Лабораторная работа 1

Формальный искусственный нейрон. Обучение по правилу Хебба.

Цель работы: приобретение и закрепление знаний и получение практических навыков работы с простейшими нейронными сетями, для обучения которых используется алгоритм Хебба.

Задание

1. Разработать формальный нейрон для распознавания двух первых букв Вашей фамилии и имени. Размер изображения не менее 25 пикселей. Представление сигнала – биполярное.
2. Обучить нейрон по правилу Хебба, при этом обучающая выборка должна содержать не менее двух эталонных изображений.
3. Разработать программное приложение под ОС Windows, использую любой высокоуровневый язык программирования, позволяющее обучать по правилу Хебба и использовать разработанный нейрон.
4. Подготовить отчет о проделанной работе с изложением основных этапов выполнения лабораторной работы.

Выполнение

1) Разработаем приложение (PyQt + numpy)

Код сетки для заполнения данных:

```
class LetterGrid(QWidget):
    def __init__(self, size=5):
        super().__init__()
        self.size = size
        self.grid = np.zeros((size, size), dtype=int)
        self.setMinimumSize(200, 200)

    def paintEvent(self, event):
        painter = QPainter(self)
        painter.setRenderHint(QPainter.Antialiasing)

        cell_size = min(self.width(), self.height()) / self.size

        # Рисуем сетку
        painter.setPen(QPen(QColor(100, 100, 100), 1))
        for i in range(self.size + 1):
            painter.drawLine(i * cell_size, 0, i * cell_size, self.size * cell_size)
            painter.drawLine(0, i * cell_size, self.size * cell_size, i * cell_size)

        # Рисуем заполненные ячейки
        for i in range(self.size):
            for j in range(self.size):
                if self.grid[i, j] == 1:
                    painter.fillRect(j * cell_size + 1, i * cell_size + 1,
                                     cell_size - 2, cell_size - 2, QColor(0, 100,
200))

    def mousePressEvent(self, event):
        cell_size = min(self.width(), self.height()) / self.size
        col = int(event.x() / cell_size)
```

```
        row = int(event.y() / cell_size)

        if 0 <= row < self.size and 0 <= col < self.size:
            self.grid[row, col] = 1 if self.grid[row, col] == 0 else 0
            self.update()

    def clear_grid(self):
        self.grid = np.zeros((self.size, self.size), dtype=int)
        self.update()

    def get_bipolar_vector(self):
        return (self.grid.flatten() * 2 - 1).astype(int)

    def set_from_vector(self, vector):
        vector = (vector + 1) // 2
        self.grid = vector.reshape((self.size, self.size))
        self.update()
```

Создается сетка 5 на 5 для отображения и редактирования.

Код формального нейрона:

```
class FormalNeuron:
    def __init__(self, input_size):
        self.weights = np.zeros(input_size + 1)
        self.input_size = input_size

    def activate(self, inputs):
        # Добавление единицы для учета смещения (bias)
        extended_inputs = np.append(inputs, 1)
        # Вычисление взвешенной суммы
        s = np.dot(self.weights, extended_inputs)
        # Пороговая функция активации
        return 1 if s > 0 else -1

    def train_hebb(self, training_set, max_epochs=100):
        for epoch in range(max_epochs):
            weights_changed = False  # Флаг изменения весов

            # Проход по всем примерам обучающей выборки
            for inputs, target in training_set:
                # Добавление смещения к входному вектору
                extended_inputs = np.append(inputs, 1)
                # Получение выхода нейрона
                output = self.activate(inputs)

                # Если выход не совпадает с целевым значением
                if output != target:
                    # Обновление весов по правилу Хебба: (дельта)w = x * d
                    self.weights += extended_inputs * target
                    weights_changed = True

            # Проверка правильности классификации после эпохи
            correct = 0
            for inputs, target in training_set:
                if self.activate(inputs) == target:
                    correct += 1

            accuracy = correct / len(training_set)

            # Если достигнута 100% точность - обучение завершено
            if accuracy == 1.0:
                return True, epoch + 1, accuracy

        # Обучение не достигло 100% точности за максимальное число эпох
        return False, max_epochs, accuracy
```

Пояснение:

__init__ - Все веса инициализируются нулями, вес смещения тоже.

activate – функция активации нейрона. Принимает: входной вектор. Возвращает: 1 если взвешенная сумма > 0, если не, то 1.

Train_hebb – Обучение нейрона. Принимает: список кортежей (вектор, целевое значение), максимальное кол-во эпох. Возвращает: success – если обучение успешно, epoch – кол-во эпох, accuracy – точность которой получилось достичь.

Остальное можно прочитать по комментариям.

