Exercise Sheet 1: Recurrent Models

Compare Vanilla Recurrent Neural Networks (RNN) with Long-Short Term Networks (LSTM). Implement a vanilla RNN and LSTM from scratch.

```python
import json
import os
import time
import math
import sys
import numpy as np
from numpy import sqrt
from datetime import datetime

import torch
import torch.nn as nn
import torch.optim as optim
import torch.utils.data as data

import matplotlib.pyplot as plt
%matplotlib inline

plt.rcParams['figure.figsize'] = [8, 5]

# set seed for reproducibility
seed = 42
np.random.seed(seed)
torch.manual_seed(seed)

# set device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# set paths
data_path = './data/'
model_path = './model/'
results_path = './results/'

# make directories if they don't exist
if not os.path.exists(data_path):
    os.makedirs(data_path)
if not os.path.exists(model_path):
    os.makedirs(model_path)
if not os.path.exists(results_path):
    os.makedirs(results_path)


# Data
p1 = [5, 10, 15, 20]
p2_acc_rnn = []
```

```
p3_acc_lstm = []

# Hyperparameters
config = {
    "input_length": 12,
    "input_dim": 1,
    "num_classes": 10,
    "num_hidden": 128,
    "batch_size": 128,
    "rnn_learning_rate": 0.001,
    "lstm_learning_rate": 0.001,
    "train_steps": 10000,
    "test_steps": 100,
    "max_norm": 1.0
}

# save config
with open(model_path + 'config.json', 'w') as file:
    json.dump(config, file)
```

Task 1: Toy Problem: Palindrome Numbers

Use a a recurrent neural network to predict the next digit of the palindrome at every timestep. This can become difficult for very long sequences since the network has to memorise information from very far away earlier timesteps. Goal is to study the memoization capability of recurrent networks.

```
class PalindromeDataset(data.Dataset):
    """ Randomly generates palindromes of a given length.
        The input is the first N-1 digits of the palindrome, the
target is the last digit.
        For short palindromes, the number of possible palindromes is
limited.
    """
    def __init__(self, seq_length):
        self.seq_length = seq_length

    def __len__(self):
        # Number of possible palindroms can be very big:
        # (10**(seq_length/2) or (10**((seq_length+1)/2)
        # Therefore we return the maximum integer value
        return sys.maxsize

    def __getitem__(self, idx):
        # Keep last digit as target label. Note: one-hot encoding for
inputs is
        # more suitable for training, but this also works.
        full_palindrome = self.generate_palindrome()
        # Split palindrome into inputs (N-1 digits) and target (1
digit)
        return full_palindrome[0:-1], int(full_palindrome[-1])

    def generate_palindrome(self):
        # Generates a single, random palindrome number of 'length'
digits.
        left = [np.random.randint(0, 10) for _ in
range(math.ceil(self.seq_length / 2))]
        left = np.asarray(left, dtype=np.float32)
        right = np.flip(left, 0) if self.seq_length % 2 == 0 else
np.flip(left[:-1], 0)
        return np.concatenate((left, right))
```

Question 1.1: Backpropagation through Time

Backpropagation by computing derivatives from L_t with respect to W_{ph}

\begin{aligned} \frac{\partial L_t}{\partial W_{ph}} = \frac{\partial L_t}{\partial h_t} \frac{\partial h_t}{\partial W_{ph}} \end{aligned}

Backpropagation by computing derivatives from L_t with respect to W_{hh} \begin{aligned} \frac{\partial L_{t}}{\partial W_{hh}} = \frac{\partial L_{t}}{\partial h_{t}} \frac{\partial h_{t}}{\partial h_{t-1} } \dots \frac{\partial h_{1}}{\partial W_{hh}} \ = \frac{\partial L_{t}}{\partial h_{t}}(\prod_{t=2}^{T} \frac{\partial h_{t}}{\partial h_{t-1}})\frac{\partial h_{1}}{\partial W_{hh}} \ = \frac{\partial L_{t}}{\partial h_{t}}(\prod_{t=2}^{T} softmax^{'}(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh}^{T-1})\frac{\partial h_{1}}{\partial W_{hh}} \ \end{aligned}

What difference do you observe in temporal dependence of the two gradients. Study the latter gradient and explain what problems might occur when training this recurrent network for a large number of timesteps.

The difference in temporal dependence of both gradients is that the first gradient depends only on the current hidden state h_t, while the second gradient depends on all previous hidden states h_{t-1}, h_{t-2}, ..., h_1. In the second gradient, the repeated multiplication of the softmax derivative and the weight matrix W_{hh} can cause the gradient to vanish or explode as the timestep t gets larger (i.e., longer sequences).

This is because the softmax derivative is always bound to very small values between zero and 0.25. As a result, the repeated multiplication with the weight matrix, which can also have a determinante less than one, during chaining with backpropagation can cause the gradient to decrease exponentially and become very small, leading to the vanishing gradient problem. On the other hand, if the values are very large, the repeated multiplication can cause the gradient to explode and become very large, leading to the exploding gradient problem. This can be clearer when looking at:

\begin{aligned} \prod_{t=1}^{T} softmax'(W_{hh}h_{t-1} + W_{xh}x_t)W_{hh} \end{aligned}

Both of these problems can make it difficult to train the network effectively for a large number of timesteps.

Question 1.2: Implement a Vanilla Recurrent Network

```python
class VanillaRNN(nn.Module):

    def __init__(
        self, seq_length, input_dim, num_hidden, num_classes,
batch_size, device=None):
        super(VanillaRNN, self).__init__()
        self.seq_length = seq_length
        self.input_dim = input_dim
        self.num_hidden = num_hidden
        self.num_classes = num_classes
        self.batch_size = batch_size

        if device is None:
            self.device = torch.device('cuda' if
torch.cuda.is_available() else 'cpu')

        self.device = device

        # Define the RNN layer
        self.hidden_state = torch.zeros(self.batch_size,
self.num_hidden)
        self.W_hx = nn.Parameter(torch.Tensor(self.input_dim,
self.num_hidden))      # input to hidden
        self.W_hh = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_hidden))     # hidden to hidden
        self.B_h = nn.Parameter(torch.Tensor(self.num_hidden))
# hidden bias
        # Define the output layer
        self.W_ph = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_classes))    # hidden to output
        self.B_y = nn.Parameter(torch.Tensor(self.num_classes))
# output bias

        # Initialize weights
        self.init_weights()

    def forward(self, x):
        # Initialize hidden state
        h_t = torch.zeros(self.num_hidden)

        for t in range(self.seq_length): # iterate over the time steps
            x_t = x[:, t].view(128,-1)
            h_t = torch.tanh(x_t @ self.W_hx + h_t @ self.W_hh +
self.B_h)

        output = h_t @ self.W_ph + self.B_y
        y = torch.softmax(output, dim=1)
        return y

    def init_weights(self):
        """ Initialize weights to avoid gradients vanishing or
exploding.
            Source: https://dennybritz.com/posts/wildml/recurrent-
neural-networks-tutorial-part-2/

        """
        # Initialize weights with uniform distribution
        n_hx = self.W_hx.size(0)  # number of incoming connections for
W_hx
        nn.init.uniform_(self.W_hx, -1 / sqrt(n_hx), 1 / sqrt(n_hx))

        n_hh = self.W_hh.size(0)  # number of incoming connections for
W_hh
        nn.init.uniform_(self.W_hh, -1 / sqrt(n_hh), 1 / sqrt(n_hh))

        n_ph = self.W_ph.size(0)  # number of incoming connections for
W_ph
        nn.init.uniform_(self.W_ph, -1 / sqrt(n_ph), 1 / sqrt(n_ph))

        # Initialize biases to zeros
        nn.init.zeros_(self.B_h)
        nn.init.zeros_(self.B_y)


    def set_grad(self, requires_grad):
        # Set requires_grad for all parameters
        for param in self.parameters():
            param.requires_grad = requires_grad

def compute_accuracy(outputs, targets):
    """ Compute the accuracy of the model's predictions."""
    # Compute accuracy of outputs compared to targets
    _, predicted = torch.max(outputs, 1)
    correct = predicted.eq(targets)
    return 100 * correct.sum().item() / targets.size(0)


def train(config:json, input_length=5, lr=0.001, step_size=1000,
gamma=0.1, type='RNN', opt='RMSprop', device=None):
    """ Train the model on the training set.
        Returns the trained model, losses and accuracies.
    """
    if device is None:
        device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')

    if input_length == 0:
        input_length = config['input_length']

    # Initialize the model that we are going to use
    if type == 'RNN':
        model = VanillaRNN(input_length, config['input_dim'],
config['num_hidden'], config['num_classes'], config['batch_size'])
    elif type == 'LSTM':
        model = LSTM(input_length, config['input_dim'],
config['num_hidden'], config['num_classes'], config['batch_size'])
    else:
        raise ValueError('Model type not supported')

    model.to(device)
    model.train()

    # Initialize the dataset and data loader (note the +1)
    dataset = PalindromeDataset(input_length + 1)
    data_loader = data.DataLoader(dataset, config['batch_size'],
num_workers=0)

    # Define the loss function and optimizer
    criterion = nn.CrossEntropyLoss()

    # Define optimizer
    if opt == 'Adam':
        optimizer = optim.Adam(model.parameters(), lr=lr)
    elif opt == 'RMSprop':
        optimizer = optim.RMSprop(model.parameters(), lr=lr)
        scheduler = optim.lr_scheduler.StepLR(optimizer,
step_size=step_size, gamma=gamma)
    else:
        raise ValueError('Optimizer not supported')

    # Train the model
    losses = []
    accuracies = []
    loss = 0.0

    for i, (inputs, targets) in enumerate(data_loader, 0):

        # Only for time measurement of step through network
        t1 = time.time()
        inputs = inputs.to(device)
        targets = targets.to(device)

        # Zero the parameter gradients
        optimizer.zero_grad()

        # Update learning rate
        if opt == 'RMSprop':
            scheduler.step()

        # Forward pass, backward pass, and optimize
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        # Backward pass
        loss.backward()

        # Clip gradients to prevent exploding gradients + update 
weights
        nn.utils.clip_grad_norm(model.parameters(), 
max_norm=config['max_norm'])
        optimizer.step()

        loss += loss.item()
        accuracy = 0.0

        # Just for time measurement
        t2 = time.time()
        examples_per_second = config['batch_size'] / float(t2 - t1)

        # Print statistics
        if i % 100 == 0:
            # Just for time measurement
            t2 = time.time()
            # print accuracy/loss here
            accuracy = compute_accuracy(outputs, targets)
            accuracies.append(accuracy)
            print(
                "[{}] Train Step {:04d}/{:04d}, Batch Size = {}, 
Examples/Sec = {:.2f}, "
                "Accuracy = {:.2f}, Loss = {:.3f}".format(
                    datetime.now().strftime("%Y-%m-%d %H:%M"),
                    i,
                    config['train_steps'],
                    config['batch_size'],
                    examples_per_second,
                    accuracy,
                    loss / 100,
                )
            )
            losses.append(loss.detach().numpy() / 100)
            loss = 0.0

        if i == config['train_steps']:
            # If you receive a PyTorch data-loader error, check this 
bug report:
            # https://github.com/pytorch/pytorch/pull/9655
            break

    print('Finished Training')
    return model, losses, accuracies
```

```python
def test(model, config:json, input_length=5, device=None):
    """ Test the model on the test set.
        Returns the accuracies.
    """
    if device is None:
        device = torch.device('cuda' if torch.cuda.is_available() else
'cpu')

    if input_length == 0:
        input_length = config['input_length']

    # Initialize the dataset and data loader (leave the +1)
    dataset = PalindromeDataset(input_length+1)
    data_loader = data.DataLoader(dataset, config['batch_size'],
num_workers=0)

    model.to(device)
    model.eval()

    # Test the model
    accuracies = []

    with torch.no_grad():
        for i, (inputs, targets) in enumerate(data_loader, 0):
            inputs = inputs.to(device)
            targets = targets.to(device)

            outputs = model(inputs)
            accuracy = 0.0
            if i % 10 == 0:
                accuracy = compute_accuracy(outputs, targets)
                accuracies.append(accuracy)
                print('Accuracy: %.4f' % accuracy)

            if i == config['test_steps']:
              # If you receive a PyTorch data-loader error, check this
bug report:
              # https://github.com/pytorch/pytorch/pull/9655
              break

    print('Finished Testing')
    return accuracies
```

Question: Why do we use gradient clipping?

Gradient clipping is a technique used to prevent the exploding gradient problem during training. This problem occurs when the gradients become too large, causing numerical instability and resulting in poor model performance. In extreme cases, the weights can take on the value of "NaN" or "Inf," which halts training altogether.

The exploding gradient problem is particularly prevalent in Recurrent Neural Networks (or RNNs) due to the nature of backpropagation through time (BPTT). In BPTT, gradients are accumulated over many time steps, and without gradient clipping, these accumulated large gradients can easily explode and destabilize training.

By clipping the gradients to a maximum norm, it can be ensured that the updates to the weights during training stay within a reasonable range. This maintains numerical stability and prevents the exploding gradient problem, leading to more robust and stable training of the network.

```
# Load the configuration
with open(model_path + 'config.json', 'r') as file:
    config = json.load(file)

def plot_loss(losses, title='Training Loss', path=None):
    """ Plot the losses of the model."""
    if path is None:
        path = results_path + 'training_loss.png'
    plt.figure(figsize=(6,4))
    plt.plot(losses)
    plt.xlabel('Steps')
    plt.ylabel('Loss')
    plt.title(title)
    plt.savefig(path)
    plt.show()

def plot_accuracy(accuracies, title='Training Accuracy', path=None):
    """ Plot the accuracies of the model."""
    if path is None:
        path = results_path + 'training_accuracy.png'
    plt.figure(figsize=(6,4))
    plt.plot(accuracies)
    plt.xlabel('Steps')
    plt.ylabel('Accuracy')
    plt.title(title)
    plt.savefig(path)
    plt.show()
```

Question 1.3: Train RNN with varying T for sequence lengths

- Train and evaluate model on Palindromes with length N = 5

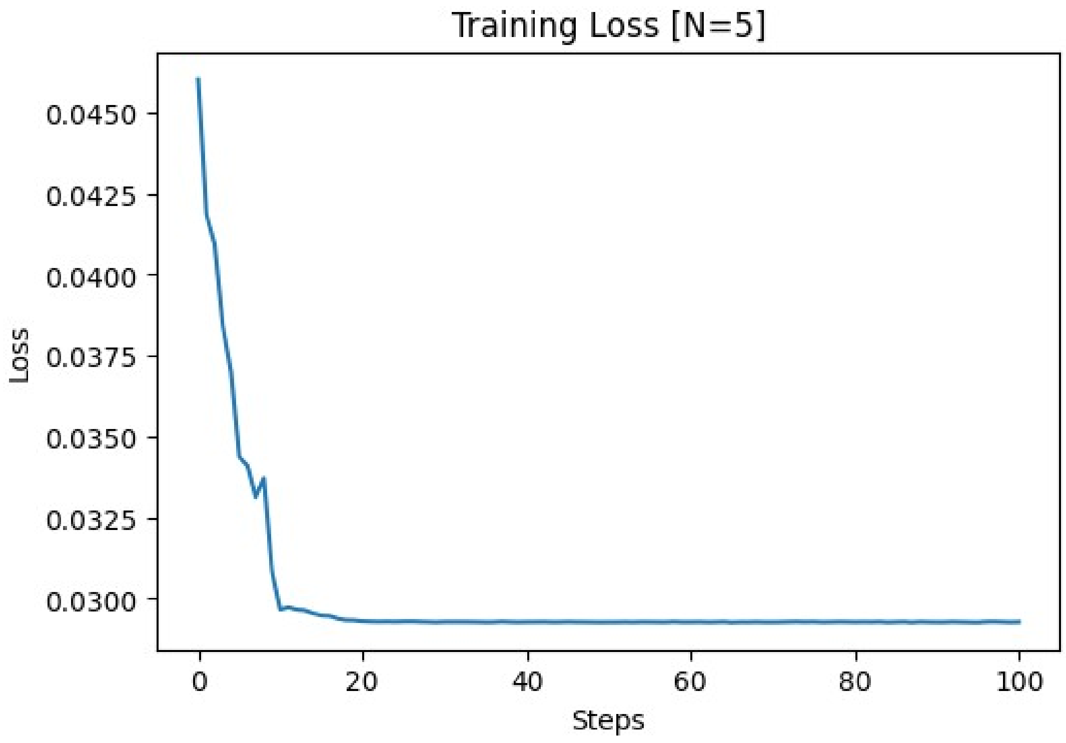
```
# Train the model on T=5
model, losses, accuracies = train(config, input_length=p1[0],
lr=config['rnn_learning_rate'], type='RNN', device=device)
```

