ka / m   Type / to search

Code   Issues 35   Pull requests 27   Actions   Projects   Security   Ir

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks or learn more about diff comparisons.

base repository: karpathy/minGPT   base: master ←...
head repository: dylanskinner65/MinGPT_UL2   compare: master

✓ **Able to merge.** These branches can be automatically merged.

Discuss and review the changes in this comparison with others.
Learn about pull requests

Create pull request

10 commits   **10** files changed   **1** contributor

Commits on Nov 2, 2023

Implemented model checkpointing.
dylanskinner65 committed 2 weeks ago   983c71d

Pushing project2a notebook.
dylanskinner65 committed 2 weeks ago   690ff74

Commits on Nov 3, 2023

Updated gitignore.
dylanskinner65 committed 2 weeks ago
9767f3b

Actually fixed .gitignore.
dylanskinner65 committed 2 weeks ago
9c310d5

More fully implemented checkpointing.
dylanskinner65 committed 2 weeks ago
6222d00

Commits on Nov 7, 2023

Pushing demo (not updating anymore).
dylanskinner65 committed last week
580be6f

Pushing training and model py files.
dylanskinner65 committed last week
567fd24

Implemented plotting, training on red pajama.
dylanskinner65 committed last week
cc27ad8

Commits on Nov 9, 2023

Implemented a way to save and plot losses.
dylanskinner65 committed 4 days ago
8ca9bba

Commits on Nov 10, 2023

Pushing files made on super computer.
dylanskinner65 committed 3 days ago
c7a725a

Showing 10 changed files with **1,170 additions** and **48 deletions**. Split **Unified**

BIN +6 KB .DS_Store

Binary file not shown.

1 🟩⬜⬜⬜⬜ .gitignore

```
3    3    *.swp
```

```
   .env
   .pylintrc
 + checkpoints/*.pth
```

199 🟩🟩🟩🟥⬜ demo.ipynb

Load diff

Large diffs are not rendered by default.

26 🟩🟩🟩🟥⬜ mingpt/model.py

```
            ))
            self.lm_head = nn.Linear(config.n_embd, config.vocab_size, bias=False)

-           # init all weights, and apply a special scaled init to the residual projections, per GPT-2 paper
-           self.apply(self._init_weights)
-           for pn, p in self.named_parameters():
-               if pn.endswith('c_proj.weight'):
-                   torch.nn.init.normal_(p, mean=0.0, std=0.02/math.sqrt(2 * config.n_layer))
+           '''This if statement is a change.'''
+           # If we are using a checkpoint, load it.
+           if config.checkpoint is not None:
+               self.checkpoint = torch.load(config.checkpoint)
+ self.transformer.load_state_dict(self.checkpoint['model_transformer'])
+ self.lm_head.load_state_dict(self.checkpoint['model_lm_head'])
+               self.iter_num = self.checkpoint['iter_num']
+               self.checkpoint_num = self.checkpoint['checkpoint_num']
+               self.saved_loss = self.checkpoint['saved_loss']
+
+           else:
+               # init all weights, and apply a special scaled init to the residual projections, per GPT-2 paper
+               self.checkpoint = None

                self.apply(self._init_weights)
```

```
+      for pn, p in self.named_parameters():
+        if pn.endswith('c_proj.weight'):
+          torch.nn.init.normal_(p, mean=0.0, std=0.02/math.sqrt(2 * config.n_layer))

    # report number of parameters (note we don't count the decoder parameters in lm_head)
    n_params = sum(p.numel() for p in self.transformer.parameters())
      {"params": [param_dict[pn] for pn in sorted(list(no_decay))], "weight_decay": 0.0},
    ]
    optimizer = torch.optim.AdamW(optim_groups, lr=train_config.learning_rate, betas=train_config.betas)
+    if self.checkpoint:
+      optimizer.load_state_dict(self.checkpoint['optimizer_state_dict']) # This is a change.
    return optimizer

  def forward(self, idx, targets=None):
    # if we are given some desired targets also calculate the loss
    loss = None
    if targets is not None:
-      loss = F.cross_entropy(logits.view(-1, logits.size(-1)), targets.view(-1), ignore_index=-1)
+      loss = F.cross_entropy(logits.view(-1, logits.size(-1)), targets.view(-1), ignore_index=-1)  # Changed class!

    return logits, loss

