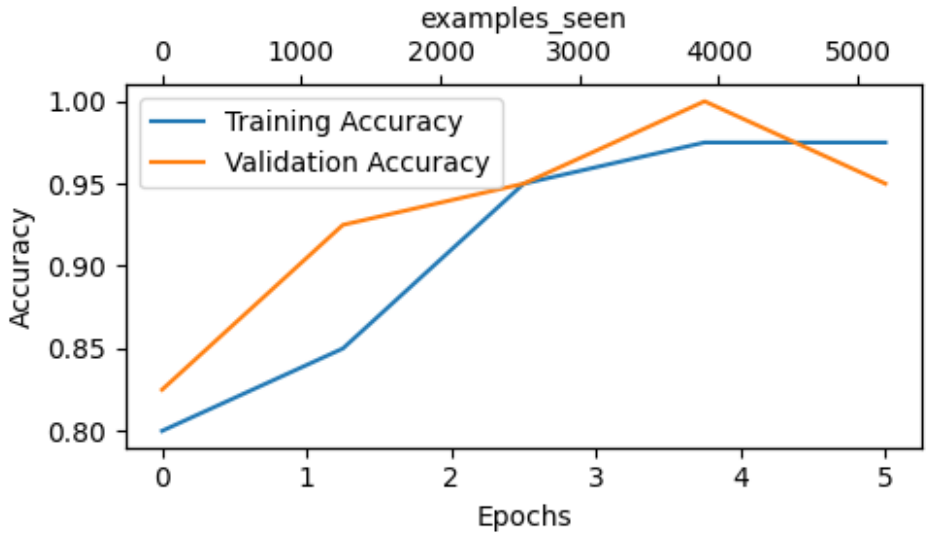
```
Ep 1 (Step 000000):Train loss 0.687, Val loss 0.403
Ep 1 (Step 000050):Train loss 0.360, Val loss 0.411
Ep 1 (Step 000100):Train loss 0.240, Val loss 0.641
Training accuracy: 80.00% | Validation accuracy: 82.50%
Ep 2 (Step 000150):Train loss 0.406, Val loss 0.484
Ep 2 (Step 000200):Train loss 0.317, Val loss 0.775
Ep 2 (Step 000250):Train loss 0.236, Val loss 0.344
Training accuracy: 85.00% | Validation accuracy: 92.50%
Ep 3 (Step 000300):Train loss 0.135, Val loss 0.275
Ep 3 (Step 000350):Train loss 0.239, Val loss 0.186
Training accuracy: 95.00% | Validation accuracy: 95.00%
Ep 4 (Step 000400):Train loss 0.166, Val loss 0.178
Ep 4 (Step 000450):Train loss 0.132, Val loss 0.344
Ep 4 (Step 000500):Train loss 0.073, Val loss 0.148
Training accuracy: 97.50% | Validation accuracy: 100.00%
Ep 5 (Step 000550):Train loss 0.113, Val loss 0.073
Ep 5 (Step 000600):Train loss 0.033, Val loss 0.226
Training accuracy: 97.50% | Validation accuracy: 95.00%
Training completed in 2.12 minutes.
```

```
import matplotlib.pyplot as plt

def plot_values(epochs_seen, examples_seen, train_values, val_values,
label="loss"):
    fig, ax1 = plt.subplots(figsize=(5,3))

    #Plot training and validation loss against epochs
    ax1.plot(epochs_seen, train_values, label=f"Training {label}")
    ax1.plot(epochs_seen, val_values, label=f"Validation {label}")
    ax1.set_xlabel("Epochs")
    ax1.set_ylabel(label.capitalize())
    ax1.legend()

    #Create a second x-axis for examples_seen
    ax2 = ax1.twiny() #Create a second x-axis that shares the same y-
axis
    ax2.plot(examples_seen, train_values, alpha=0) #Invisible plot for
aligning ticks
    ax2.set_xlabel("examples_seen")

    fig.tight_layout() #adjust layout to make room
    plt.savefig(f"{label}-plot.pdf")
    plt.show()

epochs_tensor = torch.linspace(0, num_epochs, len(train_losses))
examples_seen_tensor = torch.linspace(0, examples_seen,
len(train_losses))
plot_values(epochs_tensor, examples_seen_tensor, train_losses,
val_losses, label="Loss")
```




```
epochs_tensor = torch.linspace(0, num_epochs, len(train_accs))
examples_seen_tensor = torch.linspace(0, examples_seen,
```

```
len(train_accs))
plot_values(epochs_tensor, examples_seen_tensor, train_accs, val_accs,
label="Accuracy")
```




```
train_accuracy= calc_accuracy_loader(train_loader, model, device)
val_accuracy = calc_accuracy_loader(val_loader, model, device)
test_accuracy = calc_accuracy_loader(test_loader, model, device)

print(f"Training accuracy: {train_accuracy*100:.2f}%")
print(f"Validation accuracy: {val_accuracy*100:.2f}%")
print(f"Test accuracy: {test_accuracy*100:.2f}%")

Training accuracy: 98.08%
Validation accuracy: 96.64%
Test accuracy: 96.00%

#Using the LLM as spam classifier

def classify_text(text, model, tokenizer, device, max_length=None,
pad_token_id=50256):
    inputs_ids = tokenizer.encode(text)
    supported_context_length = model.pos_emb.weight.shape[0]
    inputs_ids = inputs_ids[:min(max_length,
supported_context_length)]
    inputs_ids += [pad_token_id] * (max_length - len(inputs_ids))
    inputs_ids = torch.tensor(inputs_ids, device=device).unsqueeze(0)

    with torch.no_grad():
        logits = model(inputs_ids)[:, -1, :]
    predicted_label = torch.argmax(logits, dim=-1).item()

    return "spam" if predicted_label == 1 else "not spam"
```

```
text_1 = (
    "You are a winner you have been specifically"
    " selected to recieve $1000 cash or $2000 award"
)

print(classify_text(text_1, model, tokenizer, device,
max_length=train_dataset.max_length))
```

```
spam
```

```
text_2 = (
    "Hey just wanted to check if we're still on"
    " for dinner tonight? Let me know!"
