Introduction

In this notebook, we'll explore how to train a lightweight NanoGPT on the Tiny Stories dataset. NanoGPT, developed by Andrej Karpathy, is a simplified variant of GPT designed for simplicity and speed. Our goal is to generate creative and coherent text based on the input data. I will also use Wandb library to track the experiments on the model.

I would like to acknowledge that I currently do not have a clear understanding of how to fine-tune and operate NanoGPT. While I am familiar with various aspects of AI and machine learning, this specific area is one I have not yet had the opportunity to explore in depth. However, I am eager to learn and improve my skills in this area, and I am confident that with the I would like to express my sincere gratitude for the assignments you have entrusted me with. While I acknowledge that not all assignments may be fully complete, and despite seeking help from ChatGPT in some instances, I am truly committed to improving my work. I deeply value your feedback and would greatly appreciate any remarks or suggestions on areas where I can improve. Your guidance is invaluable to me, and I am eager to refine my approach and develop a deeper understanding of the tasks at hand. Thank you for your continued support and the opportunity to learn from this experience.right resources and guidance, I will be able to grasp and apply the necessary techniques in fine-tuning and operating NanoGPT.

Import Libraries

```
# Clone the NanoGPT
!git clone https://github.com/karpathy/nanoGPT.git
!pip install tiktoken
!pip install wandb
```

```
Cloning into 'nanoGPT'...
remote: Enumerating objects: 682, done.
remote: Total 682 (delta 0), reused 0 (delta 0), pack-reused 682 (from 1)
Receiving objects: 100% (682/682), 952.47 KiB | 9.16 MiB/s, done.
Resolving deltas: 100% (385/385), done.
Collecting tiktoken
  Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2024.9.11)
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2024.8.3
Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 22.2 MB/s eta 0:00:00
Installing collected packages: tiktoken
Successfully installed tiktoken-0.8.0
Requirement already satisfied: wandb in /usr/local/lib/python3.10/dist-packages (0.18.7)
Requirement already satisfied: click!=8.0.0,>=7.1 in /usr/local/lib/python3.10/dist-packages (from wandb) (8.1.7)
Requirement already satisfied: docker-pycreds>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: gitpython!=3.1.29,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (3.1.43)
Requirement already satisfied: platformdirs in /usr/local/lib/python3.10/dist-packages (from wandb) (4.3.6)
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (4.25.5)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from wandb) (6.0.2)
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (2.32.3)
Requirement already satisfied: sentry-sdk>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (2.18.0)
Requirement already satisfied: setproctitle in /usr/local/lib/python3.10/dist-packages (from wandb) (1.3.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from wandb) (75.1.0)
Requirement already satisfied: typing-extensions<5,>=4.4 in /usr/local/lib/python3.10/dist-packages (from wandb) (4.12.2)
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0->wandb) (1.16.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from gitpython!=3.1.29,>=1.0.0->wandb) (4.0.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (3.4
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (2024.8.30
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.29,>=1.0
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
from tqdm.auto import tqdm
from contextlib import nullcontext
import nanoGPT.model as GPT
```

```
import wandb
import os


# Directly set your API key (not secure in public notebooks)
import wandb
wandb.login(key="2c31d7e5323a64ac198ab2499a802513a1ac5ec8")
```

```
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: W&B API key is configured. Use `wandb login --relogin` to force relogin
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
True
```

Default title text

```
# @title Default title text
class GPTConfig: # Model config from NanoGPT
    block_size: int = 124  # this helps get context for training , reduce to 124 for faster training
    vocab_size: int = 50304 # GPT-2 vocab_size of 50257, padded up to nearest multiple of 64 for efficiency
    n_layer: int = 6 # reduce layers to reduce complexity
    n_head: int = 6 # reduce heads to reduce complexity for smaller model
    n_embd: int = 768 # reduce embedding size or faster computation since model becomes 'lighter' in terms of complexity
    dropout: float = 0.0 # since we have already worked on previous steps to make model as simple as possible , dropout is kept as is
    bias: bool = False # True: bias in Linears and LayerNorms, like GPT-2. False: a bit better and faster

config = GPTConfig


config = GPTConfig()  # Create an instance of GPTConfig
wandb_config = {k: v for k, v in vars(config).items() if not callable(getattr(config, k)) and not k.startswith("__")}
wandb_config  # This will output the dictionary form of config
```

```
{}
```

```
wandb.init(project="nanogpt-tinystories", name="nanoGPT", config=wandb_config)
```

```
wandb: Currently logged in as: raj-dandekar8 (raj-dandekar8-massachusetts-institute-of-technology). Use `wandb login --relogin` to force
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241205_143748-inpu8mb0
Syncing run nanoGPT to Weights & Biases (docs)
View project at https://wandb.ai/raj-dandekar8-massachusetts-institute-of-technology/nanogpt-tinystories
View run at https://wandb.ai/raj-dandekar8-massachusetts-institute-of-technology/nanogpt-tinystories/runs/inpu8mb0
Display W&B run
```

