```
! pip install setuptools==66
```

```
! pip install d2l==1.0.0b0
```

```
!pip install matplotlib_inline
```

```
! pip install ptflops
```

```
import time
import torch
import torchvision
from torch import nn
from torchvision import transforms
from d2l import torch as d2l
from ptflops import get_model_complexity_info
```

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

D2L Stuff

```
class HyperParameters:
    def save_hyperparameters(self, ignore=[]):
        raise NotImplemented
```

```
def add_to_class(Class):
    def wrapper(obj):
        setattr(Class, obj.__name__, obj)
    return wrapper
```

```
class ProgressBoard(d2l.HyperParameters):
    def __init__(self, xlabel=None, ylabel=None, xlim=None,
                 ylim=None, xscale='linear', yscale='linear',
                 ls=['-', '--', '-.', ':'], colors=['C0', 'C1', 'C2', 'C3'],
                 fig=None, axes=None, figsize=(3.5, 2.5), display=True):
        self.save_hyperparameters()

    def draw(self, x, y, label, every_n=1):
        raise NotImplemented
```

```
class Module(nn.Module, d2l.HyperParameters):
    def __init__(self, plot_train_per_epoch=2, plot_valid_per_epoch=1):
        super().__init__()
        self.save_hyperparameters()
        self.board = ProgressBoard()
    def loss(self, y_hat, y):
        raise NotImplementedError

    def forward(self, X):
        assert hasattr(self, 'net'), 'Neural network is defined'
        return self.net(X)

    def plot(self, key, value, train):
        """Plot a point in animation."""
        assert hasattr(self, 'trainer'), 'Trainer is not inited'
        self.board.xlabel = 'epoch'
        if train:
            x = self.trainer.train_batch_idx / \
                self.trainer.num_train_batches
            n = self.trainer.num_train_batches / \
                self.plot_train_per_epoch
        else:
            x = self.trainer.epoch + 1
            n = self.trainer.num_val_batches / \
```

```
                self.plot_valid_per_epoch
        self.board.draw(x, value.to(d2l.cpu()).detach().numpy(),
                        ('train_' if train else 'val_') + key,
                        every_n=int(n))
    def training_step(self, batch):
        l = self.loss(self(*batch[:-1]), batch[-1])
        self.plot('loss', l, train=True)
        return l

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

    def configure_optimizers(self):
        raise NotImplementedError

class Trainer(d2l.HyperParameters):
    def __init__(self, max_epochs, num_gpus=0, gradient_clip_val=0):
        self.save_hyperparameters()
        assert num_gpus == 0, 'No GPU support yet'

    def prepare_data(self, data):
        self.train_dataloader = data.train_dataloader()
        self.val_dataloader = data.val_dataloader()
        self.num_train_batches = len(self.train_dataloader)
        self.num_val_batches = (len(self.val_dataloader)
                                if self.val_dataloader is not None else 0)

    def prepare_model(self, model):
        model.trainer = self
        model.board.xlim = [0, self.max_epochs]
        self.model = model

    def fit(self, model, data):
        self.prepare_data(data)
        self.prepare_model(model)
        self.optim = model.configure_optimizers()
        self.epoch = 0
        self.train_batch_idx = 0
        self.val_batch_idx = 0
        for self.epoch in range(self.max_epochs):
            self.fit_epoch()

    def fit_epoch(self):
        raise NotImplementedError


@d2l.add_to_class(d2l.Trainer)
def prepare_batch(self, batch):
    return batch

@d2l.add_to_class(d2l.Trainer)
def fit_epoch(self):
    self.model.train()
    #self.totalLoss = 0
    #self.epochLoss = 0
    #self.totalValLoss = 0
    #self.valEpochLoss = 0
    #self.lossHist = []

    for batch in self.train_dataloader:
        loss = self.model.training_step(self.prepare_batch(batch))
        #self.totalLoss = self.totalLoss + loss.item()
        self.optim.zero_grad()
        with torch.no_grad():
            loss.backward()
            if self.gradient_clip_val > 0:  # To be discussed later
                self.clip_gradients(self.gradient_clip_val, self.model)
            self.optim.step()
        self.train_batch_idx += 1

    #self.loaderLength = len(self.train_dataloader)
    #self.epochLoss = self.totalLoss / self.loaderLength
```

```
    #print("Epoch loss: ", self.epochLoss)
    #self.lossHist.append(self.epochLoss)
    #globEpochLoss.append(self.epochLoss)

    if self.val_dataloader is None:
        return
    self.model.eval()
    for batch in self.val_dataloader:
        with torch.no_grad():
            self.model.validation_step(self.prepare_batch(batch))
        #valLoss = self.model.validation_step(self.prepare_batch(batch))
        #self.totalValLoss = self.totalValLoss + valLoss
        self.val_batch_idx += 1

    #self.valLoaderLength = len(self.val_dataloader)
    #self.valEpochLoss = self.totalValLoss / self.valLoaderLength
    #globValLoss.append(self.valEpochLoss)


class DataModule(d2l.HyperParameters):
    def __init__(self, root='../data', num_workers=4):
        self.save_hyperparameters()

    def get_dataloader(self, train):
        raise NotImplementedError

    def train_dataloader(self):
        return self.get_dataloader(train=True)

    def val_dataloader(self):
        return self.get_dataloader(train=False)
```

