Homework-4

Name: Hanumantha Rao Vakkalanka

Email: hvakkala@uncc.edu

Student ID: 801333188

Course: ECGR 5106 Real Time ML

Lab Number: Spring 2023

```
!pip install torch torchvision
!pip install d2l==1.0.0a1.post0
!pip install matplotlib_inline
```

```
    ollecting qtconsole
     Downloading qtconsole-5.4.1-py3-none-any.whl (120 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 120.9/120.9 KB 13.4 MB/s eta 0:00:00
    equirement already satisfied: nbconvert in /usr/local/lib/python3.9/dist-packages (from jupyter->d2l==1.0.0a1.post0) (6.5.4)
    equirement already satisfied: importlib-resources>=3.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.pos
    equirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (2
    equirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (8.4.0)
    equirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (3.0.9
    equirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (1.4.
    equirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (0.11.0)
    equirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (1.0.7
    equirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (4.39
    equirement already satisfied: ckaging>=20.0 in /usr/local/lib/python3.9/dist-packages (from matplotlib->d2l==1.0.0a1.post0) (23.0)
    equirement already satisfied: aitlets in /usr/local/lib/python3.9/dist-packages (from matplotlib-inline->d2l==1.0.0a1.post0) (5.7.1
    equirement already satisfied: tz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas->d2l==1.0.0a1.post0) (2022.7.1)
    equirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0) (3.4)
    equirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0) (1.
    equirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0) (2022.
    equirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->d2l==1.0.0a1.post0)
    equirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.9/dist-packages (from importlib-metadata>=4.8.0->gym->d2l==1.0.0a1.
    equirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7->matplotlib->d2l==1.0.0a1
    equirement already satisfied: ipython-genutils in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0a1.post0
    equirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0a1.post0) (6
    equirement already satisfied: jupyter-client in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: ipython>=5.0.0 in /usr/local/lib/python3.9/dist-packages (from ipykernel->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/python3.9/dist-packages (from ipywidgets->jupyter->d2l==1.0
    equirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.9/dist-packages (from ipywidgets->jupyter->d2l==1.0
    equirement already satisfied: pygments in /usr/local/lib/python3.9/dist-packages (from jupyter-console->jupyter->d2l==1.0.0a1.post0)
    equirement 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-co
    equirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: bleach in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (6.0.0)
    equirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.p
    equirement already satisfied: jupyterlab-pygments in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.po
    equirement already satisfied: jupyter-core>=4.7 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post
    equirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (4.9.2)
    equirement already satisfied: nbclient>=0.5.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: nbformat>=5.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (
    equirement already satisfied: tinycss2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (1.2.1
    equirement already satisfied: jinja2>=3.0 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (3.
    equirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post
    equirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.pos
    equirement already satisfied: defusedxml in /usr/local/lib/python3.9/dist-packages (from nbconvert->jupyter->d2l==1.0.0a1.post0) (0.7
    equirement already satisfied: pyzmq>=17 in /usr/local/lib/python3.9/dist-packages (from notebook->jupyter->d2l==1.0.0a1.post0) (23.2.
    equirement already satisfied: terminado>=0.8.3 in /usr/local/lib/python3.9/dist-packages (from notebook->jupyter->d2l==1.0.0a1.post0)
    equirement already satisfied: argon2-cffi in /usr/local/lib/python3.9/dist-packages (from notebook->jupyter->d2l==1.0.0a1.post0) (21.
    equirement already satisfied: Send2Trash>=1.5.0 in /usr/local/lib/python3.9/dist-packages (from notebook->jupyter->d2l==1.0.0a1.post0
    equirement already satisfied: prometheus-client in /usr/local/lib/python3.9/dist-packages (from notebook->jupyter->d2l==1.0.0a1.post0
    ollecting qtpy>=2.0.1
     Downloading QtPy-2.3.1-py3-none-any.whl (84 kB)
        ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.9/84.9 KB 10.2 MB/s eta 0:00:00
    equirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.9/dist-packages (from ipython>=5.0.0->ipykernel->jupyter->d2
    equirement already satisfied: pickleshare in /usr/local/lib/python3.9/dist-packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.
    equirement already satisfied: backcall in /usr/local/lib/python3.9/dist-packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.0
    equirement already satisfied: decorator in /usr/local/lib/python3.9/dist-packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.0.
    equirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.9/dist-packages (from ipython>=5.0.0->ipykernel->jupyter->d2l==1.
    ollecting jedi>=0.16
```