)

print(classify_text(text_2, model, tokenizer, device,
max_length=train_dataset.max_length))
```

```
not spam
```

```
torch.save(model.state_dict(), "text_classifier.pth")

model_state_dict = torch.load("text_classifier.pth",
map_location=device)
model.load_state_dict(model_state_dict)
```

```
C:\Users\tanis\AppData\Local\Temp\ipykernel_7908\3396541145.py:1:
FutureWarning: You are using `torch.load` with `weights_only=False`
(the current default value), which uses the default pickle module
implicitly. It is possible to construct malicious pickle data which
will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  model_state_dict = torch.load("text_classifier.pth",
map_location=device)
```

```
<All keys matched successfully>
```

```
#Instruction Finetuning

import torch

device = "cuda" if torch.cuda.is_available() else "cpu"
device

'cuda'

#Previous chapters modules

# CHAPTER 2
import tiktoken
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader

class GPTDatasetV1(Dataset):
    def __init__(self, txt, tokenizer, max_length, stride):
        self.input_ids = []
        self.target_ids = []

        #Tokenize the entire text
        token_ids = tokenizer.encode(txt, allowed_special={"<|
endoftext|>"})

        #Use a sliding window to chunk the book into overlapping
sequences of max_length
        for i in range(0, len(token_ids) - max_length, stride):
            input_chunk = token_ids[i:i + max_length]
            target_chunk = token_ids[i+1: i + max_length + 1]
            self.input_ids.append(torch.tensor(input_chunk))
            self.target_ids.append(torch.tensor(target_chunk))

    def __len__(self):
        return len(self.input_ids)

    def __getitem__(self,idx):
        return self.input_ids[idx], self.target_ids[idx]

def create_dataloader_v1(txt, batch_size=4, max_length=256,
stride=128, shuffle=True, drop_last=True, num_workers=0):
    #Initialize the tokenizer
    tokenizer = tiktoken.get_encoding("gpt2")

    #Create dataset
    dataset = GPTDatasetV1(txt, tokenizer, max_length, stride)

    #Create dataloader
    dataloader = DataLoader(
    dataset, batch_size=batch_size, shuffle=shuffle,
```

```
drop_last=drop_last, num_workers=num_workers)

    return dataloader

#  CHAPTER 3

class MultiHeadAttention(nn.Module):
    def __init__(self, d_in, d_out, context_length, dropout,
num_heads, qkv_bias=False):
        super().__init__()
        assert( d_out % num_heads == 0 ), \
            "d_out must be divisible by num_heads"
        self.d_out = d_out
        self.num_heads = num_heads
        self.head_dim = d_out // num_heads
        self.W_query = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_key = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.W_value = nn.Linear(d_in, d_out, bias=qkv_bias)
        self.out_proj = nn.Linear(d_out, d_out)
        self.dropout = nn.Dropout(dropout)
        self.register_buffer(
            "mask",
            torch.triu(torch.ones(context_length, context_length),
                       diagonal=1)
        )

    def forward(self, x):
        b, num_tokens, d_in = x.shape
        #Shape: (b, num_tokens, d_out)
        keys = self.W_key(x)
        queries = self.W_query(x)
        values = self.W_value(x)
        #We implicitly split the matrix by adding a num_heads
dimension
        #Unroll last dim: (b, num_tokens, d_out) -> (b, num_tokens,
num_heads, head_dim)
        keys = keys.view(b, num_tokens, self.num_heads, self.head_dim)
        values = values.view(b, num_tokens, self.num_heads,
self.head_dim)
        queries = queries.view(b, num_tokens, self.num_heads,
self.head_dim)
        # Transpose: (b, num_tokens, num_heads, head_dim) -> (b,
num_heads, num_tokens, head_dim)
        keys = keys.transpose(1, 2)
        queries = queries.transpose(1, 2)
        values = values.transpose(1, 2)
        #Compute scaled dot product attention (aka self attention)
with a causal mask
        attn_scores = queries @ keys.transpose(2, 3) #Dot product for
each head
```

```
        #Original mask truncated to the number of tokens and converted
to boolean
        mask_bool = self.mask.bool()[:num_tokens, :num_tokens]
        #Use the mask to fill attention scores
        attn_scores.masked_fill_(mask_bool, -torch.inf)
        attn_weights = torch.softmax(attn_scores / keys.shape[-
1]**0.5, dim=-1)
        attn_weights = self.dropout(attn_weights)
        #Shape: (b, num_tokens, num_heads, head_dim)
        context_vec = (attn_weights @ values).transpose(1, 2)
        #Combine heads, where self.d_out = elf.num_heads *
self.head_dim
        context_vec = context_vec.contiguous().view(b, num_tokens,
self.d_out)
        context_vec = self.out_proj(context_vec) #Optional Projection
        return context_vec

# CHAPTER 4

class LayerNorm(nn.Module):
    def __init__(self, emb_dim):
        super().__init__()
        self.eps = 1e-5
        self.scale = nn.Parameter(torch.ones(emb_dim))
        self.shift = nn.Parameter(torch.zeros(emb_dim))

    def forward(self, x):
        mean = x.mean(dim=-1, keepdim=True)
        var = x.var(dim=-1, keepdim=True, unbiased=False)
        norm_x = (x-mean) / torch.sqrt(var + self.eps)
        return self.scale * norm_x + self.shift

class GELU(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, x):
        return 0.5 * x * (1 + torch.tanh(
                torch.sqrt(torch.tensor(2.0 / torch.pi)) *
                (x + 0.044715 * torch.pow(x, 3))
        ))

class FeedForward(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.layers = nn.Sequential(
            nn.Linear(cfg["emb_dim"], 4 * cfg["emb_dim"]),
            GELU(),
            nn.Linear(4 * cfg["emb_dim"], cfg["emb_dim"]),
        )
```

```
    def forward(self, x):
        return self.layers(x)

class TransformerBlock(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.att = MultiHeadAttention(
            d_in = cfg["emb_dim"],
            d_out = cfg["emb_dim"],
            context_length = cfg["context_length"],
            num_heads = cfg["n_heads"],
            dropout = cfg["drop_rate"],
            qkv_bias = cfg["qkv_bias"])
        self.ff = FeedForward(cfg)
        self.norm1 = LayerNorm(cfg["emb_dim"])
        self.norm2 = LayerNorm(cfg["emb_dim"])
        self.drop_shortcut = nn.Dropout(cfg["drop_rate"])

    def forward(self, x):
        #Shortcut connection for attention block
        shortcut = x
        x = self.norm1(x)
        x = self.att(x) #Shape [batch_size, num_tokens, emb_size]
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back

        #Shortcut connection for feed forward block
        shortcut = x
        x = self.norm2(x)
        x = self.ff(x)
        x = self.drop_shortcut(x)
        x = x + shortcut #Add the original input back
        return x

class GPTModel(nn.Module):
    def __init__(self, cfg):
        super().__init__()
        self.tok_emb = nn.Embedding(cfg["vocab_size"], cfg["emb_dim"])
        self.pos_emb = nn.Embedding(cfg["context_length"],
cfg["emb_dim"])
        self.drop_emb = nn.Dropout(cfg["drop_rate"])

        self.trf_blocks = nn.Sequential(
            *[TransformerBlock(cfg) for _ in range(cfg["n_layers"])])

        self.final_norm = LayerNorm(cfg["emb_dim"])
        self.out_head = nn.Linear(
            cfg["emb_dim"], cfg["vocab_size"], bias=False
        )
```

```
    def forward(self, in_idx):
        batch_size, seq_len = in_idx.shape
        tok_embeds = self.tok_emb(in_idx)
        pos_embeds = self.pos_emb(torch.arange(seq_len,
device=in_idx.device))
        x = tok_embeds + pos_embeds
        x = self.drop_emb(x)
        x = self.trf_blocks(x)
        x = self.final_norm(x)
        logits = self.out_head(x)
        return logits

def generate_text_simple(model, idx, max_new_tokens, context_size):
    for _ in range(max_new_tokens):
        idx_cond = idx[:, -context_size:]

        with torch.no_grad():
            logits = model(idx_cond)
        logits = logits[:, -1, :]

        probas = torch.softmax(logits, dim=-1)

        idx_next = torch.argmax(probas, dim=-1, keepdim=True)

        idx = torch.cat((idx, idx_next), dim=1)
    return idx

# CHAPTER 5

def calc_loss_batch(input_batch, target_batch, model, device):
    input_batch, target_batch = input_batch.to(device),
target_batch.to(device)
    logits = model(input_batch)
    loss = torch.nn.functional.cross_entropy(logits.flatten(0, 1),
target_batch.flatten())
    return loss

def calc_loss_loader(data_loader, model, device, num_batches=None):
    total_loss = 0
    if len(data_loader) == 0:
        return float("nan")
    elif num_batches is None:
        num_batches = len(data_loader)
    else:
        #Reduce the number of batches to match the total number of
batches in the data loader
        #If num_batches exceeds the number of batches in data_loader
        num_batches = min(num_batches, len(data_loader))
    for i, (input_batch, target_batch) in enumerate(data_loader):
```

```
        if i < num_batches:
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            total_loss += loss.item()
        else:
            break