Код главного приложения:

```
class MainWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Formal Neuron with Hebb Learning")
        self.setGeometry(100, 100, 1000, 700)

        self.grid_size = 5
        self.neuron = FormalNeuron(self.grid_size * self.grid_size)
        self.training_set = []

        self.init_ui()

    def init_ui(self):
        central_widget = QWidget()
        self.setCentralWidget(central_widget)

        main_layout = QHBoxLayout()
        central_widget.setLayout(main_layout)

        # Левая панель - создание и обучение
        left_panel = QVBoxLayout()

        # Группа для фамилии
        surname_group = QGroupBox("Буква ФАМИЛИИ (Класс +1)")
        surname_layout = QVBoxLayout()
        self.surname_grid = LetterGrid(self.grid_size)
        surname_layout.addWidget(self.surname_grid)

        surname_btns = QHBoxLayout()
        clear_surname_btn = QPushButton("Очистить")
        clear_surname_btn.clicked.connect(self.surname_grid.clear_grid)
        surname_btns.addWidget(clear_surname_btn)

        add_surname_btn = QPushButton("Добавить в обучение")
        add_surname_btn.clicked.connect(lambda: self.add_to_training_set(1))
        surname_btns.addWidget(add_surname_btn)

        surname_layout.addLayout(surname_btns)
        surname_group.setLayout(surname_layout)
        left_panel.addWidget(surname_group)

        # Группа для имени
        name_group = QGroupBox("Буква ИМЕНИ (Класс -1)")
        name_layout = QVBoxLayout()
        self.name_grid = LetterGrid(self.grid_size)
        name_layout.addWidget(self.name_grid)
```