Saved successfully!

```
%matplotlib inline
import time
import torch
import torchvision
from torchvision import transforms
```

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

d2l.use_svg_display()
```

```
/usr/local/lib/python3.9/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

```
!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/
Collecting ptflops
  Downloading ptflops-0.6.9.tar.gz (12 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: torch in /usr/local/lib/python3.9/dist-packages (from ptflops) (1.13.1+cu116)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.9/dist-packages (from torch->ptflops) (4.5.0)
Building wheels for collected packages: ptflops
  Building wheel for ptflops (setup.py) ... done
  Created wheel for ptflops: filename=ptflops-0.6.9-py3-none-any.whl size=11711 sha256=f229fea58a1905d4e87c98d84e5b57f87628a7b2090f35a01
  Stored in directory: /root/.cache/pip/wheels/86/07/9f/879035d99d7b639bbc564d23fed862a679aee7d1a2dced8c2e
Successfully built ptflops
Installing collected packages: ptflops
Successfully installed ptflops-0.6.9
```



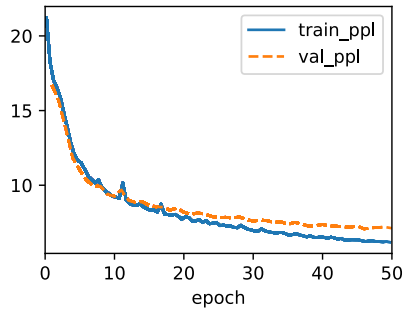
```
class GRU(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens):
        # Initialize the class GRU as a subclass of d2l.RNN
        d2l.Module.__init__(self)
        # Call the parent class constructor to initialize the module and save the hyperparameters
        self.save_hyperparameters()
        # Create an instance variable to store the hyperparameters for later use
        self.rnn = nn.GRU(num_inputs, num_hiddens)
        # Initialize a PyTorch GRU module with the given number of input and hidden units
```

```
data = d2l.TimeMachine(batch_size=1024, num_steps=32)
```

```
Downloading ../data/timemachine.txt from http://d2l-data.s3-accelerate.amazonaws.com/timemachine.txt...
```

```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=16)
# Instantiate a GRU object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 16 units
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
# Create a RNNLM object with the GRU as the underlying recurrent layer, and with the vocabulary size of the data as the number of output unit
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Instantiate a Trainer object with a maximum of 50 epochs, a gradient clipping value of 1, and using 1 GPU.
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Use the trained model to predict the next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```

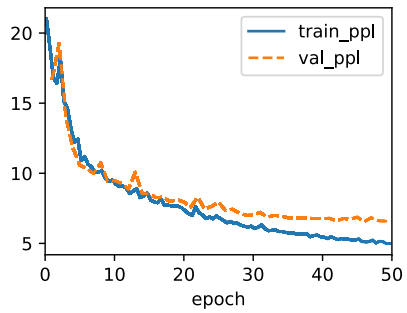
```
'it has and the the the the'
```




```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32)
# Instantiate a GRU object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 32 units
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
```

```
# Create a RNNLM object with the GRU as the underlying recurrent layer, and with the vocabulary size of the data as the number of output unit
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Instantiate a Trainer object with a maximum of 50 epochs, a gradient clipping value of 1, and using 1 GPU.
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Use the trained model to predict the next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```

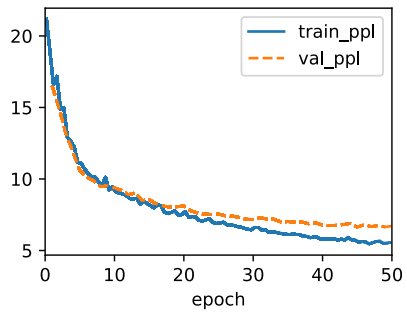
```
'it has so have and the tim'
```




```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=24)
# Instantiate a GRU object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 24 units
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
# Create a RNNLM object with the GRU as the underlying recurrent layer, and with the vocabulary size of the data as the number of output unit
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Instantiate a Trainer object with a maximum of 50 epochs, a gradient clipping value of 1, and using 1 GPU.
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Use the trained model to predict the next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```