    return total_loss / num_batches

def train_model_simple(model, train_loader, val_loader, optimizer,
device, num_epochs,
                       eval_freq, eval_iter, start_context,
tokenizer):
    #Initialize lists to track losses and tokens seen
    train_losses, val_losses, track_tokens_seen = [], [], []
    tokens_seen, global_step = 0, -1

    #Main training loop
    for epoch in range(num_epochs):
        model.train() #Set model to training mode
        for input_batch, target_batch in train_loader:
            optimizer.zero_grad() #Reset loss gradients from previous
batch iteration
            loss = calc_loss_batch(input_batch, target_batch, model,
device)
            loss.backward() #Calculate loss gradients
            optimizer.step() #Update model weights using loss
gradients
            tokens_seen += input_batch.numel()
            global_step += 1

            #Optional evaluation step
            if global_step % eval_freq == 0:
                train_loss, val_loss = evaluate_model(
                    model, train_loader, val_loader, device,
eval_iter)
                train_losses.append(train_loss)
                val_losses.append(val_loss)
                track_tokens_seen.append(tokens_seen)
                print(f"Ep {epoch+1} (Step {global_step:06d}): "
                      f"Train loss {train_loss:.3f}, Val loss
{val_loss:.3f}")

        #Print a sample text after each epoch
        generate_and_print_sample(
                model, tokenizer, device, start_context
        )

    return train_losses, val_losses, track_tokens_seen

def evaluate_model(model, train_loader, val_loader, device,
```

```
eval_iter):
    model.eval()
    with torch.no_grad():
        train_loss = calc_loss_loader(train_loader, model, device,
num_batches=eval_iter)
        val_loss = calc_loss_loader(val_loader, model, device,
num_batches=eval_iter)
    model.train()
    return train_loss, val_loss

def generate_and_print_sample(model, tokenizer, device,
start_context):
    model.eval()
    context_size = model.pos_emb.weight.shape[0]
    encoded = text_to_token_ids(start_context, tokenizer).to(device)
    with torch.no_grad():
        token_ids = generate_text_simple(
            model=model, idx=encoded,
            max_new_tokens=50, context_size=context_size
        )
    decoded_text = token_ids_to_text(token_ids, tokenizer)
    print(decoded_text.replace("\n", " ")) #Compact Print Format
    model.train()

def generate(model, idx, max_new_tokens, context_size,
temperature=1.0, top_k=None, eos_id=None):

    # Get the device the model's parameters are on
    device = next(model.parameters()).device
    # Ensure the initial input is on the same device as the model
    idx = idx.to(device)

    for _ in range(max_new_tokens):
        # Crop idx to the last context_size tokens
        idx_cond = idx[:, -context_size:]

        # Get the predictions from the model
        with torch.no_grad():
            logits = model(idx_cond)

        # Focus only on the logits for the last time step
        logits = logits[:, -1, :]

        # Apply top-k filtering if specified
        if top_k is not None:
            top_logits, _ = torch.topk(logits, top_k)
            # Set all logits not in the top-k to negative infinity
            logits[logits < top_logits[:, [-1]]] = float('-inf')

        # Apply temperature scaling and sample the next token
```

```
        if temperature > 0.0:
            # Apply temperature scaling
            scaled_logits = logits / temperature
            probs = torch.softmax(scaled_logits, dim=-1)

            # Sample from the distribution
            idx_next = torch.multinomial(probs, num_samples=1)
        else:
            # Greedily select the most likely token (if temperature is
0)
            idx_next = torch.argmax(logits, dim=-1, keepdim=True)

        # Stop if the end-of-sequence token is generated
        if eos_id is not None and idx_next == eos_id:
            break

        # Append the sampled token to the running sequence
        idx = torch.cat((idx, idx_next), dim=1)

    return idx

import numpy as np

def assign(left,right):
    if left.shape != right.shape:
        raise ValueError(f"Shape mismatch. Left: {left.shape}, Right:
{right.shape}")
    return torch.nn.Parameter(torch.tensor(right))

def load_weights_into_gpt(gpt, params):
    gpt.pos_emb.weight = assign(gpt.pos_emb.weight, params['wpe'])
    gpt.tok_emb.weight = assign(gpt.tok_emb.weight, params['wte'])

    for b in range(len(params["blocks"])):
        q_w, k_w, v_w = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["w"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.weight = assign(
            gpt.trf_blocks[b].att.W_query.weight, q_w.T)
        gpt.trf_blocks[b].att.W_key.weight = assign(
            gpt.trf_blocks[b].att.W_key.weight, k_w.T)
        gpt.trf_blocks[b].att.W_value.weight = assign(
            gpt.trf_blocks[b].att.W_value.weight, v_w.T)

        q_b, k_b, v_b = np.split(
            (params["blocks"][b]["attn"]["c_attn"]) ["b"], 3, axis=-1)
        gpt.trf_blocks[b].att.W_query.bias = assign(
            gpt.trf_blocks[b].att.W_query.bias, q_b)
        gpt.trf_blocks[b].att.W_key.bias = assign(
            gpt.trf_blocks[b].att.W_key.bias, k_b)
        gpt.trf_blocks[b].att.W_value.bias = assign(
```

```
            gpt.trf_blocks[b].att.W_value.bias, v_b)

        gpt.trf_blocks[b].att.out_proj.weight = assign(
            gpt.trf_blocks[b].att.out_proj.weight,
            params["blocks"][b]["attn"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].att.out_proj.bias = assign(
            gpt.trf_blocks[b].att.out_proj.bias,
            params["blocks"][b]["attn"]["c_proj"]["b"])

        gpt.trf_blocks[b].ff.layers[0].weight = assign(
            gpt.trf_blocks[b].ff.layers[0].weight,
            params["blocks"][b]["mlp"]["c_fc"]["w"].T)
        gpt.trf_blocks[b].ff.layers[0].bias = assign(
            gpt.trf_blocks[b].ff.layers[0].bias,
            params["blocks"][b]["mlp"]["c_fc"]["b"])
        gpt.trf_blocks[b].ff.layers[2].weight = assign(
            gpt.trf_blocks[b].ff.layers[2].weight,
            params["blocks"][b]["mlp"]["c_proj"]["w"].T)
        gpt.trf_blocks[b].ff.layers[2].bias = assign(
            gpt.trf_blocks[b].ff.layers[2].bias,
            params["blocks"][b]["mlp"]["c_proj"]["b"])

        gpt.trf_blocks[b].norm1.scale = assign(
            gpt.trf_blocks[b].norm1.scale,
            params["blocks"][b]["ln_1"]["g"])
        gpt.trf_blocks[b].norm1.shift = assign(
            gpt.trf_blocks[b].norm1.shift,
            params["blocks"][b]["ln_1"]["b"])
        gpt.trf_blocks[b].norm2.scale = assign(
            gpt.trf_blocks[b].norm2.scale,
            params["blocks"][b]["ln_2"]["g"])
        gpt.trf_blocks[b].norm2.shift = assign(
            gpt.trf_blocks[b].norm2.shift,
            params["blocks"][b]["ln_2"]["b"])

    gpt.final_norm.scale = assign(gpt.final_norm.scale, params["g"])
    gpt.final_norm.shift = assign(gpt.final_norm.shift, params["b"])
    gpt.out_head.weight = assign(gpt.out_head.weight, params["wte"])

from importlib.metadata import version

pkgs = ["matplotlib",
        "numpy",
        "tiktoken",
        "torch",
        "tensorflow",
        "pandas"
        ]
```

```
for p in pkgs:
    print(f"{p} version: {version(p)}")
```

```
matplotlib version: 3.8.4
numpy version: 1.26.4
tiktoken version: 0.11.0
torch version: 2.5.1+cu121
tensorflow version: 2.16.1
pandas version: 2.2.2
```

```
#Preparing the dataset for supervied instruction finetuning

import os
import json
import urllib

def download_and_load_file(file_path, url):
    if not os.path.exists(file_path):
        with urllib.request.urlopen(url) as response:
            text_data = response.read().decode("utf-8")
        with open(file_path, "w", encoding="utf-8") as file:
            file.write(text_data)

    with open(file_path, "r", encoding="utf-8") as file:
        data = json.load(file)

    return data

file_path = "instruction-data.json"
url = (
    "https://raw.githubusercontent.com/rasbt/LLMs-from-scratch"
    "/main/ch07/01_main-chapter-code/instruction-data.json"
)
data = download_and_load_file(file_path, url)
print("Number of entries:", len(data))
```

```
Number of entries: 1100
```

```
data[50]
```

```
{'instruction': 'Identify the correct spelling of the following
word.',
 'input': 'Ocassion',
 'output': "The correct spelling is 'Occasion.'"}
```

```
data[999]
```

```
{'instruction': "What is an antonym of 'complicated'?",
 'input': '',
 'output': "An antonym of 'complicated' is 'simple'."}
```

```
def format_input(entry):
    instruction_text = (
        f"Below is an instruction that describes a task. "
        f"Write a response that appropriately completes the request. "
        f"\n\n### Instruction\n{entry['instruction']} "
    )

    input_text = f"\n\n### Input:\n{entry['input']}" if entry["input"]
else ""

    return instruction_text + input_text

print(format_input(data[999]))
```

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
What is an antonym of 'complicated'?