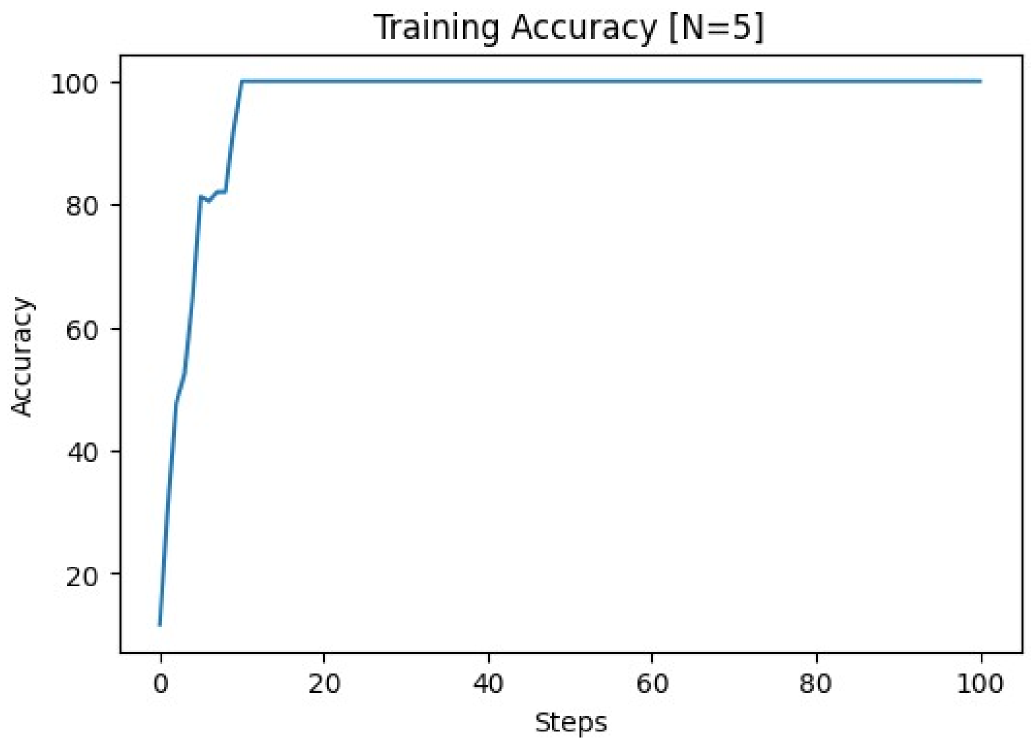
```
/usr/local/lib/python3.10/dist-packages/torch/optim/
lr_scheduler.py:143: UserWarning: Detected call of
`lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and
later, you should call them in the opposite order: `optimizer.step()`
before `lr_scheduler.step()`.  Failure to do this will result in
PyTorch skipping the first value of the learning rate schedule. See
```

```
more details at https://pytorch.org/docs/stable/optim.html#how-to-
adjust-learning-rate
  warnings.warn("Detected call of `lr_scheduler.step()` before
`optimizer.step()`. "
<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 07:55] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 702.83, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:55] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 7367.52, Accuracy = 31.25, Loss = 0.042
[2024-06-27 07:55] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 9658.56, Accuracy = 47.66, Loss = 0.041
[2024-06-27 07:55] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 31573.21, Accuracy = 52.34, Loss = 0.038
[2024-06-27 07:55] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 24329.13, Accuracy = 64.84, Loss = 0.037
[2024-06-27 07:55] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 29203.16, Accuracy = 81.25, Loss = 0.034
[2024-06-27 07:55] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 32744.02, Accuracy = 80.47, Loss = 0.034
[2024-06-27 07:55] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 27695.17, Accuracy = 82.03, Loss = 0.033
[2024-06-27 07:55] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 30849.33, Accuracy = 82.03, Loss = 0.034
[2024-06-27 07:55] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 30038.10, Accuracy = 92.19, Loss = 0.031
[2024-06-27 07:55] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 30785.65, Accuracy = 100.00, Loss = 0.030
[2024-06-27 07:55] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 32034.78, Accuracy = 100.00, Loss = 0.030
[2024-06-27 07:55] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 30213.91, Accuracy = 100.00, Loss = 0.030
[2024-06-27 07:55] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 23401.23, Accuracy = 100.00, Loss = 0.030
[2024-06-27 07:55] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 20717.41, Accuracy = 100.00, Loss = 0.030
[2024-06-27 07:56] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 32071.14, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 31748.72, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 33040.24, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 27753.87, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 30159.59, Accuracy = 100.00, Loss = 0.029
```

```
[2024-06-27 07:56] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 30840.47, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 31090.51, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 31799.50, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 31871.23, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 33154.51, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 30402.11, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 32250.31, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 33752.73, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 21901.48, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 23252.24, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 20123.35, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 31619.70, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 33368.82, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 33003.68, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 32587.01, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 25237.20, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 27268.94, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 33277.81, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 31375.78, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 32742.02, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 29222.24, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 31647.66, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 32384.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 14503.75, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 20848.55, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 19799.78, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 30514.43, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 33083.00, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 27825.80, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 31549.09, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 33594.33, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 32541.58, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 28907.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 34269.81, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 26547.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 31049.15, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 31714.96, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 31562.08, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 24902.40, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 22939.28, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 19516.19, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 29801.33, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 32015.68, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 32822.09, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 31113.93, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 34744.43, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 29947.62, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 32353.32, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 29187.28, Accuracy = 100.00, Loss = 0.029
```

```
[2024-06-27 07:56] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 25011.46, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 31856.10, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 30063.33, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 31863.67, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 16393.00, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 19558.85, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 30488.44, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 21684.74, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 28734.26, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 32384.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 31954.70, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 30745.10, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 25532.45, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 19709.64, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 31475.11, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 26509.53, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:56] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 32824.10, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 32531.72, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 31266.14, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 20070.69, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 20663.19, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 31630.88, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 25288.31, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 31015.07, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 30632.83, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 28087.84, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 31642.06, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 31229.77, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 33109.52, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 30268.42, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 32136.41, Accuracy = 100.00, Loss = 0.029
[2024-06-27 07:57] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 30414.17, Accuracy = 100.00, Loss = 0.029
Finished Training

# Plot the losses
plot_loss(losses, title='Training Loss [N=5]', path=results_path +
'training_loss_5_rnn.png')

# Plot accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=5]',
path=results_path + 'training_accuracy_5_rnn.png')
```

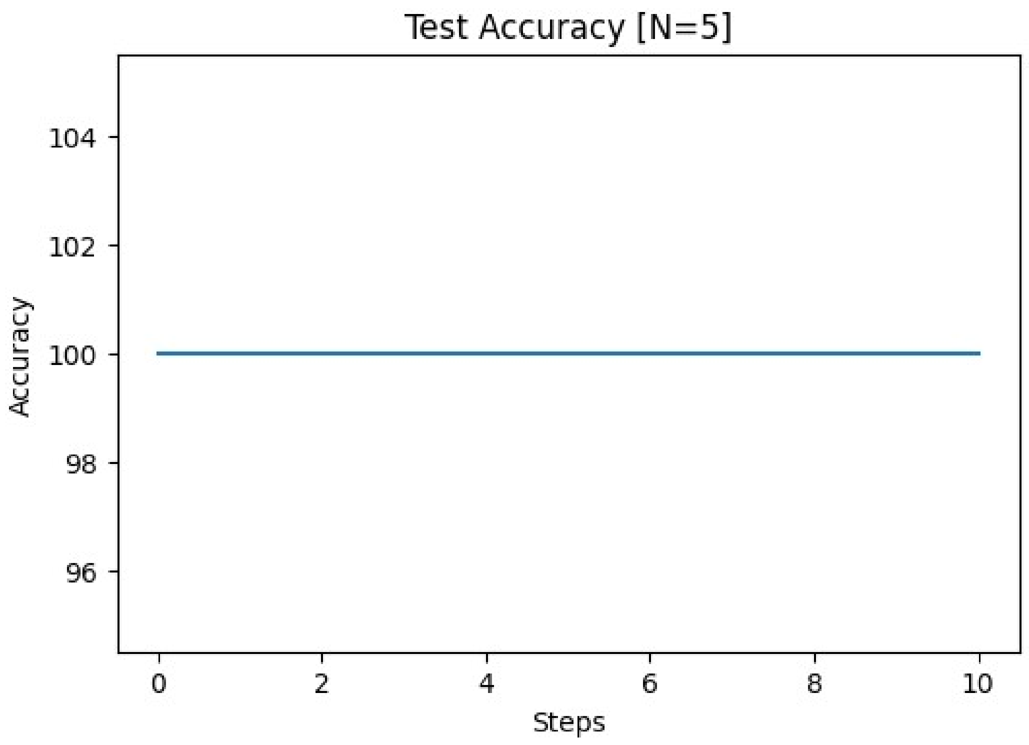




```
# Test the model
test_accuracies = test(model, input_length=p1[0], config=config,
device=device)

# Add accuracies
p2_acc_rnn.append(np.mean(test_accuracies))
```

```
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=5]',
path=results_path + 'test_accuracy_5_rnn.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



```
Average test accuracy: 100.00%
```

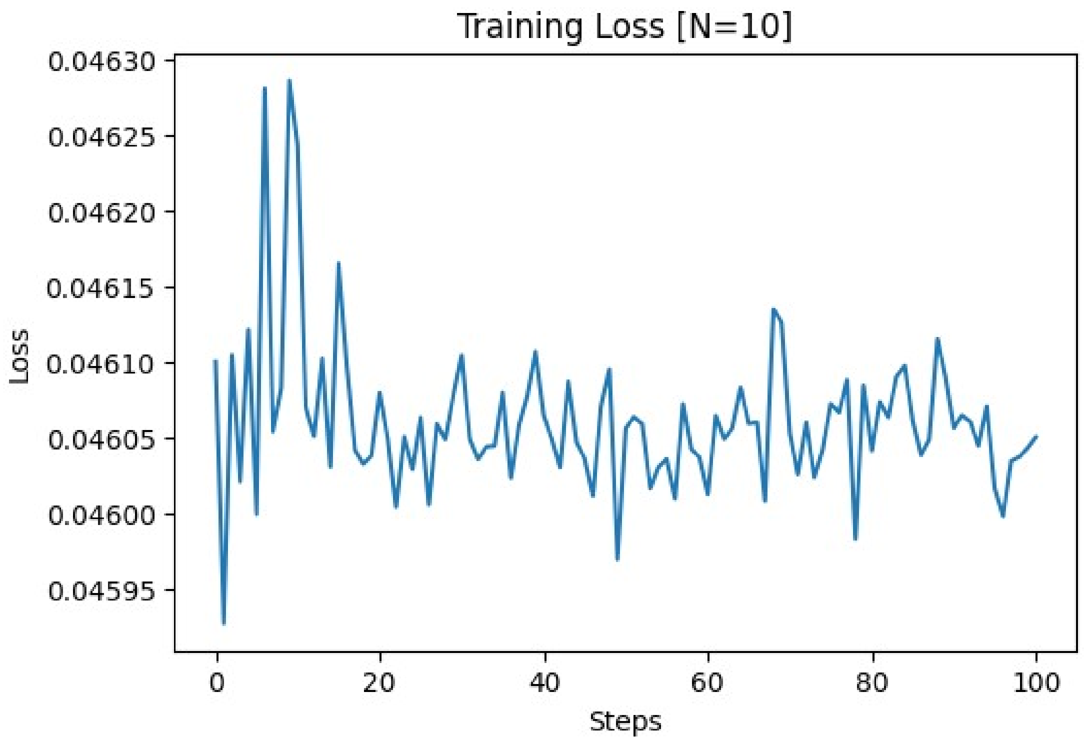
Task 2: Vanilla RNN in PyTorch

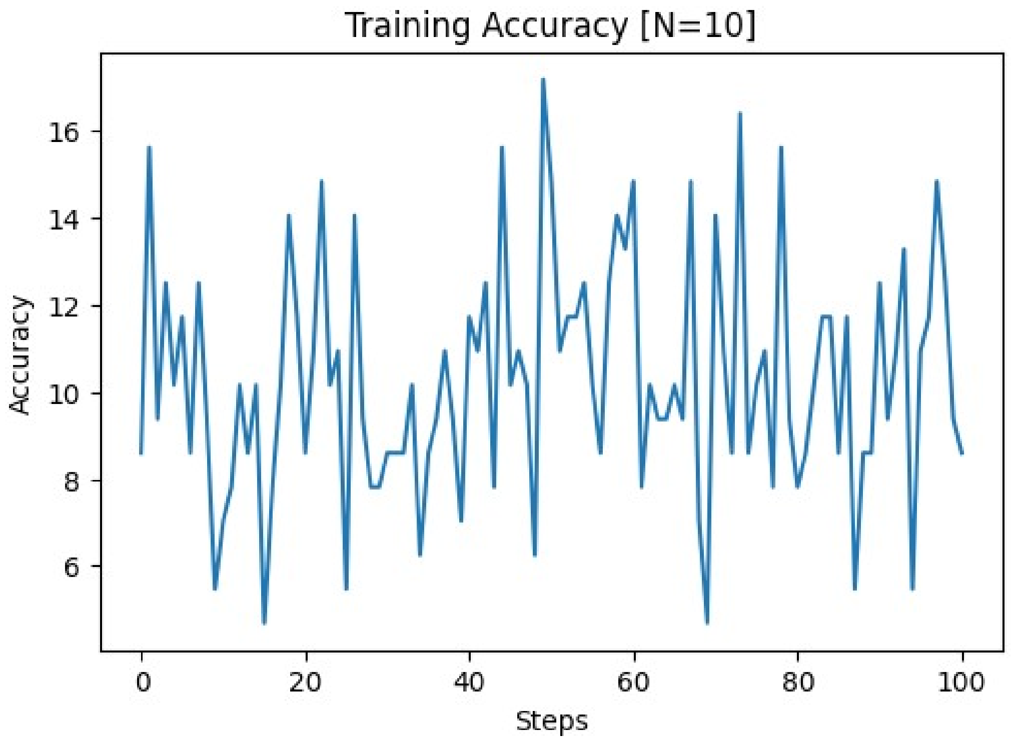
```
# Train the model on T=10
model, losses, accuracies = train(config, input_length=p1[1],
lr=config['rnn_learning_rate'], type='RNN', device=device)

[2024-06-27 07:57] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 15176.13, Accuracy = 8.59, Loss = 0.046

<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 07:57] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 12583.11, Accuracy = 15.62, Loss = 0.046
[2024-06-27 07:57] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 12137.89, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:57] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 18090.47, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:57] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 19475.13, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:57] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 19355.07, Accuracy = 11.72, Loss = 0.046
```

```
[2024-06-27 07:57] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 19094.85, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 19780.08, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:57] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 18382.84, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:57] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 19109.81, Accuracy = 5.47, Loss = 0.046
[2024-06-27 07:57] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 13225.05, Accuracy = 7.03, Loss = 0.046
[2024-06-27 07:57] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 11971.70, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:57] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 16501.33, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:57] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 20118.83, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 20855.03, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:57] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 19884.84, Accuracy = 4.69, Loss = 0.046
[2024-06-27 07:57] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 19721.95, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:57] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 18862.73, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:57] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 19248.20, Accuracy = 14.06, Loss = 0.046
[2024-06-27 07:57] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 20255.46, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:57] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 13445.30, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 17236.68, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:57] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 16235.85, Accuracy = 14.84, Loss = 0.046
[2024-06-27 07:57] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 20369.97, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:57] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 4574.45, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:57] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 18967.35, Accuracy = 5.47, Loss = 0.046
[2024-06-27 07:57] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 19477.25, Accuracy = 14.06, Loss = 0.046
[2024-06-27 07:57] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 19639.70, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:57] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 18391.03, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:57] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 10645.44, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:57] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 19160.28, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 19352.98, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 10693.36, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:57] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 19967.68, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 20292.97, Accuracy = 6.25, Loss = 0.046
[2024-06-27 07:58] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 16333.66, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:58] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 19597.41, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:58] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 18989.49, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:58] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 13079.74, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:58] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 16627.57, Accuracy = 7.03, Loss = 0.046
[2024-06-27 07:58] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 19391.42, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:58] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 16310.83, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:58] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 19497.76, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:58] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 17508.18, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:58] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 13754.63, Accuracy = 15.62, Loss = 0.046
[2024-06-27 07:58] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 18589.07, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 13276.72, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:58] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 16580.84, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 20247.82, Accuracy = 6.25, Loss = 0.046
[2024-06-27 07:58] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 17481.39, Accuracy = 17.19, Loss = 0.046
[2024-06-27 07:58] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 18868.69, Accuracy = 14.84, Loss = 0.046
[2024-06-27 07:58] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 16399.01, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:58] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 15302.44, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:58] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 19468.77, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:58] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 16614.70, Accuracy = 12.50, Loss = 0.046
```