```
'it has is and and the time'
```



(2)

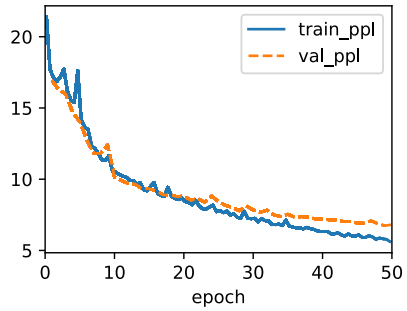
```
class LSTM(d2l.RNN):
    # Define a class LSTM that extends the d2l.RNN class
    def __init__(self, num_inputs, num_hiddens):
        # Initialize the object
        d2l.Module.__init__(self)
        # Call the superclass constructor
        self.save_hyperparameters()
        # Save the hyperparameters of the model
        self.rnn = nn.LSTM(num_inputs, num_hiddens)
        # Initialize the LSTM layer with a given number of input units and a given number of hidden units

    def forward(self, inputs, H_C=None):
        # Define the forward pass through the LSTM layer
        return self.rnn(inputs, H_C)
        # Return the output and the final hidden and cell states of the LSTM layer, given the input and the initial hidden and cell states
```

```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32)
# Instantiate an LSTM object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 32 units
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
# Create a RNNLM object with the LSTM as the underlying recurrent layer, and with the vocabulary size of the data as the number of output uni
```

```
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Use the trained model to predict the next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```

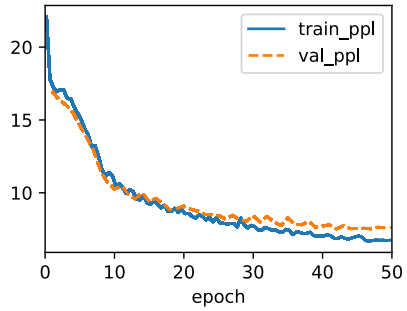
```
'it has i said the time the'
```




```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=16)
# Instantiate an LSTM object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 64 units
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
# Create a RNNLM object with the LSTM as the underlying recurrent layer, and with the vocabulary size of the data as the number of output uni
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
ab, d2l.try_gpu())
he next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```

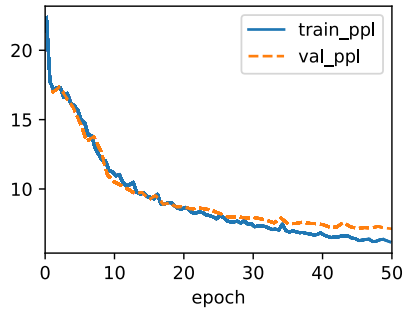
Saved successfully!

```
'it has the the the the the'
```




```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=24)
# Instantiate an LSTM object with a number of input units equal to the vocabulary size of the data, and a hidden state size of 64 units
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
# Create a RNNLM object with the LSTM as the underlying recurrent layer, and with the vocabulary size of the data as the number of output uni
trainer.fit(model, data)
# Train the model using the Trainer object on the given data.
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Use the trained model to predict the next 20 words given the starting phrase "it has", using the vocabulary of the data and trying to use a
```

```
'it has of the the the the '
```



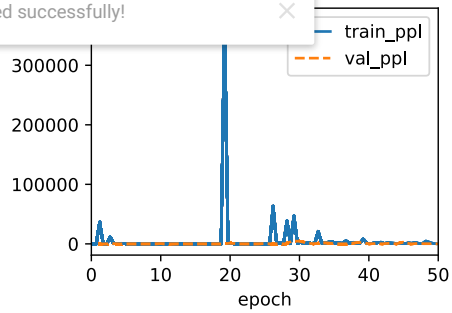
(3)

