Managing greenhouse gas emissions using a neural network from gas generating plants

Ekaterina Kulakova*, and *Elena* Muravyova

Ufa State Petroleum Technological University, Sterlitamak 453103, Russia

Abstract. This article describes a method for developing a neural network model for the control system of gas generator sets, which are an efficient and environmentally friendly way of energy production. Gas generator sets are used to convert solid fuels such as coal or wood into synthesis gas, which can then be used to generate electricity or other types of thermal energy. The article describes a method for developing a neural network model for a wood chip gas generator for preparing a fuel recipe. The article also discusses the advantages of gas generator sets, such as the possibility of using inexpensive and affordable fuels, reducing emissions of greenhouse gases and other harmful substances, as well as improving energy efficiency. The article provides useful information about gas generator sets and their role in providing environmentally friendly and efficient energy.

1 Introduction

In the modern world, the problem of energy security and environmental sustainability is becoming more and more urgent [1-3].

Gas generators operate on the principle of gasification, in which fuel is converted into gas with a high content of hydrogen and carbon, which can be used to generate electricity.

The neural network control system for gas generator sets is based on machine learning algorithms that allow neural networks to analyze data on the operation of gas generators, optimize process parameters and predict possible deviations and breakdowns. Such a system is capable of automatically regulating the operation of gas generators and making decisions based on the data obtained [4].

The practical application of gas generators is already showing its effectiveness and prospects. This article will discuss the basic principles of operation of gas generators, as well as the development of a neural network model of the control system.

The main parts of a gas generator set are [5-6]:

a) a gas generator, into which burned fuel is loaded, giving combustible gas;
b) cleaners for coarse and fine purification of gas from mechanical impurities;
c) liquid cooler;
d) a mixer where a gas-air combustible mixture is prepared;
e) an electric fan for igniting fuel during the initial preparation of the gas generator for operation;

* Corresponding author: kulakova87@list.ru

f) gas pipelines connecting separate parts of the gas generator set.

Solid fuels are burned to produce gas generator gas, more often wood and wood waste; peat, brown coals, anthracite, straw briquettes can also be used [7].

The conversion of solid fuel into gaseous fuel occurs in a gas generator, and depending on the direction of the incoming air and exhaust gas flows, direct, horizontal and reverse, or inverted, gasification processes are distinguished.

The fuel is loaded into the gas generator through the upper hatch. The combustion zone in the fuel is located in the lower part of the gas generator. [8-9].

The generator gas obtained in this way rises to the nozzle, from where it is directed through purifiers and a cooler to the mixer.

Burn temperature and gas release can be controlled factors in the combustion of materials [10].

Burn, especially incomplete combustion, can lead to environmental pollution, the release of toxins and other negative consequences.

A number of basic gases are released during the combustion of wood chips. These gases include [11-14]:

Carbon dioxide (CO_2): Carbon dioxide is the main gas that is released during the burning of wood. [15].

Carbon monoxide (CO): Carbon monoxide is formed when wood materials are incompletely burned.

Water vapor (H_2O): Burn wood also creates water vapor, which is formed by the oxidation of hydrogen contained in the wood [14-15].

Methane (CH_4): Methane can be formed during the burning of wood, especially in the presence of high temperatures.

Nitrogen oxides (NO_X): Nitrogen oxides include nitrogen oxide (NO) and nitrogen dioxide (NO_2).

2 Materials and methods

The Python programming language has been chosen as the tool for creating a neural network model, as it is flexible and logical [16-19].

Goal. To develop a neural network model for the preparation of a fuel recipe for gas generator sets, forming a list and number of chips of various types for the target value of the emission of gases generated during combustion.

Research methods. To achieve the goal of this project, we will develop a neural network based on the machine learning framework for Python - PyTorch.

3 Results and discussion

A PyTorch neural network model has been developed for a wood chip gas generator for preparing a fuel recipe for gas generator sets, forming a list and number of chips of various types. The neural model is possible for practical application.

The neural network is developed based on the Python programming language in the integrated Pycharm environment.

Three layers are defined in the neural model class. Each layer of the nn.Linear class accepts two parameters as input: the number of inputs and the number of outputs. The number of inputs and outputs are calculated from the learning dictionary.

The class of the neural model is shown in Figure 1.

```
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        return out
```

Fig. 1. Description of the neural model class.

1. 1 Input layer.