```
[2024-06-27 07:58] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 11466.95, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 19198.65, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:58] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 16452.28, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:58] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 16652.84, Accuracy = 14.06, Loss = 0.046
[2024-06-27 07:58] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 19440.57, Accuracy = 13.28, Loss = 0.046
[2024-06-27 07:58] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 15820.10, Accuracy = 14.84, Loss = 0.046
[2024-06-27 07:58] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 19062.99, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:58] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 19291.09, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 18981.44, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:58] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 12274.98, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:58] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 11586.23, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 16313.80, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:58] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 16390.50, Accuracy = 14.84, Loss = 0.046
[2024-06-27 07:58] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 12173.39, Accuracy = 7.03, Loss = 0.046
[2024-06-27 07:58] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 18335.13, Accuracy = 4.69, Loss = 0.046
[2024-06-27 07:58] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 20562.68, Accuracy = 14.06, Loss = 0.046
[2024-06-27 07:58] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 19831.22, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:58] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 19366.24, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:58] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 13209.76, Accuracy = 16.41, Loss = 0.046
[2024-06-27 07:58] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 10679.32, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:58] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 19268.24, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:58] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 19966.19, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:59] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 19112.53, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:59] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 19528.97, Accuracy = 15.62, Loss = 0.046
[2024-06-27 07:59] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 18791.42, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:59] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 16051.39, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:59] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 18203.95, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 13634.82, Accuracy = 10.16, Loss = 0.046
[2024-06-27 07:59] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 10393.20, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 19385.12, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 20435.88, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 15689.26, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 19222.02, Accuracy = 5.47, Loss = 0.046
[2024-06-27 07:59] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 20521.80, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 19293.86, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 19080.60, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:59] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 11632.67, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:59] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 10509.98, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:59] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 19870.13, Accuracy = 13.28, Loss = 0.046
[2024-06-27 07:59] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 18917.23, Accuracy = 5.47, Loss = 0.046
[2024-06-27 07:59] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 15525.47, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:59] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 19595.97, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 14350.23, Accuracy = 14.84, Loss = 0.046
[2024-06-27 07:59] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 10785.09, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:59] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 19744.43, Accuracy = 9.38, Loss = 0.046
[2024-06-27 07:59] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 20152.81, Accuracy = 8.59, Loss = 0.046
Finished Training

# Plot the losses
plot_loss(losses, title='Training Loss [N=10]', path=results_path +
'training_loss_10_rnn.png')

# Plot the accuracies
```

```
plot_accuracy(accuracies, title='Training Accuracy [N=10]',
path=results_path + 'training_accuracy_10_rnn.png')
```

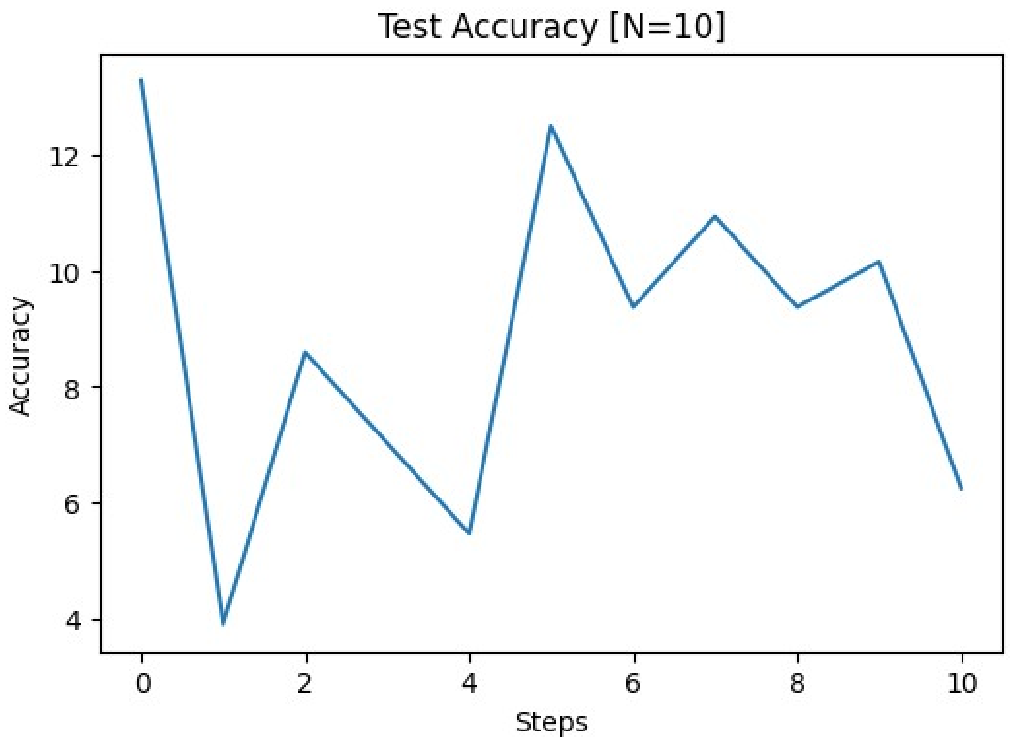




```python
# Test the model
test_accuracies = test(model, input_length=p1[1], config=config,
device=device)
```

```
Accuracy: 13.2812
Accuracy: 3.9062
Accuracy: 8.5938
Accuracy: 7.0312
Accuracy: 5.4688
Accuracy: 12.5000
Accuracy: 9.3750
Accuracy: 10.9375
Accuracy: 9.3750
Accuracy: 10.1562
Accuracy: 6.2500
Finished Testing
```

```python
# Add accuracies
p2_acc_rnn.append(np.mean(test_accuracies))

# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=10]',
path=results_path + 'test_accuracy_10_rnn.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



```
Average test accuracy: 8.81%
```

```python
# Train the model on T=15
model, losses, accuracies = train(config, input_length=p1[2],
lr=config['rnn_learning_rate'], type='RNN', device=device)

# Plot the losses
plot_loss(losses, title='Training Loss [N=15]', path=results_path +
'training_loss_15_rnn.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=15]',
path=results_path + 'training_accuracy_15_rnn.png')
```

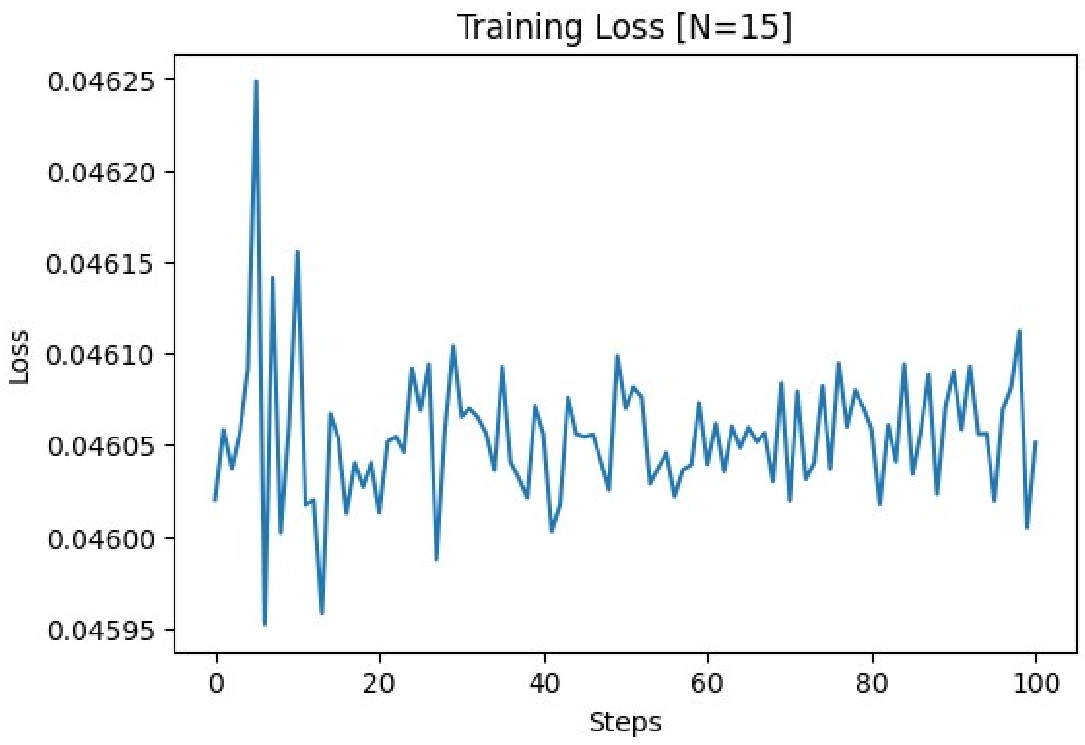
```
[2024-06-27 07:59] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 5759.37, Accuracy = 12.50, Loss = 0.046
```

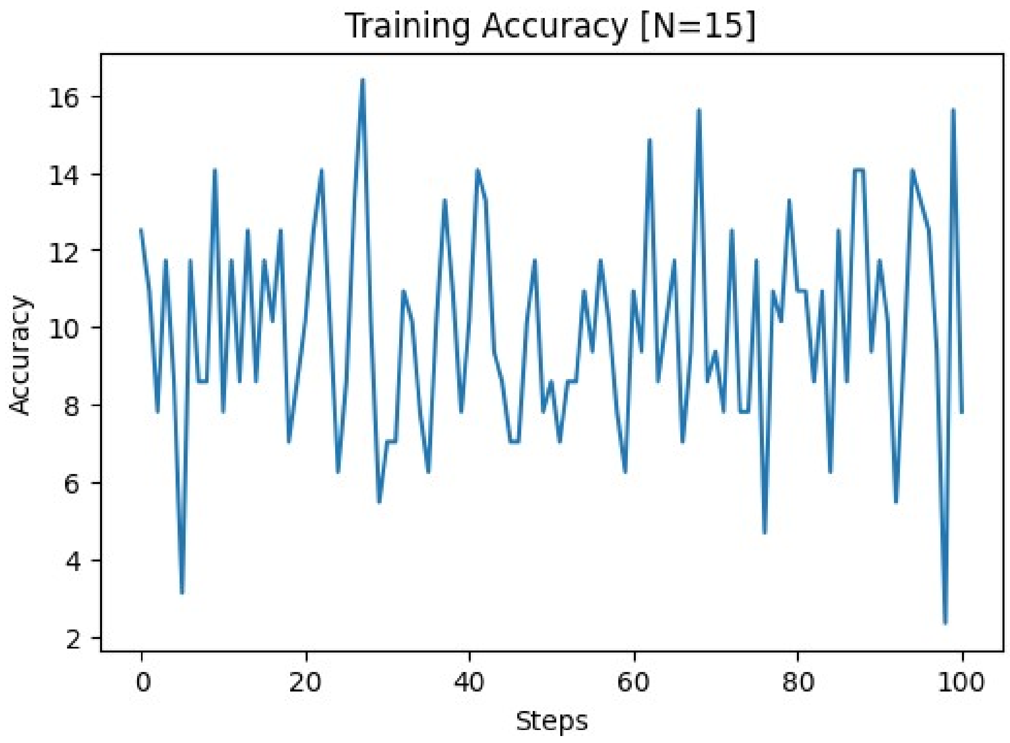
```
<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])
```

```
[2024-06-27 07:59] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 14320.76, Accuracy = 10.94, Loss = 0.046
[2024-06-27 07:59] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 14171.07, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:59] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 14823.73, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 13677.54, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 12818.35, Accuracy = 3.12, Loss = 0.046
[2024-06-27 07:59] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 11575.48, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 7423.44, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 7309.64, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 14857.78, Accuracy = 14.06, Loss = 0.046
[2024-06-27 07:59] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 12486.24, Accuracy = 7.81, Loss = 0.046
[2024-06-27 07:59] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 14097.02, Accuracy = 11.72, Loss = 0.046
[2024-06-27 07:59] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 14747.99, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 14106.65, Accuracy = 12.50, Loss = 0.046
[2024-06-27 07:59] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 13860.46, Accuracy = 8.59, Loss = 0.046
[2024-06-27 07:59] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 8919.60, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:00] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 12045.84, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 12045.30, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:00] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 13738.80, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:00] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 12780.21, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:00] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 12546.06, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 14399.11, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:00] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 9474.81, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:00] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 13491.59, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 13886.63, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:00] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 13360.65, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:00] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 13407.03, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:00] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 13890.94, Accuracy = 16.41, Loss = 0.046
[2024-06-27 08:00] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 13409.37, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 14162.47, Accuracy = 5.47, Loss = 0.046
[2024-06-27 08:00] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 7501.13, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:00] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 13963.56, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:00] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 13881.24, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:00] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 14167.33, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 13824.41, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:00] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 13934.56, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:00] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 12262.93, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 9415.48, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:00] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 13978.47, Accuracy = 10.94, Loss = 0.046
```

```
[2024-06-27 08:00] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 13813.74, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:00] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 14035.84, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 13063.51, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:00] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 11800.92, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:00] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 8631.78, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:00] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 14166.58, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:00] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 8420.45, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:00] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 13787.49, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:00] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 13882.32, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:00] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 13764.51, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:00] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 14075.58, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:00] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 13831.89, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 13986.48, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:01] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 8845.97, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 13793.51, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 14521.80, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 9980.68, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:01] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 11880.04, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:01] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 13294.81, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:01] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 13854.38, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:01] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 8107.87, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:01] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 8740.98, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 9339.97, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:01] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 12778.38, Accuracy = 14.84, Loss = 0.046
[2024-06-27 08:01] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 14220.24, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 13296.78, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:01] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 12754.70, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:01] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 14547.38, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:01] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 9129.99, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:01] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 13203.91, Accuracy = 15.62, Loss = 0.046
[2024-06-27 08:01] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 14141.21, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 14085.92, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:01] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 13982.83, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:01] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 12582.52, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:01] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 14735.84, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:01] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 8579.36, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:01] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 14178.55, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:01] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 13859.38, Accuracy = 4.69, Loss = 0.046
[2024-06-27 08:01] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 11332.85, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 14142.32, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:01] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 13448.00, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:01] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 13739.15, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 13785.72, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 9339.65, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:01] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 14495.53, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:01] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 13167.96, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:01] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 13694.99, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:02] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 14351.77, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:02] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 13940.72, Accuracy = 14.06, Loss = 0.046
```

```
[2024-06-27 08:02] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 14001.80, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:02] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 8500.85, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:02] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 9285.69, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:02] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 8267.57, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:02] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 14019.35, Accuracy = 5.47, Loss = 0.046
[2024-06-27 08:02] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 13015.05, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:02] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 13310.63, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:02] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 13696.04, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:02] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 9219.99, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:02] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 13844.37, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:02] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 14083.34, Accuracy = 2.34, Loss = 0.046
[2024-06-27 08:02] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 10177.26, Accuracy = 15.62, Loss = 0.046
[2024-06-27 08:02] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 14093.32, Accuracy = 7.81, Loss = 0.046
Finished Training
```