```
class RNN(d2l.Module):
    # Define a new class called RNN that inherits from the d2l.Module class
    def __init__(self, num_inputs, num_hiddens):
        # Define the constructor method for the RNN class, taking in the number of input units and number of hidden units as arguments
        super().__init__()
        # Call the constructor of the superclass (d2l.Module)
        self.save_hyperparameters()
        # Save the hyperparameters of the RNN (num_inputs and num_hiddens) for later reference
        self.rnn = nn.RNN(num_inputs, num_hiddens)
        # Create an RNN layer with the given number of input and hidden units and store it as an attribute of the RNN object

    def forward(self, inputs, H=None):
        # Define the forward method of the RNN class, taking in inputs and an optional hidden state H as arguments
        return self.rnn(inputs, H)
        # Return the output and hidden state from passing the inputs and hidden state through the RNN layer defined in the constructor
```

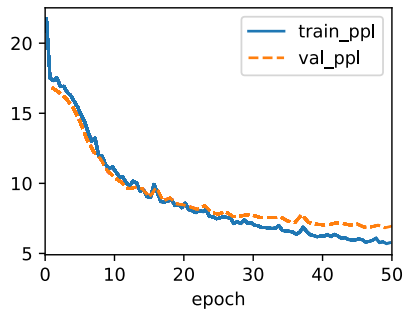
```
rnn = RNN(num_inputs=len(data.vocab), num_hiddens=32)
# Create an RNN object with the specified number of input and hidden units
model = d2l.RNNLM(rnn, vocab_size=len(data.vocab), lr=4)
# Create a language model using the RNN object and the vocabulary size, with a learning rate of 4
trainer.fit(model, data)
# Train the language model using the specified Trainer object and training data
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Generate a sequence of 20 words using the trained language model, starting with the phrase "it has" and using the provided vocabulary, and
```

Saved successfully!




```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32)
# Create an LSTM object with the specified number of input and hidden units
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
# Create a language model using the LSTM object and the vocabulary size, with a learning rate of 4
trainer.fit(model, data)
# Train the language model using the specified Trainer object and training data
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# Generate a sequence of 20 words using the trained language model, starting with the phrase "it has" and using the provided vocabulary, and
```

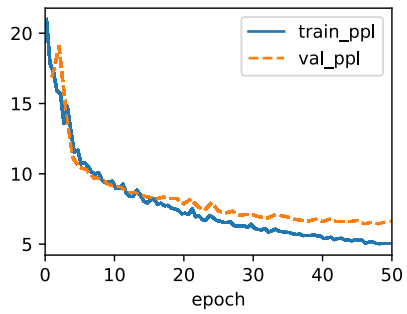
```
'it has and the the the pro'
```




```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32)
# create a GRU model with 64 hidden units
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
# create an RNNLM model with the GRU model, vocabulary size, and learning rate
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# create a trainer object with 50 maximum epochs, gradient clipping value of 1, and use 1 GPU
trainer.fit(model, data)
# train the model with the data using the trainer
```

```
model.predict("it has", 20, data.vocab, d2l.try_gpu())
# generate 20 words starting with "it has" using the trained model and the vocabulary in the data. Try to use a GPU if available.
```

```
'it has expect and the time'
```



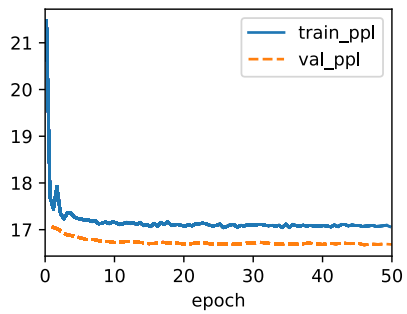
Problem 2(1)

```
class LSTM(d2l.RNN):
    # Define a class LSTM that extends the d2l.RNN class
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout = 0):
        # Initialize the object
        d2l.Module.__init__(self)
        # Call the constructor of the parent class to initialize hyperparameters
        self.save_hyperparameters()
        # Save the hyperparameters
        self.rnn = nn.LSTM(num_inputs, num_hiddens,  num_layers, dropout = dropout)
        # Initialize the LSTM layer with a given number of input units and a given number of hidden units
```

Saved successfully!