Tiny Stories datasets and preprocessing

TinyStories, a synthetic dataset of short stories that only contain words that a typical 3 to 4-year-olds usually understand, generated by GPT-3.5 and GPT-4. We show that TinyStories can be used to train and evaluate LMs that are much smaller than the state-of-the-art models (below 10 million total parameters)

```
!pip install datasets
```

```
from datasets import load_dataset
```

```
ds = load_dataset("roneneldan/TinyStories")
```

```
Collecting datasets
  Downloading datasets-3.1.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.6)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.11.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0->data
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2024.8.3
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1.16
Downloading datasets-3.1.0-py3-none-any.whl (480 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 29.9 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.3 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 16.1 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 13.3 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 17.9 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which is incompatible.
Successfully installed datasets-3.1.0 dill-0.3.8 fsspec-2024.9.0 multiprocess-0.70.16 xxhash-3.5.0
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secre
```

```
import tiktoken
import os
import numpy as np
from tqdm.auto import tqdm

enc = tiktoken.get_encoding("gpt2")

# Some functions from https://github.com/karpathy/nanoGPT/blob/master/data/openwebtext/prepare.py

def process(example):
    ids = enc.encode_ordinary(example['text'])  # Ignore special tokens
    out = {'ids': ids, 'len': len(ids)}
    return out

# Check dataset size before tokenizing
train_size = len(ds['train'])
val_size = len(ds['validation'])

# Print dataset size for reference
print(f"Training set size: {train_size} examples")
print(f"Validation set size: {val_size} examples")

# Adjust batch size and parallelization based on dataset size
```

```
num_proc = 8 if train_size > 100000 else 4
total_batches = 1024 if train_size > 100000 else 512

# Only tokenize and process if the binary files do not already exist
if not os.path.exists("train.bin"):
    tokenized = ds.map(
        process,
        remove_columns=['text'],
        desc="Tokenizing the splits",
        num_proc=num_proc,
    )

    # Concatenate all the ids in each dataset into one large file we can use for training
    for split, dset in tokenized.items():
        arr_len = np.sum(dset['len'], dtype=np.uint64)
        filename = f'{split}.bin'
        dtype = np.uint16  # (GPT-2 vocab size is 50256, so 16-bit unsigned integers are enough)
        arr = np.memmap(filename, dtype=dtype, mode='w+', shape=(arr_len,))

        idx = 0
        for batch_idx in tqdm(range(total_batches), desc=f'Writing {filename}'):
            # Efficient batch processing to save memory
            batch = dset.shard(num_shards=total_batches, index=batch_idx, contiguous=True).with_format('numpy')
            arr_batch = np.concatenate(batch['ids'])
            arr[idx: idx + len(arr_batch)] = arr_batch
            idx += len(arr_batch)

        # Ensure the memory-mapped array is saved to disk
        arr.flush()

else:
    print("Binary files already exist. Skipping tokenization and writing.")
```

```
Training set size: 2119719 examples
Validation set size: 21990 examples
Tokenizing the splits (num_proc=8): 100%  2119719/2119719 [07:39<00:00, 4761.11 examples/s]
Tokenizing the splits (num_proc=8): 100%  21990/21990 [00:06<00:00, 4540.97 examples/s]
Writing train.bin: 100%  1024/1024 [00:26<00:00, 48.47it/s]
Writing validation.bin: 100%  1024/1024 [00:02<00:00, 421.92it/s]
```