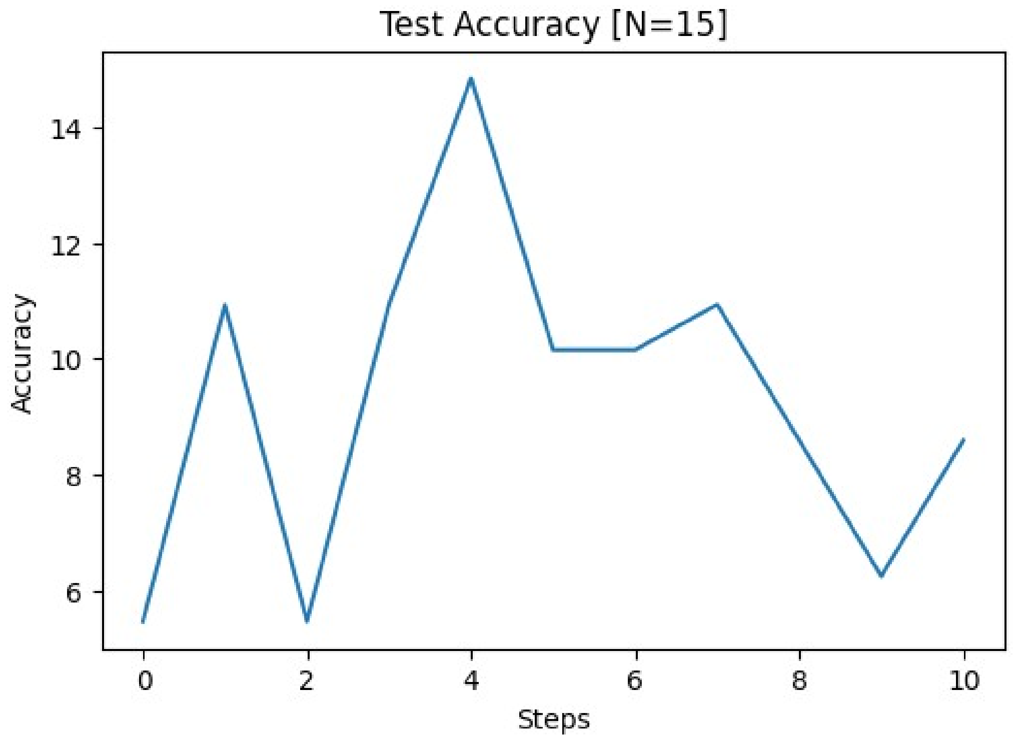


```
# Test the model
test_accuracies = test(model, input_length=p1[2], config=config,
device=device)
```

```
# Add accuracies
p2_acc_rnn.append(np.mean(test_accuracies))
```

```
Accuracy: 5.4688
Accuracy: 10.9375
Accuracy: 5.4688
Accuracy: 10.9375
Accuracy: 14.8438
Accuracy: 10.1562
Accuracy: 10.1562
Accuracy: 10.9375
Accuracy: 8.5938
Accuracy: 6.2500
Accuracy: 8.5938
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=15]',
path=results_path + 'test_accuracy_15_rnn.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



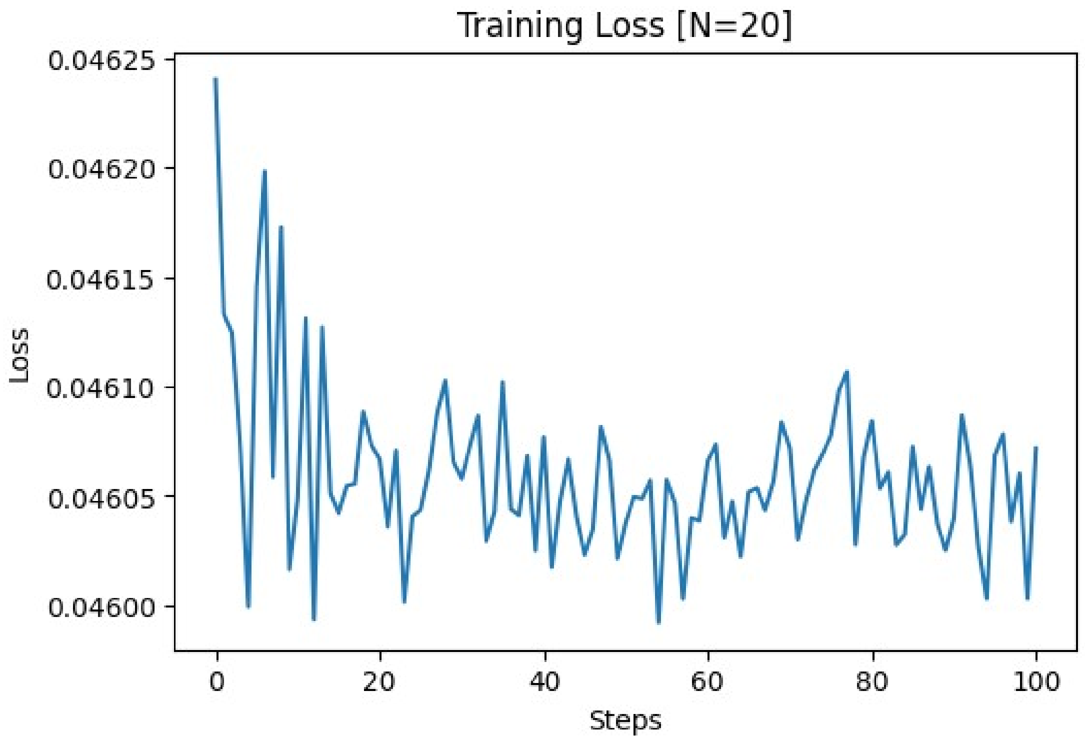
```
Average test accuracy: 9.30%
```

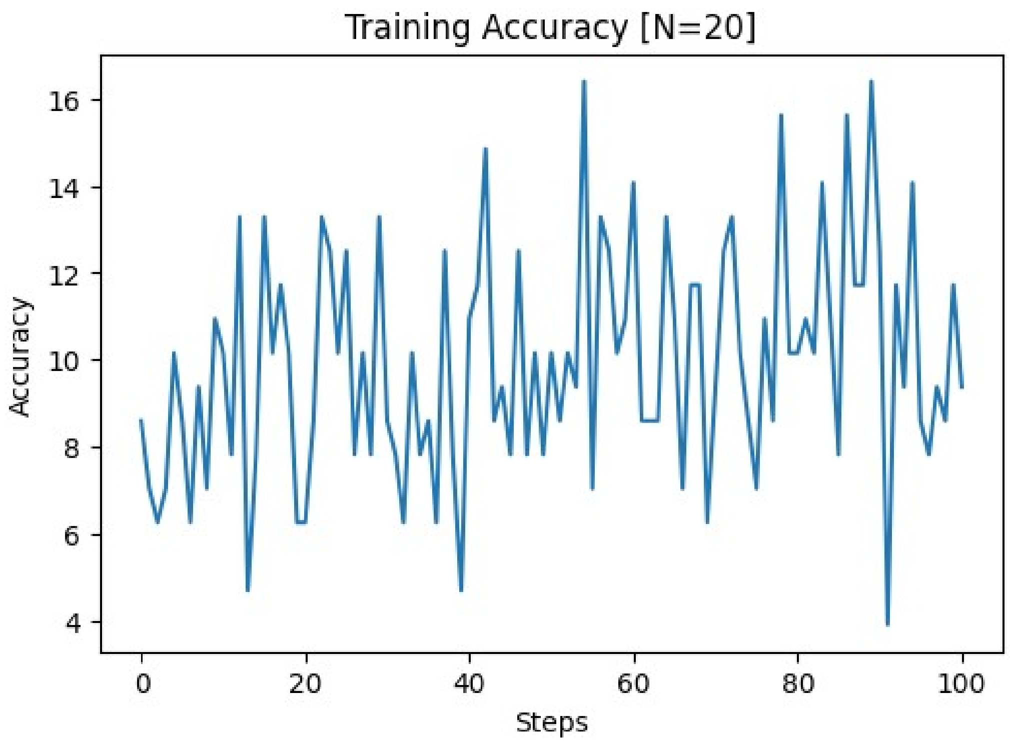
```
# Train the model on T=20
model, losses, accuracies = train(config, input_length=p1[3],
lr=config['rnn_learning_rate'], type='RNN', device=device)

[2024-06-27 08:02] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 7704.14, Accuracy = 8.59, Loss = 0.046

<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 08:02] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 11238.90, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:02] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 6670.61, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:02] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 10870.25, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:02] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 10892.09, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:02] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 8867.74, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:02] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 9449.12, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:02] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 3893.70, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:02] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 10436.23, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:02] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 10560.22, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:02] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 10975.36, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:02] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 10897.39, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:02] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 8427.32, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:02] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 6514.63, Accuracy = 4.69, Loss = 0.046
[2024-06-27 08:03] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 8643.04, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 10341.74, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:03] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 7966.03, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:03] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 11054.69, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:03] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 11393.21, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:03] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 7138.77, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:03] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 10902.04, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:03] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 11022.23, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:03] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 10896.29, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:03] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 10725.19, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:03] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 10957.44, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:03] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 6142.27, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:03] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 11277.62, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 11033.79, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:03] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 8864.52, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 10845.66, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:03] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 10754.41, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:03] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 7380.99, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 10342.54, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:03] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 10965.05, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:03] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 10489.24, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 10912.45, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:03] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 9290.19, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:03] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 6599.68, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:03] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 10418.41, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:03] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 7925.23, Accuracy = 4.69, Loss = 0.046
[2024-06-27 08:03] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 10639.32, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:03] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 10855.75, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:04] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 7886.35, Accuracy = 14.84, Loss = 0.046
[2024-06-27 08:04] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 6861.76, Accuracy = 8.59, Loss = 0.046
```

```
[2024-06-27 08:04] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 11124.32, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:04] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 11076.13, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:04] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 10904.47, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:04] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 8893.16, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:04] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 10394.60, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:04] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 6903.23, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:04] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 11271.93, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:04] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 10914.89, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:04] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 11145.80, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:04] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 9379.95, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:04] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 10834.93, Accuracy = 16.41, Loss = 0.046
[2024-06-27 08:04] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 7779.27, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:04] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 11052.18, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:04] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 10840.18, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:04] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 9162.87, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:04] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 11148.11, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:04] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 9483.84, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:04] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 10953.42, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:04] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 10585.20, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:04] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 10879.06, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:04] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 11053.55, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:04] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 10822.70, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:04] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 6148.39, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:04] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 10695.70, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:04] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 10840.84, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:04] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 11246.67, Accuracy = 6.25, Loss = 0.046
[2024-06-27 08:05] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 10858.60, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:05] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 11349.14, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:05] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 7278.03, Accuracy = 13.28, Loss = 0.046
[2024-06-27 08:05] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 10110.18, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:05] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 10335.97, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:05] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 10715.13, Accuracy = 7.03, Loss = 0.046
[2024-06-27 08:05] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 10769.08, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:05] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 11226.68, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:05] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 7217.46, Accuracy = 15.62, Loss = 0.046
[2024-06-27 08:05] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 10400.24, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:05] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 10764.11, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:05] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 10247.19, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:05] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 10952.97, Accuracy = 10.16, Loss = 0.046
[2024-06-27 08:05] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 10830.56, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:05] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 6888.35, Accuracy = 10.94, Loss = 0.046
[2024-06-27 08:05] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 9805.50, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:05] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 11544.12, Accuracy = 15.62, Loss = 0.046
[2024-06-27 08:05] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 11024.04, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:05] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 10740.43, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:05] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 10149.75, Accuracy = 16.41, Loss = 0.046
[2024-06-27 08:05] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 5701.20, Accuracy = 12.50, Loss = 0.046
[2024-06-27 08:05] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 11014.54, Accuracy = 3.91, Loss = 0.046
[2024-06-27 08:05] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 11251.15, Accuracy = 11.72, Loss = 0.046
```

```
[2024-06-27 08:05] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 10911.79, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:05] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 10964.38, Accuracy = 14.06, Loss = 0.046
[2024-06-27 08:05] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 10975.81, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:05] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 6642.63, Accuracy = 7.81, Loss = 0.046
[2024-06-27 08:05] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 10485.56, Accuracy = 9.38, Loss = 0.046
[2024-06-27 08:06] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 10655.79, Accuracy = 8.59, Loss = 0.046
[2024-06-27 08:06] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 7437.64, Accuracy = 11.72, Loss = 0.046
[2024-06-27 08:06] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 10882.59, Accuracy = 9.38, Loss = 0.046
Finished Training
```

```
# Plot the losses
plot_loss(losses, title='Training Loss [N=20]', path=results_path +
'training_loss_20_rnn.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=20]',
path=results_path + 'training_accuracy_20_rnn.png')
```

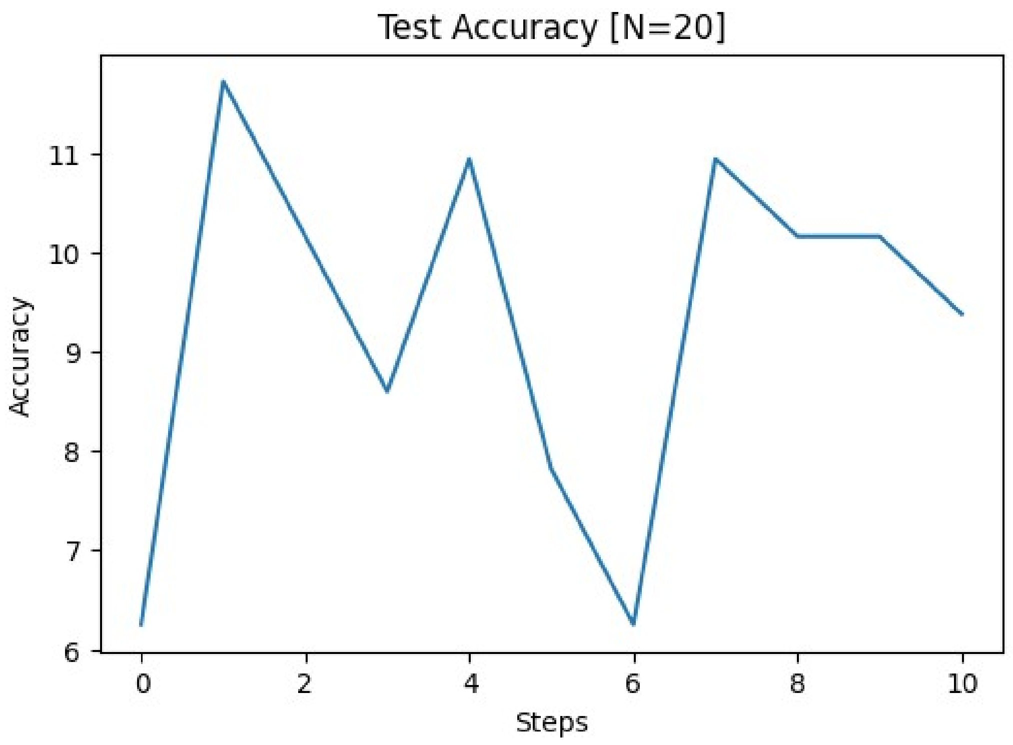




```
# Test the model
test_accuracies = test(model, input_length=p1[3], config=config,
device=device)

# Add accuracies
p2_acc_rnn.append(np.mean(test_accuracies))
```

```
Accuracy: 6.2500
Accuracy: 11.7188
Accuracy: 10.1562
Accuracy: 8.5938
Accuracy: 10.9375
Accuracy: 7.8125
Accuracy: 6.2500
Accuracy: 10.9375
Accuracy: 10.1562
Accuracy: 10.1562
Accuracy: 9.3750
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=20]',
path=results_path + 'test_accuracy_20_rnn.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



```
Average test accuracy: 9.30%
```

Question 1.4: Optimization Methods (momentum, adaptive learning rate)

Vanilla Gradient Descent computes gradients of loss function with respect to the model weights and updates them which can make the loss function slowly oscillate towards vertical axes. This happens since no history about the previously computed gradients are kept track off, making the gradient steps less deterministic at each step, which can lead to a slower convergence. Due to this osicllation a large learning rate could lead to disconvergence.

Hence, to achieve faster and more stable convergence, it would be desirable for the loss function to take larger steps in the horizontal direction while taking smaller steps in the vertical direction. Momentum is a technique used to accomplish this. It incorporates an exponentially weighted moving average of the gradient values (momentum of the previous step), which helps the optimizer maintain a consistent direction in the parameter space. By doing so, the momentum term increases for dimensions whose gradients point in the same directions, while reducing updates for dimensions whose gradients frequently change directions. Consequently, this results in faster convergence and reduced oscillation.

Adam Optimizer

Adam is an *adaptive learningrate* optimization algorithm that combines the ideas from RMSprop and Stochastic Gradient Descent with momentum. It utilizes the squared gradients to scale the learning rate, similar to RMSprop, and employs a moving average of the gradients instead of the gradients themselves, like momentum. This combination of techniques allows Adam to adapt the learning rate for each parameter individually, leading to more efficient optimization and faster convergence in many cases.

\begin{aligned} m_t & = \beta_1 m_{t-1} + (1 - \beta_1) g_t \ v_t & = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \ \hat{m}_t & = \frac{m_t}{1 - \beta_1^t} \ \hat{v}t & = |frac{v_t}{1 - |beta_2^t} | |theta_t & = |theta{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{aligned}

- m_t and v_t represent the first moment (mean) and second moment (uncentered variance) estimates, respectively.
- $ß_1$ and $ß_2$ are the exponential decay rates for the moment estimates.
- g_t denotes the gradient at time step t.
- $\widehat{m}_t$ and $\widehat{v}_t$ are the bias-corrected first moment and second moment estimates, respectively. The bias correction is applied to counteract the effect of initialization on the moment estimates.
- The term $1 - \beta_1^t$ and $1 - \beta_2^t$ in the denominators of $\widehat{m}_t$ and $\widehat{v}_t$ are used to adjust for the decay of the past gradients as t increases.
- θ_t is the parameter value at time step t. It is updated using the previous parameter value $\theta_t - 1$, the learning rate α, and the ratio of the bias-corrected first moment to the square root of the bias-corrected second moment.
- The term $\sqrt{\widehat{v}_t} + \epsilon$ in the denominator of the update rule for θ_t is used to prevent division by zero and to improve numerical stability. The small positive constant ϵ is typically set to a value such as 10^{-8}.