```
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32, num_layers=4)
 # Define LSTM with 4 layers
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
 # Define RNNLM model with the LSTM defined above
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
 # Define trainer with maximum epochs, gradient clipping, and number of GPUs
trainer.fit(model, data)
 # Train the model using the data and the trainer
model.predict("it has", 20, data.vocab, d2l.try_gpu())
 # Generate text by giving an initial text and the model's learned vocabulary
```

```
'it has                    '
```

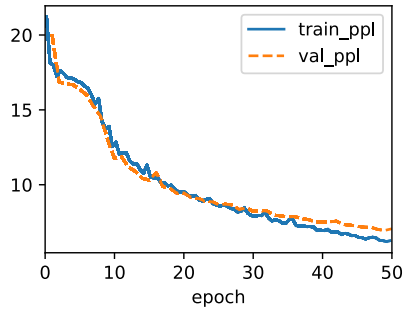


(2)

```
class GRU(d2l.RNN):
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout=0):
        # Call the constructor of the parent class to initialize hyperparameters
        d2l.Module.__init__(self)
        # Save the hyperparameters
        self.save_hyperparameters()
        # Define the GRU layer
        self.rnn = nn.GRU(num_inputs, num_hiddens, num_layers,
                          dropout=dropout)
```

```
# Define a GRU with 2 layers, each with 32 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=2)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

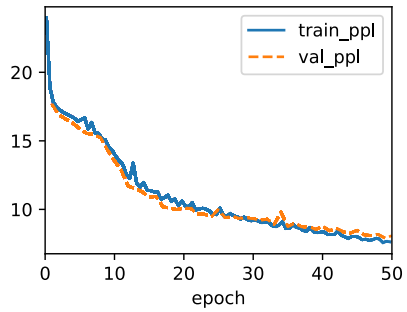
```
'it has a man the the the t'
```



Saved successfully!

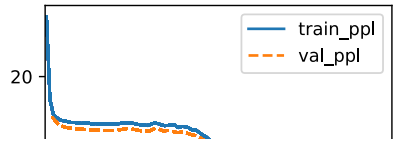
```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32, num_layers=1)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has the the the the the'
```



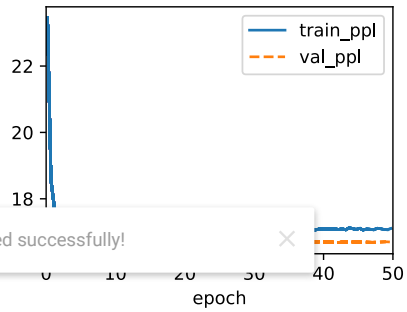

```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32, num_layers=2)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has the the the the the'
```



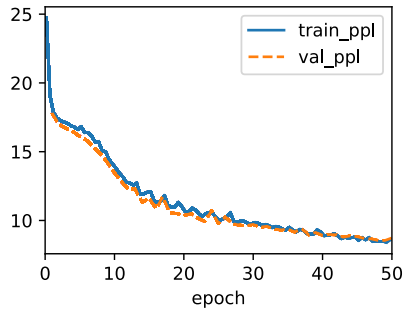

```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=64, num_layers=4)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has                    '
```



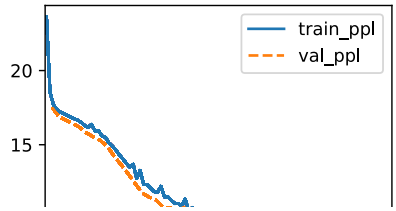

```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=16, num_layers=1)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has and the the the the'
```