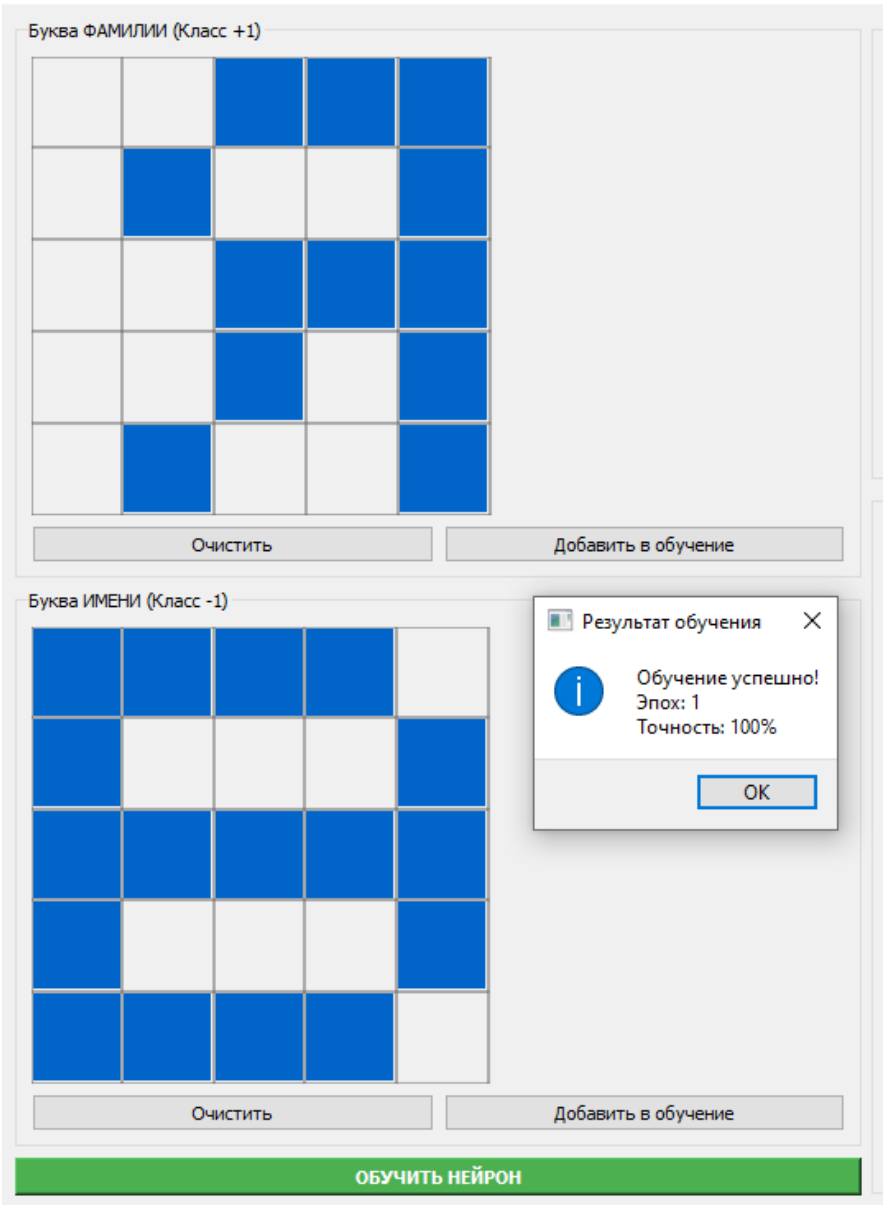
```
        name_btns = QHBoxLayout()
        clear_name_btn = QPushButton("Очистить")
        clear_name_btn.clicked.connect(self.name_grid.clear_grid)
        name_btns.addWidget(clear_name_btn)

        add_name_btn = QPushButton("Добавить в обучение")
        add_name_btn.clicked.connect(lambda: self.add_to_training_set(-1))
        name_btns.addWidget(add_name_btn)

        name_layout.addLayout(name_btns)
        name_group.setLayout(name_layout)
        left_panel.addWidget(name_group)

        # Кнопка обучения
        train_btn = QPushButton("ОБУЧИТЬ НЕЙРОН")
        train_btn.setStyleSheet("QPushButton { background-color: #4CAF50; color:
white; font-weight: bold; }")
        train_btn.clicked.connect(self.train_neuron)
        left_panel.addWidget(train_btn)

        # Правая панель - тестирование и информация
        right_panel = QVBoxLayout()

        # Тестирование
        test_group = QGroupBox("ТЕСТИРОВАНИЕ")
        test_layout = QVBoxLayout()
        self.test_grid = LetterGrid(self.grid_size)
        test_layout.addWidget(self.test_grid)

        test_btns = QHBoxLayout()
        clear_test_btn = QPushButton("Очистить")
        clear_test_btn.clicked.connect(self.test_grid.clear_grid)
        test_btns.addWidget(clear_test_btn)

        test_btn = QPushButton("Распознать")
        test_btn.clicked.connect(self.test_neuron)
        test_btn.setStyleSheet("QPushButton { background-color: #2196F3; color: white;
}")
        test_btns.addWidget(test_btn)

        test_layout.addLayout(test_btns)

        self.result_label = QLabel("Результат: Нейрон не обучен")
        self.result_label.setStyleSheet("QLabel { font-weight: bold; font-size: 14px;
}")
        test_layout.addWidget(self.result_label)

        test_group.setLayout(test_layout)
        right_panel.addWidget(test_group)

        # Информация
        info_group = QGroupBox("ИНФОРМАЦИЯ")
        info_layout = QVBoxLayout()

        self.info_text = QTextEdit()
        self.info_text.setReadOnly(True)
        self.info_text.setPlainText(
            "ИНСТРУКЦИЯ:\n"
            "1. Создайте несколько вариантов буквы фамилии (класс +1)\n"
            "2. Создайте несколько вариантов буквы имени (класс -1)\n"
            "3. Нажмите 'Обучить нейрон'\n"
            "4. Протестируйте на новых изображениях\n\n"
        )
        info_layout.addWidget(self.info_text)

        self.weights_label = QLabel("Весовые коэффициенты: Не обучено")
        info_layout.addWidget(self.weights_label)

        self.stats_label = QLabel("Статистика: Не обучено")
        info_layout.addWidget(self.stats_label)
```

```
        info_group.setLayout(info_layout)
        right_panel.addWidget(info_group)