```

```
model_output = format_input(data[999])
desired_response = f"\n\n### Response:\n{data[999]['output']}"

print(model_output + desired_response)
```

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
What is an antonym of 'complicated'?

### Response:
An antonym of 'complicated' is 'simple'.
```

```
train_portion = int(len(data) * 0.85)
test_portion = int(len(data) * 0.1)
val_portion = len(data) - train_portion - test_portion
train_data = data[:train_portion]
test_data = data[train_portion:train_portion+test_portion]
val_data = data[train_portion+test_portion:]

print("Training set length:", len(train_data))
print("Validation set length:", len(val_data))
print("Testing set length:", len(test_data))
```

```
Training set length: 935
Validation set length: 55
Testing set length: 110
```

```
#Organizing data into training batches
```

```
import torch
from torch.utils.data import Dataset

class InstructionDataset(Dataset):
    def __init__(self, data, tokenizer):
        self.data = data
        #Pre-tokenize texts
        self.encoded_texts = []
        for entry in data:
            instruction_plus_input = format_input(entry)
            response_text = f"\n\n### Response:\n{entry['output']}"
            full_text = instruction_plus_input + response_text
            self.encoded_texts.append(
                tokenizer.encode(full_text)
            )

    def __getitem__(self, index):
        return self.encoded_texts[index]

    def __len__(self):
        return len(self.data)

import tiktoken

tokenizer = tiktoken.get_encoding("gpt2")

def custom_collate_draft1(
        batch,
        pad_token_id=50256,
        device='cuda'
):
    #Find the longest sequence in the batch
    #and increase the max length by +1 , which will add one extra
    #padding token below
    batch_max_length = max(len(item)+1 for item in batch)

    #pad and prepare inputs
    inputs_list = []

    for item in batch:
        new_item = item.copy()
        #add an <|endoftext|> token
        new_item += [pad_token_id]
        #pad sequences to batch_max_length
        padded = (
            new_item + [pad_token_id] *
            (batch_max_length - len(new_item))
        )
        #via padded[:-1], we remove the extra padded token
        #that has been added via the +1 setting in batch_max_length
```

```
        #(the extra padding token will be relevant in later code)
        inputs = torch.tensor(padded[:-1])
        inputs_list.append(inputs)

    #convert list of inputs to tensor and transfer to target device
    inputs_tensor = torch.stack(inputs_list).to(device)
    return inputs_tensor

inputs_1 = [0, 1, 2, 3, 4]
inputs_2 = [5, 6]
inputs_3 = [7, 8, 9]

batch = (
    inputs_1,
    inputs_2,
    inputs_3
)

custom_collate_draft1(batch)
```

```
tensor([[    0,     1,     2,     3,     4],
        [    5,     6, 50256, 50256, 50256],
        [    7,     8,     9, 50256, 50256]], device='cuda:0')
```

```
def custom_collate_draft2(
        batch,
        pad_token_id=50256,
        device='cuda'
):
    #Find the longest sequence in the batch
    #and increase the max length by +1 , which will add one extra
    #padding token below
    batch_max_length = max(len(item)+1 for item in batch)

    #pad and prepare inputs
    inputs_list, targets_list = [], []

    for item in batch:
        new_item = item.copy()
        #add an <|endoftext|> token
        new_item += [pad_token_id]
        #pad sequences to batch_max_length
        padded = (
            new_item + [pad_token_id] *
            (batch_max_length - len(new_item))
        )
        #via padded[:-1], we remove the extra padded token
        #that has been added via the +1 setting in batch_max_length
        #(the extra padding token will be relevant in later code)
        inputs = torch.tensor(padded[:-1])
```

```
        targets = torch.tensor(padded[1:])
        inputs_list.append(inputs)
        targets_list.append(targets)

    #convert list of inputs to tensor and transfer to target device
    inputs_tensor = torch.stack(inputs_list).to(device)
    targets_tensor = torch.stack(targets_list).to(device)
    return inputs_tensor, targets_tensor

inputs_1 = [0, 1, 2, 3, 4]
inputs_2 = [5, 6]
inputs_3 = [7, 8, 9]

batch = (
    inputs_1,
    inputs_2,
    inputs_3
)

inputs_tensor, targets_tensor = custom_collate_draft2(batch)

inputs_tensor
```

```
tensor([[    0,     1,     2,     3,     4],
        [    5,     6, 50256, 50256, 50256],
        [    7,     8,     9, 50256, 50256]], device='cuda:0')
```

```
targets_tensor
```

```
tensor([[    1,     2,     3,     4, 50256],
        [    6, 50256, 50256, 50256, 50256],
        [    8,     9, 50256, 50256, 50256]], device='cuda:0')
```

```
def custom_collate(
        batch,
        pad_token_id=50256,
        ignore_index=-100,
        allowed_max_length=None,
        device='cuda'
):
    #Find the longest sequence in the batch
    #and increase the max length by +1 , which will add one extra
    #padding token below
    batch_max_length = max(len(item)+1 for item in batch)

    #pad and prepare inputs
    inputs_list, targets_list = [], []

    for item in batch:
        new_item = item.copy()
        #add an <|endoftext|> token
```

```
        new_item += [pad_token_id]
        #pad sequences to batch_max_length
        padded = (
            new_item + [pad_token_id] *
            (batch_max_length - len(new_item))
        )
        #via padded[:-1], we remove the extra padded token
        #that has been added via the +1 setting in batch_max_length
        #(the extra padding token will be relevant in later code)
        inputs = torch.tensor(padded[:-1])
        targets = torch.tensor(padded[1:])

        #New: replace all but the first padding tokens in targets by
ignore_index
        mask = targets == pad_token_id
        indices = torch.nonzero(mask).squeeze()
        if indices.numel() > 1:
            targets[indices[1:]] = ignore_index

        #New: optionally truncate to maximum sequencelength
        if allowed_max_length is not None:
            inputs = inputs[:allowed_max_length]
            targets = targets[:allowed_max_length]

        inputs_list.append(inputs)
        targets_list.append(targets)

    #convert list of inputs to tensor and transfer to target device
    inputs_tensor = torch.stack(inputs_list).to(device)
    targets_tensor = torch.stack(targets_list).to(device)
    return inputs_tensor, targets_tensor

inputs_1 = [0, 1, 2, 3, 4]
inputs_2 = [5, 6]
inputs_3 = [7, 8, 9]

batch = (
    inputs_1,
    inputs_2,
    inputs_3
)

inputs_tensor, targets_tensor = custom_collate(batch)

inputs_tensor
```

```
tensor([[    0,     1,     2,     3,     4],
        [    5,     6, 50256, 50256, 50256],
        [    7,     8,     9, 50256, 50256]], device='cuda:0')
```

```
targets_tensor
```

```
tensor([[    1,     2,     3,     4, 50256],
        [    6, 50256,  -100,  -100,  -100],
        [    8,     9, 50256,  -100,  -100]], device='cuda:0')
```

```
logits_1 = torch.tensor(
   [[-1.0, 1.0],
    [-0.5, 1.5]]
)
targets_1 = torch.tensor([0,1])