Training

```
def get_batch(split):
    """
    This function fetches a batch of data using chunking for improved I/O efficiency.
    Optimization includes:
    - Preloading larger chunks of data into memory.
    - Fetching batches sequentially from the preloaded chunk to reduce I/O overhead.
    """
    # Use the correct file depending on the split
    data_file = 'train.bin' if split == 'train' else 'validation.bin'

    # Preload a chunk of data into memory for more efficient access
    chunk_size = block_size * batch_size * 10  # Example: Load 10 batches at once (adjust as needed)
    data = np.memmap(data_file, dtype=np.uint16, mode='r')

    # Randomly sample a starting index, ensuring that we don't go past the chunk size
    ix = torch.randint(len(data) - chunk_size, (batch_size,))

    # We create a chunk of data and slice it to get the batches
    chunk = data[ix[0]: ix[0] + chunk_size]  # Load a chunk from the data

    # Efficiently slice the chunk into individual batches for X and Y
    x = torch.from_numpy(chunk[:batch_size * block_size]).reshape(batch_size, block_size).to(torch.int64)
    y = torch.from_numpy(chunk[batch_size: batch_size + batch_size * block_size]).reshape(batch_size, block_size).to(torch.int64)

    # Move to GPU (with non-blocking for faster transfer)
    if device_type == 'cuda':
        x, y = x.pin_memory().to(device, non_blocking=True), y.pin_memory().to(device, non_blocking=True)
    else:
```

```
            x, y = x.to(device), y.to(device)

        return x, y



    # Notes:
    # 1. Memory-Mapped File (`np.memmap`) is used to handle large datasets efficiently by reading them in chunks.
    # 2. Pinning memory (`pin_memory()`) is applied only to CPU tensors to speed up data transfer to GPU.
    #    This avoids errors when trying to pin already GPU-bound tensors.
    # 3. The `get_batch()` function selects random batches from the dataset (`train.bin` or `validation.bin`).
    #    The slicing method ensures minimal memory overhead by reading large chunks in a single operation.
    # 4. The `to(device)` method ensures tensors are moved to the right device (GPU or CPU).
    #    This is needed when training on a CUDA-enabled device. For non-CUDA (CPU), tensors are processed directly.

    def get_batch(split, chunk_size=1024):
        """
        This function fetches a batch of data from a memory-mapped file.
        Optimization includes:
        - Avoiding excessive memory access (sequential access improves performance).
        - Minimizing tensor conversions and unnecessary operations.
        """
        # Use the correct file depending on the split
        data_file = 'train.bin' if split == 'train' else 'validation.bin'
        data = np.memmap(data_file, dtype=np.uint16, mode='r')

        # Randomly sample indices for the batch
        ix = torch.randint(len(data) - block_size, (batch_size,))

        # Fetch the batches efficiently by using slicing instead of multiple individual numpy accesses
        x = torch.tensor([data[i:i+block_size] for i in ix], dtype=torch.int64)
        y = torch.tensor([data[i+1:i+1+block_size] for i in ix], dtype=torch.int64)

        # Transfer to device, pinning memory only for CPU tensors
        if device_type == 'cuda':
            # Pin only the CPU tensor for fast GPU transfer, then move to GPU
            x, y = x.to(device), y.to(device)
        else:
            # If on CPU, pin memory
            x, y = x.pin_memory(), y.pin_memory()

        return x, y



    from torch.optim.lr_scheduler import LinearLR, SequentialLR, CosineAnnealingLR

    # Instantiate the model
    nanoGPT = GPT.GPT(config)

    # AdamW Optimizer
    optimizer = torch.optim.AdamW(nanoGPT.parameters(), lr=learning_rate, betas=(0.9, 0.98), eps=1e-9)

    # Learning Rate Scheduler for Warmup (LinearLR)
    # Linearly increases the learning rate for the first 'warmup_steps' iterations.
    # This helps prevent large gradient updates at the beginning of training.
    scheduler_warmup = LinearLR(optimizer, total_iters=warmup_steps)

    # Learning Rate Scheduler for Decay (CosineAnnealingLR)
    # After the warmup period, the learning rate will decay using a cosine function.
    # The learning rate gradually decreases to 'eta_min' after the 'max_iters' iterations.
    scheduler_decay = CosineAnnealingLR(optimizer, T_max=max_iters - warmup_steps, eta_min=min_lr)

    # SequentialLR Scheduler to combine warmup and decay
    # First applies 'scheduler_warmup' for the 'warmup_steps' iterations,
    # and then switches to 'scheduler_decay' after the warmup phase.
    scheduler = SequentialLR(optimizer, schedulers=[scheduler_warmup, scheduler_decay], milestones=[warmup_steps])

    # GradScaler for Mixed-Precision Training
    # The GradScaler helps scale the gradients when training with mixed precision (e.g., 'float16').
    # This prevents overflow or underflow issues that could arise during backpropagation with lower precision.
    # It is enabled only if the model is using 'float16' or 'bfloat16', which are used to optimize memory usage and training speed.
    scaler = torch.cuda.amp.GradScaler(enabled=(dtype == 'float16'))
```

```
number of parameters: 81.11M
<ipython-input-21-e26af8e5a6bb>:28: FutureWarning: `torch.cuda.amp.GradScaler(args...)` is deprecated. Please use `torch.amp.GradScaler(
  scaler = torch.cuda.amp.GradScaler(enabled=(dtype == 'float16'))
```

```
# Initialize the configuration object
config = GPTConfig()

# Set the configuration parameters manually
config.block_size = block_size
config.vocab_size = 50257  # Example vocab size, replace with your value
config.n_embd = 512        # Example embedding size, replace with your value
config.n_layer = 8         # Example number of layers, replace with your value
config.n_head = 8          # Example number of attention heads, replace with your value
config.dropout = 0.1       # Example dropout rate, replace with your value

# Create the model instance
nanoGPT = GPT.GPT(config)
```

```
number of parameters: 50.91M
```

```
import matplotlib.pyplot as plt

# Check if the lists are not empty
if len(train_loss_list) == 0 or len(validation_loss_list) == 0:
    print("Training or Validation loss list is empty.")
else:
    # Convert losses from tensor to numpy for plotting
    train_loss_list_converted = [i.cpu().detach().numpy() for i in train_loss_list]
    validation_loss_list_converted = [i.cpu().detach().numpy() for i in validation_loss_list]

    # Plot training loss and validation loss
    plt.plot(train_loss_list_converted, 'g', label='Train Loss')  # Green for train loss
    plt.plot(validation_loss_list_converted, 'r', label='Validation Loss')  # Red for validation loss

    # Labels and title
    plt.xlabel("Steps (Every 100 epochs)")  # Label for x-axis
    plt.ylabel("Loss")  # Label for y-axis
    plt.title("Training and Validation Loss Over Time")  # Title of the plot

    # Show the legend
    plt.legend()

    # Display the plot
    plt.show()
```

```
Training or Validation loss list is empty.
```