Task 3: Long-Short Term Network (LSTM) in PyTorch

```
class LSTM(nn.Module):

    def __init__(self, seq_length, input_dim, num_hidden, num_classes,
batch_size=128, device=None):
        super(LSTM, self).__init__()

        self.seq_length = seq_length
        self.input_dim = input_dim
        self.num_hidden = num_hidden
        self.num_classes = num_classes
        self.batch_size = batch_size

        if device is None:
          device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')

        self.device = device

        # Hidden Layer
        self.W_gx = nn.Parameter(torch.Tensor(self.input_dim,
self.num_hidden))
        self.W_gh = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_hidden))
        self.B_g = nn.Parameter(torch.Tensor(self.num_hidden))

        # Cell State
        # (1) Input gate
        self.W_ix = nn.Parameter(torch.Tensor(self.input_dim,
self.num_hidden))
        self.W_ih = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_hidden))
        self.B_i = nn.Parameter(torch.Tensor(self.num_hidden))

        # (2) Forget gate
        self.W_fx = nn.Parameter(torch.Tensor(self.input_dim,
self.num_hidden))
        self.W_fh = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_hidden))
        self.B_f = nn.Parameter(torch.Tensor(self.num_hidden))

        # (3) Output gate
        self.W_ox = nn.Parameter(torch.Tensor(self.input_dim,
self.num_hidden))
        self.W_oh = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_hidden))
        self.B_o = nn.Parameter(torch.Tensor(self.num_hidden))

        # Output Layer
        self.W_ph = nn.Parameter(torch.Tensor(self.num_hidden,
self.num_classes))
        self.B_y = nn.Parameter(torch.Tensor(self.num_classes))

        # Initialize weights
        self.init_weights()


    def forward(self, x):
        # Initialize hidden state and cell state
        h_t = torch.zeros(self.batch_size, self.num_hidden,
device=self.device)
        c_t = torch.zeros(self.batch_size, self.num_hidden,
device=self.device)

        for t in range(self.seq_length):
            x_t = x[:, t].view(self.batch_size, -1)

            # Compute the hidden state
            i_t = torch.sigmoid(x_t @ self.W_ix + h_t @ self.W_ih +
self.B_i)
            f_t = torch.sigmoid(x_t @ self.W_fx + h_t @ self.W_fh +
self.B_f)
            o_t = torch.sigmoid(x_t @ self.W_ox + h_t @ self.W_oh +
self.B_o)
            g_t = torch.tanh(x_t @ self.W_gx + h_t @ self.W_gh +
self.B_g)
```

```
            c_t = f_t * c_t + i_t * g_t
            h_t = o_t * torch.tanh(c_t)

        # Compute the output
        output = h_t @ self.W_ph + self.B_y
        y = torch.softmax(output, dim=1)
        return y

    def init_weights(self):
        """ Initialize weights to avoid gradients vanishing or
exploding.
            Source: https://dennybritz.com/posts/wildml/recurrent-
neural-networks-tutorial-part-2/
        """
        n_gx = self.W_gx.size(0)
        nn.init.uniform_(self.W_gx, -1 / sqrt(n_gx), 1 / sqrt(n_gx))
        n_gh = self.W_gh.size(0)

        nn.init.uniform_(self.W_gh, -1 / sqrt(n_gh), 1 / sqrt(n_gh))
        n_ix = self.W_ix.size(0)
        nn.init.uniform_(self.W_ix, -1 / sqrt(n_ix), 1 / sqrt(n_ix))

        n_ih = self.W_ih.size(0)
        nn.init.uniform_(self.W_ih, -1 / sqrt(n_ih), 1 / sqrt(n_ih))
        n_fx = self.W_fx.size(0)
        nn.init.uniform_(self.W_fx, -1 / sqrt(n_fx), 1 / sqrt(n_fx))

        n_fh = self.W_fh.size(0)
        nn.init.uniform_(self.W_fh, -1 / sqrt(n_fh), 1 / sqrt(n_fh))
        n_ox = self.W_ox.size(0)
        nn.init.uniform_(self.W_ox, -1 / sqrt(n_ox), 1 / sqrt(n_ox))

        n_oh = self.W_oh.size(0)
        nn.init.uniform_(self.W_oh, -1 / sqrt(n_oh), 1 / sqrt(n_oh))
        n_ph = self.W_ph.size(0)
        nn.init.uniform_(self.W_ph, -1 / sqrt(n_ph), 1 / sqrt(n_ph))

        nn.init.zeros_(self.B_g)
        nn.init.zeros_(self.B_i)
        nn.init.zeros_(self.B_f)
        nn.init.zeros_(self.B_o)
        nn.init.zeros_(self.B_y)


    def init_hidden(self):
        # Initialize hidden state
        self.hidden_state = torch.zeros(self.batch_size,
self.self.num_hidden, device=self.device)

    def set_grad(self, requires_grad):
        # Set requires_grad for all parameters
        for param in self.parameters():
            param.requires_grad = requires_grad
```

Question 1.5(a): LSTM Gates

Idea is that humans don't understand a sequence of information from scratch every time, but that there is a contextual understanding based on previous context (time-sequence datat). Hence Recurrent networks have a *recursive step* or recurrence in them which persists prior information along the way. In theory RNNs should be able to learn very long-term dependcies, but in practice as describe above (Vanishing Gradient Problem) they are not.

Long Short Term Memory Networks and Long-Term Dependencies

However, such long-term dependencies might on one hand connect previous infromation to the present taks, but equally for certain tasks only the persent information might be needed. LSTM networks avoid the Vanishing Gradient problem by keeping a cell state c_t which allows the gradient to flow backwards through the network without the need to go through any learned NN weight layers. Information can flow along like on a conveyabelt. Thereby information can be added, updated or removed.

LSTM Gates - The gated memory cell

(1) Input Gate: Decides which values will be updated in the cell state.

(2) Forget Gate: Decides which information can be discarded from the cell state.

(3) Output Gate: Decides which information will be added to the cell state.

Question 1.5(b): LSTM Trainable Parameters

(1) For each gate, there are $d \, x \, n$ parameters in the input weight matrix; $n \, x \, n$ parameters in the recurrent weight matrix and n parameters in the bias term:

\begin{aligned} d \times n + n \times n + n \end{aligned}

(2) Since there are 3 gates (input, forget, and output) and one cell state, the total number of trainable parameters is:

\begin{aligned} 3 \times (d \times n + n \times n + n) + (d \times n + n \times n + n) = 4 \times (d \times n + n \times n + n). \end{aligned}

The formula for the total number of trainable parameters in the LSTM cell is:

\begin{aligned} 4 \times (d \times n + n \times n + n) \end{aligned}

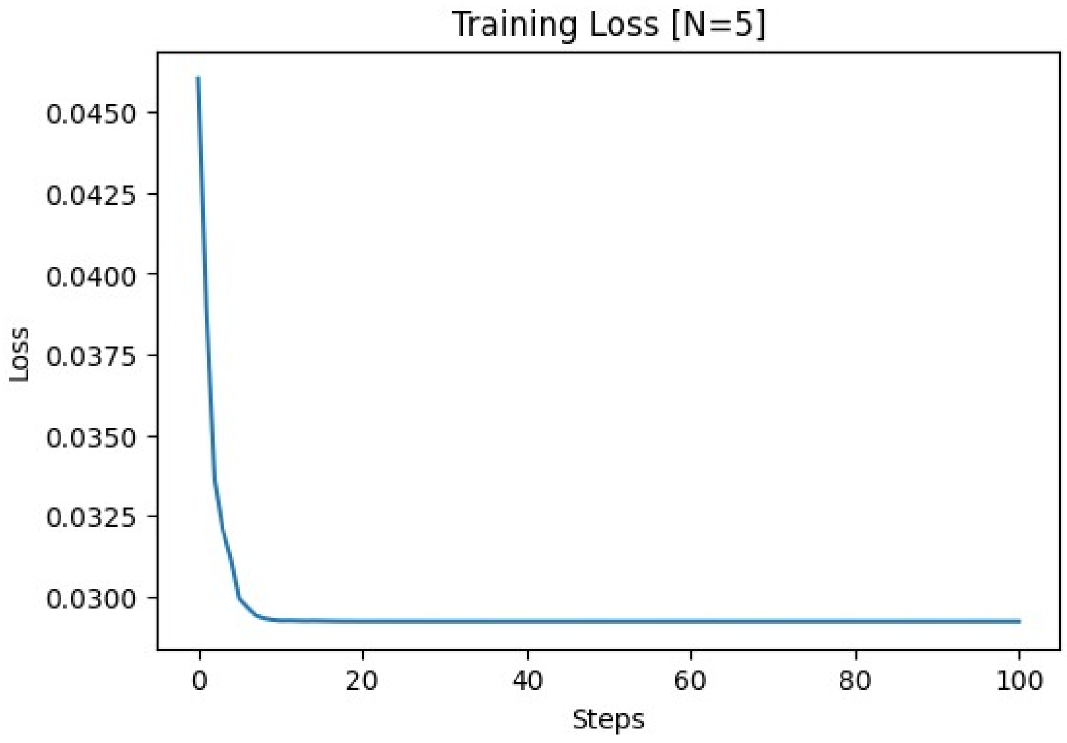
Question 1.6: Implement a LSTM network

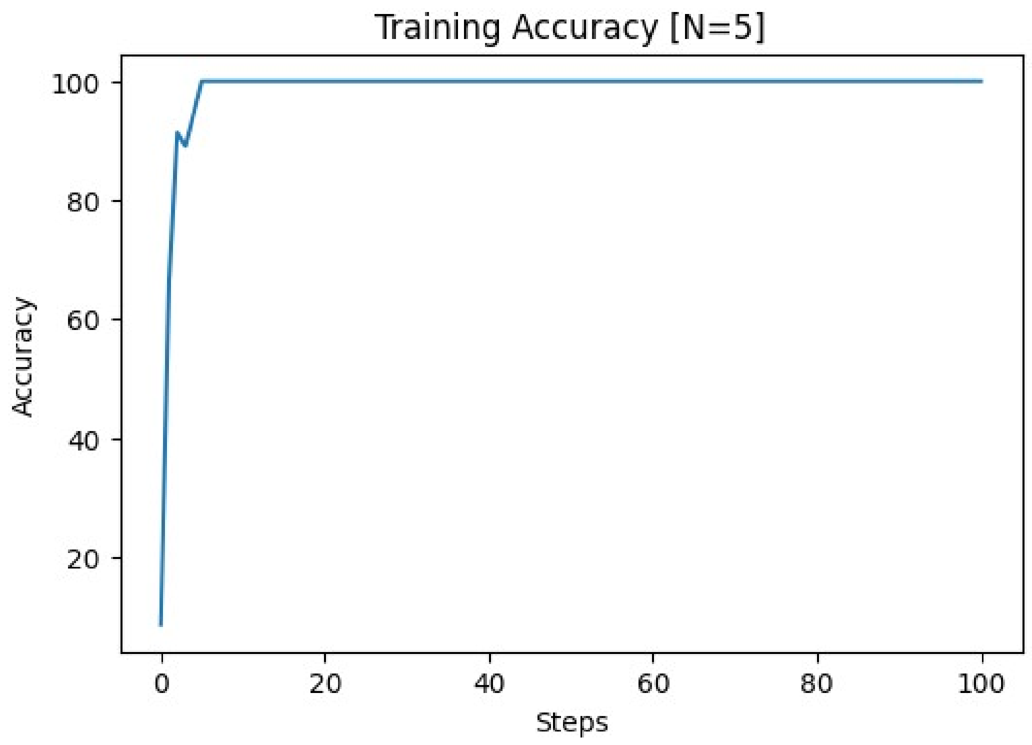
```
# Train the model T=5
model, losses, accuracies = train(config, input_length=p1[0],
lr=config['lstm_learning_rate'], type='LSTM', device=device)
```

```
[2024-06-27 08:06] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 4771.63, Accuracy = 8.59, Loss = 0.046

<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 08:06] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 9570.57, Accuracy = 66.41, Loss = 0.039
[2024-06-27 08:06] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 10036.66, Accuracy = 91.41, Loss = 0.034
[2024-06-27 08:06] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 9308.39, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:06] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 9998.15, Accuracy = 94.53, Loss = 0.031
[2024-06-27 08:06] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 10097.82, Accuracy = 100.00, Loss = 0.030
[2024-06-27 08:06] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 5484.49, Accuracy = 100.00, Loss = 0.030
[2024-06-27 08:06] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 10107.33, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 9608.26, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 10160.50, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 9552.35, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 8248.64, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 9505.84, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 6918.26, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 8165.71, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 10304.43, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 9206.71, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 10398.43, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 10283.31, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 8301.31, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 5366.29, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 8392.15, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 9911.95, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 8473.34, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 9494.58, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 7853.47, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:06] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 6259.35, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 10545.08, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 8810.84, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 9828.66, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 10124.67, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 9754.02, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 8191.63, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 6899.06, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 10017.93, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 9751.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 9801.21, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 9257.83, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 9988.85, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 9105.68, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 6672.93, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 9179.79, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 10206.09, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 9946.29, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 9975.68, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 10000.76, Accuracy = 100.00, Loss = 0.029
```

```
[2024-06-27 08:07] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 9813.03, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 6221.85, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 9990.71, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 10171.67, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 9062.18, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 10079.62, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 9092.57, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 8421.77, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 6282.87, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 9890.04, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 9973.08, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 8484.99, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 9548.79, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:07] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 9866.23, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 6186.72, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 9661.16, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 9401.47, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 10426.50, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 10012.14, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 10046.80, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 10046.61, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 6894.27, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 9991.08, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 9857.90, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 6438.46, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 9715.54, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 9542.68, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 10304.43, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 2419.18, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 9133.41, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 9411.85, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 10074.14, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 9044.17, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 2221.72, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 3077.38, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 9593.49, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 10239.18, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 5504.67, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 9826.14, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:08] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 10039.47, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 8607.29, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 10244.46, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 8559.39, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 8437.65, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 6047.27, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 10190.59, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 8819.96, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 8711.62, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 9442.64, Accuracy = 100.00, Loss = 0.029
```

```
[2024-06-27 08:09] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 8457.99, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 1902.90, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 5804.51, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 10230.01, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 5107.46, Accuracy = 100.00, Loss = 0.029
[2024-06-27 08:09] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 10351.91, Accuracy = 100.00, Loss = 0.029
Finished Training

# Plot the losses
plot_loss(losses, title='Training Loss [N=5]', path=results_path +
'training_loss_5_lstm.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=5]',
path=results_path + 'training_accuracy_5_lstm.png')
```

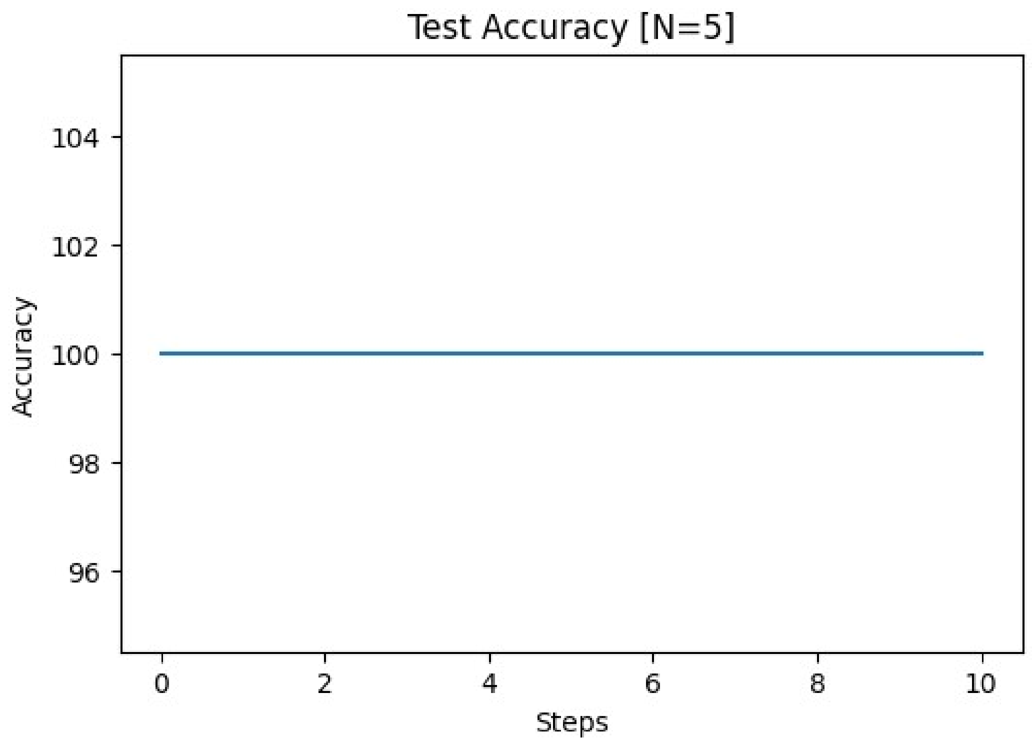




```
# Test the model
test_accuracies = test(model, input_length=p1[0], config=config,
device=device)

# Add accuracies
p3_acc_lstm.append(np.mean(test_accuracies))
```

```
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Accuracy: 100.0000
Finished Testing
```