        main_layout.addLayout(left_panel, 1)
        main_layout.addLayout(right_panel, 1)

    def add_to_training_set(self, target):
        if target == 1:
            inputs = self.surname_grid.get_bipolar_vector()
            self.training_set.append((inputs, target))
            QMessageBox.information(self, "Добавлено", "Добавлен пример класса ФАМИЛИЯ
(+1)")
        else:
            inputs = self.name_grid.get_bipolar_vector()
            self.training_set.append((inputs, target))
            QMessageBox.information(self, "Добавлено", "Добавлен пример класса ИМЯ (-
1)")

        self.update_stats()

    def update_stats(self):
        stats_text = f"Обучающая выборка: {len(self.training_set)} примеров\n"
        class1_count = sum(1 for _, target in self.training_set if target == 1)
        class2_count = len(self.training_set) - class1_count
        stats_text += f"Класс +1 (Фамилия): {class1_count}\n"
        stats_text += f"Класс -1 (Имя): {class2_count}"
        self.stats_label.setText(stats_text)

    def train_neuron(self):
        if len(self.training_set) < 2:
            QMessageBox.warning(self, "Ошибка", "Нужно хотя бы 2 примера для
обучения!")
            return

        success, epochs, accuracy = self.neuron.train_hebb(self.training_set)

        if success:
            msg = f"Обучение успешно!\nЭпох: {epochs}\nТочность: 100%"
            self.result_label.setText("Результат: Нейрон обучен ✓")
            self.result_label.setStyleSheet("QLabel { color: green; font-weight: bold;
}")
        else:
            msg = f"Обучение не завершено!\nЭпох: {epochs}\nЛучшая точность: {accuracy
* 100:.1f}%"
            self.result_label.setText("Результат: Нейрон не полностью обучен")
            self.result_label.setStyleSheet("QLabel { color: orange; font-weight:
bold; }")

        QMessageBox.information(self, "Результат обучения", msg)

        # Обновляем информацию о весах
        weights_text = "Весовые коэффициенты:\n"
        weights_text += ", ".join([f"{w:.2f}" for w in self.neuron.weights])
        self.weights_label.setText(weights_text)

    def test_neuron(self):
        if len(self.training_set) == 0:
            QMessageBox.warning(self, "Ошибка", "Сначала обучите нейрон!")
            return

        test_input = self.test_grid.get_bipolar_vector()
        result = self.neuron.activate(test_input)

        if result == 1:
            self.result_label.setText("Результат: Похоже на ФАМИЛИЮ (+1)")
            self.result_label.setStyleSheet("QLabel { color: blue; font-weight: bold;
}")
        else:
            self.result_label.setText("Результат: Похоже на ИМЯ (-1)")
            self.result_label.setStyleSheet("QLabel { color: red; font-weight: bold;
}")
```