Inference

```
#Load the model
nanoGPT = GPT.GPT(config)
device =  "cuda" if torch.cuda.is_available() else "cpu"
best_model_params_path = "best_model_params.pt"
nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states
```

```
number of parameters: 50.91M
<ipython-input-29-d4e888674873>:5: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), whic
  nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-29-d4e888674873> in <cell line: 5>()
      3 device =  "cuda" if torch.cuda.is_available() else "cpu"
      4 best_model_params_path = "best_model_params.pt"
----> 5 nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states

                                 2 frames
/usr/local/lib/python3.10/dist-packages/torch/serialization.py in __init__(self, name, mode)
    638 class _open_file(_opener):
    639     def __init__(self, name, mode):
--> 640         super().__init__(open(name, mode))
    641
    642     def __exit__(self, *args):

FileNotFoundError: [Errno 2] No such file or directory: 'best_model_params.pt'
```

Introduction

In this notebook, we'll explore how to train a lightweight NanoGPT on the Tiny Stories dataset. NanoGPT, developed by Andrej Karpathy, is a simplified variant of GPT designed for simplicity and speed. Our goal is to generate creative and coherent text based on the input data. I will also use Wandb library to track the experiments on the model.

```
**I would like to acknowledge that I currently do not have a clear understanding of how to fine-tune and operate NanoGPT. While I am fami
various aspects of AI and machine learning, this specific area is one
not yet had the opportunity to explore in depth. However, I am eager t
and improve my skills in this area, and I am confident that with the r
resources and guidance, I will be able to grasp and apply the necessar
techniques in fine-tuning and operating NanoGPT.**
```

I would like to acknowledge that I currently do not have a clear understanding of how to fine-tune and operate NanoGPT. While I am familiar with various aspects of AI and machine learning, this specific area is one I have not yet had the opportunity to explore in depth. However, I am eager to learn and improve my skills in this area, and I am confident that with the right resources and guidance, I will be able to grasp and apply the necessary techniques in fine-tuning and operating NanoGPT.