Preparing the Dataset

```
import collections
import random
import re


class TimeMachine(d2l.DataModule):
    """The Time Machine dataset."""
    def _download(self):
        fname = d2l.download(d2l.DATA_URL + 'timemachine.txt', self.root,
                             '090b5e7e70c295757f55df93cb0a180b9691891a')
        with open(fname) as f:
            return f.read()

data = TimeMachine()
raw_text = data._download()
raw_text[:60]
```

```
Downloading ../data/timemachine.txt from http://d2l-data.s3-accelerate.amazonaws.com/timemachine.txt...
'The Time Machine, by H. G. Wells [1898]\n\n\n\n\nI\n\n\nThe Time Tra'
```

```
@d2l.add_to_class(TimeMachine)
def _preprocess(self, text):
    return re.sub('[^A-Za-z]+', ' ', text).lower()

text = data._preprocess(raw_text)
text[:60]
```

```
'the time machine by h g wells i the time traveller for so it'
```

```
@d2l.add_to_class(TimeMachine)
def _tokenize(self, text):
    return list(text)

tokens = data._tokenize(text)
','.join(tokens[:30])
```

```
't,h,e, ,t,i,m,e, ,m,a,c,h,i,n,e, ,b,y, ,h, ,g, ,w,e,l,l,s, '
```

```
class Vocab:
    """Vocabulary for text."""
    def __init__(self, tokens=[], min_freq=0, reserved_tokens=[]):
        # Flatten a 2D list if needed
        if tokens and isinstance(tokens[0], list):
            tokens = [token for line in tokens for token in line]
        # Count token frequencies
        counter = collections.Counter(tokens)
        self.token_freqs = sorted(counter.items(), key=lambda x: x[1],
                                  reverse=True)
        # The list of unique tokens
        self.idx_to_token = list(sorted(set(['<unk>'] + reserved_tokens + [
            token for token, freq in self.token_freqs if freq >= min_freq])))
        self.token_to_idx = {token: idx
                             for idx, token in enumerate(self.idx_to_token)}

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

    def __getitem__(self, tokens):
        if not isinstance(tokens, (list, tuple)):
            return self.token_to_idx.get(tokens, self.unk)
        return [self.__getitem__(token) for token in tokens]

    def to_tokens(self, indices):
        if hasattr(indices, '__len__') and len(indices) > 1:
            return [self.idx_to_token[int(index)] for index in indices]
        return self.idx_to_token[indices]

    @property
    def unk(self):  # Index for the unknown token
        return self.token_to_idx['<unk>']


vocab = Vocab(tokens)
indices = vocab[tokens[:10]]
print('indices:', indices)
print('words:', vocab.to_tokens(indices))
```

```
indices: [21, 9, 6, 0, 21, 10, 14, 6, 0, 14]
words: ['t', 'h', 'e', ' ', 't', 'i', 'm', 'e', ' ', 'm']
```

```
@d2l.add_to_class(TimeMachine)
def build(self, raw_text, vocab=None):
    tokens = self._tokenize(self._preprocess(raw_text))
    if vocab is None: vocab = Vocab(tokens)
    corpus = [vocab[token] for token in tokens]
    return corpus, vocab

corpus, vocab = data.build(raw_text)
len(corpus), len(vocab)
```

```
(173428, 28)
```

