Санкт-Петербургский государственный электротехнический университет им. В.И. Ульянова (Ленина)

РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА ДЛЯ ВИЗУАЛЬНОГО ПРОЕКТИРОВАНИЯ АРХИТЕКТУР НЕЙРОННЫХ СЕТЕЙ

Выполнил: Бахеров Даниил Владиславович, гр. 7382
Руководитель: Жукова Наталия Александровна, к.т.н., доцент
Консультант: Жангиров Тимур Рафаилович, ассистент каф. МОЭВМ, СПбГЭТУ «ЛЭТИ»

Санкт-Петербург, 2021

ЦЕЛЬ И ЗАДАЧИ

Актуальность: существующие программные средства для создания нейронных сетей

- не предназначены для людей, которые только начинают развиваться в области машинного обучения,
- имеют ограничения на количество создаваемых элементов архитектуры нейронной сети.

Цель: разработка программного средства для визуального проектирования архитектур нейронных сетей.

Задачи:

1) Сравнение аналогов

2) Определение функциональных требований

3) Разработка архитектуры программного средства

4) Разработка программного средства

СРАВНЕНИЕ АНАЛОГОВ

Аналоги	Критерии		
	1	2	3
Программное средство на C#	-	+	+
Программное средство на Visual Basic 6	+	-	-
Программное средство на Visual Basic for Application	+	+	+
Программное средство на C++	+	+	+
Нейропакет QwikNet32	+	-	-

Критерии:

1) Графическое представление нейронной сети

2) Возможность создания >10 слоев и >30 нейронов на слое

3) Возможность создания разных типов нейронных сетей

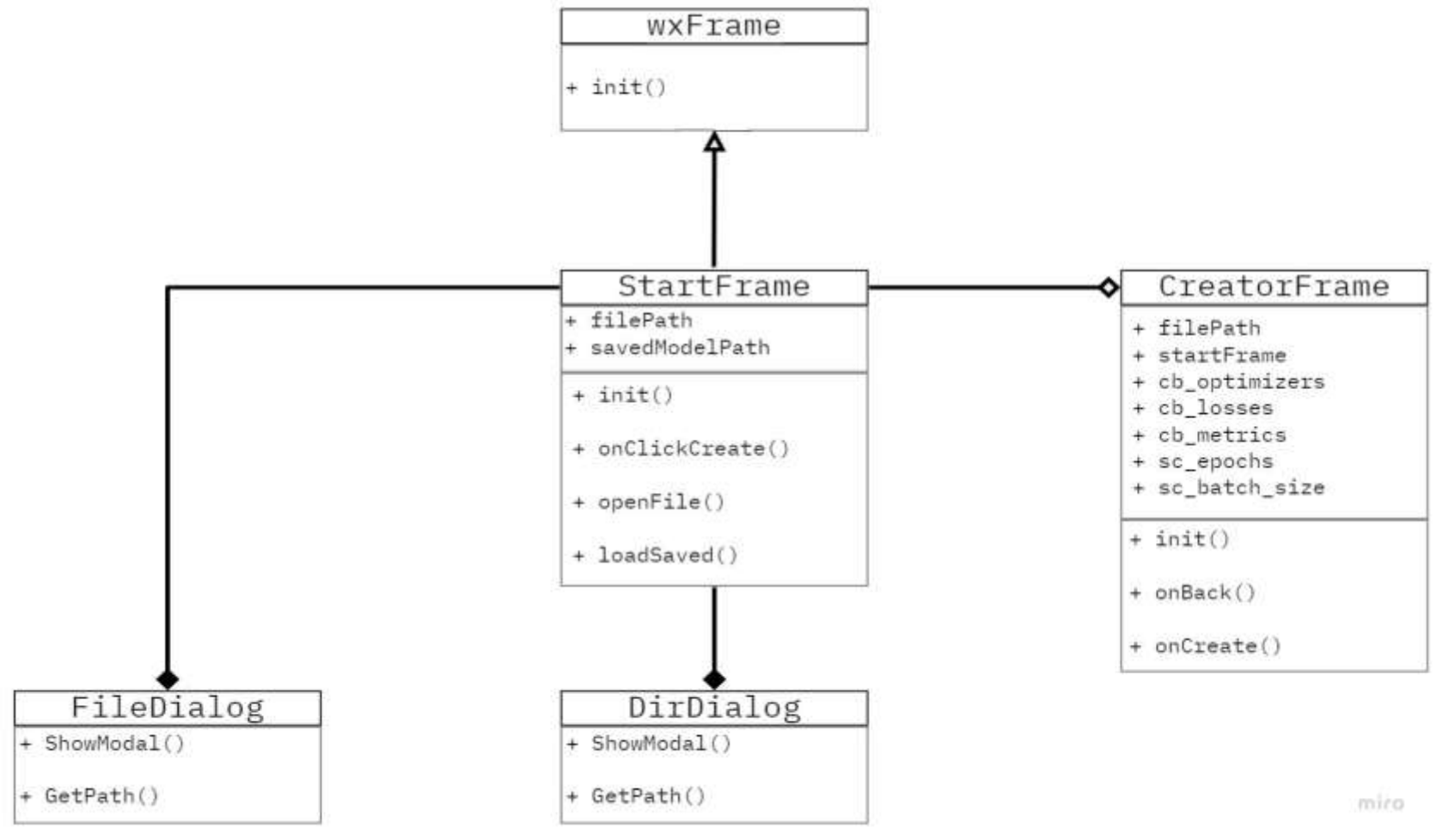
ОПРЕДЕЛЕНИЕ ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