Import Libraries

```
# Clone the NanoGPT
!git clone https://github.com/karpathy/nanoGPT.git
!pip install tiktoken
!pip install wandb
```

```
Cloning into 'nanoGPT'...
remote: Enumerating objects: 682, done.
remote: Total 682 (delta 0), reused 0 (delta 0), pack-reused 682 (from 1)
Receiving objects: 100% (682/682), 952.47 KiB | 9.16 MiB/s, done.
Resolving deltas: 100% (385/385), done.
Collecting tiktoken
  Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.6 kB)
Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2024.9.11)
Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2.32.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (3.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0->tiktoken) (2024.8.3
Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.2/1.2 MB 22.2 MB/s eta 0:00:00
Installing collected packages: tiktoken
Successfully installed tiktoken-0.8.0
Requirement already satisfied: wandb in /usr/local/lib/python3.10/dist-packages (0.18.7)
Requirement already satisfied: click!=8.0.0,>=7.1 in /usr/local/lib/python3.10/dist-packages (from wandb) (8.1.7)
Requirement already satisfied: docker-pycreds>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (0.4.0)
Requirement already satisfied: gitpython!=3.1.29,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (3.1.43)
Requirement already satisfied: platformdirs in /usr/local/lib/python3.10/dist-packages (from wandb) (4.3.6)
Requirement already satisfied: protobuf!=4.21.0,!=5.28.0,<6,>=3.19.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (4.25.5)
Requirement already satisfied: psutil>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages (from wandb) (6.0.2)
Requirement already satisfied: requests<3,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (2.32.3)
Requirement already satisfied: sentry-sdk>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from wandb) (2.18.0)
Requirement already satisfied: setproctitle in /usr/local/lib/python3.10/dist-packages (from wandb) (1.3.4)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from wandb) (75.1.0)
Requirement already satisfied: typing-extensions<5,>=4.4 in /usr/local/lib/python3.10/dist-packages (from wandb) (4.12.2)
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0->wandb) (1.16.0)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from gitpython!=3.1.29,>=1.0.0->wandb) (4.0.1
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (3.4
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0->wandb) (2024.8.30
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.10/dist-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.29,>=1.0
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
from tqdm.auto import tqdm
from contextlib import nullcontext
```

```
import nanoGPT.model as GPT
import wandb
import os


# Directly set your API key (not secure in public notebooks)
import wandb
wandb.login(key="2c31d7e5323a64ac198ab2499a802513a1ac5ec8")
```

```
wandb: Using wandb-core as the SDK backend.  Please refer to https://wandb.me/wandb-core for more information.
wandb: W&B API key is configured. Use `wandb login --relogin` to force relogin
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the command line.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
True
```

Default title text

```
# @title Default title text
class GPTConfig: # Model config from NanoGPT
    block_size: int = 124  # this helps get context for training , reduce to 124 for faster training
    vocab_size: int = 50304 # GPT-2 vocab_size of 50257, padded up to nearest multiple of 64 for efficiency
    n_layer: int = 6 # reduce layers to reduce complexity
    n_head: int = 6 # reduce heads to reduce complexity for smaller model
    n_embd: int = 768 # reduce embedding size or faster computation since model becomes 'lighter' in terms of complexity
    dropout: float = 0.0 # since we have already worked on previous steps to make model as simple as possible , dropout is kept as is
    bias: bool = False # True: bias in Linears and LayerNorms, like GPT-2. False: a bit better and faster

config = GPTConfig


config = GPTConfig()  # Create an instance of GPTConfig
wandb_config = {k: v for k, v in vars(config).items() if not callable(getattr(config, k)) and not k.startswith("__")}
wandb_config  # This will output the dictionary form of config
```

```
{}
```

```
wandb.init(project="nanogpt-tinystories", name="nanoGPT", config=wandb_config)
```

```
wandb: Currently logged in as: raj-dandekar8 (raj-dandekar8-massachusetts-institute-of-technology). Use `wandb login --relogin` to force
Tracking run with wandb version 0.18.7
Run data is saved locally in /content/wandb/run-20241205_143748-inpu8mb0
Syncing run nanoGPT to Weights & Biases (docs)
View project at https://wandb.ai/raj-dandekar8-massachusetts-institute-of-technology/nanogpt-tinystories
View run at https://wandb.ai/raj-dandekar8-massachusetts-institute-of-technology/nanogpt-tinystories/runs/inpu8mb0
Display W&B run
```

