Real Time Machine Learning - Homework 4

Name : Tarun Reddy Challa

Student ID : 801318301

GitHub Link : https://github.com/tarunreddy03/RTML

Real Time Machine Learning - Homework 4

Name : Tarun Reddy Challa

Student ID : 801318301

GitHub Link : https://github.com/tarunreddy03/RTML

```
!pip install setuptools==66
!pip install d2l==1.0.0-beta0

import cv2
import numpy as np
import pandas as pd
from tqdm import tqdm
from datetime import datetime
from matplotlib import pyplot as plt

import torch
from torch import nn
from d2l import torch as d2l
from torch.nn import functional as F

!pip install ptflops
import ptflops
from ptflops import get_model_complexity_info
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: setuptools==66 in /usr/local/lib/python3.9/dist-packages (66.0.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: d2l==1.0.0-beta0 in /usr/local/lib/python3.9/dist-packages (1.0.0b0)
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (1.4.4)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (3.7.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (1.10.1)
Requirement already satisfied: matplotlib-inline in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (0.1.6)
Requirement already satisfied: gpytorch in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (1.9.1)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (2.27.1)
Requirement already satisfied: gym==0.21.0 in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (0.21.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (1.22.4)
Requirement already satisfied: jupyter in /usr/local/lib/python3.9/dist-packages (from d2l==1.0.0-beta0) (1.0.0)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.9/dist-packages (from gym==0.21.0->d2l==1.0.0-beta0) (2.
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (from gpytorch->d2l==1.0.0-beta0) (1.2.2)
Requirement already satisfied: linear-operator>=0.2.0 in /usr/local/lib/python3.9/dist-packages (from gpytorch->d2l==1.0.0-beta0) (0
Requirement already satisfied: qtconsole in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (5.4.1)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (6.5.4)
Requirement already satisfied: notebook in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (6.3.0)
Requirement already satisfied: jupyter-console in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (6.1.0)
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (7.7.1)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0-beta0) (5.5.6)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (3.0.9
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (4.39
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (1.4.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (23.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (2
Requirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-bet
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (1.0.7
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0-beta0) (8.4.0)
Requirement already satisfied: traitlets in /usr/local/lib/python3.9/dist-packages (from matplotlib-inline->d2l==1.0.0-beta0) (5.7.1
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->d2l==1.0.0-beta0) (2022.7.1)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0-beta0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0-beta0) (1.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0-beta0) (2022.
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0-beta0) (3.4)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-packages (from importlib-resources>=3.2.0->matplotlib->d
Requirement already satisfied: torch>=1.11 in /usr/local/lib/python3.9/dist-packages (from linear-operator>=0.2.0->gpytorch->d2l==1.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matplotlib->d2l==1.0.0
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0-beta0
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0-beta0) (6
Requirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0-beta0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0-beta0)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from ipywidgets->jupyter->d2l==1
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.9/dist-packages (from ipywidgets->jupyter->d2l==1
Requirement already satisfied: pygments in /usr/local/lib/python3.9/dist-packages (from jupyter-console->jupyter->d2l==1.0.0-beta0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from jupyter-
Requirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta0)
```

```
Requirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta0) (
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta0) (1.2.1
Requirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta0) (0.7
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-b
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta0)
Requirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0-beta
```

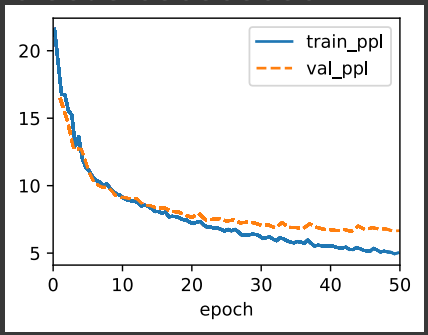
IN CLASS EXAMPLE FOR GRU

```
data = d2l.TimeMachine(batch_size=1024, num_steps=32)
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
```

```
class GRU(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.GRU(num_inputs, num_hiddens)
```

```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32)
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

```
'time travel the the the the t'
```



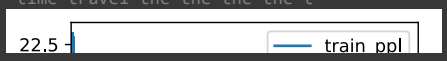
Q 1.1 In this homework, we focus on the language model we did in the lectures.Use the GRU example, adjust the hyperparameters (fully connected network and the number of hidden states) and analyze their influence on running time, perplexity, training and validation loss, and the output sequence

sol: Adjusting the num_hiddens=3, num_layers=4, dropout=1