1) Графический пользовательский интерфейс для построения нейронной сети

2) Открытый исходный код

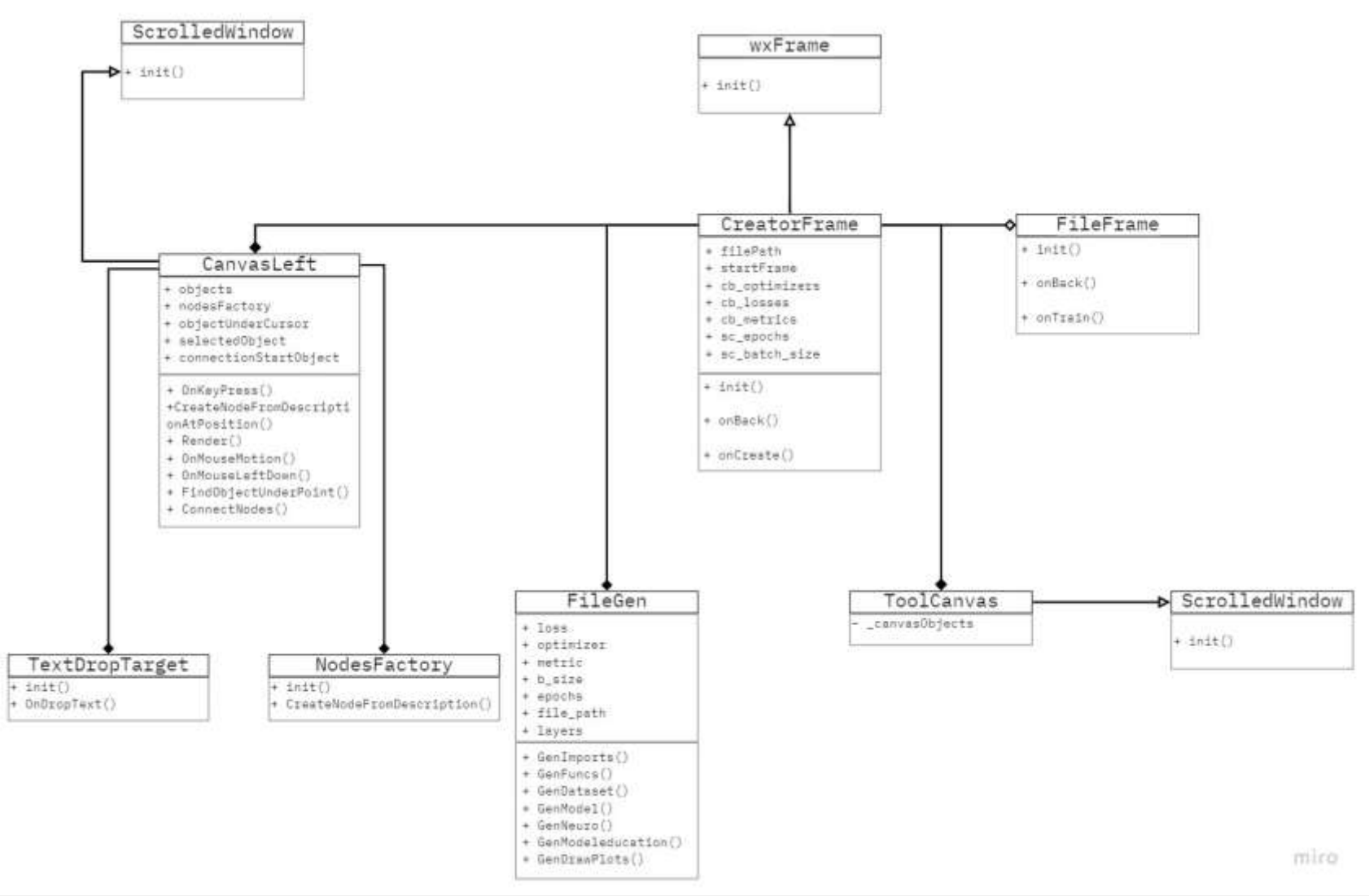
3) Проектирование различных типов нейронных сетей без ограничения на количество создаваемых элементов