Tiny Stories datasets and preprocessing

TinyStories, a synthetic dataset of short stories that only contain words that a typical 3 to 4-year-olds usually understand, generated by GPT-3.5 and GPT-4. We show that TinyStories can be used to train and evaluate LMs that are much smaller than the state-of-the-art models (below 10 million total parameters)

```
!pip install datasets
```

```
from datasets import load_dataset
```

```
ds = load_dataset("roneneldan/TinyStories")
```

```
Collecting datasets
  Downloading datasets-3.1.0-py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from datasets) (3.16.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from datasets) (1.26.4)
Requirement already satisfied: pyarrow>=15.0.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (17.0.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from datasets) (2.2.2)
Requirement already satisfied: requests>=2.32.2 in /usr/local/lib/python3.10/dist-packages (from datasets) (2.32.3)
Requirement already satisfied: tqdm>=4.66.3 in /usr/local/lib/python3.10/dist-packages (from datasets) (4.66.6)
Collecting xxhash (from datasets)
  Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py310-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.9.0,>=2023.1.0 (from fsspec[http]<=2024.9.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.9.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from datasets) (3.11.2)
Requirement already satisfied: huggingface-hub>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from datasets) (0.26.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from datasets) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from datasets) (6.0.2)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (2.4.3)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.1.0)
Requirement already satisfied: propcache>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (0.2.0)
Requirement already satisfied: yarl<2.0,>=1.17.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.17.2)
Requirement already satisfied: async-timeout<6.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.23.0->d
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2.2.3
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.32.2->datasets) (2024.
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas->datasets) (1
Downloading datasets-3.1.0-py3-none-any.whl (480 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 480.6/480.6 kB 29.9 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 116.3/116.3 kB 11.3 MB/s eta 0:00:00
Downloading fsspec-2024.9.0-py3-none-any.whl (179 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 179.3/179.3 kB 16.1 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py310-none-any.whl (134 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 134.8/134.8 kB 13.3 MB/s eta 0:00:00
Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.1/194.1 kB 17.9 MB/s eta 0:00:00
Installing collected packages: xxhash, fsspec, dill, multiprocess, datasets
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2024.10.0
    Uninstalling fsspec-2024.10.0:
      Successfully uninstalled fsspec-2024.10.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sour
gcsfs 2024.10.0 requires fsspec==2024.10.0, but you have fsspec 2024.9.0 which is incompatible.
Successfully installed datasets-3.1.0 dill-0.3.8 fsspec-2024.9.0 multiprocess-0.70.16 xxhash-3.5.0
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as se
```

```
import tiktoken
import os
import numpy as np
from tqdm.auto import tqdm

enc = tiktoken.get_encoding("gpt2")

# Some functions from https://github.com/karpathy/nanoGPT/blob/master/data/openwebtext/prepare.py

def process(example):
    ids = enc.encode_ordinary(example['text'])  # Ignore special tokens
    out = {'ids': ids, 'len': len(ids)}
    return out

# Check dataset size before tokenizing
train_size = len(ds['train'])
val_size = len(ds['validation'])

# Print dataset size for reference
print(f"Training set size: {train_size} examples")
print(f"Validation set size: {val_size} examples")

# Adjust batch size and parallelization based on dataset size
```

```
num_proc = 8 if train_size > 100000 else 4
total_batches = 1024 if train_size > 100000 else 512

# Only tokenize and process if the binary files do not already exist
if not os.path.exists("train.bin"):
    tokenized = ds.map(
        process,
        remove_columns=['text'],
        desc="Tokenizing the splits",
        num_proc=num_proc,
    )

    # Concatenate all the ids in each dataset into one large file we can use for training
    for split, dset in tokenized.items():
        arr_len = np.sum(dset['len'], dtype=np.uint64)
        filename = f'{split}.bin'
        dtype = np.uint16  # (GPT-2 vocab size is 50256, so 16-bit unsigned integers are enough)
        arr = np.memmap(filename, dtype=dtype, mode='w+', shape=(arr_len,))

        idx = 0
        for batch_idx in tqdm(range(total_batches), desc=f'Writing {filename}'):
            # Efficient batch processing to save memory
            batch = dset.shard(num_shards=total_batches, index=batch_idx, contiguous=True).with_format('numpy')
            arr_batch = np.concatenate(batch['ids'])
            arr[idx: idx + len(arr_batch)] = arr_batch
            idx += len(arr_batch)

        # Ensure the memory-mapped array is saved to disk
        arr.flush()

else:
    print("Binary files already exist. Skipping tokenization and writing.")
```

```
Training set size: 2119719 examples
Validation set size: 21990 examples
Tokenizing the splits (num_proc=8): 100%  2119719/2119719 [07:39<00:00, 4761.11 examples/s]
Tokenizing the splits (num_proc=8): 100%  21990/21990 [00:06<00:00, 4540.97 examples/s]
Writing train.bin: 100%  1024/1024 [00:26<00:00, 48.47it/s]
Writing validation.bin: 100%  1024/1024 [00:02<00:00, 421.92it/s]
```