Accepts input data such as the dimensions and characteristics of the gas generator, the properties of sawdust, air supply parameters and other related parameters.

2. 2 Hidden layers.

You can use multiple layers with different numbers of neurons to extract complex dependencies from the input data.

Each layer can use an activation function such as ReLU (Rectified Linear Unit) or hyperbolic tangent to introduce non-linearity into the model.

3. Output layer.

Determines the output values of the neural network, such as the efficiency of the gas generator set, the amount of gas produced, or other indicators.

It is possible to use various activation functions depending on the required output values.

4. Training and optimization.

The method of gradient descent and back propagation of the error was used to train the neural network.

Optimization of the model includes the selection of optimal hyperparameters, such as the learning rate and the number of neurons in the layers.

5. Validation and testing:

After training, the model must be tested on new data to assess its accuracy and effectiveness.

Validation of the model will help determine how well it generalizes to new examples and avoids overfitting.

The number of inputs of the first layer will be equal to the number of gas molecules, the number of outputs is 8.

The number of inputs and outputs of the hidden layer is 8.

The number of inputs of the last layer is equal to the number of outputs of the hidden layer, and the number of outputs is equal to the number of types of chips.

The architecture of the developed neural model is shown in Figure 2.

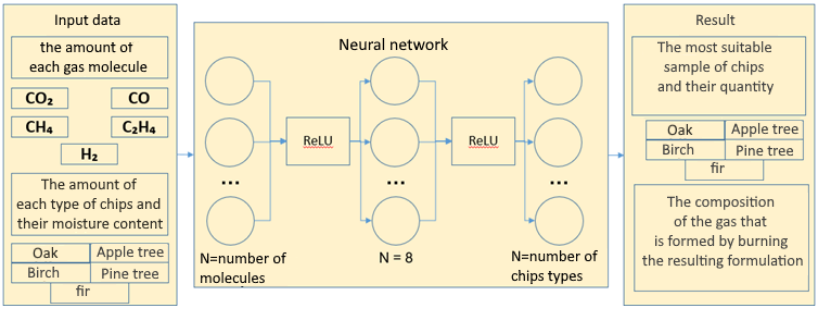

Fig. 2. The architecture of the neural model.

To develop a neural network for a sawdust gas generator set, it is important to determine the input data that will be used for training and network operation.

1. Dimensions and characteristics of the gas generator:
   - The volume of the gas generator is not more than 5.6 tons;
   - Reactor temperature from 1300℃-1700℃.
2. Properties of sawdust:
   - Moisture content of sawdust (Table 1).

Table 1. The effect of moisture content of wood chips on the release of pyrolysis gas.

Type of sawdust	The amount of gas released during the combustion of 1 kg of sawdust at humidity, m^3						
	10%	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**
Oak	3.65	3.12	2.61	2.14	1.64	1.12	0.77
Apple Tree	3.54	3.06	2.58	2.06	1.58	1.01	0.76
Birch tree	3.48	2.96	2.52	2	1.53	0.96	0.75
Pine	3.41	2.87	2.45	1.91	1.47	0.97	0.74
Spruce	3.34	2.74	2.36	1.85	1.31	0.98	0.73

3. Other auxiliary data

are the results of the analysis of the composition of the generated gas (Table 2).

When developing a neural network, it is necessary to determine the most relevant input parameters that will best represent the processes in a sawdust gas generator.

Table 2. Composition of gases formed during pyrolysis of wood.

Components of gases	**CO_2**	**CO**	**CH_4**	**C_2H_4**	**H_2**	**CO_2**	**CO**
Oak	50	29.6	15.6	1.8	3	50	29.6
Apple Tree	49.5	29	16.9	1.6	3	49.5	29
Birch tree	49	28.4	18.2	1.4	3	49	28.4
Pine	48.5	28.5	18	1	3	48.5	28.5
Spruce	48	28	19	1	4	48	28

The above data was grouped by tree type and written to a file in JSON format. As an example, the JSON object of the oak tree is shown in Figure 3.


```
"oak": {
  "humidity": {
    "10": 3.65,
    "15": 3.12,
    "20": 2.61,
    "25": 2.14,
    "30": 1.64,
    "35": 1.12,
    "40": 0.77
  },
  "structure": {
    "CO2": 50,
    "CO": 29.6,
    "CH4": 15.6,
    "C2H4": 1.8,
    "H2": 3
  }
},
```


Fig. 3. The JSON object of the oak tree.