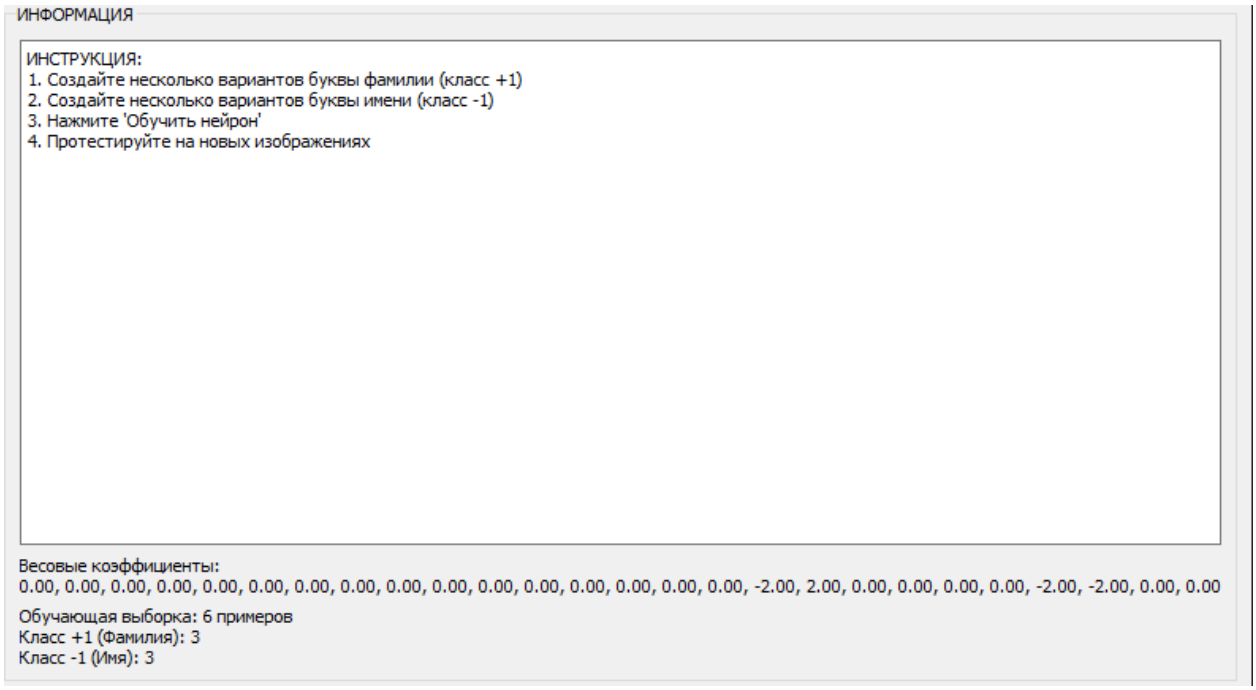
```
def main():
    app = QApplication(sys.argv)
    window = MainWindow()
    window.show()
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```

Главное окно приложения с интерфесом (кнопки и тд.).

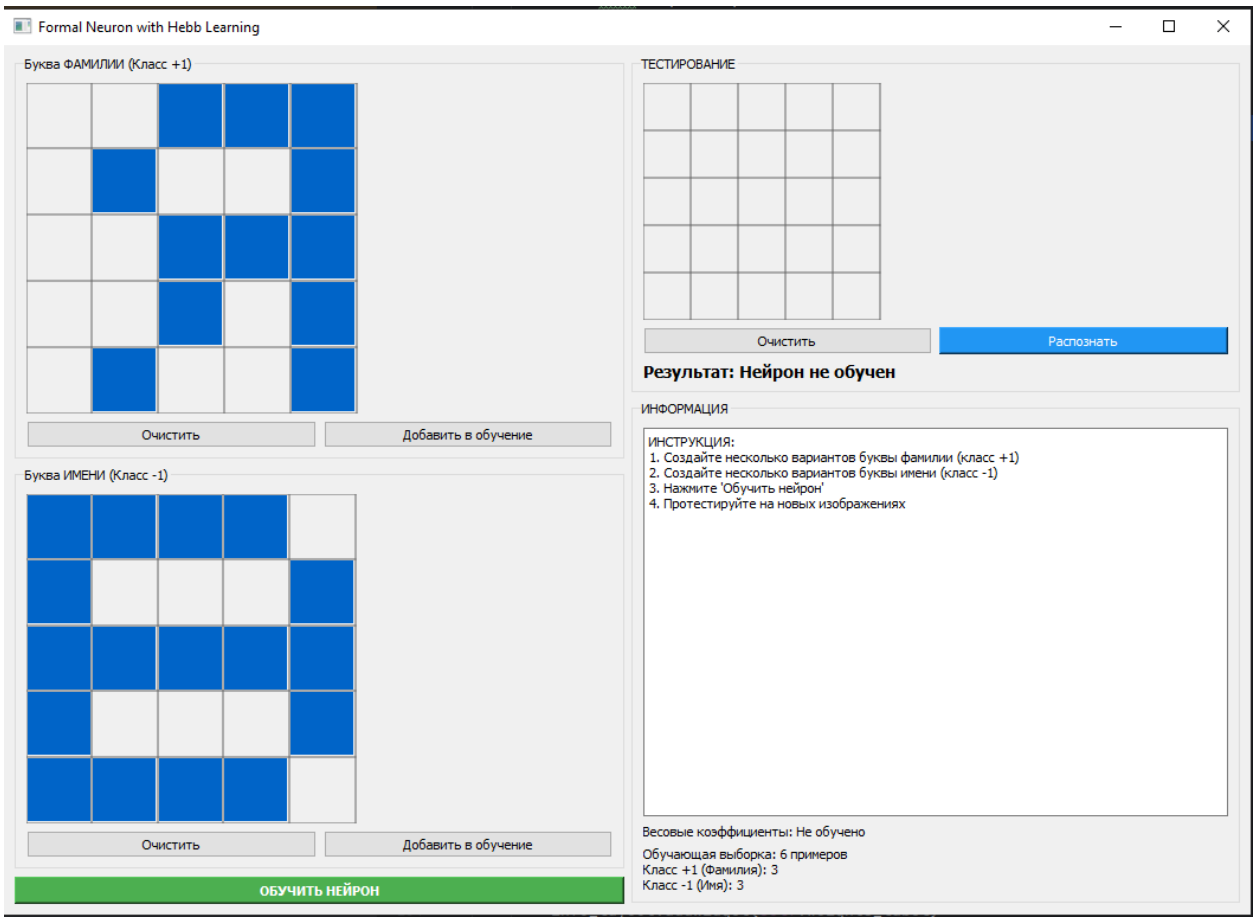
2) Обучим нейрон на трех произвольных изображениях букв (В. Я.)



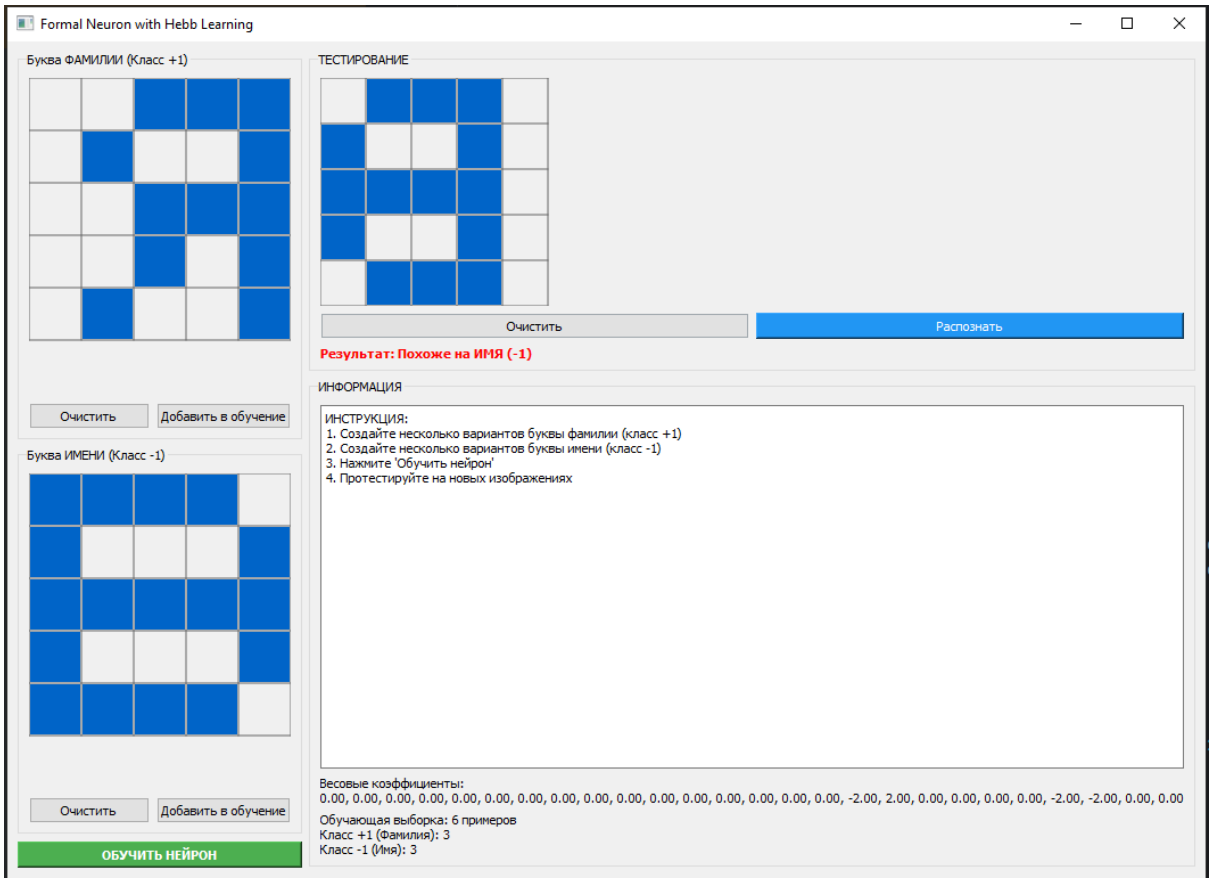
Информационная часть окна о обученном нейроне и инструкция

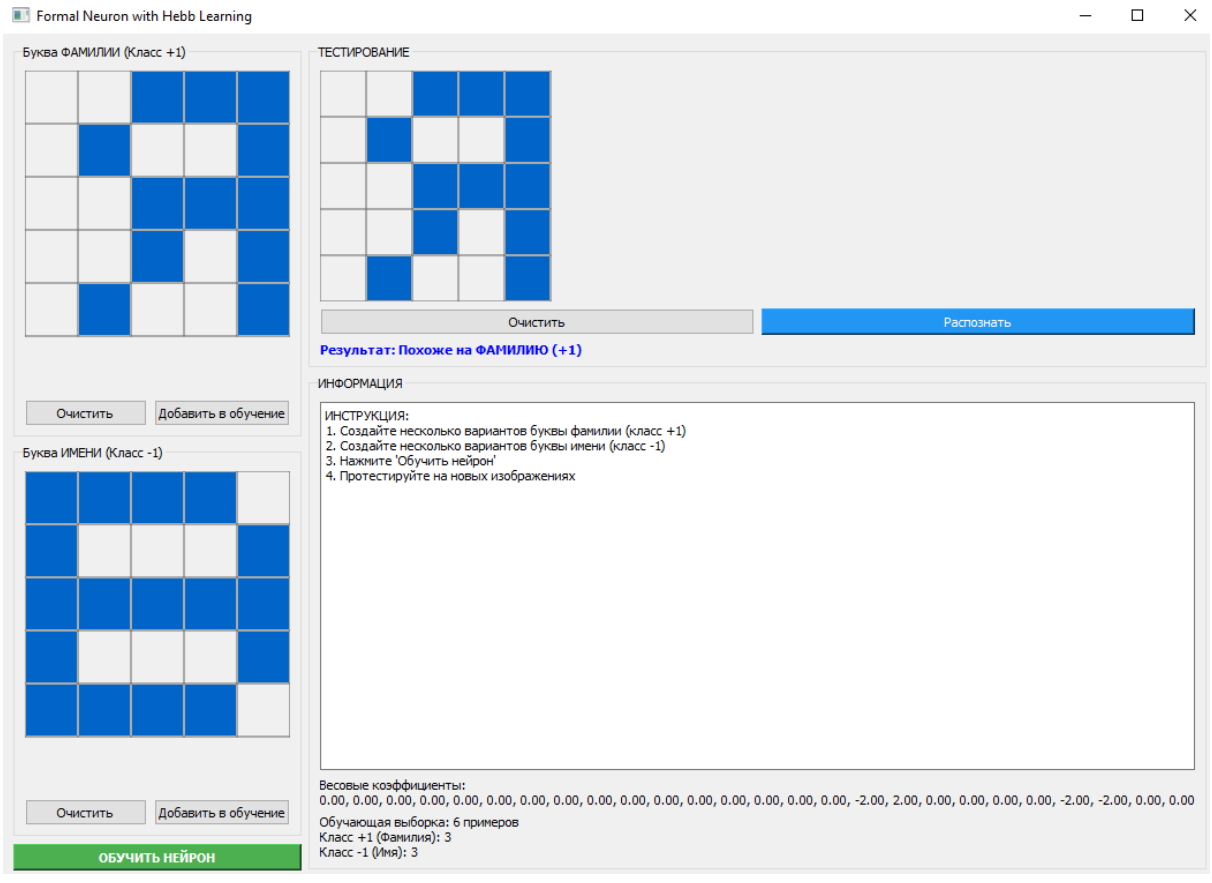


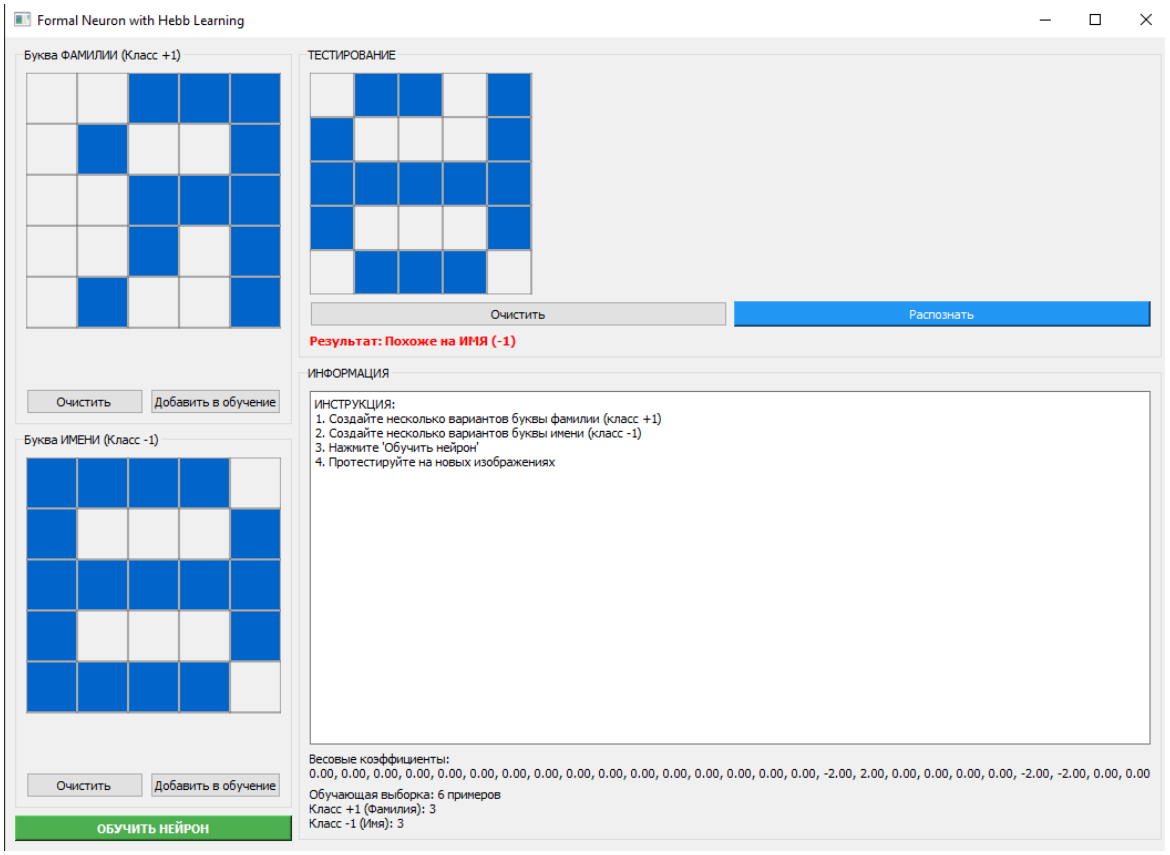