```

41 🟩🟩🟩🟩⬜ mingpt/trainer.py

```
import torch
from torch.utils.data.dataloader import DataLoader
from mingpt.utils import CfgNode as CN
+import numpy as np

class Trainer:

    )

```

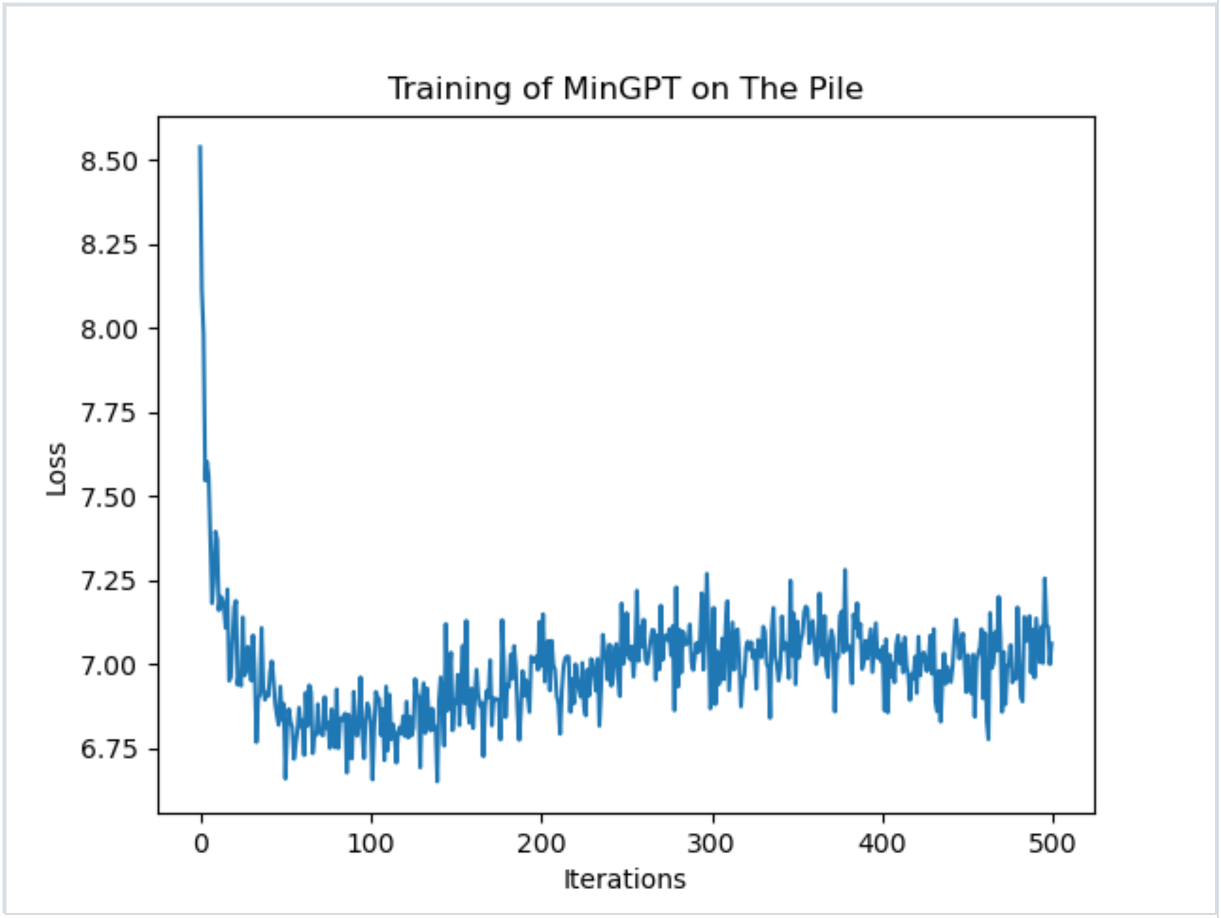
```diff
         model.train()
-        self.iter_num = 0
+        self.iter_num = model.iter_num if hasattr(model, 'iter_num') else 0  # This is a change
+        self.iter_list = model.iter_list if hasattr(model, 'iter_list') else []  # This is a change
+        self.since_last_save = 0  # This is a change
+        self.checkpoint_num = model.checkpoint_num if hasattr(model, 'checkpoint_num') else 0   # This is a change
         self.iter_time = time.time()
+        self.saved_loss = model.saved_loss if hasattr(model, 'saved_loss') else []  # This is a change
         data_iter = iter(train_loader)
+        checkpoint_name = config.checkpoint_name if hasattr(config, 'checkpoint_name') else 'checkpoint'  # This is a change
+
+        # Define loss
+        self.loss = self.saved_loss[-1] if self.saved_loss else np.inf  # This is a change
+        self.curr_loss = []
+
         while True:

             # fetch the next batch (x, y) and re-init iterator if needed
                 batch = next(data_iter)
             batch = [t.to(self.device) for t in batch]
             x, y = batch
+            x = x.squeeze(0)  # This is a change.
+            y = y.squeeze(0)  # This is a change.
+
+            # Get previous loss.
+            prev_loss = self.loss  # This is a change.