Training

```
def get_batch(split):
    """
    This function fetches a batch of data using chunking for improved I/O efficiency.
    Optimization includes:
    - Preloading larger chunks of data into memory.
    - Fetching batches sequentially from the preloaded chunk to reduce I/O overhead.
    """
    # Use the correct file depending on the split
    data_file = 'train.bin' if split == 'train' else 'validation.bin'

    # Preload a chunk of data into memory for more efficient access
    chunk_size = block_size * batch_size * 10  # Example: Load 10 batches at once (adjust as needed)
    data = np.memmap(data_file, dtype=np.uint16, mode='r')

    # Randomly sample a starting index, ensuring that we don't go past the chunk size
    ix = torch.randint(len(data) - chunk_size, (batch_size,))

    # We create a chunk of data and slice it to get the batches
    chunk = data[ix[0]: ix[0] + chunk_size]  # Load a chunk from the data

    # Efficiently slice the chunk into individual batches for X and Y
    x = torch.from_numpy(chunk[:batch_size * block_size]).reshape(batch_size, block_size).to(torch.int64)
    y = torch.from_numpy(chunk[batch_size: batch_size + batch_size * block_size]).reshape(batch_size, block_size).to(torch.int64)

    # Move to GPU (with non-blocking for faster transfer)
    if device_type == 'cuda':
        x, y = x.pin_memory().to(device, non_blocking=True), y.pin_memory().to(device, non_blocking=True)
    else:
```

```
            x, y = x.to(device), y.to(device)

        return x, y



    # Notes:
    # 1. Memory-Mapped File (`np.memmap`) is used to handle large datasets efficiently by reading them in chunks.
    # 2. Pinning memory (`pin_memory()`) is applied only to CPU tensors to speed up data transfer to GPU.
    #    This avoids errors when trying to pin already GPU-bound tensors.
    # 3. The `get_batch()` function selects random batches from the dataset (`train.bin` or `validation.bin`).
    #    The slicing method ensures minimal memory overhead by reading large chunks in a single operation.
    # 4. The `to(device)` method ensures tensors are moved to the right device (GPU or CPU).
    #    This is needed when training on a CUDA-enabled device. For non-CUDA (CPU), tensors are processed directly.

    def get_batch(split, chunk_size=1024):
        """
        This function fetches a batch of data from a memory-mapped file.
        Optimization includes:
        - Avoiding excessive memory access (sequential access improves performance).
        - Minimizing tensor conversions and unnecessary operations.
        """
        # Use the correct file depending on the split
        data_file = 'train.bin' if split == 'train' else 'validation.bin'
        data = np.memmap(data_file, dtype=np.uint16, mode='r')

        # Randomly sample indices for the batch
        ix = torch.randint(len(data) - block_size, (batch_size,))

        # Fetch the batches efficiently by using slicing instead of multiple individual numpy accesses
        x = torch.tensor([data[i:i+block_size] for i in ix], dtype=torch.int64)
        y = torch.tensor([data[i+1:i+1+block_size] for i in ix], dtype=torch.int64)

        # Transfer to device, pinning memory only for CPU tensors
        if device_type == 'cuda':
            # Pin only the CPU tensor for fast GPU transfer, then move to GPU
            x, y = x.to(device), y.to(device)
        else:
            # If on CPU, pin memory
            x, y = x.pin_memory(), y.pin_memory()

        return x, y



    from torch.optim.lr_scheduler import LinearLR, SequentialLR, CosineAnnealingLR

    # Instantiate the model
    nanoGPT = GPT.GPT(config)

    # AdamW Optimizer
    optimizer = torch.optim.AdamW(nanoGPT.parameters(), lr=learning_rate, betas=(0.9, 0.98), eps=1e-9)

    # Learning Rate Scheduler for Warmup (LinearLR)
    # Linearly increases the learning rate for the first 'warmup_steps' iterations.
    # This helps prevent large gradient updates at the beginning of training.
    scheduler_warmup = LinearLR(optimizer, total_iters=warmup_steps)