```
class adjGRU(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.GRU(num_inputs, num_hiddens)
```

```
gru = adjGRU(num_inputs=len(data.vocab), num_hiddens=3, num_layers=4, dropout=1)
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=3)
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

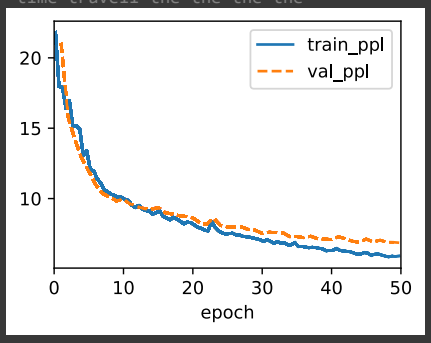
```
'time travel the the the the t'
```



Adjusting num_hiddens=24, num_layers=12, dropout=0

```
gru = adjGRU(num_inputs=len(data.vocab), num_hiddens=24, num_layers=12, dropout=0)
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=3)
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

```
'time travell the the the the '
```



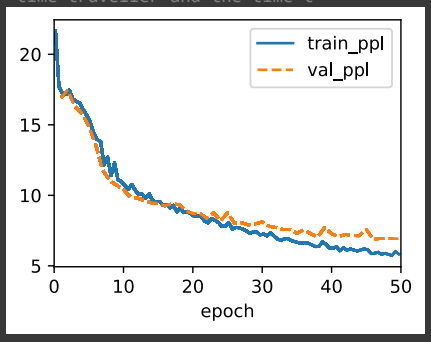
Q 1.2 Use the LSTM example, adjust the hyperparameters (fully connected network and the number of hidden states) and analyze their influence on running time, perplexity, training and validation loss, and the output sequence

Sol: We used the LTSM model in this section. The baseline model with 32 hidden layers has low training and validation perplexity but produced mediocre predictions. These are the baseline model's results

```
class LSTM(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens)

    def forward(self, inputs, H_C=None):
        return self.rnn(inputs, H_C)
```

```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32)
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict('time travel', 18, data.vocab, d2l.try_gpu())
```

```
'time traveller and the time t'
```



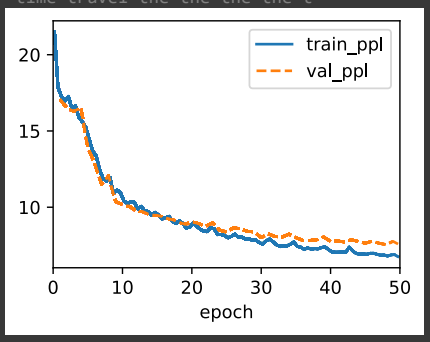
Adjusting LSTM num_hiddens=16, num_layers=10, dropout=0)

```
class adjLSTM(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens)
```

```
    def forward(self, inputs, H_C=None):
        return self.rnn(inputs, H_C)
```

```
lstm = adjLSTM(num_inputs=len(data.vocab), num_hiddens=16, num_layers=10, dropout=0)
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict('time travel', 18, data.vocab, d2l.try_gpu())
```

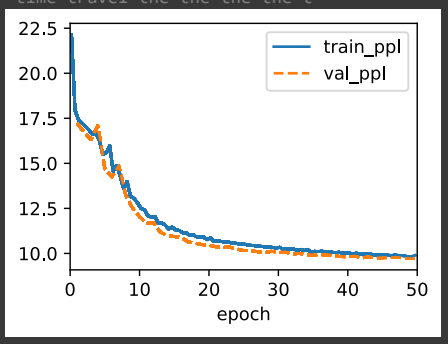
```
'time travel the the the the t'
```



Adjusting LSTM num_hiddens=4, num_layers=2, dropout=1

```
lstm = adjLSTM(num_inputs=len(data.vocab), num_hiddens=4, num_layers=2, dropout=1)
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict('time travel', 18, data.vocab, d2l.try_gpu())
```