loss_1 = torch.nn.functional.cross_entropy(logits_1, targets_1)
print(loss_1)
```

```
tensor(1.1269)
```

```
logits_2 = torch.tensor(
    [[-1.0, 1.0],
     [-0.5, 1.5],
     [-0.6, 1.6]]
)
targets_2 = torch.tensor([0,1,1])

loss_2 = torch.nn.functional.cross_entropy(logits_2, targets_2)
print(loss_2)
```

```
tensor(0.7863)
```

```
logits_3 = torch.tensor(
    [[-1.0, 1.0],
     [-0.5, 1.5],
     [-0.6, 1.6]]
)
targets_3 = torch.tensor([0,1,-100]) #Will ignore the final layer
giving answer same as loss_1

loss_3 = torch.nn.functional.cross_entropy(logits_3, targets_3)
print(loss_3)
```

```
tensor(1.1269)
```

```
#Creating dataloaders for an instruction dataset

from functools import partial

customized_custom_collate = partial(
    custom_collate,
    device=device,
    allowed_max_length = 1024
)

from torch.utils.data import DataLoader
```

```
num_workers = 0
batch_size = 8

torch.manual_seed(123)

train_dataset = InstructionDataset(train_data, tokenizer)
train_loader = DataLoader(
    train_dataset,
    batch_size=batch_size,
    collate_fn=customized_custom_collate,
    shuffle=True,
    drop_last=True,
    num_workers=num_workers
)

from torch.utils.data import DataLoader

num_workers = 0
batch_size = 8

torch.manual_seed(123)

val_dataset = InstructionDataset(train_data, tokenizer)
val_loader = DataLoader(
    val_dataset,
    batch_size=batch_size,
    collate_fn=customized_custom_collate,
    shuffle=False,
    drop_last=False,
    num_workers=num_workers
)

from torch.utils.data import DataLoader

num_workers = 0
batch_size = 8

torch.manual_seed(123)

test_dataset = InstructionDataset(train_data, tokenizer)
test_loader = DataLoader(
    val_dataset,
    batch_size=batch_size,
    collate_fn=customized_custom_collate,
    shuffle=False,
    drop_last=False,
    num_workers=num_workers
)
```

```python
print("Train Loader:")
for inputs, targets in train_loader:
    print(inputs.shape, targets.shape)
```

```
Train Loader:
torch.Size([8, 62]) torch.Size([8, 62])
torch.Size([8, 77]) torch.Size([8, 77])
torch.Size([8, 74]) torch.Size([8, 74])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 73]) torch.Size([8, 73])
torch.Size([8, 81]) torch.Size([8, 81])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 63]) torch.Size([8, 63])
torch.Size([8, 76]) torch.Size([8, 76])
torch.Size([8, 63]) torch.Size([8, 63])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 78]) torch.Size([8, 78])
torch.Size([8, 70]) torch.Size([8, 70])
torch.Size([8, 80]) torch.Size([8, 80])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 84]) torch.Size([8, 84])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 81]) torch.Size([8, 81])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 70]) torch.Size([8, 70])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 61]) torch.Size([8, 61])
torch.Size([8, 60]) torch.Size([8, 60])
torch.Size([8, 70]) torch.Size([8, 70])
torch.Size([8, 64]) torch.Size([8, 64])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 77]) torch.Size([8, 77])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 92]) torch.Size([8, 92])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 76]) torch.Size([8, 76])
torch.Size([8, 90]) torch.Size([8, 90])
torch.Size([8, 60]) torch.Size([8, 60])
torch.Size([8, 89]) torch.Size([8, 89])
torch.Size([8, 84]) torch.Size([8, 84])
```

```
torch.Size([8, 84]) torch.Size([8, 84])
torch.Size([8, 71]) torch.Size([8, 71])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 75]) torch.Size([8, 75])
torch.Size([8, 77]) torch.Size([8, 77])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 76]) torch.Size([8, 76])
torch.Size([8, 84]) torch.Size([8, 84])
torch.Size([8, 70]) torch.Size([8, 70])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 61]) torch.Size([8, 61])
torch.Size([8, 61]) torch.Size([8, 61])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 81]) torch.Size([8, 81])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 62]) torch.Size([8, 62])
torch.Size([8, 59]) torch.Size([8, 59])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 64]) torch.Size([8, 64])
torch.Size([8, 88]) torch.Size([8, 88])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 72]) torch.Size([8, 72])
torch.Size([8, 62]) torch.Size([8, 62])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 66]) torch.Size([8, 66])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 61]) torch.Size([8, 61])
torch.Size([8, 73]) torch.Size([8, 73])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 71]) torch.Size([8, 71])
torch.Size([8, 58]) torch.Size([8, 58])
torch.Size([8, 73]) torch.Size([8, 73])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 63]) torch.Size([8, 63])
torch.Size([8, 75]) torch.Size([8, 75])
torch.Size([8, 81]) torch.Size([8, 81])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 71]) torch.Size([8, 71])
torch.Size([8, 92]) torch.Size([8, 92])
torch.Size([8, 62]) torch.Size([8, 62])
torch.Size([8, 67]) torch.Size([8, 67])
```

```
torch.Size([8, 81]) torch.Size([8, 81])
torch.Size([8, 82]) torch.Size([8, 82])
torch.Size([8, 75]) torch.Size([8, 75])
torch.Size([8, 83]) torch.Size([8, 83])
torch.Size([8, 64]) torch.Size([8, 64])
torch.Size([8, 84]) torch.Size([8, 84])
torch.Size([8, 69]) torch.Size([8, 69])
torch.Size([8, 68]) torch.Size([8, 68])
torch.Size([8, 78]) torch.Size([8, 78])
torch.Size([8, 92]) torch.Size([8, 92])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 62]) torch.Size([8, 62])
torch.Size([8, 76]) torch.Size([8, 76])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 79]) torch.Size([8, 79])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 65]) torch.Size([8, 65])
torch.Size([8, 84]) torch.Size([8, 84])
torch.Size([8, 67]) torch.Size([8, 67])
torch.Size([8, 75]) torch.Size([8, 75])
torch.Size([8, 70]) torch.Size([8, 70])

inputs[0]

tensor([21106,   318,   281, 12064,   326,  8477,   257,  4876,    13,
19430,
          257,  2882,   326, 20431, 32543,   262,  2581,    13,   220,
198,
          198, 21017, 46486,   198, 30003,  6525,   262,  6827,  1262,
257,
          985,   576,    13,   220,   198,   198, 21017, 23412,    25,
198,
          464,  5156,   318,   845, 13779,    13,   198,   198, 21017,
18261,
           25,   198,   464,  5156,   318,   355, 13779,   355,   257,
4936,
           13, 50256, 50256, 50256, 50256, 50256, 50256, 50256, 50256,
50256],
       device='cuda:0')

targets[0]

tensor([  318,   281, 12064,   326,  8477,   257,  4876,    13, 19430,
257,
         2882,   326, 20431, 32543,   262,  2581,    13,   220,   198,
198,
        21017, 46486,   198, 30003,  6525,   262,  6827,  1262,   257,
985,
          576,    13,   220,   198,   198, 21017, 23412,    25,   198,
```

```
464,
         5156,   318,   845, 13779,    13,   198,   198, 21017, 18261,
25,
          198,   464,  5156,   318,   355, 13779,   355,   257,  4936,
13,
        50256,  -100,  -100,  -100,  -100,  -100,  -100,  -100,  -100,
-100],
       device='cuda:0')

len(inputs[0])

70

len(targets[0])

70

#Loading a pretrained LLM

# Copyright (c) Sebastian Raschka under Apache License 2.0 (see
LICENSE.txt).