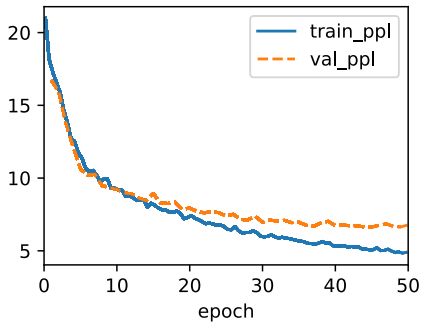
1a) GRU

```
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)


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


# Default network from example
```

```
gru = GRU(num_inputs=len(data.vocab), num_hiddens=32)
model = d2l.RNNLM(gru, vocab_size=len(data.vocab), lr=4)
trainer.fit(model, data)
# Train time: 1min 44s
```




```
model.predict('it has', 20, data.vocab, d2l.try_gpu())
```

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

```
model.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are alwaysed the time the time'
```

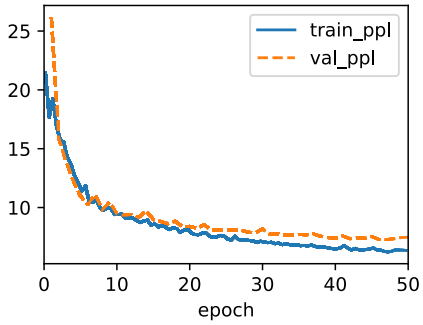
```
model.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across the time the time t'
```

```
macs, params = get_model_complexity_info(gru, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: variables __flops__ or __params__ are already defined for the moduleGRU ptflops can affect your code!
Warning: module GRU is treated as a zero-op.
GRU(
  5.95 k, 100.000% Params, 6.32 MMac, 100.000% MACs,
  (rnn): GRU(5.95 k, 100.000% Params, 6.32 MMac, 100.000% MACs, 28, 32)
)
Computational complexity:  6.32 MMac
Number of parameters:  5.95 k
```

```
# Halfing the number of hidden states to 16
gru16 = GRU(num_inputs=len(data.vocab), num_hiddens=16)
gruModel16 = d2l.RNNLM(gru16, vocab_size=len(data.vocab), lr=4)
trainer.fit(gruModel16, data)
# Training time: 1min 31sec
```




```
gruModel16.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has mong the the the th'
```

```
gruModel16.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always the the the the the'
```

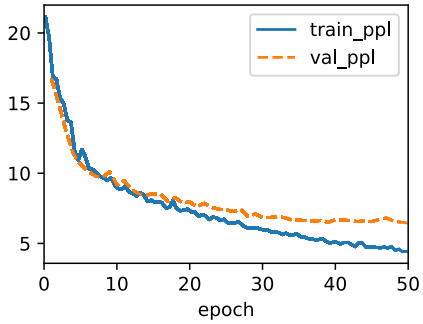
```
gruModel16.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked acrossing the the the the '
```

```
macs, params = get_model_complexity_info(gru16, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module GRU is treated as a zero-op.
GRU(
  2.21 k, 100.000% Params, 2.38 MMac, 100.000% MACs,
  (rnn): GRU(2.21 k, 100.000% Params, 2.38 MMac, 100.000% MACs, 28, 16)
)
Computational complexity:  2.38 MMac
Number of parameters:  2.21 k
```

```
# Increasing the number of hidden states to 48
gru48 = GRU(num_inputs=len(data.vocab), num_hiddens=48)
gruModel48 = d2l.RNNLM(gru48, vocab_size=len(data.vocab), lr=4)
trainer.fit(gruModel48, data)
# Training time: 2mins 14sec
```




```
gruModel48.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has interical mentered '
```

```
gruModel48.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always this space the psyc'
```

```
gruModel48.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

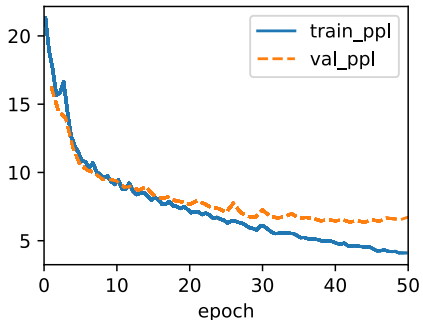
```
'he looked across for instance the ti'
```