```
'time travel the the the the t'
```



In the first test, we modified the model to have 16 hidden layers, and in the second test, we changed it to have 4 hidden layers. The former once more produced the best prediction while somewhat increasing the difficulty of training and validation.

Q 1.3 Compare runtime for training and inference, computational and mode size complexities, training and validation loss, and the output sequence (try a few examples)for rnn.RNN, rnn.LSTM and rnn.GRU implementations with each other use the same hyperparameters for your comparison.

Sol:

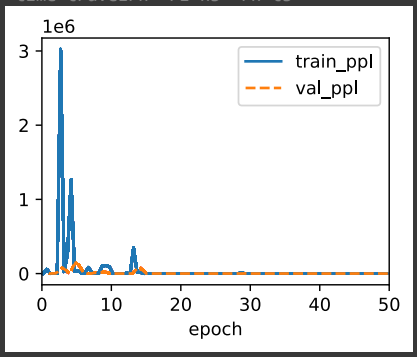
```
class RNNLMScratch(d2l.Classifier):
    """The RNN-based language model implemented from scratch."""
    def __init__(self, rnn, vocab_size, lr=0.01):
        super().__init__()
        self.save_hyperparameters()
        self.init_params()

    def init_params(self):
        self.W_hq = nn.Parameter(
            torch.randn(
                self.rnn.num_hiddens, self.vocab_size) * self.rnn.sigma)
        self.b_q = nn.Parameter(torch.zeros(self.vocab_size))

    def training_step(self, batch):
        l = self.loss(self(*batch[:-1]), batch[-1])
        self.plot('ppl', torch.exp(l), train=True)
```

```
        return l

    def validation_step(self, batch):
        l = self.loss(self(*batch[:-1]), batch[-1])
        self.plot('ppl', torch.exp(l), train=False)
```

```
class RNN(d2l.Module):
    def __init__(self, num_inputs, num_hiddens):
        super().__init__()
        self.save_hyperparameters()
        self.rnn = nn.RNN(num_inputs, num_hiddens)

    def forward(self, inputs, H=None):
        return self.rnn(inputs, H)
```

```
rnn = RNN(num_inputs=len(data.vocab), num_hiddens=64)
model = d2l.RNNLM(rnn, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

```
'time travelrh  rl ns  rn ts  '
```

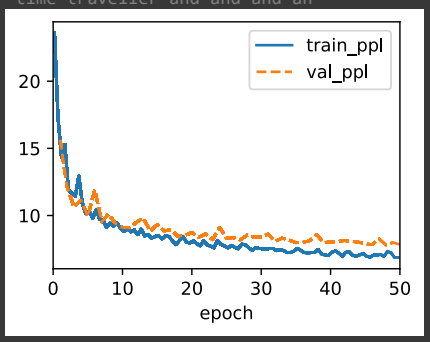


Adjusting RNN num_hiddens=18, num_layers=10, dropout=1

```
class adjRNN(d2l.Module):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout):
        super().__init__()
        self.save_hyperparameters()
        self.rnn = nn.RNN(num_inputs, num_hiddens)

    def forward(self, inputs, H=None):
        return self.rnn(inputs, H)
```

```
rnn = adjRNN(num_inputs=len(data.vocab), num_layers=10, num_hiddens=18, dropout=1)
model = d2l.RNNLM(rnn, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

```
'time traveller and and and an'
```

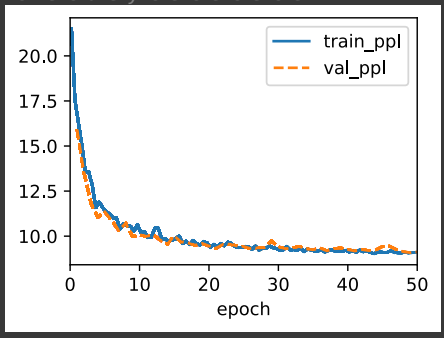


Adjusting RNN num_hiddens=6, num_layers=2, dropout=0

```
rnn = adjRNN(num_inputs=len(data.vocab), num_layers=2, num_hiddens=6, dropout=0)
model = d2l.RNNLM(rnn, vocab_size=len(data.vocab), lr=4)
```

```
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

```
'time travely the the the the '
```



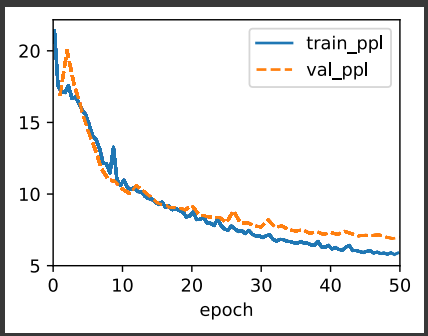
In this section, we compared the perplexity and forecasts of our top LSTM and GRU models to those of an RNN model. 64 hidden layers seems to produce a better forecast even if the perplexity was larger, so we did that. Once more, the LSTM and GRU models produced comparable outcomes, but the RNN performed appallingly. Although there was little confusion, the prognosis was awful.