After collecting the necessary data, it is required to assemble a dataset that will be used in training and testing the neural network. The dataset should consist of records with input data and the result.

The input data for the neural network is a set of values in the form of the percentage gas concentration required to obtain, the total volume of gas, as well as the amount and humidity of sawdust of a particular type of tree.

The result of the neural network will be the amount of each type of sawdust, after burning which the desired gas concentration will be obtained.

A function was written to obtain data for training and testing a neural network. It generates pseudo-random input values (Figure 4).

```
needStructure = {
    "CO2": random.randint( a: 48,  b: 240),
    "CO": random.randint( a: 28,  b: 140),
    "CH4": random.randint( a: 19,  b: 95),
    "C2H4": random.randint( a: 1,  b: 5),
    "H2": random.randint( a: 4,  b: 20)
}
hasTree = {
    "oak": {
        "count": random.randint( a: 0,  b: 5),
        "humidity": random.randint( a: 10,  b: 40),
    },
    "appletree": {
        "count": random.randint( a: 0,  b: 5),
        "humidity": random.randint( a: 10,  b: 40),
    },
    "birch": {
        "count": random.randint( a: 0,  b: 5),
        "humidity": random.randint( a: 10,  b: 40),
    },
```

Fig. 4. Code for generating input values randomly.

Then, the generated dataset is written to two files: "testDataset.json" and "trainDataset.json". An example of how dataset generation works is shown in Figure 5.


```
"X": {
    "needStructure": {"CO2": 24,"CO": 36,"CH4": 15,"C2H4": 20,"H2": 5
    },
    "hasTree": {
        "oak": {"count": 5,"humidity": 27},
        "appletree": {"count": 0,"humidity": 11},
        "birch": {"count": 4,"humidity": 32},
        "pine": {"count": 2,"humidity": 18},
        "fir": {"count": 2,"humidity": 40}
    },
    "allVolume": 210.162
},
"Y": {
    "perfectCombine": {"oak": 3,"appletree": 0,"birch": 0,"pine": 2,"fir": 0},
    "structure": {"CO2": 61.033,"CO": 87.16,"CH4": 23.202,"C2H4": 51.538,"H2": 13.201}
}
```


Fig. 5. Example of data from a dataset.

The entire amount of data cannot be passed through a neural network at once. Therefore, the epochs are divided into a number of training facilities. There are 10 such objects in the implemented neural model.

The initial number of training epochs is 1000.

In each epoch, the neural network is trained and the final learning result is saved. The learning results are a calculated graph. Part of the neural network learning function and the result output in the terminal are shown in Figure 6 and Figure 7.

```
# Model Training
for epoch in range(num_epochs):
    for (dataX, resultY) in train_loader:
        dataX = dataX.to(device)
        resultY = resultY.to(dtype=torch.long).to(device)

        # Step forward
        outputs = model(dataX)
        loss = criterion(outputs, resultY)

        # Step back and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch + 1) % 100 == 0:
        print(f'Epoch" [{epoch + 1}/{num_epochs}], Loss: {loss.item():.4f}')
print(f'The Final Loss: {loss.item():.4f}')
```

Fig. 6. Neural network learning function.

```
Epoch" [0/1000], Loss: 96.3300
Epoch" [100/1000], Loss: 80.3600
Epoch" [200/1000], Loss: 84.5400
Epoch" [300/1000], Loss: 80.0900
Epoch" [400/1000], Loss: 92.3600
Epoch" [500/1000], Loss: 64.7600
Epoch" [600/1000], Loss: 60.0000
Epoch" [700/1000], Loss: 45.1400
Epoch" [800/1000], Loss: 59.3800
Epoch" [900/1000], Loss: 17.7800
Epoch" [1000/1000], Loss: 19.2500
```

Fig. 7. Output of the neural network learning function.