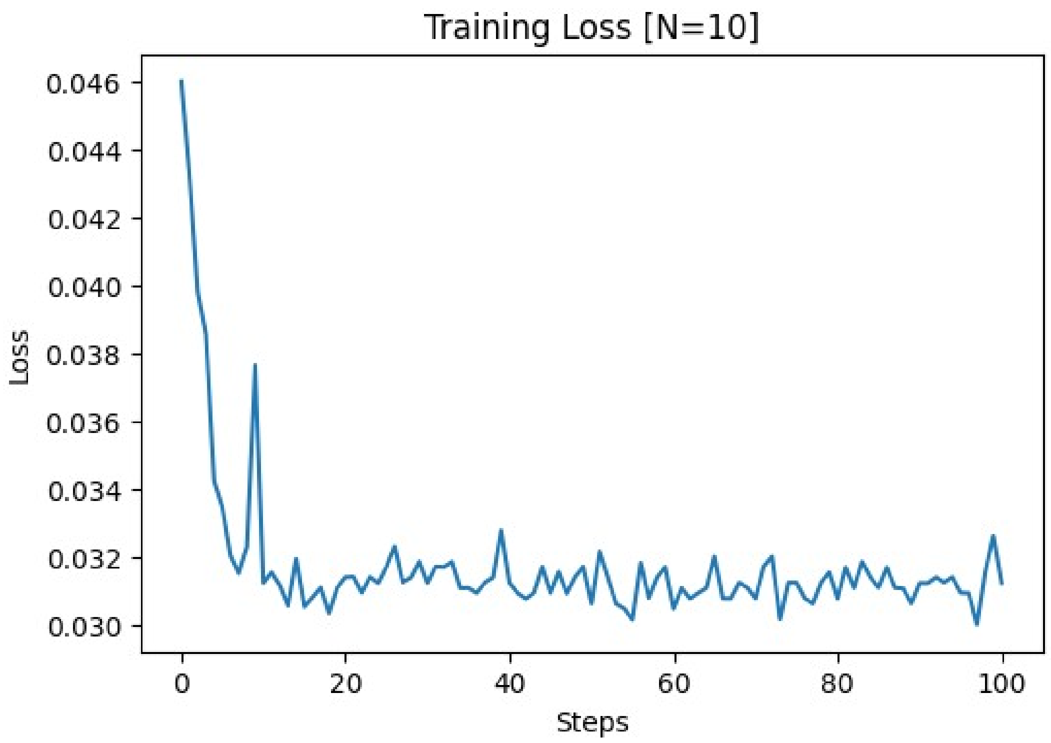
```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=5]',
path=results_path + 'test_accuracy_5_lstm.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```

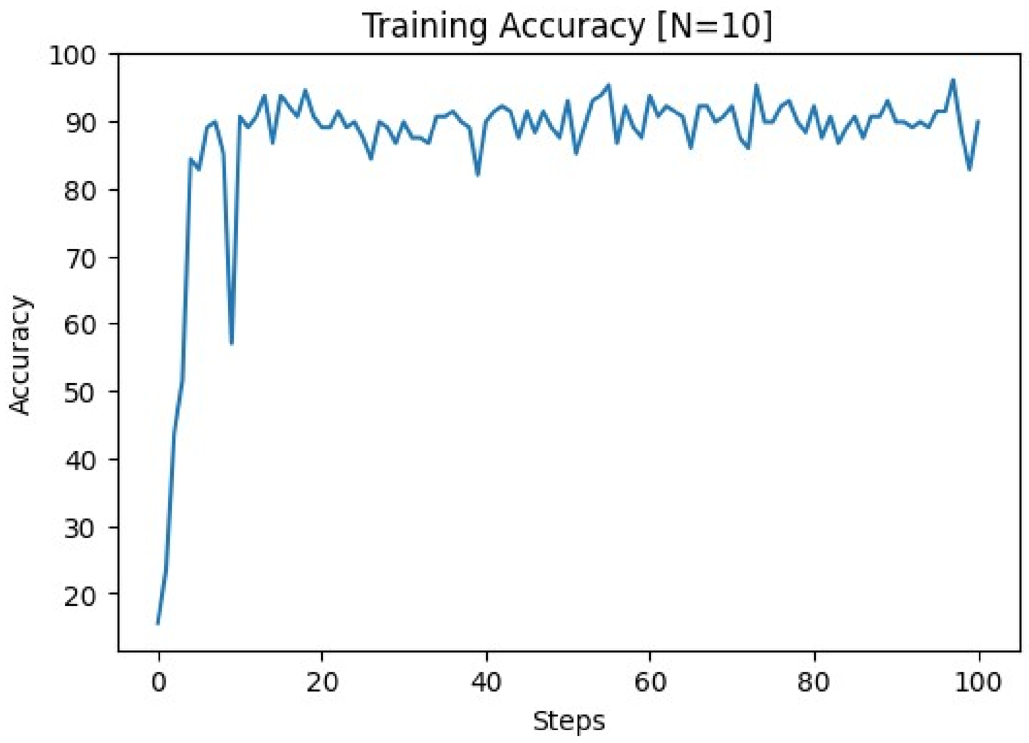


```
Average test accuracy: 100.00%

# Train the model T=10
model, losses, accuracies = train(config, input_length=p1[1],
lr=config['lstm_learning_rate'], type='LSTM', device=device)

[2024-06-27 08:09] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 4939.97, Accuracy = 15.62, Loss = 0.046

<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 08:09] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 5338.86, Accuracy = 23.44, Loss = 0.043
[2024-06-27 08:09] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 5320.98, Accuracy = 43.75, Loss = 0.040
[2024-06-27 08:09] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 5669.53, Accuracy = 51.56, Loss = 0.039
[2024-06-27 08:09] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 4585.94, Accuracy = 84.38, Loss = 0.034
[2024-06-27 08:09] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 5202.14, Accuracy = 82.81, Loss = 0.033
[2024-06-27 08:09] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 4812.74, Accuracy = 89.06, Loss = 0.032
```

```
[2024-06-27 08:09] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 4857.68, Accuracy = 89.84, Loss = 0.032
[2024-06-27 08:10] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 2519.75, Accuracy = 85.16, Loss = 0.032
[2024-06-27 08:10] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 4994.47, Accuracy = 57.03, Loss = 0.038
[2024-06-27 08:10] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 4887.13, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:10] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 4739.58, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:10] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 5218.67, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:10] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 4845.27, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:10] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 5128.69, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:10] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 5379.79, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:10] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 4518.93, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:10] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 5240.73, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:10] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 4418.22, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:10] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 3269.30, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:10] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 5327.53, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:10] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 4658.68, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:10] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 5419.44, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:10] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 3681.46, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:10] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 4533.96, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:10] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 5204.96, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:11] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 5474.48, Accuracy = 84.38, Loss = 0.032
[2024-06-27 08:11] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 5171.07, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:11] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 5501.68, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:11] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 5399.54, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:11] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 5148.80, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:11] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 4890.16, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:11] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 5374.78, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:11] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 5350.52, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:11] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 3487.81, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:11] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 5306.20, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:11] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 5187.86, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:11] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 5355.11, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:11] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 3604.52, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:11] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 5327.10, Accuracy = 82.03, Loss = 0.033
[2024-06-27 08:11] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 5530.87, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:11] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 5347.85, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:11] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 3109.09, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:11] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 5388.27, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:12] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 5407.64, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:12] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 5515.65, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:12] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 5167.09, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:12] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 5375.75, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:12] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 5418.23, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:12] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 2747.90, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:12] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 3083.25, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:12] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 3068.06, Accuracy = 85.16, Loss = 0.032
[2024-06-27 08:12] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 5316.87, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:12] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 5466.56, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:12] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 5365.22, Accuracy = 93.75, Loss = 0.030
[2024-06-27 08:12] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 2303.38, Accuracy = 95.31, Loss = 0.030
```

```
[2024-06-27 08:12] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 4465.96, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:12] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 3343.62, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:12] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 3197.66, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:13] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 5058.76, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:13] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 4780.68, Accuracy = 93.75, Loss = 0.030
[2024-06-27 08:13] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 5372.63, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:13] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 3381.27, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:13] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 4714.03, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:13] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 4928.59, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:13] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 5407.48, Accuracy = 85.94, Loss = 0.032
[2024-06-27 08:13] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 4154.61, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:13] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 5291.87, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:13] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 5376.88, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:13] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 4782.77, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:13] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 5070.75, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:13] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 5355.32, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:13] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 3741.42, Accuracy = 85.94, Loss = 0.032
[2024-06-27 08:13] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 4945.57, Accuracy = 95.31, Loss = 0.030
[2024-06-27 08:13] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 5378.44, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:14] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 5095.34, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:14] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 3662.00, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:14] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 5396.66, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:14] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 3425.25, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:14] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 3561.88, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:14] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 5368.55, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:14] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 3345.16, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:14] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 4845.89, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:14] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 5321.93, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:14] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 3444.33, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:14] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 5270.36, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:14] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 5112.18, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:14] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 5420.47, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:14] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 3487.83, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:14] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 5353.77, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:15] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 5006.72, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:15] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 5273.57, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:15] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 3212.06, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:15] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 5477.16, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:15] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 1095.53, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:15] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 5351.53, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:15] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 3025.37, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:15] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 5436.39, Accuracy = 96.09, Loss = 0.030
[2024-06-27 08:15] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 4882.46, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:15] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 5367.53, Accuracy = 82.81, Loss = 0.033
[2024-06-27 08:15] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 3503.94, Accuracy = 89.84, Loss = 0.031
Finished Training

# Plot the losses
plot_loss(losses, title='Training Loss [N=10]', path=results_path +
'training_loss_10_lstm.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=10]',
path=results_path + 'training_accuracy_10_lstm.png')
```



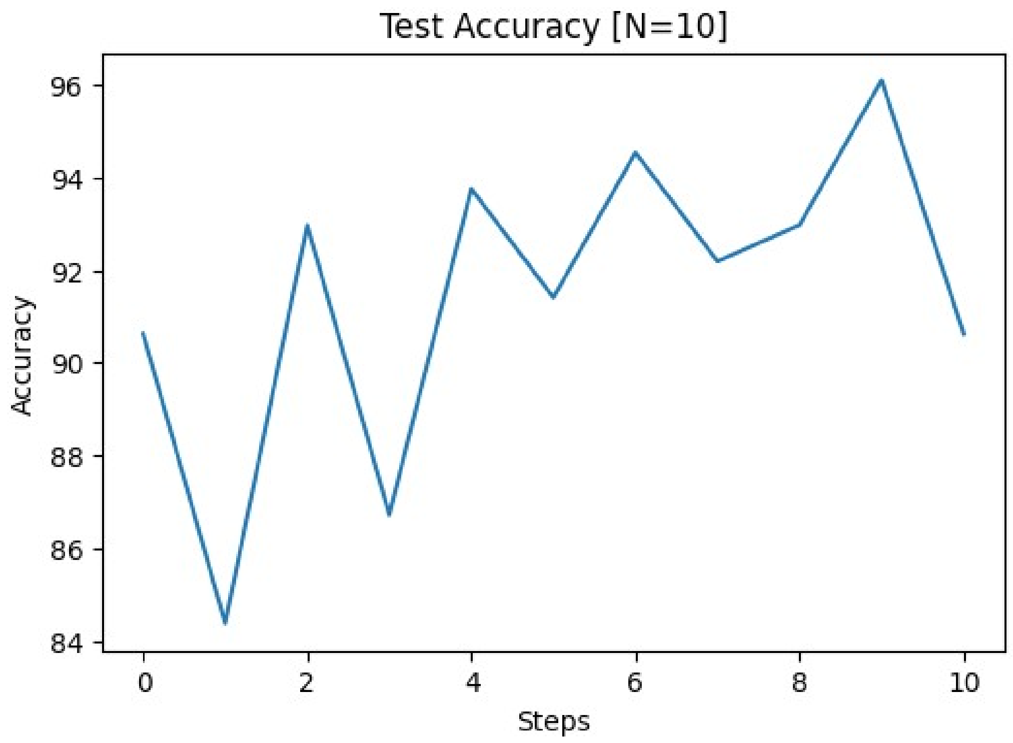


```
# Test the model
test_accuracies = test(model, input_length=p1[1], config=config,
device=device)
```

```
# Add accuracies
p3_acc_lstm.append(np.mean(test_accuracies))
```

```
Accuracy: 90.6250
Accuracy: 84.3750
Accuracy: 92.9688
Accuracy: 86.7188
Accuracy: 93.7500
Accuracy: 91.4062
Accuracy: 94.5312
Accuracy: 92.1875
Accuracy: 92.9688
Accuracy: 96.0938
Accuracy: 90.6250
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=10]',
path=results_path + 'test_accuracy_10_lstm.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



```
Average test accuracy: 91.48%
```

```python
# Train the model T=15
model, losses, accuracies = train(config, input_length=p1[2],
lr=config['lstm_learning_rate'], type='LSTM', device=device)
```

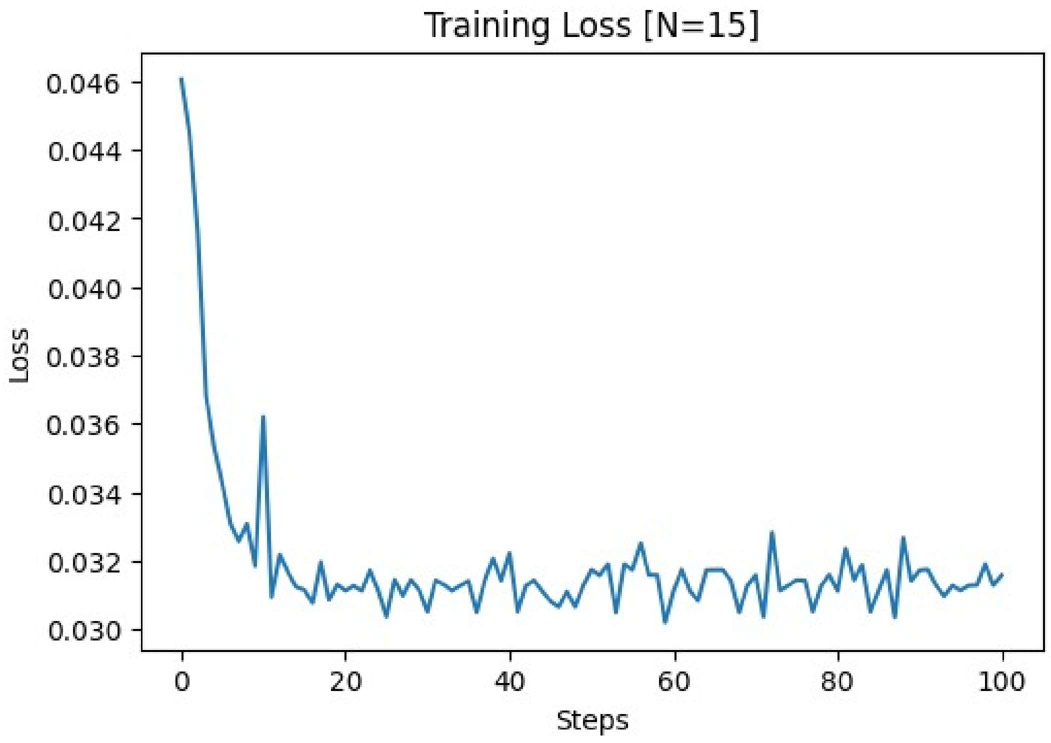
```
[2024-06-27 08:15] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 3328.30, Accuracy = 8.59, Loss = 0.046
```