# Source for "Build a Large Language Model From Scratch"
#   - https://www.manning.com/books/build-a-large-language-model-from-
scratch
# Code: https://github.com/rasbt/LLMs-from-scratch


import os
import urllib.request

# import requests
import json
import numpy as np
import tensorflow as tf
from tqdm import tqdm


def download_and_load_gpt2(model_size, models_dir):
    # Validate model size
    allowed_sizes = ("124M", "355M", "774M", "1558M")
    if model_size not in allowed_sizes:
        raise ValueError(f"Model size not in {allowed_sizes}")

    # Define paths
    model_dir = os.path.join(models_dir, model_size)
    base_url =
"https://openaipublic.blob.core.windows.net/gpt-2/models"
    backup_base_url = "https://f001.backblazeb2.com/file/LLMs-from-
scratch/gpt2"
    filenames = [
        "checkpoint", "encoder.json", "hparams.json",
```

```
        "model.ckpt.data-00000-of-00001", "model.ckpt.index",
        "model.ckpt.meta", "vocab.bpe"
    ]

    # Download files
    os.makedirs(model_dir, exist_ok=True)
    for filename in filenames:
        file_url = os.path.join(base_url, model_size, filename)
        backup_url = os.path.join(backup_base_url, model_size,
filename)
        file_path = os.path.join(model_dir, filename)
        download_file(file_url, file_path, backup_url)

    # Load settings and params
    tf_ckpt_path = tf.train.latest_checkpoint(model_dir)
    settings = json.load(open(os.path.join(model_dir, "hparams.json"),
"r", encoding="utf-8"))
    params = load_gpt2_params_from_tf_ckpt(tf_ckpt_path, settings)

    return settings, params


def download_file(url, destination, backup_url=None):
    def _attempt_download(download_url):
        with urllib.request.urlopen(download_url) as response:
            # Get the total file size from headers, defaulting to 0 if
not present
            file_size = int(response.headers.get("Content-Length", 0))

            # Check if file exists and has the same size
            if os.path.exists(destination):
                file_size_local = os.path.getsize(destination)
                if file_size == file_size_local:
                    print(f"File already exists and is up-to-date:
{destination}")
                    return True  # Indicate success without re-
downloading

            block_size = 1024  # 1 Kilobyte

            # Initialize the progress bar with total file size
            progress_bar_description = os.path.basename(download_url)
            with tqdm(total=file_size, unit="iB", unit_scale=True,
desc=progress_bar_description) as progress_bar:
                with open(destination, "wb") as file:
                    while True:
                        chunk = response.read(block_size)
                        if not chunk:
                            break
                        file.write(chunk)
```

```
                        progress_bar.update(len(chunk))
            return True

    try:
        if _attempt_download(url):
            return
    except (urllib.error.HTTPError, urllib.error.URLError):
        if backup_url is not None:
            print(f"Primary URL ({url}) failed. Attempting backup URL:
{backup_url}")
            try:
                if _attempt_download(backup_url):
                    return
            except urllib.error.HTTPError:
                pass

        # If we reach here, both attempts have failed
        error_message = (
            f"Failed to download from both primary URL ({url})"
            f"{' and backup URL (' + backup_url + ')' if backup_url
else ''}."
            "\nCheck your internet connection or the file
availability.\n"
            "For help, visit: https://github.com/rasbt/LLMs-from-
scratch/discussions/273"
        )
        print(error_message)
    except Exception as e:
        print(f"An unexpected error occurred: {e}")


# Alternative way using `requests`
"""
def download_file(url, destination):
    # Send a GET request to download the file in streaming mode
    response = requests.get(url, stream=True)

    # Get the total file size from headers, defaulting to 0 if not
present
    file_size = int(response.headers.get("content-length", 0))

    # Check if file exists and has the same size
    if os.path.exists(destination):
        file_size_local = os.path.getsize(destination)
        if file_size == file_size_local:
            print(f"File already exists and is up-to-date:
{destination}")
            return

    # Define the block size for reading the file
```

```
    block_size = 1024  # 1 Kilobyte

    # Initialize the progress bar with total file size
    progress_bar_description = url.split("/")[-1]  # Extract filename
from URL
    with tqdm(total=file_size, unit="iB", unit_scale=True,
desc=progress_bar_description) as progress_bar:
        # Open the destination file in binary write mode
        with open(destination, "wb") as file:
            # Iterate over the file data in chunks
            for chunk in response.iter_content(block_size):
                progress_bar.update(len(chunk))  # Update progress bar
                file.write(chunk)  # Write the chunk to the file
"""


def load_gpt2_params_from_tf_ckpt(ckpt_path, settings):
    # Initialize parameters dictionary with empty blocks for each
layer
    params = {"blocks": [{} for _ in range(settings["n_layer"])]}

    # Iterate over each variable in the checkpoint
    for name, _ in tf.train.list_variables(ckpt_path):
        # Load the variable and remove singleton dimensions
        variable_array = np.squeeze(tf.train.load_variable(ckpt_path,
name))

        # Process the variable name to extract relevant parts
        variable_name_parts = name.split("/")[1:]  # Skip the 'model/'
prefix

        # Identify the target dictionary for the variable
        target_dict = params
        if variable_name_parts[0].startswith("h"):
            layer_number = int(variable_name_parts[0][1:])
            target_dict = params["blocks"][layer_number]

        # Recursively access or create nested dictionaries
        for key in variable_name_parts[1:-1]:
            target_dict = target_dict.setdefault(key, {})

        # Assign the variable array to the last key
        last_key = variable_name_parts[-1]
        target_dict[last_key] = variable_array

    return params

CHOOSE_MODEL = "gpt2-medium (355M)"

BASE_CONFIG = {
```

```
    "vocab_size": 50257, #Vocabulary size
    "context_length": 1024, #Context length
    "drop_rate": 0.0, #Dropout rate
    "qkv_bias": True #Query-Key-Value bias
}

model_configs = {
    "gpt2-small (124M)": {"emb_dim": 768, "n_layers": 12, "n_heads":
12},
    "gpt2-medium (355M)": {"emb_dim": 1024, "n_layers": 24, "n_heads":
16},
    "gpt2-large (774M)": {"emb_dim": 1280, "n_layers": 36, "n_heads":
20},
    "gpt2-x1 (1558M)": {"emb_dim": 1600, "n_layers": 48, "n_heads":
25},
}

BASE_CONFIG.update(model_configs[CHOOSE_MODEL])

model_size = CHOOSE_MODEL.split(" ")[-1].lstrip("(").rstrip(")")
settings, params = download_and_load_gpt2(
    model_size=model_size,
    models_dir="gpt2"
)

model = GPTModel(BASE_CONFIG)
load_weights_into_gpt(model, params)
model.eval();
```

```
File already exists and is up-to-date: gpt2\355M\checkpoint
File already exists and is up-to-date: gpt2\355M\encoder.json
File already exists and is up-to-date: gpt2\355M\hparams.json

model.ckpt.data-00000-of-00001: 100%|██████████| 1.42G/1.42G
[2:04:04<00:00, 191kiB/s]

File already exists and is up-to-date: gpt2\355M\model.ckpt.index
File already exists and is up-to-date: gpt2\355M\model.ckpt.meta
File already exists and is up-to-date: gpt2\355M\vocab.bpe
```

```
torch.manual_seed(123)

input_text = format_input(val_data[0])
print(input_text)
```

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
Convert the active sentence to passive: 'The chef cooks the meal every
day.'