Q 2 Building the model by replacing the GRU with an LSTM

--REPLACING GRU WITH AN LSTM IS SAME AS Q 1.2 TAKING Q 1.2 WE GET

To examine how the training and usage of our data were affected, we applied LSTM GRU models to a deep neural network in this part. In order to solve this issue, we initially evaluated a standard 32-layer LSTM model. The confusion of training and validation was higher than previously and only started to decline in the later epochs. The forecast was less accurate than earlier LSTM models.

```
class LSTM(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens)

    def forward(self, inputs, H_C=None):
        return self.rnn(inputs, H_C)

lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32)
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
```



Q 2.2 Comparing runtime for training and inference, computational and mode size complexities, and the output strings for nn.LSTM and rnn.GRU implementations

```
import time

# Define input sequence and hyperparameters
input_seq_len = 50
hidden_size = 64
batch_size = 16
num_layers = 2

# Define LSTM model
lstm_model = torch.nn.LSTM(input_size=1, hidden_size=hidden_size, num_layers=num_layers)
```

```
lstm_model = torch.nn.LSTM(input_size=1, hidden_size=hidden_size, num_layers=num_layers)
# Define GRU model
gru_model = torch.nn.GRU(input_size=1, hidden_size=hidden_size, num_layers=num_layers)

# Generate random input data
input_data = torch.randn(input_seq_len, batch_size, 1)

# Train and time LSTM model
start_time = time.time()
lstm_out, _ = lstm_model(input_data)
loss = torch.mean(lstm_out)
loss.backward()
end_time = time.time()
lstm_train_time = end_time - start_time

# Inference and time LSTM model
start_time = time.time()
lstm_out, _ = lstm_model(input_data)
end_time = time.time()
lstm_inference_time = end_time - start_time

# Train and time GRU model
start_time = time.time()
gru_out, _ = gru_model(input_data)
loss = torch.mean(gru_out)
loss.backward()
end_time = time.time()
gru_train_time = end_time - start_time

# Inference and time GRU model
start_time = time.time()
gru_out, _ = gru_model(input_data)
end_time = time.time()
gru_inference_time = end_time - start_time

# Compute model sizes
lstm_model_size = sum(p.numel() for p in lstm_model.parameters())
gru_model_size = sum(p.numel() for p in gru_model.parameters())

# Compare output strings
lstm_output_string = str(lstm_out.detach().numpy())
gru_output_string = str(gru_out.detach().numpy())

# Print results
print("LSTM train time:", lstm_train_time)
print("LSTM inference time:", lstm_inference_time)
print("GRU train time:", gru_train_time)
print("GRU inference time:", gru_inference_time)
print("LSTM model size:", lstm_model_size)
print("GRU model size:", gru_model_size)
print("LSTM output string:", lstm_output_string)
print("GRU output string:", gru_output_string)
```

```
LSTM train time: 0.04556608200073242
LSTM inference time: 0.020972013473510742
GRU train time: 0.03356361389160156
GRU inference time: 0.009248733520507812
LSTM model size: 50432
GRU model size: 37824
LSTM output string: [[[-0.02202267 -0.01259168 -0.02612131 ... -0.02531106 -0.05288368
   -0.03348327]
  [-0.0221755  -0.01270712 -0.0260102  ... -0.02541824 -0.05308561
   -0.03334538]
  [-0.0211253  -0.01197402 -0.02678187 ... -0.02465654 -0.05173292
   -0.03423838]
  ...
  [-0.0214461  -0.01218298 -0.02654434 ... -0.02489476 -0.05213714
   -0.03397916]
  [-0.02367932 -0.01399851 -0.02494982 ... -0.02636959 -0.05514858
   -0.03184713]
  [-0.02171363 -0.01236734 -0.02634736 ... -0.02509007 -0.05248046
   -0.03375384]]

 [[-0.03177744 -0.02570961 -0.0374683  ... -0.04225768 -0.07847
   -0.04932812]
  [-0.0340411  -0.02798934 -0.03533737 ... -0.04409608 -0.08199383
   -0.0473148 ]
  [-0.03224982 -0.02616622 -0.03708808 ... -0.04305071 -0.07926976
   -0.04843595]
```

```
        ...
        [-0.03344662 -0.02744996 -0.0359998  ... -0.04386961 -0.08109507
         -0.04745707]
        [-0.03334219 -0.02740166 -0.03593312 ... -0.04272868 -0.08069384
         -0.04859996]
        [-0.03132051 -0.02530124 -0.03789398 ... -0.04199495 -0.07783253
         -0.04954597]]

       [[-0.0357587  -0.03639214 -0.04285477 ... -0.05174653 -0.08946674
         -0.05597039]
        [-0.04160872 -0.0431502  -0.03601929 ... -0.0561632  -0.09935361
         -0.05170431]
        [-0.03795388 -0.03826882 -0.04069474 ... -0.05360558 -0.09265089
         -0.05436734]
        ...
        [-0.03731362 -0.03838263 -0.04134405 ... -0.05210509 -0.09153415
         -0.05512777]
        [-0.03841144 -0.03951558 -0.03964695 ... -0.05361768 -0.09384558
         -0.05466664]
        [-0.03569326 -0.03617063 -0.04297871 ... -0.05191599 -0.08936016
         -0.05578634]]

       ...

       [[-0.04960123 -0.07376627 -0.03166776 ... -0.06898189 -0.11790223
         -0.04973987]
        [-0.04422221 -0.06499436 -0.04086899 ... -0.06589317 -0.10700628
         -0.05225989]
        [-0.0463783  -0.07055568 -0.0348895  ... -0.0668018  -0.11310998
         -0.05405071]
        ...
        [-0.04860405 -0.07108512 -0.03417033 ... -0.06883562 -0.11523098
```