```
macs, params = get_model_complexity_info(gru48, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module GRU is treated as a zero-op.
GRU(
  11.23 k, 100.000% Params, 11.85 MMac, 100.000% MACs,
  (rnn): GRU(11.23 k, 100.000% Params, 11.85 MMac, 100.000% MACs, 28, 48)
)
Computational complexity:  11.85 MMac
Number of parameters:  11.23 k
```

```
# Increasing the number of hidden states to 64
gru64 = GRU(num_inputs=len(data.vocab), num_hiddens=64)
gruModel64 = d2l.RNNLM(gru64, vocab_size=len(data.vocab), lr=4)
```

```
trainer.fit(gruModel64, data)
# Training time: 2min 43sec
```




```
gruModel64.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has in thing that that '
```

```
gruModel64.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always dimension in i sume'
```

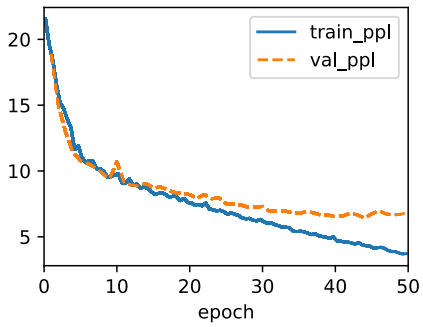
```
gruModel64.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across the psychologist yo'
```

```
macs, params = get_model_complexity_info(gru64, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module GRU is treated as a zero-op.
GRU(
  18.05 k, 100.000% Params, 18.94 MMac, 100.000% MACs,
  (rnn): GRU(18.05 k, 100.000% Params, 18.94 MMac, 100.000% MACs, 28, 64)
)
Computational complexity:  18.94 MMac
Number of parameters:  18.05 k
```

```
# Increasing the number of hidden states to 128
gru128 = GRU(num_inputs=len(data.vocab), num_hiddens=128)
gruModel128 = d2l.RNNLM(gru128, vocab_size=len(data.vocab), lr=4)
trainer.fit(gruModel128, data)
# Training time: 5min 33sec
```




```
gruModel128.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has it and the time tra'
```

```
gruModel128.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always the time traveller '
```

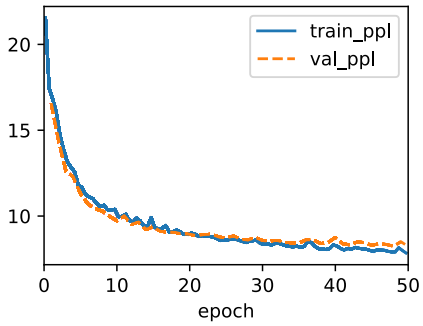
```
gruModel128.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked acrossed the provention is'
```

```
macs, params = get_model_complexity_info(gru128, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module GRU is treated as a zero-op.
GRU(
  60.67 k, 100.000% Params, 63.05 MMac, 100.000% MACs,
  (rnn): GRU(60.67 k, 100.000% Params, 63.05 MMac, 100.000% MACs, 28, 128)
)
Computational complexity:  63.05 MMac
Number of parameters:  60.67 k
```

```
# Decreasing the number of hidden states to 8
gru8 = GRU(num_inputs=len(data.vocab), num_hiddens=8)
gruModel8 = d2l.RNNLM(gru8, vocab_size=len(data.vocab), lr=4)
trainer.fit(gruModel8, data)
# Training time: 1min 29sec
```




```
gruModel8.predict('it has', 20, data.vocab, d2l.try_gpu())
```

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

```
gruModel8.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always the the the the the'
```

```
gruModel8.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across the the the the the'
```

```
macs, params = get_model_complexity_info(gru8, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

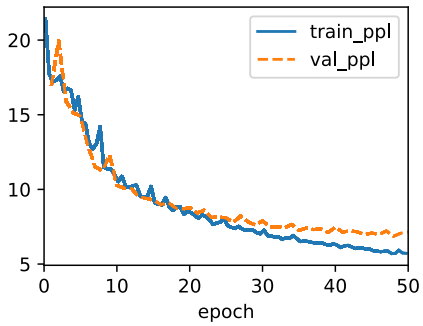
```
Warning: module GRU is treated as a zero-op.
GRU(
  912, 100.000% Params, 991.23 KMac, 100.000% MACs,
  (rnn): GRU(912, 100.000% Params, 991.23 KMac, 100.000% MACs, 28, 8)
)
Computational complexity:  991.23 KMac
Number of parameters:  912
```

