Практическая работа по дисциплине "Нейроэволюционные вычисления"

Малкин Артем студент гр. 8ВМ22 НИ ТПУ

19 Мая 2023

Номенклатура

NEAT	NeuroEvolution of Augmenting Topology
CUDA	Compute Unified Device Architecture

1 Постановка задачи

Эта секция описывает задачи, поставленные в рамках практической работы.

1.1 Программный код

В рамках практической работы был разработан программный код с применением модуля PyTorch для реализации нейроэволюционного алгоритма NEAT. Программный код был разработан на языке Python.

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision.datasets import MNIST
from torch.utils.data import DataLoader
from torchvision.transforms import ToTensor
import neat

# Define the PyTorch-based neural network class
class NeuralNetwork(nn.Module):
    def __init__(self, input_size, output_size):
        super(NeuralNetwork, self).__init__()
        self.fc = nn.Linear(input_size, 64)
        self.relu = nn.ReLU()
        self.out = nn.Linear(64, output_size)

    def forward(self, x):
        x = self.fc(x)
        x = self.relu(x)
```

```python
        x = self.out(x)
        return x

# Define the fitness evaluation function
def eval_fitness(genomes, config):
    for genome_id, genome in genomes:
        net = neat.nn.FeedForwardNetwork.create(genome, config)
        criterion = nn.CrossEntropyLoss()
        optimizer = optim.Adam(net.parameters(), lr=0.01)
        for batch_images, batch_labels in train_loader:
            batch_images = batch_images.cuda()
            batch_labels = batch_labels.cuda()

            optimizer.zero_grad()
            outputs = net.activate(batch_images.view(batch_images.size(0), -1))
            loss = criterion(outputs, batch_labels)
            loss.backward()
            optimizer.step()

        # Evaluate the fitness
        correct = 0
        total = 0
        for test_images, test_labels in test_loader:
            test_images = test_images.cuda()
            test_labels = test_labels.cuda()

            outputs = net.activate(test_images.view(test_images.size(0), -1))
            _, predicted = torch.max(outputs.data, 1)
            total += test_labels.size(0)
            correct += (predicted == test_labels).sum().item()

        accuracy = correct / total
        genome.fitness = accuracy

# Load MNIST dataset
train_dataset = MNIST(root='./data', train=True, transform=ToTensor(), download=True)
train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True)
test_dataset = MNIST(root='./data', train=False, transform=ToTensor(), download=True)
test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False)

```

```python
    # Configure NEAT
    config_path = 'neat-config-file.cfg'  # Specify the path to your NEAT configuration file
    config = neat.config.Config(neat.DefaultGenome, neat.DefaultReproduction, neat.DefaultSpeciesSet,
                                neat.DefaultStagnation, config_path)

    # Create the population
    population = neat.Population(config)

    # Add a reporter to display the progress during evolution
    reporter = neat.StdOutReporter(True)
    population.add_reporter(reporter)

    # Run NEAT
    best_genome = population.run(eval_fitness, 100)

    # Retrieve the best neural network from the evolved population
    best_net = neat.nn.FeedForwardNetwork.create(best_genome, config)

    # Test the best neural network
    with torch.no_grad():
        correct = 0
        total = 0
        for test_images, test_labels in test_loader:
            test_images = test_images.cuda()
            test_labels = test_labels.cuda()

            outputs = best_net.activate(test_images.view(test_images.size(0), -1))
            _, predicted = torch.max(outputs.data, 1)
            total += test_labels.size(0)
            correct += (predicted == test_labels).sum().item()

        accuracy = correct / total
        print("Accuracy of the best network: {:.2f}%".format(accuracy * 100))
```

Listing 1: Программный код реализации алгоритма NEAT с применением модуля PyTorch

Aircraft Characteristics	Values
Total Weight	2.917 kg
Endurance	$\approx$ 15 minutes
Center of Gravity	17.75 in. aft of aircraft nose

Table 1: UAS performance specifications.

1.2 ВСЁ, что ниже, (и таблица выше) - не моё, а fake-информация, не относящаяся к NEAT

The goals of this experiment are as follows:

- Flight test a small unmanned aerial system (UAS).
- Demonstrate viability of using an aviation tranponder onboard a UAS.

The aircraft used in the experiment is shown in Figure 1. This was flight tested on several occasions [1] and uses autonomous algorithms to perform search and rescue [2].

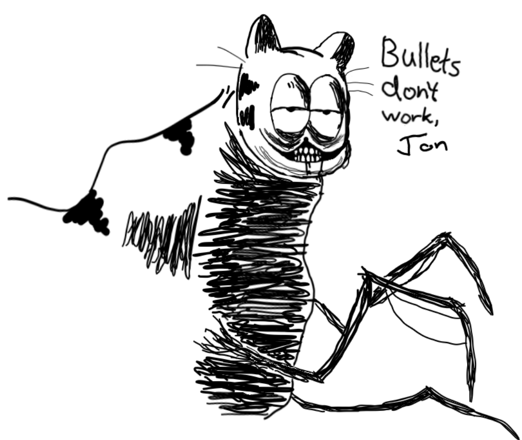

Figure 1: Bullets don't work, Jon.

Characteristics of the aircraft are shown in Table 1.
The ground control station (GCS) is shown in Figure 2.

1.3 Algorithms

An example of an equation is given in Eq. 1.

$$K = P_{predicted}H^T\left(HP_{predicted}H^T + R\right)^{-1} \tag{1}$$

Figure 2: Operators at the GCS.

2 Conclusions

Here are the conclusions. Note that this is a separate .tex file which is included into the larger document.

References

[1] Lum, C. W., Larson, R. S., Handley, W., Lui, S., and Caratao, Z., "Flight Testing an ADS-B Equipped sUAS in GPS-Denied Environments," *Proceedings of the AIAA Flight Testing Conference*, Denver, CO, June 2017.

[2] Lum, C. W., Vagners, J., and Rysdyk, R. T., "Search Algorithm for Teams of Heterogeneous Agents with Coverage Guarantees," *AIAA Journal of Aerospace Computing, Information, and Communication*, Vol. 7, January 2010, pp. 1–31.