             # forward the model
             logits, self.loss = model(x, y)
+            self.curr_loss.append(self.loss.detach())

             # backprop and update the parameters
             model.zero_grad(set_to_none=True)
             tnow = time.time()
             self.iter_dt = tnow - self.iter_time
             self.iter_time = tnow
+
+            # Save when we last saved the weights.
+            self.since_last_save += 1  # This is a change.
+
+            '''All of this is a change.'''
+            if self.loss <= prev_loss and self.since_last_save >= config.checkpoint_iters:
+                self.since_last_save = 0
+
+                # Create and save our checkpoint
+                checkpoint = {
+                    'model_transformer': model.transformer.state_dict(),
+                    'model_lm_head': model.lm_head.state_dict(),
+                    'optimizer_state_dict': self.optimizer.state_dict(),
+                    'loss': self.loss,
+                    'iter_num': self.iter_num,
+                    'iter_list': self.iter_list.append(self.iter_num),
+                    'checkpoint_num': self.checkpoint_num,
+                    'saved_loss': self.saved_loss.append(self.loss)
+                }
+                torch.save(checkpoint, f'checkpoints/{checkpoint_name}_{self.checkpoint_num}.pth')
+                self.checkpoint_num += 1
+

             # termination conditions
             if config.max_iters is not None and self.iter_num >= config.max_iters:
```