1b) LSTM

```
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)


# Default from example
lstm = LSTM(num_inputs=len(data.vocab), num_hiddens=32)
lstmModel = d2l.RNNLM(lstm, vocab_size=len(data.vocab), lr=4)
trainer.fit(lstmModel, data)
# Train time: 2min 30sec
```




```
lstmModel.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
    'it has the traveller and t'
```

```
lstmModel.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
    'we are always of the time travell'
```

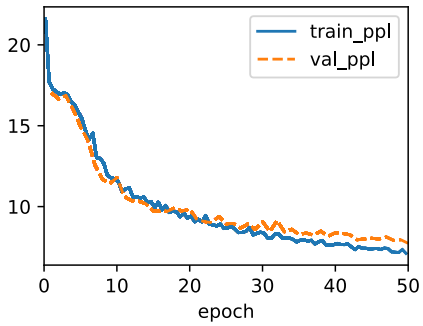
```
lstmModel.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
    'he looked acrossions and the time tr'
```

```
macs, params = get_model_complexity_info(lstm, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
    Warning: module LSTM is treated as a zero-op.
    LSTM(
      7.94 k, 100.000% Params, 8.45 MMac, 100.000% MACs,
      (rnn): LSTM(7.94 k, 100.000% Params, 8.45 MMac, 100.000% MACs, 28, 32)
    )
    Computational complexity:  8.45 MMac
    Number of parameters:  7.94 k
```

```
# Halving the number of hidden states to 16
lstm16 = LSTM(num_inputs=len(data.vocab), num_hiddens=16)
LSTMmodel16 = d2l.RNNLM(lstm16, vocab_size=len(data.vocab), lr=4)
trainer.fit(LSTMmodel16, data)
# Training time: 2min 3sec
```



```
LSTMmodel16.predict('it has', 20, data.vocab, d2l.try_gpu())
```

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

```
LSTMmodel16.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always of the the the the '
```

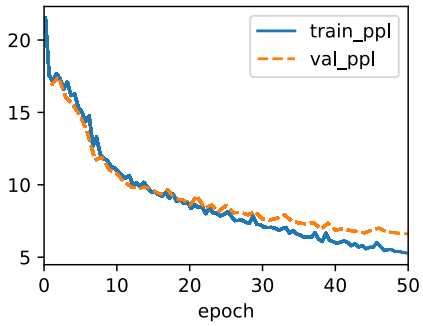
```
LSTMmodel16.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across of the the the the '
```

```
macs, params = get_model_complexity_info(lstm16, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module LSTM is treated as a zero-op.
LSTM(
  2.94 k, 100.000% Params, 3.18 MMac, 100.000% MACs,
  (rnn): LSTM(2.94 k, 100.000% Params, 3.18 MMac, 100.000% MACs, 28, 16)
)
Computational complexity:  3.18 MMac
Number of parameters:  2.94 k
```

```
# Increasing the number of hidden states to 48
lstm48 = LSTM(num_inputs=len(data.vocab), num_hiddens=48)
LSTMmodel48 = d2l.RNNLM(lstm48, vocab_size=len(data.vocab), lr=4)
trainer.fit(LSTMmodel48, data)
# Training time: 3min 1sec
```




```
LSTMmodel48.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has and the that the th'
```

```
LSTMmodel48.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always and the that the th'
```