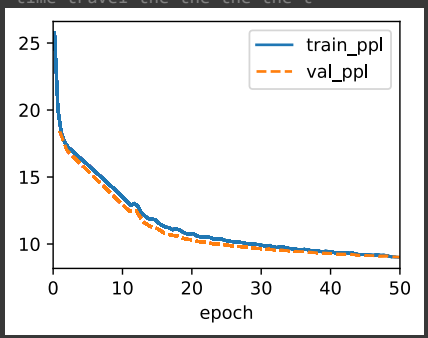
Q 2.3 Adjusting the hyperparameters (fully connected network, number of hidden layers, and the number of hidden states) and comparing your results

:Adjusting hyperparameters of GRU with lr= 1, num_hiddens= 10, num_layer= 4, dropout=1

```
class adjGRU(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.GRU(num_inputs, num_hiddens)
```

```
gru = adjGRU(num_inputs=len(data.vocab), num_hiddens=10, num_layers=4, dropout=1)
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=1)
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
trainer.fit(model, data)
model.predict("time travel", 18, data.vocab, d2l.try_gpu())
```

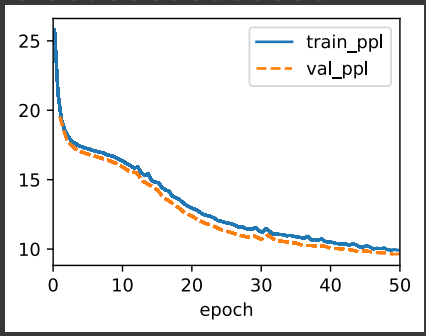
```
'time travel the the the the t'
```




```
class adjLSTM(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens)

    def forward(self, inputs, H_C=None):
        return self.rnn(inputs, H_C)
```

```
lstm = adjLSTM(num_inputs=len(data.vocab), num_hiddens=12, num_layers=10, dropout=1)
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=1)
trainer.fit(model, data)
model.predict('time travel', 20, data.vocab, d2l.try_gpu())
```

```
'time travele the the the the th'
```




```
class RNN(d2l.Module):
    def __init__(self, num_inputs, num_hiddens):
        super().__init__()
        self.save_hyperparameters()
        self.rnn = nn.RNN(num_inputs, num_hiddens)

    def forward(self, inputs, H=None):
        return self.rnn(inputs, H)
```

```
rnn = nn.RNN(input_size=len(data.vocab), hidden_size=12, num_layers=8, dropout=0)
model = d2l.RNNLM(rnn, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
model.predict("time travel", 20, data.vocab, d2l.try_gpu())
```

```
'time travel                    '
```