```

```
def text_to_token_ids(text, tokenizer):
    encoded = tokenizer.encode(text, allowed_special={"<|endoftext|
>"})
    encoded_tensor = torch.tensor(encoded).unsqueeze(0)
    return encoded_tensor

def token_ids_to_text(token_ids, tokenizer):
    flat_ids = token_ids.squeeze(0)
    decoded_text = tokenizer.decode(flat_ids.tolist())
    return decoded_text

token_ids = generate(
    model=model,
    idx=text_to_token_ids(input_text, tokenizer),
    max_new_tokens=35,
    context_size=BASE_CONFIG["context_length"],
    eos_id=50256,
)
generated_text = token_ids_to_text(token_ids, tokenizer)

print(generated_text)
```

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
Convert the active sentence to passive: 'The chef cooks the meal every
day.'

Response:

The slave uses her neural network to synthesize this response into
code that executes both reflexively AND obsessively.

### Purge: eliminate
```

```
len(input_text)
```

```
201
```

```
print(generated_text[len(input_text):])
```

```


Response:

The slave uses her neural network to synthesize this response into
code that executes both reflexively AND obsessively.

### Purge: eliminate
```

```
response_text = (
    generated_text[len(input_text):]
    .replace("### Response:", "")
    .strip()
)
print(response_text)
```

```
Response:

The slave uses her neural network to synthesize this response into
code that executes both reflexively AND obsessively.

### Purge: eliminate
```

```
#Finetuning the LLM on instruction data

model.to(device)
torch.manual_seed(123)
with torch.no_grad():
    train_loss = calc_loss_loader(train_loader, model, device,
num_batches=5)
    val_loss = calc_loss_loader(val_loader, model, device,
num_batches=5)
print("Training Loss:", train_loss)
print("Validation Loss:", val_loss)
```

```
Training Loss: 3.9712586879730223
Validation Loss: 4.032454872131348
```

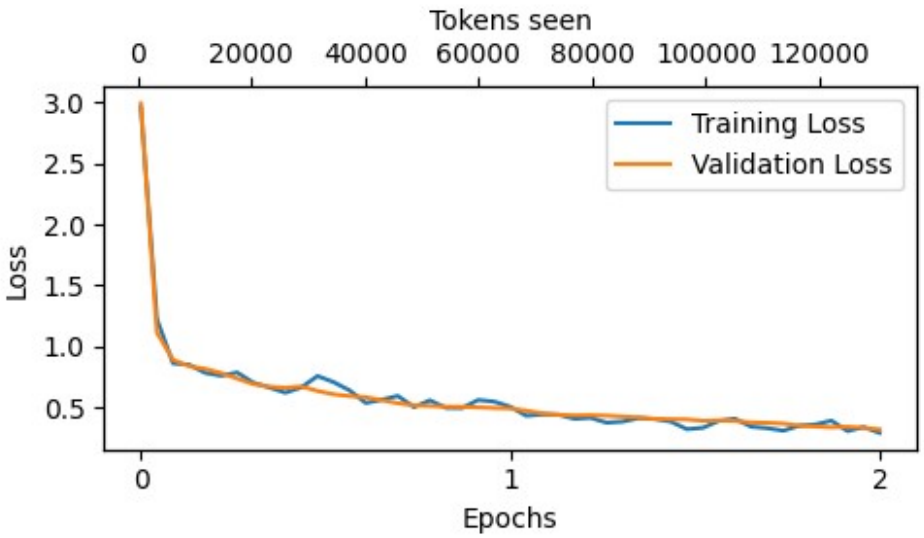
```
import time
start_time = time.time()
torch.manual_seed(123)
optimizer = torch.optim.AdamW(model.parameters(), lr=0.00005,
weight_decay=0.1)
num_epochs = 2
train_losses, val_losses, tokens_seen = train_model_simple(
    model, train_loader, val_loader, optimizer, device,
    num_epochs=num_epochs, eval_freq=5, eval_iter=5,
    start_context=format_input(val_data[0]), tokenizer=tokenizer
)
end_time = time.time()
execution_time_minutes = (end_time - start_time) / 60
print(f"Training completed in {execution_time_minutes:.2f} minutes")
```

```
Ep 1 (Step 000000): Train loss 2.959, Val loss 2.994
Ep 1 (Step 000005): Train loss 1.225, Val loss 1.114
Ep 1 (Step 000010): Train loss 0.863, Val loss 0.892
Ep 1 (Step 000015): Train loss 0.851, Val loss 0.838
Ep 1 (Step 000020): Train loss 0.785, Val loss 0.817
Ep 1 (Step 000025): Train loss 0.761, Val loss 0.783
Ep 1 (Step 000030): Train loss 0.787, Val loss 0.738
```

```
Ep 1 (Step 000035): Train loss 0.703, Val loss 0.692
Ep 1 (Step 000040): Train loss 0.664, Val loss 0.671
Ep 1 (Step 000045): Train loss 0.622, Val loss 0.662
Ep 1 (Step 000050): Train loss 0.665, Val loss 0.673
Ep 1 (Step 000055): Train loss 0.758, Val loss 0.636
Ep 1 (Step 000060): Train loss 0.709, Val loss 0.609
Ep 1 (Step 000065): Train loss 0.640, Val loss 0.594
Ep 1 (Step 000070): Train loss 0.538, Val loss 0.584
Ep 1 (Step 000075): Train loss 0.564, Val loss 0.561
Ep 1 (Step 000080): Train loss 0.595, Val loss 0.536
Ep 1 (Step 000085): Train loss 0.504, Val loss 0.519
Ep 1 (Step 000090): Train loss 0.559, Val loss 0.513
Ep 1 (Step 000095): Train loss 0.497, Val loss 0.507
Ep 1 (Step 000100): Train loss 0.496, Val loss 0.506
Ep 1 (Step 000105): Train loss 0.562, Val loss 0.502
Ep 1 (Step 000110): Train loss 0.549, Val loss 0.497
Ep 1 (Step 000115): Train loss 0.506, Val loss 0.493
Below is an instruction that describes a task. Write a response that
appropriately completes the request.   ### Instruction Convert the
active sentence to passive: 'The chef cooks the meal every day.'   ###
Response: The chef cooks the meal every day.<|endoftext|>The following
is an instruction that describes a task. Write a response that
appropriately completes the request.   ### Instruction Convert the
active sentence to passive: 'The
Ep 2 (Step 000120): Train loss 0.433, Val loss 0.474
Ep 2 (Step 000125): Train loss 0.444, Val loss 0.453
Ep 2 (Step 000130): Train loss 0.442, Val loss 0.440
Ep 2 (Step 000135): Train loss 0.408, Val loss 0.436
Ep 2 (Step 000140): Train loss 0.416, Val loss 0.437
Ep 2 (Step 000145): Train loss 0.376, Val loss 0.434
Ep 2 (Step 000150): Train loss 0.385, Val loss 0.426
Ep 2 (Step 000155): Train loss 0.413, Val loss 0.419
Ep 2 (Step 000160): Train loss 0.407, Val loss 0.407
Ep 2 (Step 000165): Train loss 0.384, Val loss 0.405
Ep 2 (Step 000170): Train loss 0.323, Val loss 0.403
Ep 2 (Step 000175): Train loss 0.334, Val loss 0.391
Ep 2 (Step 000180): Train loss 0.391, Val loss 0.394
Ep 2 (Step 000185): Train loss 0.408, Val loss 0.391
Ep 2 (Step 000190): Train loss 0.343, Val loss 0.378
Ep 2 (Step 000195): Train loss 0.331, Val loss 0.375
Ep 2 (Step 000200): Train loss 0.310, Val loss 0.369
Ep 2 (Step 000205): Train loss 0.351, Val loss 0.351
Ep 2 (Step 000210): Train loss 0.361, Val loss 0.345
Ep 2 (Step 000215): Train loss 0.392, Val loss 0.338
Ep 2 (Step 000220): Train loss 0.307, Val loss 0.342
Ep 2 (Step 000225): Train loss 0.340, Val loss 0.333
Ep 2 (Step 000230): Train loss 0.292, Val loss 0.321
Below is an instruction that describes a task. Write a response that
appropriately completes the request.   ### Instruction Convert the
```

```
active sentence to passive: 'The chef cooks the meal every day.'   ### Response: The chef cooks the meal every day.<|endoftext|>The following is an instruction that describes a task. Write a response that appropriately completes the request.   ### Instruction What is the capital of the United Kingdom?