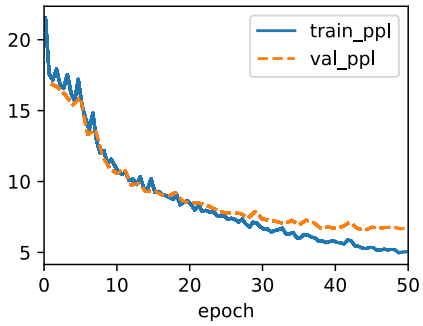
```
LSTMmodel48.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked acrossed and the traveller'
```

```
macs, params = get_model_complexity_info(lstm48, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module LSTM is treated as a zero-op.
LSTM(
  14.98 k, 100.000% Params, 15.83 MMac, 100.000% MACs,
  (rnn): LSTM(14.98 k, 100.000% Params, 15.83 MMac, 100.000% MACs, 28, 48)
)
Computational complexity:  15.83 MMac
Number of parameters:  14.98 k
```

```
# Increasing the number of hidden states to 64
lstm64 = LSTM(num_inputs=len(data.vocab), num_hiddens=64)
LSTMmodel64 = d2l.RNNLM(lstm64, vocab_size=len(data.vocab), lr=4)
trainer.fit(LSTMmodel64, data)
# Training time: 3min 54sec
```




```
LSTMmodel64.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has inour and wather an'
```

```
LSTMmodel64.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always of and wathereard h'
```

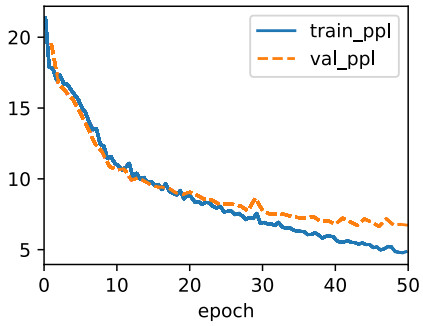
```
LSTMmodel64.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across have and the time t'
```

```
macs, params = get_model_complexity_info(lstm64, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module LSTM is treated as a zero-op.
LSTM(
  24.06 k, 100.000% Params, 25.3 MMac, 100.000% MACs,
  (rnn): LSTM(24.06 k, 100.000% Params, 25.3 MMac, 100.000% MACs, 28, 64)
)
Computational complexity:  25.3 MMac
Number of parameters:  24.06 k
```

```
# Increasing the number of hidden states to 128
lstm128 = LSTM(num_inputs=len(data.vocab), num_hiddens=128)
LSTMmodel128 = d2l.RNNLM(lstm128, vocab_size=len(data.vocab), lr=4)
trainer.fit(LSTMmodel128, data)
# Training time: 7min 32sec
```




```
LSTMmodel128.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has in and in and the t'
```

```
LSTMmodel128.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are alwayser and the the grome'
```

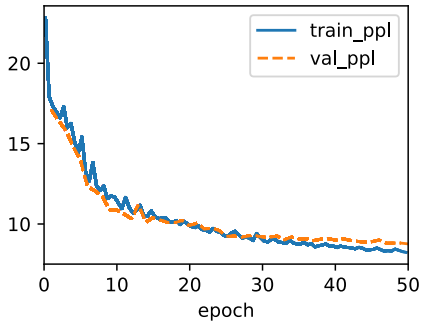
```
LSTMmodel128.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across of the thing the ti'
```

```
macs, params = get_model_complexity_info(lstm128, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module LSTM is treated as a zero-op.
LSTM(
  80.9 k, 100.000% Params, 84.15 MMac, 100.000% MACs,
  (rnn): LSTM(80.9 k, 100.000% Params, 84.15 MMac, 100.000% MACs, 28, 128)
)
Computational complexity:  84.15 MMac
Number of parameters:  80.9 k
```

```
# Decreasing the number of hidden states to 8
lstm8 = LSTM(num_inputs=len(data.vocab), num_hiddens=8)
LSTMmodel8 = d2l.RNNLM(lstm8, vocab_size=len(data.vocab), lr=4)
trainer.fit(LSTMmodel8, data)
# Training time: 2min
```




```
LSTMmodel8.predict('it has', 20, data.vocab, d2l.try_gpu())
```

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

```
LSTMmodel8.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always and and and and and'
```

```
LSTMmodel8.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across and and and and and'
```

```
macs, params = get_model_complexity_info(lstm8, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

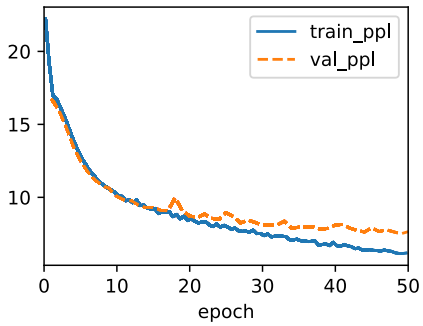
```
Warning: module LSTM is treated as a zero-op.
LSTM(
  1.22 k, 100.000% Params, 1.33 MMac, 100.000% MACs,
  (rnn): LSTM(1.22 k, 100.000% Params, 1.33 MMac, 100.000% MACs, 28, 8)
)
Computational complexity:  1.33 MMac
Number of parameters:  1.22 k
```

RNN Comparison