16 🟩🟩🟩🟩🟩 mingpt_jobscript

```
@@ -0,0 +1,16 @@
+ #!/bin/bash --login
+
+ #SBATCH --time=03:00:00   # walltime
+ #SBATCH --ntasks=1   # number of processor cores (i.e. tasks)
+ #SBATCH --nodes=1   # number of nodes
+ #SBATCH --mem=120G   # memory per CPU core
+ #SBATCH --gpus=1  # num gpus
+ #SBATCH --qos=cs
+
+ # Set the max number of threads to use for programs using OpenMP. Should be <= ppn. Does nothing if the program doesn't use OpenMP.
+ export OMP_NUM_THREADS=$SLURM_CPUS_ON_NODE
```

```
+
+ # LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE
+ mamba activate mingpt
+ python ~/compute/MinGPT_UL2/project2a.py
+
```

BIN +37.5 KB mingpt_loss.png



506 🟩🟩🟩🟩🟩 output_results.txt

Load diff

Large diffs are not rendered by default.

341 ■■■■■ project2a.ipynb

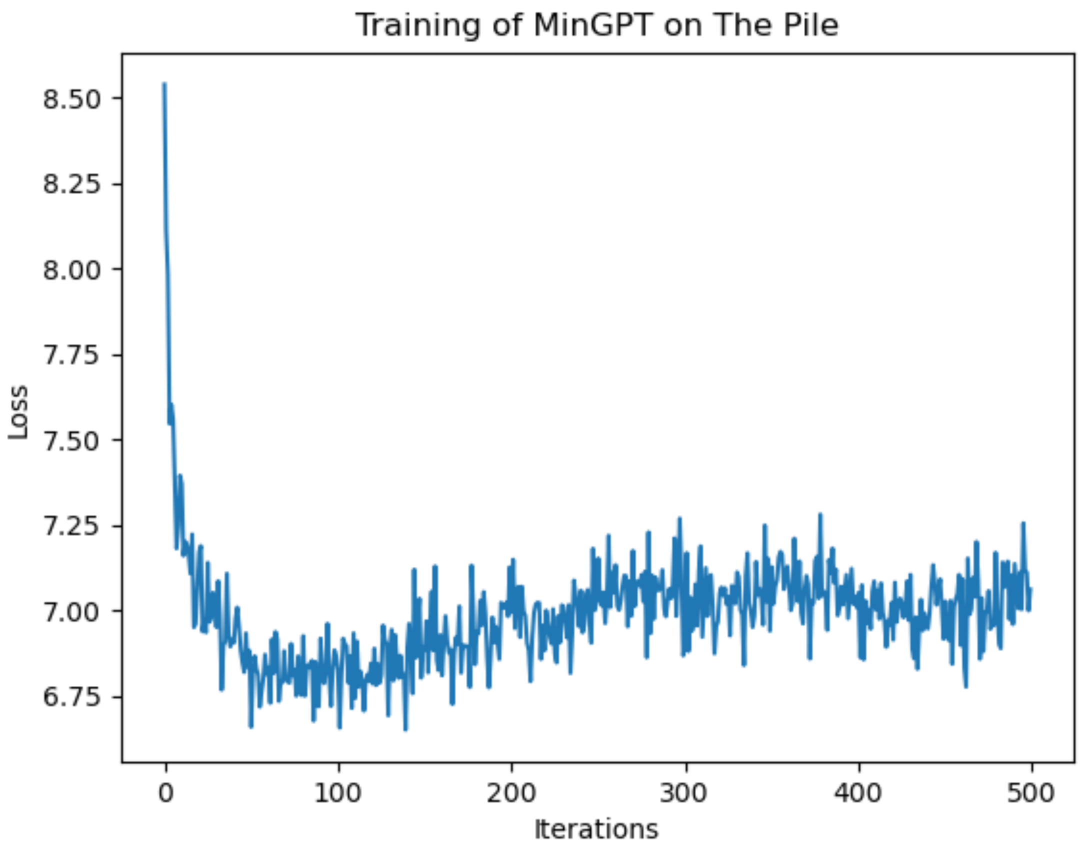
Load diff

Large diffs are not rendered by default.