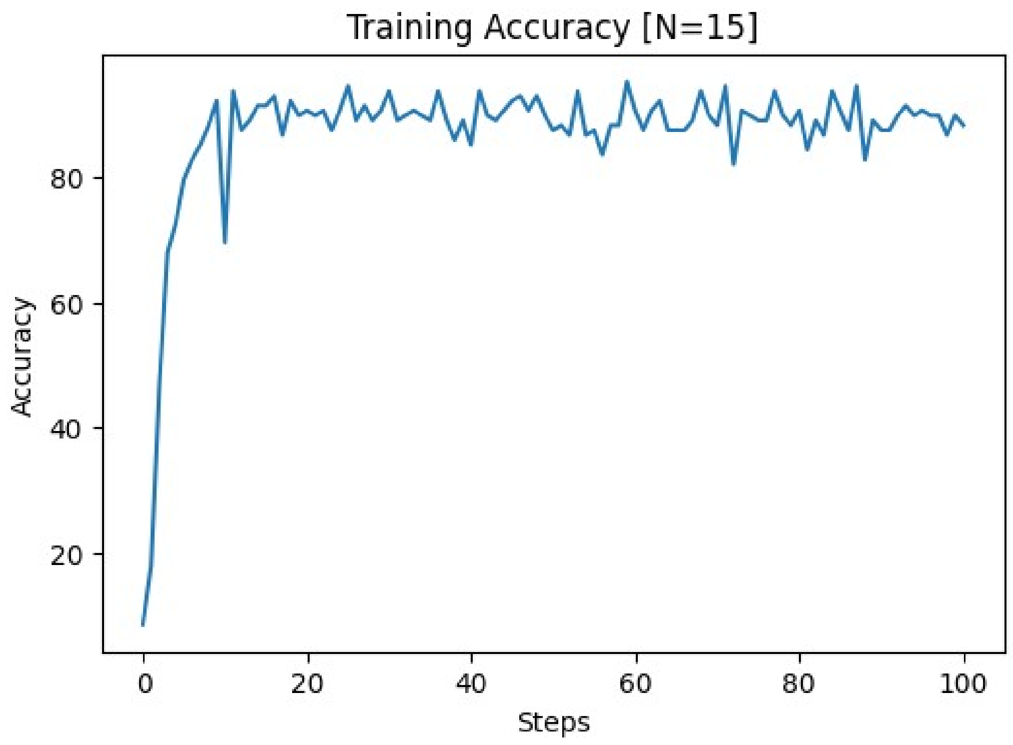
```
<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])
```

```
[2024-06-27 08:15] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 3688.14, Accuracy = 17.97, Loss = 0.044
[2024-06-27 08:15] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 2476.59, Accuracy = 46.88, Loss = 0.042
[2024-06-27 08:15] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 3552.26, Accuracy = 67.97, Loss = 0.037
[2024-06-27 08:15] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 3437.49, Accuracy = 72.66, Loss = 0.035
[2024-06-27 08:16] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 1154.24, Accuracy = 79.69, Loss = 0.034
[2024-06-27 08:16] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 2742.31, Accuracy = 82.81, Loss = 0.033
[2024-06-27 08:16] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 2241.15, Accuracy = 85.16, Loss = 0.033
[2024-06-27 08:16] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 2469.51, Accuracy = 88.28, Loss = 0.033
[2024-06-27 08:16] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 3477.12, Accuracy = 92.19, Loss = 0.032
[2024-06-27 08:16] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 3520.30, Accuracy = 69.53, Loss = 0.036
[2024-06-27 08:16] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 3271.29, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:16] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 3758.92, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:16] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 3607.95, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:16] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 2138.20, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:16] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 3509.07, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:16] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 3667.35, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:17] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 2290.06, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:17] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 3224.97, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:17] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 3476.40, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:17] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 3561.43, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:17] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 3314.96, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:17] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 2281.10, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:17] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 2093.96, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:17] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 2188.89, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:17] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 3736.31, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:17] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 1775.01, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:17] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 3088.91, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:18] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 3293.81, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:18] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 2066.62, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:18] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 3686.52, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:18] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 2934.92, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:18] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 3725.35, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:18] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 3702.94, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:18] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 2101.03, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:18] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 3298.73, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:18] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 3739.25, Accuracy = 93.75, Loss = 0.030
[2024-06-27 08:18] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 3047.15, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:18] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 3081.37, Accuracy = 85.94, Loss = 0.032
[2024-06-27 08:18] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 2396.00, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:19] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 3278.56, Accuracy = 85.16, Loss = 0.032
[2024-06-27 08:19] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 3669.18, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:19] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 2155.94, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:19] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 3375.34, Accuracy = 89.06, Loss = 0.031
```

```
[2024-06-27 08:19] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 3562.49, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:19] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 3691.92, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:19] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 3416.36, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:19] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 2236.91, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:19] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 3316.58, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:19] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 3042.43, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:19] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 3413.19, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:19] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 3773.45, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:20] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 3643.58, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:20] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 2991.94, Accuracy = 93.75, Loss = 0.030
[2024-06-27 08:20] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 3308.32, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:20] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 2226.31, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:20] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 3315.06, Accuracy = 83.59, Loss = 0.033
[2024-06-27 08:20] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 3657.16, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:20] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 3431.14, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:20] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 3529.58, Accuracy = 95.31, Loss = 0.030
[2024-06-27 08:20] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 3714.57, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:20] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 3640.62, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:20] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 3074.97, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:20] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 2263.34, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:20] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 3508.55, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:21] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 3686.67, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:21] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 3654.74, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:21] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 3168.71, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:21] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 3745.96, Accuracy = 93.75, Loss = 0.030
[2024-06-27 08:21] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 3694.61, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:21] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 3670.03, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:21] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 2131.81, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:21] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 2437.23, Accuracy = 82.03, Loss = 0.033
[2024-06-27 08:21] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 3635.78, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:21] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 3432.59, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:21] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 3706.26, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:21] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 3123.59, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:22] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 3621.19, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:22] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 3688.19, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:22] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 3436.57, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:22] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 3690.85, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:22] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 2302.52, Accuracy = 84.38, Loss = 0.032
[2024-06-27 08:22] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 3476.00, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:22] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 3618.63, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:22] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 2298.57, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:22] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 2774.31, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:22] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 3611.13, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:22] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 3349.46, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:22] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 3598.45, Accuracy = 82.81, Loss = 0.033
[2024-06-27 08:22] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 3629.79, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:23] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 3715.40, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:23] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 3354.48, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:23] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 2493.10, Accuracy = 89.84, Loss = 0.031
```

```
[2024-06-27 08:23] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 3327.20, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:23] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 3679.59, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:23] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 2118.46, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:23] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 3602.51, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:23] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 3684.29, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:23] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 3720.67, Accuracy = 86.72, Loss = 0.032
[2024-06-27 08:23] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 3584.18, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:23] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 3685.30, Accuracy = 88.28, Loss = 0.032
Finished Training

# Plot the losses
plot_loss(losses, title='Training Loss [N=15]', path=results_path +
'training_loss_15_lstm.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=15]',
path=results_path + 'training_accuracy_15_lstm.png')
```

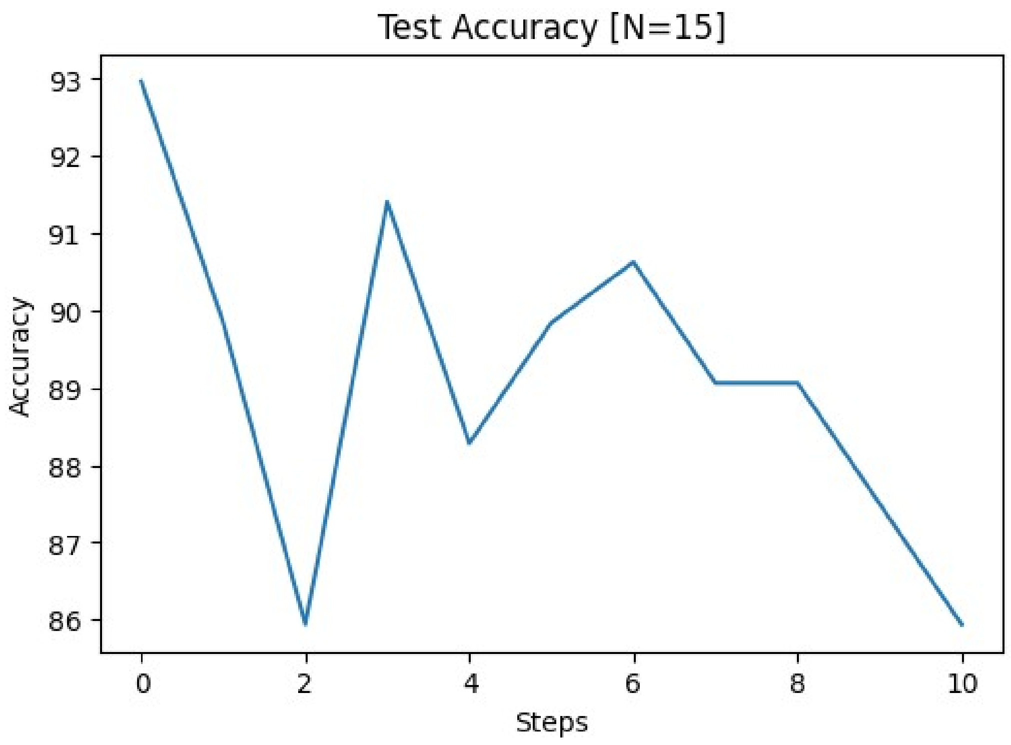




```
# Test the model
test_accuracies = test(model, input_length=p1[2], config=config,
device=device)

# Add accuracies
p3_acc_lstm.append(np.mean(test_accuracies))
```

```
Accuracy: 92.9688
Accuracy: 89.8438
Accuracy: 85.9375
Accuracy: 91.4062
Accuracy: 88.2812
Accuracy: 89.8438
Accuracy: 90.6250
Accuracy: 89.0625
Accuracy: 89.0625
Accuracy: 87.5000
Accuracy: 85.9375
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=15]',
path=results_path + 'test_accuracy_15_lstm.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```

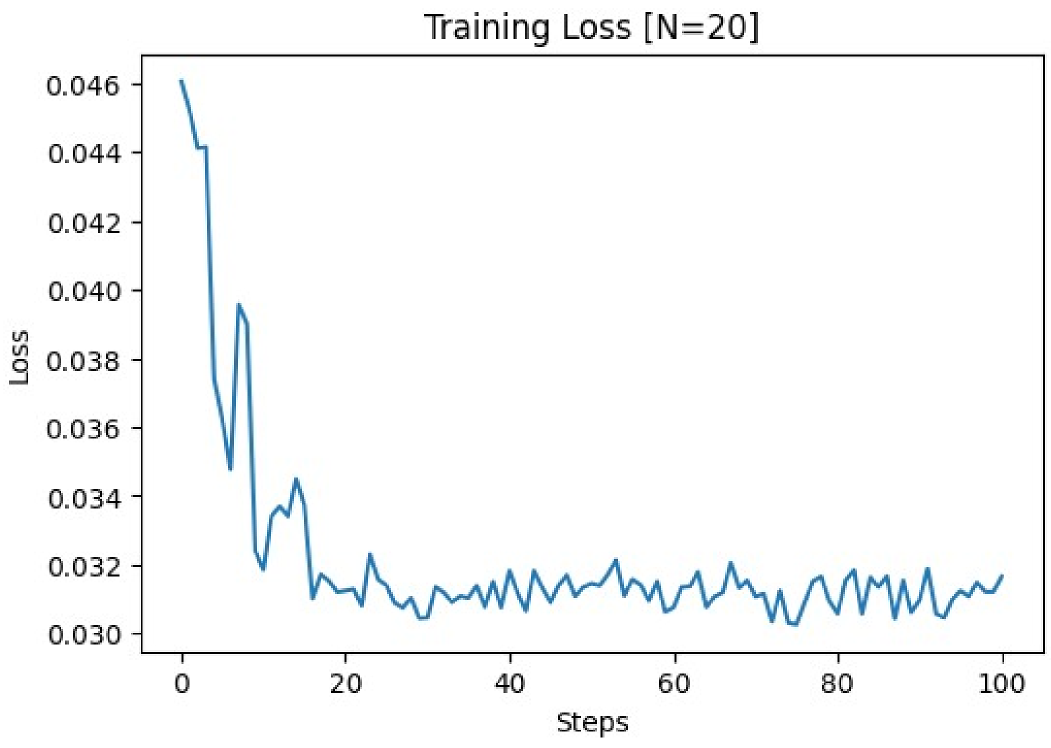


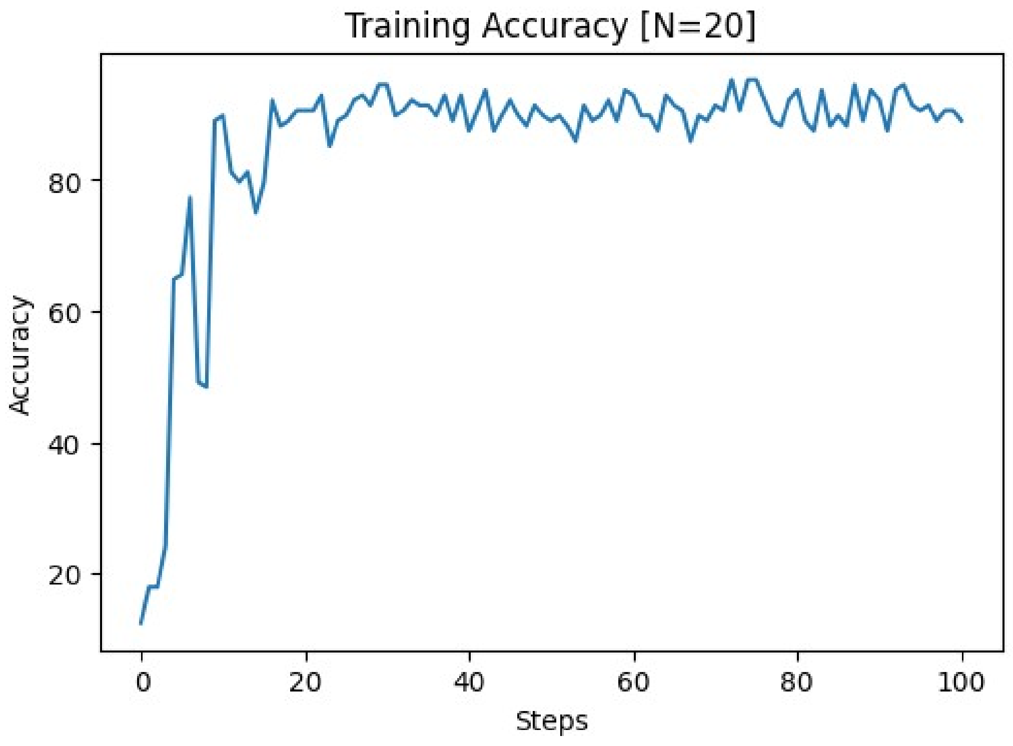
```
Average test accuracy: 89.13%

# Train the model T=20
model, losses, accuracies = train(config, input_length=p1[3],
lr=config['lstm_learning_rate'], type='LSTM', device=device)

[2024-06-27 08:23] Train Step 0000/10000, Batch Size = 128,
Examples/Sec = 2728.46, Accuracy = 12.50, Loss = 0.046

<ipython-input-5-69509e95475b>:64: UserWarning:
torch.nn.utils.clip_grad_norm is now deprecated in favor of
torch.nn.utils.clip_grad_norm_.
  nn.utils.clip_grad_norm(model.parameters(),
max_norm=config['max_norm'])

[2024-06-27 08:23] Train Step 0100/10000, Batch Size = 128,
Examples/Sec = 2872.18, Accuracy = 17.97, Loss = 0.045
[2024-06-27 08:24] Train Step 0200/10000, Batch Size = 128,
Examples/Sec = 1608.76, Accuracy = 17.97, Loss = 0.044
[2024-06-27 08:24] Train Step 0300/10000, Batch Size = 128,
Examples/Sec = 2788.32, Accuracy = 24.22, Loss = 0.044
[2024-06-27 08:24] Train Step 0400/10000, Batch Size = 128,
Examples/Sec = 1737.74, Accuracy = 64.84, Loss = 0.037
[2024-06-27 08:24] Train Step 0500/10000, Batch Size = 128,
Examples/Sec = 2213.54, Accuracy = 65.62, Loss = 0.036
[2024-06-27 08:24] Train Step 0600/10000, Batch Size = 128,
Examples/Sec = 1853.41, Accuracy = 77.34, Loss = 0.035
```