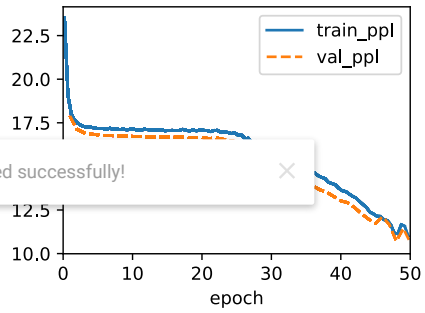

```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=16, num_layers=1)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has and an a the the th'
```




```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=16, num_layers=2)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

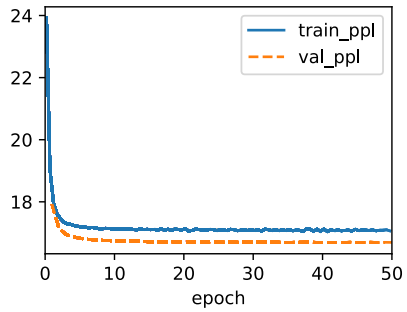
```
'it has the the the the the'
```



Saved successfully!

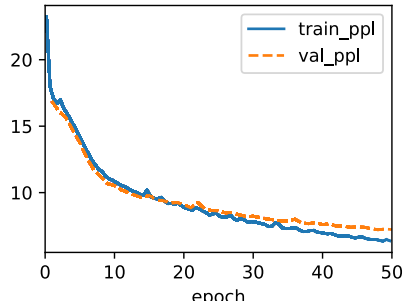
```
# Define an LSTM with 1 layer, 64 hidden units, and input size of the vocabulary size
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=16, num_layers=4)
# Define a RNN language model with the LSTM and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has                    '
```



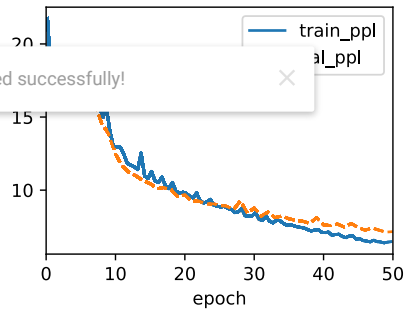

```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=1)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has in and the the the '
```




```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=2)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

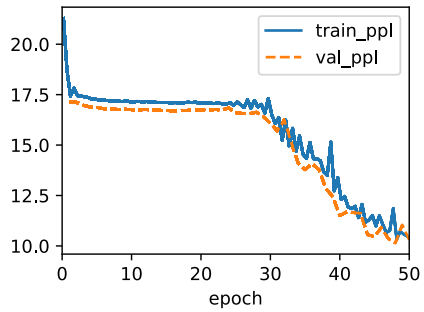
```
'it has a man the the the t'
```



Saved successfully!

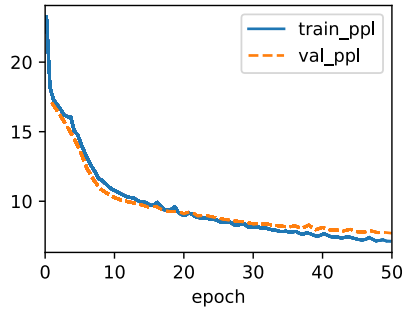
```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=4)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has the thennn the then'
```



```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens= 16, num_layers=1)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

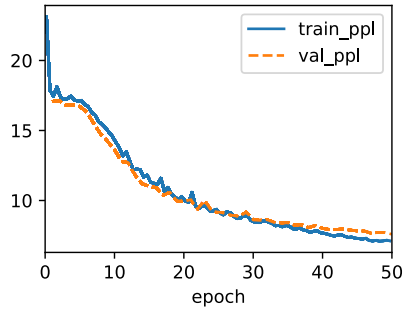
```
'it has and the the the the'
```




```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
                                                  , num_hiddens= 16, num_layers=2)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
                                                  n(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

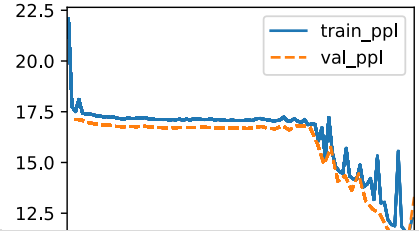
Saved successfully!

```
'it has a mens the the the '
```




```
# Define a GRU with 1 layer, 64 hidden units, and input size of the vocabulary size
gru = GRU(num_inputs=len(data.vocab), num_hiddens= 16, num_layers=4)
# Define a RNN language model with the GRU and the vocabulary size as input and learning rate of 2
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=2)
# Define a trainer with max_epochs of 50, gradient clip value of 1, and 1 GPU for training
trainer = d2l.Trainer(max_epochs=50, gradient_clip_val=1, num_gpus=1)
# Fit the model with the trainer and data
trainer.fit(model, data)
# Generate a sequence of 20 tokens with the given prefix "it has" using the trained model and the vocabulary
model.predict("it has", 20, data.vocab, d2l.try_gpu())
```

```
'it has are the are the are'
```



✓ 2m 44s completed at 3:56 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.

Saved successfully!