```
class RNN(d2l.Module):
    """The RNN model implemented with high-level APIs."""
    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)


rnn64 = RNN(num_inputs=len(data.vocab), num_hiddens=64)
RNNmodel64 = d2l.RNNLM(rnn64, vocab_size=len(data.vocab), lr=1)
trainer.fit(RNNmodel64, data)
# Training time: 2min 11sec
```




```
RNNmodel64.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has and hin the proun t'
```

```
RNNmodel64.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always all has in the time'
```

```
RNNmodel64.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across his said the time t'
```

```
macs, params = get_model_complexity_info(rnn64, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

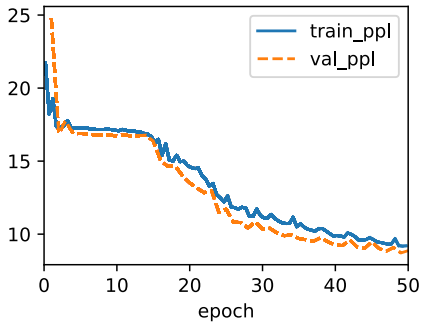
```
Warning: module RNN is treated as a zero-op.
RNN(
  6.02 k, 100.000% Params, 6.23 MMac, 100.000% MACs,
  (rnn): RNN(6.02 k, 100.000% Params, 6.23 MMac, 100.000% MACs, 28, 64)
)
Computational complexity:  6.23 MMac
Number of parameters:  6.02 k
```

2a) Deep GRU

```
class deepGRU(d2l.RNN):
    """The multi-layer GRU model."""
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout=0.5):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.GRU(num_inputs, num_hiddens, num_layers,
                          dropout=dropout)


# GRU w/ 3 layers
deepGru3 = deepGRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=3)
deepGRUModel3 = d2l.RNNLM(deepGru3, vocab_size=len(data.vocab), lr=2)
```

```
trainer.fit(deepGRUModel3, data)
# Train time: 3min 14sec
```




```
macs, params = get_model_complexity_info(deepGru3, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
    Warning: variables __flops__ or __params__ are already defined for the moduleGRU ptflops can affect your code!
    Warning: module deepGRU is treated as a zero-op.
    deepGRU(
      18.62 k, 100.000% Params, 19.76 MMac, 100.000% MACs,
      (rnn): GRU(18.62 k, 100.000% Params, 19.76 MMac, 100.000% MACs, 28, 32, num_layers=3, dropout=0.5)
    )
    Computational complexity:  19.76 MMac
    Number of parameters:  18.62 k
```

```
deepGRUModel3.predict('it has', 20, data.vocab, d2l.try_gpu())
```

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

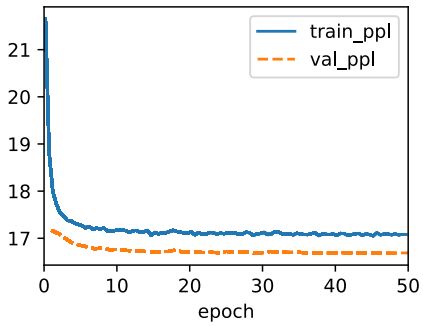
```
deepGRUModel3.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
    'we are always the the the the the'
```

```
deepGRUModel3.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
    'he looked across the the the the the'
```

```
# GRU w/ 5 layers
deepGru5 = deepGRU(num_inputs=len(data.vocab), num_hiddens=32, num_layers=5)
deepGRUModel5 = d2l.RNNLM(deepGru5, vocab_size=len(data.vocab), lr=2)
trainer.fit(deepGRUModel5, data)
# Train time: 5min 30sec
```




```
macs, params = get_model_complexity_info(deepGru5, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
    Warning: module deepGRU is treated as a zero-op.
```

```
deepGRU(
  31.3 k, 100.000% Params, 33.19 MMac, 100.000% MACs,
  (rnn): GRU(31.3 k, 100.000% Params, 33.19 MMac, 100.000% MACs, 28, 32, num_layers=5, dropout=0.5)
)
Computational complexity:  33.19 MMac
Number of parameters:  31.3 k
```

```
deepGRUModel5.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has                    '
```

```
deepGRUModel5.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
'we are always             '
```