```
[2024-06-27 08:24] Train Step 0700/10000, Batch Size = 128,
Examples/Sec = 2653.05, Accuracy = 49.22, Loss = 0.040
[2024-06-27 08:24] Train Step 0800/10000, Batch Size = 128,
Examples/Sec = 2658.21, Accuracy = 48.44, Loss = 0.039
[2024-06-27 08:24] Train Step 0900/10000, Batch Size = 128,
Examples/Sec = 2782.98, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:24] Train Step 1000/10000, Batch Size = 128,
Examples/Sec = 2669.13, Accuracy = 89.84, Loss = 0.032
[2024-06-27 08:24] Train Step 1100/10000, Batch Size = 128,
Examples/Sec = 2717.30, Accuracy = 81.25, Loss = 0.033
[2024-06-27 08:25] Train Step 1200/10000, Batch Size = 128,
Examples/Sec = 2574.28, Accuracy = 79.69, Loss = 0.034
[2024-06-27 08:25] Train Step 1300/10000, Batch Size = 128,
Examples/Sec = 2661.58, Accuracy = 81.25, Loss = 0.033
[2024-06-27 08:25] Train Step 1400/10000, Batch Size = 128,
Examples/Sec = 2800.29, Accuracy = 75.00, Loss = 0.034
[2024-06-27 08:25] Train Step 1500/10000, Batch Size = 128,
Examples/Sec = 2066.89, Accuracy = 79.69, Loss = 0.034
[2024-06-27 08:25] Train Step 1600/10000, Batch Size = 128,
Examples/Sec = 2781.54, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:25] Train Step 1700/10000, Batch Size = 128,
Examples/Sec = 2374.54, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:25] Train Step 1800/10000, Batch Size = 128,
Examples/Sec = 2729.46, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:25] Train Step 1900/10000, Batch Size = 128,
Examples/Sec = 2784.90, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:25] Train Step 2000/10000, Batch Size = 128,
Examples/Sec = 2658.94, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:26] Train Step 2100/10000, Batch Size = 128,
Examples/Sec = 2780.26, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:26] Train Step 2200/10000, Batch Size = 128,
Examples/Sec = 2679.37, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:26] Train Step 2300/10000, Batch Size = 128,
Examples/Sec = 2805.35, Accuracy = 85.16, Loss = 0.032
[2024-06-27 08:26] Train Step 2400/10000, Batch Size = 128,
Examples/Sec = 2806.41, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:26] Train Step 2500/10000, Batch Size = 128,
Examples/Sec = 1720.70, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:26] Train Step 2600/10000, Batch Size = 128,
Examples/Sec = 2700.03, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:26] Train Step 2700/10000, Batch Size = 128,
Examples/Sec = 1870.43, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:26] Train Step 2800/10000, Batch Size = 128,
Examples/Sec = 2747.14, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:26] Train Step 2900/10000, Batch Size = 128,
Examples/Sec = 2742.62, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:26] Train Step 3000/10000, Batch Size = 128,
Examples/Sec = 2798.20, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:27] Train Step 3100/10000, Batch Size = 128,
Examples/Sec = 2874.12, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:27] Train Step 3200/10000, Batch Size = 128,
Examples/Sec = 2826.23, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:27] Train Step 3300/10000, Batch Size = 128,
Examples/Sec = 2722.00, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:27] Train Step 3400/10000, Batch Size = 128,
Examples/Sec = 2494.13, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:27] Train Step 3500/10000, Batch Size = 128,
Examples/Sec = 2720.51, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:27] Train Step 3600/10000, Batch Size = 128,
Examples/Sec = 2798.93, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:27] Train Step 3700/10000, Batch Size = 128,
Examples/Sec = 2360.28, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:27] Train Step 3800/10000, Batch Size = 128,
Examples/Sec = 2698.52, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:27] Train Step 3900/10000, Batch Size = 128,
Examples/Sec = 2827.21, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:27] Train Step 4000/10000, Batch Size = 128,
Examples/Sec = 2738.79, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:28] Train Step 4100/10000, Batch Size = 128,
Examples/Sec = 2824.11, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:28] Train Step 4200/10000, Batch Size = 128,
Examples/Sec = 1708.70, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:28] Train Step 4300/10000, Batch Size = 128,
Examples/Sec = 2834.53, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:28] Train Step 4400/10000, Batch Size = 128,
Examples/Sec = 1897.69, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:28] Train Step 4500/10000, Batch Size = 128,
Examples/Sec = 2788.20, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:28] Train Step 4600/10000, Batch Size = 128,
Examples/Sec = 1848.44, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:28] Train Step 4700/10000, Batch Size = 128,
Examples/Sec = 2751.42, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:28] Train Step 4800/10000, Batch Size = 128,
Examples/Sec = 2643.40, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:28] Train Step 4900/10000, Batch Size = 128,
Examples/Sec = 2749.07, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:28] Train Step 5000/10000, Batch Size = 128,
Examples/Sec = 1915.58, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:29] Train Step 5100/10000, Batch Size = 128,
Examples/Sec = 2823.04, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:29] Train Step 5200/10000, Batch Size = 128,
Examples/Sec = 2786.13, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:29] Train Step 5300/10000, Batch Size = 128,
Examples/Sec = 2586.16, Accuracy = 85.94, Loss = 0.032
[2024-06-27 08:29] Train Step 5400/10000, Batch Size = 128,
Examples/Sec = 1866.36, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:29] Train Step 5500/10000, Batch Size = 128,
Examples/Sec = 2807.37, Accuracy = 89.06, Loss = 0.032
```

```
[2024-06-27 08:29] Train Step 5600/10000, Batch Size = 128,
Examples/Sec = 2711.11, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:29] Train Step 5700/10000, Batch Size = 128,
Examples/Sec = 2831.16, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:29] Train Step 5800/10000, Batch Size = 128,
Examples/Sec = 2447.01, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:29] Train Step 5900/10000, Batch Size = 128,
Examples/Sec = 2753.25, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:29] Train Step 6000/10000, Batch Size = 128,
Examples/Sec = 2563.35, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:30] Train Step 6100/10000, Batch Size = 128,
Examples/Sec = 2812.74, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:30] Train Step 6200/10000, Batch Size = 128,
Examples/Sec = 2472.26, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:30] Train Step 6300/10000, Batch Size = 128,
Examples/Sec = 1690.29, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:30] Train Step 6400/10000, Batch Size = 128,
Examples/Sec = 2713.92, Accuracy = 92.97, Loss = 0.031
[2024-06-27 08:30] Train Step 6500/10000, Batch Size = 128,
Examples/Sec = 1736.91, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:30] Train Step 6600/10000, Batch Size = 128,
Examples/Sec = 2819.26, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:30] Train Step 6700/10000, Batch Size = 128,
Examples/Sec = 1792.14, Accuracy = 85.94, Loss = 0.032
[2024-06-27 08:30] Train Step 6800/10000, Batch Size = 128,
Examples/Sec = 2702.62, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:30] Train Step 6900/10000, Batch Size = 128,
Examples/Sec = 2085.35, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:30] Train Step 7000/10000, Batch Size = 128,
Examples/Sec = 2753.42, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:31] Train Step 7100/10000, Batch Size = 128,
Examples/Sec = 2623.84, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:31] Train Step 7200/10000, Batch Size = 128,
Examples/Sec = 2786.88, Accuracy = 95.31, Loss = 0.030
[2024-06-27 08:31] Train Step 7300/10000, Batch Size = 128,
Examples/Sec = 2852.98, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:31] Train Step 7400/10000, Batch Size = 128,
Examples/Sec = 2846.31, Accuracy = 95.31, Loss = 0.030
[2024-06-27 08:31] Train Step 7500/10000, Batch Size = 128,
Examples/Sec = 2371.06, Accuracy = 95.31, Loss = 0.030
[2024-06-27 08:31] Train Step 7600/10000, Batch Size = 128,
Examples/Sec = 2682.30, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:31] Train Step 7700/10000, Batch Size = 128,
Examples/Sec = 2675.29, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:31] Train Step 7800/10000, Batch Size = 128,
Examples/Sec = 2649.04, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:31] Train Step 7900/10000, Batch Size = 128,
Examples/Sec = 2774.54, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:31] Train Step 8000/10000, Batch Size = 128,
Examples/Sec = 1646.62, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:32] Train Step 8100/10000, Batch Size = 128,
Examples/Sec = 2838.93, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:32] Train Step 8200/10000, Batch Size = 128,
Examples/Sec = 1669.33, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:32] Train Step 8300/10000, Batch Size = 128,
Examples/Sec = 2806.14, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:32] Train Step 8400/10000, Batch Size = 128,
Examples/Sec = 1772.52, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:32] Train Step 8500/10000, Batch Size = 128,
Examples/Sec = 2863.19, Accuracy = 89.84, Loss = 0.031
[2024-06-27 08:32] Train Step 8600/10000, Batch Size = 128,
Examples/Sec = 2366.02, Accuracy = 88.28, Loss = 0.032
[2024-06-27 08:32] Train Step 8700/10000, Batch Size = 128,
Examples/Sec = 2734.31, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:32] Train Step 8800/10000, Batch Size = 128,
Examples/Sec = 2819.02, Accuracy = 89.06, Loss = 0.032
[2024-06-27 08:32] Train Step 8900/10000, Batch Size = 128,
Examples/Sec = 2744.77, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:32] Train Step 9000/10000, Batch Size = 128,
Examples/Sec = 2634.02, Accuracy = 92.19, Loss = 0.031
[2024-06-27 08:33] Train Step 9100/10000, Batch Size = 128,
Examples/Sec = 2820.35, Accuracy = 87.50, Loss = 0.032
[2024-06-27 08:33] Train Step 9200/10000, Batch Size = 128,
Examples/Sec = 2791.38, Accuracy = 93.75, Loss = 0.031
[2024-06-27 08:33] Train Step 9300/10000, Batch Size = 128,
Examples/Sec = 2771.00, Accuracy = 94.53, Loss = 0.030
[2024-06-27 08:33] Train Step 9400/10000, Batch Size = 128,
Examples/Sec = 2405.06, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:33] Train Step 9500/10000, Batch Size = 128,
Examples/Sec = 1758.69, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:33] Train Step 9600/10000, Batch Size = 128,
Examples/Sec = 2812.67, Accuracy = 91.41, Loss = 0.031
[2024-06-27 08:33] Train Step 9700/10000, Batch Size = 128,
Examples/Sec = 1947.32, Accuracy = 89.06, Loss = 0.031
[2024-06-27 08:33] Train Step 9800/10000, Batch Size = 128,
Examples/Sec = 2850.42, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:33] Train Step 9900/10000, Batch Size = 128,
Examples/Sec = 1623.72, Accuracy = 90.62, Loss = 0.031
[2024-06-27 08:33] Train Step 10000/10000, Batch Size = 128,
Examples/Sec = 2684.93, Accuracy = 89.06, Loss = 0.032
Finished Training
```

```
# Plot the losses
plot_loss(losses, title='Training Loss [N=20]', path=results_path +
'training_loss_20_lstm.png')

# Plot the accuracies
plot_accuracy(accuracies, title='Training Accuracy [N=20]',
path=results_path + 'training_accuracy_20_lstm.png')
```



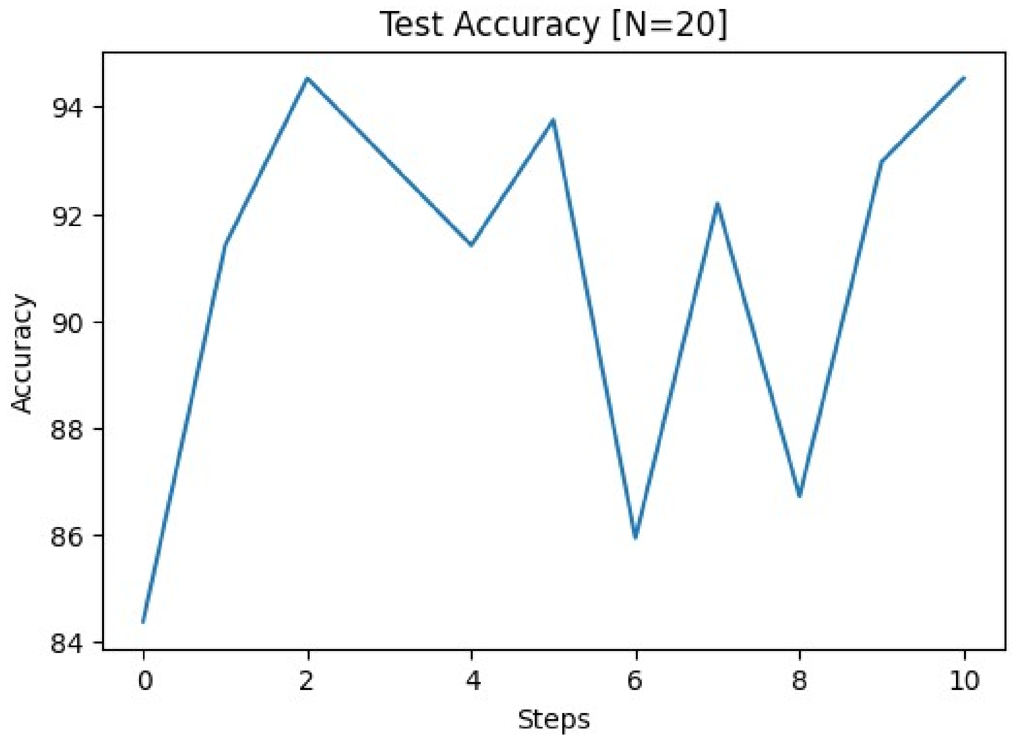


```
# Test the model
test_accuracies = test(model, input_length=p1[3], config=config,
device=device)
```

```
# Add accuracies
p3_acc_lstm.append(np.mean(test_accuracies))
```

```
Accuracy: 84.3750
Accuracy: 91.4062
Accuracy: 94.5312
Accuracy: 92.9688
Accuracy: 91.4062
Accuracy: 93.7500
Accuracy: 85.9375
Accuracy: 92.1875
Accuracy: 86.7188
Accuracy: 92.9688
Accuracy: 94.5312
Finished Testing
```

```
# plot the test accuracies
plot_accuracy(test_accuracies, title='Test Accuracy [N=20]',
path=results_path + 'test_accuracy_20_lstm.png')

# Average accuracy over all Steps
print(f"Average test accuracy: {np.mean(test_accuracies):.2f}%")
```



```
Average test accuracy: 90.98%
```

Question 1.6: Comparison LSTM & Vanilla RNN

Write down a comparison with the vanilla RNN and think of reasons for the different behavior on the Palindrome prediction task.

Convergence speed:

- LSTM network generally converges faster than the Vanilla RNN during training due to its gating mechanism, which helps mitigate the vanishing gradient problem.
- The Vanilla RNN is more susceptible to the Vanishing Gradient problem since it's gradients have to be propagated through sevaral learnable layers during BPTT who's values might become arbitrarily small or large. For both training loops a gradient clipping is used to prevent the models from exploding gradients.

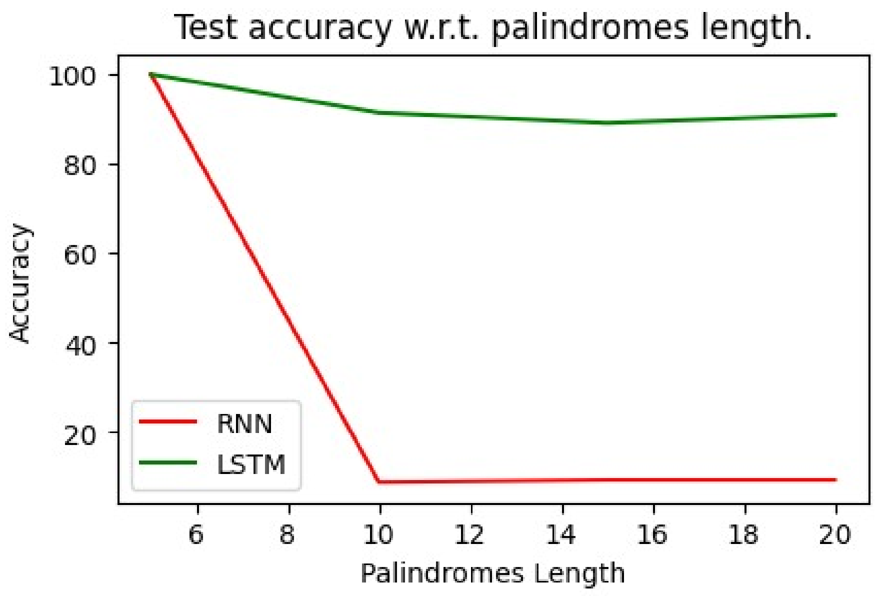
Performance with longer sequences:

- Vanilla RNNs: The accuracy tends to decrease, and the loss increases during training with longer sequences due to the vanishing gradient problem. Gradient clipping is used to prevent gradients from exploding, but it does not address the vanishing gradient issue noticable in Vanilla RNNs which prevents the Vanilla RNN to *memorise* the necessary information from an earlier timestep in order to solve the palindrome task.
- LSTM network: The LSTM also exhibits reduced accuracy with increasing lengths of palindrome sequences. However, in comparison to the Vanilla RNN this is less pronounced. The network needs to learn parameters to preserve the right amount of information from a very early stage onwards, which can be challenging.

LSTM's updating information:

- It may be possible that the LSTM network struggles to learn to update the right information and not remove relevant signals, especially when dealing with longer sequences. This might be due to the difficulty in optimizing the complex interactions between the input, forget, and output gates.

```python
plt.figure(figsize=(5,3))

plt.title('Test accuracy w.r.t. palindromes length.')
# Plot RNN avg. test acc
plt.plot(p1, p2_acc_rnn, color='red', label='RNN')
# Plot LSTM avg. test acc
plt.plot(p1, p3_acc_lstm, color='green', label='LSTM')

plt.xlabel('Palindromes Length')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


Test accuracy w.r.t. palindromes length.
Accuracy
Palindromes Length
RNN
LSTM
100
80
60
40
20
6
8
10
12
14
16
18
20