Интерфейс приложения в целом

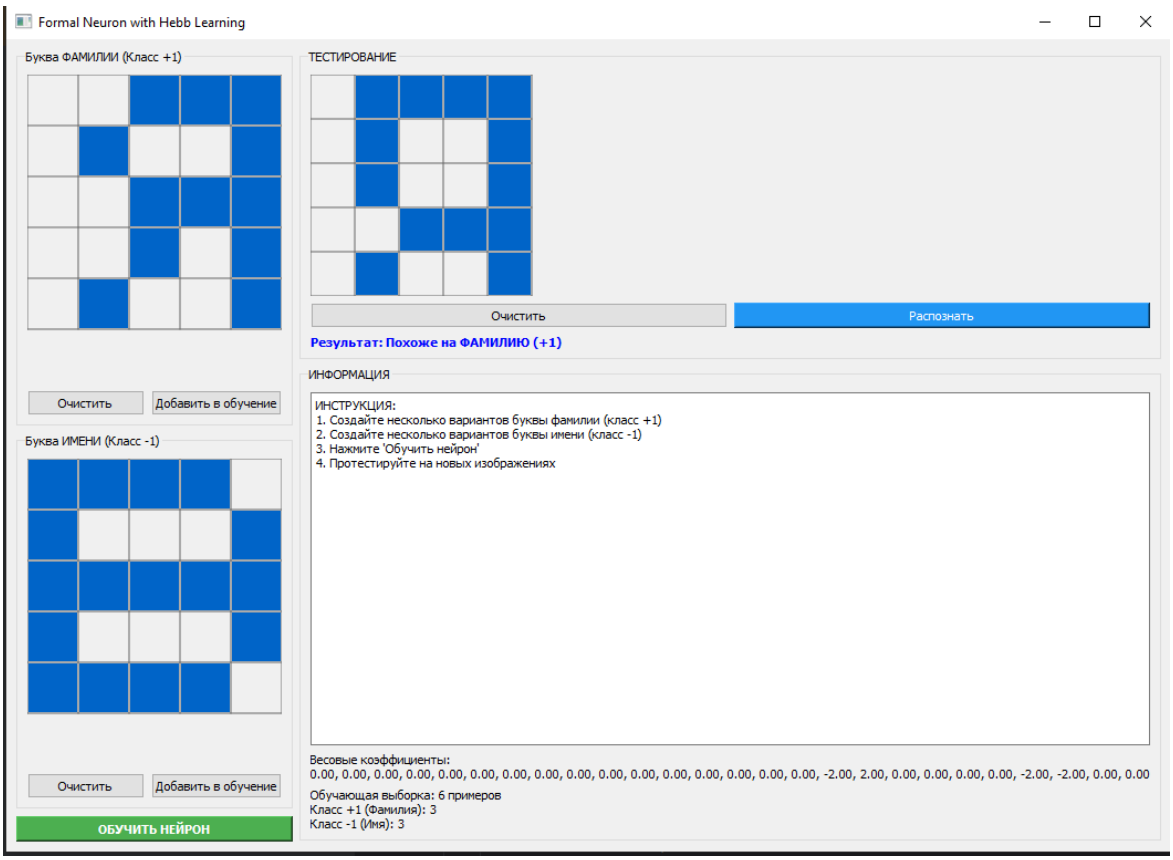


Результаты


Formal Neuron with Hebb Learning
Буква ФАМИЛИИ (Класс +1)
Очистить
Добавить в обучение
Буква ИМЕНИ (Класс -1)
Очистить
Добавить в обучение
ОБУЧИТЬ НЕЙРОН
ТЕСТИРОВАНИЕ
Очистить
Распознать
Результат: Похоже на ИМЯ (-1)
ИНФОРМАЦИЯ
ИНСТРУКЦИЯ:
1. Создайте несколько вариантов буквы фамилии (класс +1)
2. Создайте несколько вариантов буквы имени (класс -1)
3. Нажмите 'Обучить нейрон'
4. Протестируйте на новых изображениях
Весовые коэффициенты:
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, -2.00, 2.00, 0.00, 0.00, 0.00, 0.00, -2.00, -2.00, 0.00, 0.00
Обучающая выборка: 6 примеров
Класс +1 (Фамилия): 3
Класс -1 (Имя): 3


Formal Neuron with Hebb Learning
Буква ФАМИЛИИ (Класс +1)
Очистить
Добавить в обучение
Буква ИМЕНИ (Класс -1)
Очистить
Добавить в обучение
ОБУЧИТЬ НЕЙРОН
ТЕСТИРОВАНИЕ
Очистить
Распознать
Результат: Похоже на ФАМИЛИЮ (+1)
ИНФОРМАЦИЯ
ИНСТРУКЦИЯ:
1. Создайте несколько вариантов буквы фамилии (класс +1)
2. Создайте несколько вариантов буквы имени (класс -1)
3. Нажмите 'Обучить нейрон'
4. Протестируйте на новых изображениях
Весовые коэффициенты:
0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, 0.00, -2.00, 2.00, 0.00, 0.00, 0.00, 0.00, -2.00, -2.00, 0.00, 0.00
Обучающая выборка: 6 примеров
Класс +1 (Фамилия): 3
Класс -1 (Имя): 3





Как видим приложение функционирует, нейрон отрабатывает стабильно, хоть и не с самой лучшей точностью. Чтобы повысить точность можно добавить больше данных для обучения, добавить нейроны в связь, поработать с инициализацией нейронов (например, случайно).