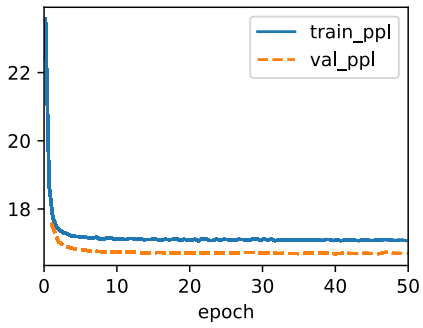
```
deepGRUModel5.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
'he looked across          '
```

2b) Deep LSTM

```
class deepLSTM(d2l.RNN):
    """The multi-layer LSTM model."""
    def __init__(self, num_inputs, num_hiddens, num_layers, dropout=0.5):
        d2l.Module.__init__(self)
        self.save_hyperparameters()
        self.rnn = nn.LSTM(num_inputs, num_hiddens, num_layers,
                           dropout=dropout)
```

```
# LSTM w/ 3 layers
deepLSTM3 = deepLSTM(num_inputs=len(data.vocab), num_hiddens=32, num_layers=3)
deepLSTMModel3 = d2l.RNNLM(deepLSTM3, vocab_size=len(data.vocab), lr=2)
trainer.fit(deepLSTMModel3, data)
# Train time: 4min 33sec
```




```
macs, params = get_model_complexity_info(deepLSTM3, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
Warning: module deepLSTM is treated as a zero-op.
deepLSTM(
  24.83 k, 100.000% Params, 26.41 MMac, 100.000% MACs,
  (rnn): LSTM(24.83 k, 100.000% Params, 26.41 MMac, 100.000% MACs, 28, 32, num_layers=3, dropout=0.5)
)
Computational complexity:  26.41 MMac
Number of parameters:  24.83 k
```

```
deepLSTMModel3.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
'it has                    '
```

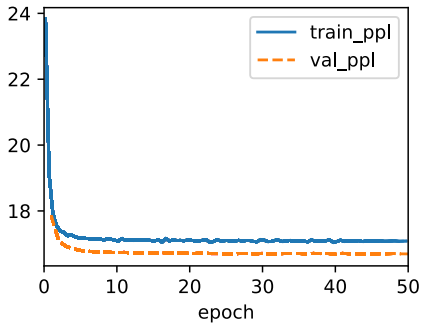
```
deepLSTMModel3.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
    'we are always                    '
```

```
deepLSTMModel3.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
    'he looked across                 '
```

```
# LSTM w/ 5 layers
deepLSTM5 = deepLSTM(num_inputs=len(data.vocab), num_hiddens=32, num_layers=5)
deepLSTMModel5 = d2l.RNNLM(deepLSTM5, vocab_size=len(data.vocab), lr=2)
trainer.fit(deepLSTMModel5, data)
# Train time: 6min 37sec
```




```
macs, params = get_model_complexity_info(deepLSTM5, (1024, 28), as_strings=True,
                                         print_per_layer_stat=True, verbose=True)
print('Computational complexity: ', macs)
print('Number of parameters: ', params)
```

```
    Warning: module deepLSTM is treated as a zero-op.
    deepLSTM(
      41.73 k, 100.000% Params, 44.37 MMac, 100.000% MACs,
      (rnn): LSTM(41.73 k, 100.000% Params, 44.37 MMac, 100.000% MACs, 28, 32, num_layers=5, dropout=0.5)
    )
    Computational complexity:  44.37 MMac
    Number of parameters:  41.73 k
```

```
deepLSTMModel5.predict('it has', 20, data.vocab, d2l.try_gpu())
```

```
    'it has                   '
```

```
deepLSTMModel5.predict('we are always', 20, data.vocab, d2l.try_gpu())
```

```
    'we are always                    '
```

```
deepLSTMModel5.predict('he looked across', 20, data.vocab, d2l.try_gpu())
```

```
    'he looked across                 '
```

✓ 0s completed at 6:54 PM