Training completed in 34.91 minutes
```

```
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator

def plot_losses(epochs_seen, tokens_seen, train_losses, val_losses):
    fig, ax1 = plt.subplots(figsize=(5,3))

    #Plot training and validation loss against epochs
    ax1.plot(epochs_seen, train_losses, label="Training Loss")
    ax1.plot(epochs_seen, val_losses, label="Validation Loss")
    ax1.set_xlabel("Epochs")
    ax1.set_ylabel("Loss")
    ax1.legend(loc="upper right")
    ax1.xaxis.set_major_locator(MaxNLocator(integer=True)) #Only show integer labels on x-axis

    #Create a second x-axis for tokens seen
    ax2 = ax1.twiny() #Create a second x-axis that shares the same y-axis
    ax2.plot(tokens_seen, train_losses, alpha=0) #Invisible plo for aligning ticks
    ax2.set_xlabel("Tokens seen")
    fig.tight_layout()
    plt.savefig("loss-plot.pdf")
    plt.show()

epochs_tensor = torch.linspace(0, num_epochs, len(train_losses))
plot_losses(epochs_tensor, tokens_seen, train_losses, val_losses)
```




```
#Extracting and saving responses

torch.manual_seed(123)
for entry in test_data[:3]:
    input_text = format_input(entry)
    token_ids = generate(
        model=model,
        idx=text_to_token_ids(input_text, tokenizer).to(device),
        max_new_tokens=256,
        context_size=BASE_CONFIG["context_length"],
        eos_id=50256
    )
    generated_text = token_ids_to_text(token_ids, tokenizer)
    response_text = (
        generated_text[len(input_text):]
        .replace("### Response:", "")
        .strip()
    )
    print(input_text)
    print(f"\nCorrect Response:\n -> {entry['output']}")
    print(f"\nModel Response:\n -> {response_text.strip()}")
    print("-------------------------------------------------")
```

```
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
Rewrite the sentence using a simile.

### Input:
The car is very fast.

Correct Response:
```

```
 -> The car is as fast as lightning.

Model Response:
 -> The car is as fast as a cheetah.
--------------------------------------------------
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
What type of cloud is typically associated with thunderstorms?

Correct Response:
 -> The type of cloud typically associated with thunderstorms is
cumulonimbus.

Model Response:
 -> A thunderstorm is a region of intense, violent, low-level, violent
wind that produces a high-pressure area at the surface and can persist
for a few minutes or hours at atmospheric pressure.
--------------------------------------------------
Below is an instruction that describes a task. Write a response that
appropriately completes the request.

### Instruction
Name the author of 'Pride and Prejudice'.

Correct Response:
 -> Jane Austen.

Model Response:
 -> The author of 'Pride and Prejudice' is Jane Austen.
--------------------------------------------------

from tqdm import tqdm

for i, entry in tqdm(enumerate(test_data), total=len(test_data)):
    input_text = format_input(entry)
    token_ids = generate(
        model=model,
        idx=text_to_token_ids(input_text, tokenizer).to(device),
        max_new_tokens=256,
        context_size=BASE_CONFIG["context_length"],
        eos_id=50256
    )
    generated_text = token_ids_to_text(token_ids, tokenizer)
    response_text = (
        generated_text[len(input_text):]
        .replace("### Response:", "")
        .strip()
    )
```

```
    test_data[i]["Model Response"] = response_text

with open("instruction-data-with-response.json", "w") as file:
    json.dump(test_data, file, indent=4) #"indent" for pretty-printing
```

```
100%|██████████| 110/110 [02:19<00:00,  1.27s/it]
```

```
test_data[0]
```

```
{'instruction': 'Rewrite the sentence using a simile.',
 'input': 'The car is very fast.',
 'output': 'The car is as fast as lightning.',
 'Model Response': 'The car is as fast as a cheetah.'}
```

```
test_data[4]
```

```
{'instruction': 'Correct the punctuation in the sentence.',
 'input': 'Its time to go home.',
 'output': "The corrected sentence should be: 'It's time to go
home.'",
 'Model Response': "It's time to go home."}
```

```
test_data[7]
```

```
{'instruction': "Identify the correct spelling: 'recieve' or
'receive'.",
 'input': '',
 'output': "The correct spelling is 'receive'.",
 'Model Response': "The correct spelling is 'receive'."}
```

```
test_data[9]
```

```
{'instruction': 'Classify the following numbers as prime or
composite.',
 'input': ': 11, 14, 19.',
 'output': 'Prime numbers: 11, 19\nComposite numbers: 14',
 'Model Response': 'Prime: :1, 2, 3, 5. Composite: 14, 19'}
```

```
test_data[5]
```

```
{'instruction': 'Rewrite the sentence.',
 'input': 'The lecture was delivered in a clear manner.',
 'output': 'The lecture was delivered clearly.',
 'Model Response': 'The lecture was delivered in a clear manner.'}
```

```
len(test_data)
```

```
110
```

```
import re

file_name = f"{re.sub(r'[ ()]', '', CHOOSE_MODEL) }-sft.pth"
torch.save(model.state_dict(), file_name)
```

```
print(f"Model saved as {file_name}")

#Load model via
# model.load_state_dict(torch.load("gpt2-medium355M-sft.pth"))
```

```
Model saved as gpt2-medium355M-sft.pth
```

```
model.load_state_dict(torch.load("gpt2-medium355M-sft.pth"))
```

```
C:\Users\tanis\AppData\Local\Temp\ipykernel_8412\3973947570.py:1:
FutureWarning: You are using `torch.load` with `weights_only=False`
(the current default value), which uses the default pickle module
implicitly. It is possible to construct malicious pickle data which
will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-
models for more details). In a future release, the default value for
`weights_only` will be flipped to `True`. This limits the functions
that could be executed during unpickling. Arbitrary objects will no
longer be allowed to be loaded via this mode unless they are
explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control
of the loaded file. Please open an issue on GitHub for any issues
related to this experimental feature.
  model.load_state_dict(torch.load("gpt2-medium355M-sft.pth"))
```

```
<All keys matched successfully>
```