After launching the program, the user must enter the required gas concentration, the amount of sawdust available and their humidity. These inputs are sent to a prepared neural

network to obtain the most appropriate combination of sawdust types. An example of how the program works is shown in Figure 8.

```
Enter how much interest is required CO2: 32
Enter how much interest is required CO: 28
Enter how much interest is required CH4: 17
Enter how much interest is required C2H4: 16
Enter how much interest is required H2: 10
The entered percentage exceeds 7.0
Enter how much interest is required H2: 7
Enter the gas volume: 214
Enter the quantity oak (0-5): 0
Enter the quantity appletree (0-5): 3
Enter the humidity appletree (10-40): 17
Enter the quantity birch (0-5): 2
Enter the humidity birch (10-40): 28
Enter the quantity pine (0-5): 4
Enter the humidity pine (10-40): 24
Enter the quantity fir (0-5): 3
Enter the humidity fir (10-40): 28
Required as a percentage
{'C2H4': 16.0, 'CH4': 17.0, 'CO': 28.0, 'CO2': 32.0, 'H2': 7.0}
It is necessary in quantity
{'C2H4': 34.24, 'CH4': 36.38, 'CO': 59.92, 'CO2': 68.48, 'H2': 14.98}
The resulting structure
{'CO2': 50.657, 'CO': 49.824, 'CH4': 43.625, 'C2H4': 43.926, 'H2': 13.458}
Answer
{'oak': 5, 'appletree': 0, 'birch': 0, 'pine': 0, 'fir': 0}
```

Fig. 8. An example of how the program works.

4 Conclusion

Wood chip gas generating plants have significant potential for various industrial and commercial applications. They can be successfully used to provide energy in remote areas, as well as in agriculture, forestry and other industries where the availability of traditional energy sources is limited.

The use of a neural network control system for gas generator sets makes it possible to increase the efficiency of installations, reduce the cost of their operation and ensure more reliable and safe operation of equipment.

Working out the optimal fuel recipe for gas generator sets can be a complex and time-consuming process. A neural network system can automate this process and reduce the time and effort spent on searching and optimizing a fuel recipe.

Thus, a study and analysis of wood chip gas generator sets were conducted, which allowed us to obtain valuable information about the capabilities and efficiency of this type of energy systems, and a neural network model was developed for preparing a fuel recipe for gas generator sets. It forms a list and the number of chips of various types for the target value of the emission of gases generated during their combustion.

This study was funded by the Ministry of Science and Higher Education of the Russian Federation «PRIORITY 2030» (National Project «Science and University»)

References

1. E.S. Kulakova, A.M. Safarov, Journal of Physics: Conference Series **2373(8)** 082001 (2022)
2. E.S. Kulakova, D.A. Nuykin, AIP Conference Proceedings **2467** 030014 (2022)
3. E. Kulakova, E. Muravyova, Sensors **23(20)** 8645 (2023)
4. E. Kulakova, E. Muravyova, Nanotechnologies in Construction **14(1)**, 62-69 (2022)

5. N. Kodama, International Journal of Innovative Science and Research Technology **6**, 159-171 (2021)
6. N. Kodama, International Journal of Innovative Science and Research Technology **8**, 114-121 (2023)
7. N. Kodama, International Journal of Innovative Science and Research Technology **8**, 1285-1297 (2023)
8. N. Kodama, International Journal of Innovative Science and Research Technology **8**, 356-360 (2023)
9. A. Iwamura, T. Itoh, M. Sakano, S. Sakai, S. Kuribayashi, Tenth International Conference on Cold Fusion, 435-446 (2003)
10. N. Kodama, International Journal of Innovative Science and Research Technology **7**, 1056-1065 (2022)
11. G.H. Abd-Alla, Energy Conservation and Management **43(8)**, 1027-1042 (2022)
12. K. Ali, Q. Mohamad, Energy Procedia **110**, 459-464 (2017)
13. D.S. Domenico, D.O. Alex, R.S. José, Journal of the Brazilian Society of Mechanical Sciences and Engineering **39(6)**, 1919-1927 (2017)
14. U. Hassan, L. Upahi, M. Ibrahim, Journal of Sciences **5(1)**, 271-277 (2021)
15. A.E. Martin, J.N. Gerge, G.R. Fred, Journal of the Air Pollution Control Association **5(2)**, 103-108 (2012)
16. F. Mazzotti, M. Rochford, J. Vinci, J. Brian, K. Jennifer, C. Dove, K. Sommers, Southeastern Naturalist **15**, 63-74 (2016)
17. A. Biersack, American Anthropologist **101**, 68 - 87 (1999).
18. S. Lapp, T. Rhinehart, L. Freeland-Haynes, J. Khilnani, A. Syunkova, J. Kitzes, Methods in Ecology and Evolution **14**, 2321-2328 (2023)
19. J. Nourisa, B. Zeller-Plumhoff, R. Willumeit, Practice and Experience **52**, 7718 (2022)