    # Learning Rate Scheduler for Decay (CosineAnnealingLR)
    # After the warmup period, the learning rate will decay using a cosine function.
    # The learning rate gradually decreases to 'eta_min' after the 'max_iters' iterations.
    scheduler_decay = CosineAnnealingLR(optimizer, T_max=max_iters - warmup_steps, eta_min=min_lr)

    # SequentialLR Scheduler to combine warmup and decay
    # First applies 'scheduler_warmup' for the 'warmup_steps' iterations,
    # and then switches to 'scheduler_decay' after the warmup phase.
    scheduler = SequentialLR(optimizer, schedulers=[scheduler_warmup, scheduler_decay], milestones=[warmup_steps])

    # GradScaler for Mixed-Precision Training
    # The GradScaler helps scale the gradients when training with mixed precision (e.g., 'float16').
    # This prevents overflow or underflow issues that could arise during backpropagation with lower precision.
    # It is enabled only if the model is using 'float16' or 'bfloat16', which are used to optimize memory usage and training speed.
    scaler = torch.cuda.amp.GradScaler(enabled=(dtype == 'float16'))
```

```
number of parameters: 81.11M
<ipython-input-21-e26af8e5a6bb>:28: FutureWarning: `torch.cuda.amp.GradScaler(args...)` is deprecated. Please use `torch.amp.GradScaler(
  scaler = torch.cuda.amp.GradScaler(enabled=(dtype == 'float16'))
```

```
# Initialize the configuration object
config = GPTConfig()

# Set the configuration parameters manually
config.block_size = block_size
config.vocab_size = 50257  # Example vocab size, replace with your value
config.n_embd = 512        # Example embedding size, replace with your value
config.n_layer = 8         # Example number of layers, replace with your value
config.n_head = 8          # Example number of attention heads, replace with your value
config.dropout = 0.1       # Example dropout rate, replace with your value

# Create the model instance
nanoGPT = GPT.GPT(config)
```

```
number of parameters: 50.91M
```

```
import matplotlib.pyplot as plt

# Check if the lists are not empty
if len(train_loss_list) == 0 or len(validation_loss_list) == 0:
    print("Training or Validation loss list is empty.")
else:
    # Convert losses from tensor to numpy for plotting
    train_loss_list_converted = [i.cpu().detach().numpy() for i in train_loss_list]
    validation_loss_list_converted = [i.cpu().detach().numpy() for i in validation_loss_list]

    # Plot training loss and validation loss
    plt.plot(train_loss_list_converted, 'g', label='Train Loss')  # Green for train loss
    plt.plot(validation_loss_list_converted, 'r', label='Validation Loss')  # Red for validation loss

    # Labels and title
    plt.xlabel("Steps (Every 100 epochs)")  # Label for x-axis
    plt.ylabel("Loss")  # Label for y-axis
    plt.title("Training and Validation Loss Over Time")  # Title of the plot

    # Show the legend
    plt.legend()

    # Display the plot
    plt.show()
```

```
Training or Validation loss list is empty.
```

Inference

```
#Load the model
nanoGPT = GPT.GPT(config)
device =  "cuda" if torch.cuda.is_available() else "cpu"
best_model_params_path = "best_model_params.pt"
nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states
```

```
number of parameters: 50.91M
<ipython-input-29-d4e888674873>:5: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), whic
  nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-29-d4e888674873> in <cell line: 5>()
      3 device =  "cuda" if torch.cuda.is_available() else "cpu"
      4 best_model_params_path = "best_model_params.pt"
----> 5 nanoGPT.load_state_dict(torch.load(best_model_params_path, map_location=torch.device(device))) # load best model states

                                 2 frames
/usr/local/lib/python3.10/dist-packages/torch/serialization.py in __init__(self, name, mode)
    638 class _open_file(_opener):
    639     def __init__(self, name, mode):
--> 640         super().__init__(open(name, mode))
    641
    642     def __exit__(self, *args):

FileNotFoundError: [Errno 2] No such file or directory: 'best_model_params.pt'
```