88 ■■■■■ project2a.py

```
@@ -0,0 +1,88 @@
+ import torch
+ from transformers import GPT2Tokenizer, GPT2LMHeadModel
+ from mingpt.model import GPT
+ from mingpt.utils import set_seed
+ from datasets import load_dataset
+ import torch
+ import matplotlib.pyplot as plt
+ from torch.utils.data import Dataset, DataLoader
+ from datasets import load_dataset
+ from mingpt.trainer import Trainer
+ import numpy as np
+ set_seed(3407)
+
+ # Custom dataset class for the Red Pajama dataset
+ class RedPajamaDataset(Dataset):
+     def __init__(self, data, max_length=1024):
+         self.data = data
+         self.tokenizer =
GPT2Tokenizer.from_pretrained('gpt2_tokenizer')
+         self.tokenizer.pad_token_id = 50256
+         self.max_length = max_length
+         self.vocab_size = self.tokenizer.vocab_size
+
+     def __len__(self):
+         return len(self.data)
+
+     def __getitem__(self, idx):
+         text = self.data[idx]['text']
+         # Tokenize the text
+         tokens = self.tokenizer.encode(
+             text, add_special_tokens=True,
max_length=self.max_length, truncation=True, return_tensors='pt',
padding=True)
        # Split the tokens into chunks of max_length
        # Shift the tokens to get targets (excluding the [CLS] token)
        target_tokens = tokens[:, 1:].clone()  # Exclude the [CLS] token
        # Exclude the last token to match the shifted targets
        tokens = tokens[:, :-1]

        return tokens, target_tokens


def batch_end_callback(trainer):
    if trainer.iter_num % 100 == 0:
        print(
            f"iter_dt {trainer.iter_dt * 1000:.2f}ms; iter {trainer.iter_num}: train loss {trainer.loss.item():.5f}")


if __name__ == '__main__':

    # load in the dataset
    dataset = load_dataset(
        "json", data_files="/lustre/scratch/usr/dw87/pile_data_10.jsonl", cache_dir='pile_dataset')
    dataset = dataset['train']
    print('Loaded Dataset')
    data = RedPajamaDataset(dataset)
    print('Instatiated Dataset Class')

    # load in an instance of the model
    model_config = GPT.get_default_config()
    model_config.model_type = 'gpt2'
    model_config.vocab_size = data.vocab_size
    model_config.block_size = data.max_length - 1
    model_config.checkpoint = None
    model = GPT(model_config)

    # create a trainer object
    train_config = Trainer.get_default_config()
    # the model we're using is so small that we can go a bit faster
    train_config.learning_rate = 5e-4
    max_iters = 50000
```

```
+    train_config.max_iters = max_iters + \
+        model.iter_num if model_config.checkpoint else max_iters  # This is a change
+    train_config.num_workers = 0
+    train_config.checkpoint_iters = 10000    # This is a change
+    train_config.batch_size = 1
+    trainer = Trainer(train_config, model, data)
+
+    trainer.set_callback('on_batch_end', batch_end_callback)
+    trainer.run()
+
+    # Get the average of every 10 elements for plotting.
+    losses = [a.detach().cpu() for a in trainer.curr_loss]
+    x = 100
+    new_losses = np.mean(np.array(losses).reshape(-1, x), axis=1)
+
+    plt.plot(np.arange(len(new_losses)), new_losses)
+    plt.title('Training of MinGPT on The Pile')
+    plt.ylabel('Loss')
+    plt.xlabel('Iterations')
+    plt.savefig('mingpt_loss.png')
```



Training of MinGPT on The Pile
Loss
8.50
8.25
8.00
7.75
7.50
7.25
7.00
6.75
0
100
200
300
400
500
Iterations

```
{
  "results": {
    "arc_easy": {
      "acc": 0.43813131313131315,
      "acc_stderr": 0.010180937100600062,
      "acc_norm": 0.3947811447811448,
      "acc_norm_stderr": 0.010030038935883556
    },
    "hellaswag": {
      "acc": 0.2891854212308305,
      "acc_stderr": 0.00452457589295296,
      "acc_norm": 0.31139215295757816,
      "acc_norm_stderr": 0.004621163476949214
    },
    "openbookqa": {
      "acc": 0.164,
      "acc_stderr": 0.01657581114244669,
      "acc_norm": 0.272,
      "acc_norm_stderr": 0.019920483209566065
    },
    "piqa": {
      "acc": 0.6289445048966268,
      "acc_stderr": 0.011271222398600523,
      "acc_norm": 0.6251360174102285,
      "acc_norm_stderr": 0.011294565805619014
    }
  },
  "versions": {
    "arc_easy": 0,
    "hellaswag": 0,
    "openbookqa": 0,
    "piqa": 0
  },
  "config": {
    "model": "mingpt",
    "model_args": "pretrained=gpt2",
    "num_fewshot": 0,
    "batch_size": null,
    "batch_sizes": [],
    "device": "cuda:0",
    "no_cache": false,
    "limit": null,
    "bootstrap_iters": 100000,
    "description_dict": {}
  }
}
```

```
mingpt (pretrained=gpt2), limit: None, provide_description: False, num_fewshot: 0, batch_size: None
|   Task   |Version| Metric |Value |   |Stderr|
|----------|------:|--------|-----:|---|-----:|
|arc_easy  |      0|acc     |0.4381|±  |0.0102|
|          |       |acc_norm|0.3948|±  |0.0100|
|hellaswag |      0|acc     |0.2892|±  |0.0045|
|          |       |acc_norm|0.3114|±  |0.0046|
|openbookqa|      0|acc     |0.1640|±  |0.0166|
|          |       |acc_norm|0.2720|±  |0.0199|
|piqa      |      0|acc     |0.6289|±  |0.0113|
|          |       |acc_norm|0.6251|±  |0.0113|
```