РАЗРАБОТКА АРХИТЕКТУРЫ ПРОГРАММНОГО СРЕДСТВА



Uml-диаграмма стартового окна

РАЗРАБОТКА АРХИТЕКТУРЫ ПРОГРАММНОГО СРЕДСТВА



Uml-диаграмма окна создания нейронной сети

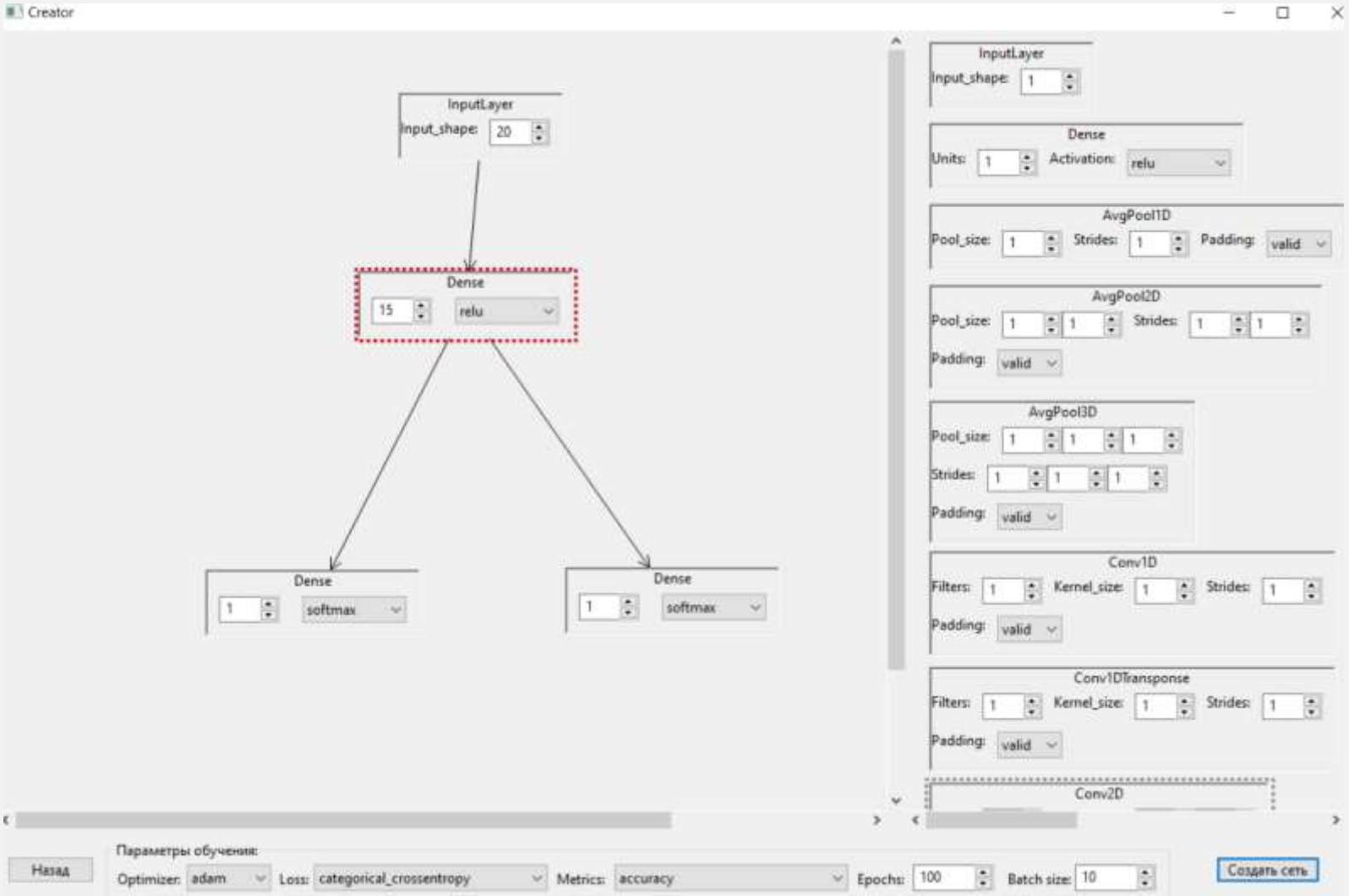
ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ

- Язык программирования Python
- Фреймворк машинного обучения Keras
- Библиотека графического интерфейса wxPython

РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

- Модуль загрузки обучающих данных
- Модуль загрузки сохраненной модели нейронной сети
- Модуль создания архитектуры нейронной сети
- Модуль генерации кода
- Модуль обучения нейронной сети
- Модуль сохранения результатов обучения

ПРИМЕР СОЗДАНИЯ АРХИТЕКТУРЫ НЕЙРОННОЙ СЕТИ



Окно создания архитектуры нейронной сети

ЗАКЛЮЧЕНИЕ

- Было проведено сравнение аналогов, в результате чего были выявлены недостатки современных решений
- На основе сравнения аналогов были определены функциональные требования к программному средству
- Были созданы uml-диаграммы классов в качестве архитектуры программного средства
- Реализовано программное средство со следующими возможностями:
  - Визуальное проектирование различных типов нейронных сетей (например рекуррентных, сверточных, нелинейных нейронных сетей) без ограничения на количество создаваемых элементов
  - Сохранение и загрузка модели нейронной сети

В качестве дальнейшего пути разработки требуется реализовать автоматическое преобразование обучающих данных, сохранение и загрузку графической архитектуры нейронной сети.

АПРОБАЦИЯ РАБОТЫ

- Бахеров Д.В., Жангиров Т.Р. Разработка архитектуры программного обеспечения для визуального проектирования архитектур нейронных сетей // Научные труды кафедры МОЭВМ. 2021. 95 с.
- Репозиторий проекта. URL: https://github.com/baheroff/neurocreator

ЗАПАСНЫЕ СЛАЙДЫ

ГЕНЕРАЦИЯ КОДА

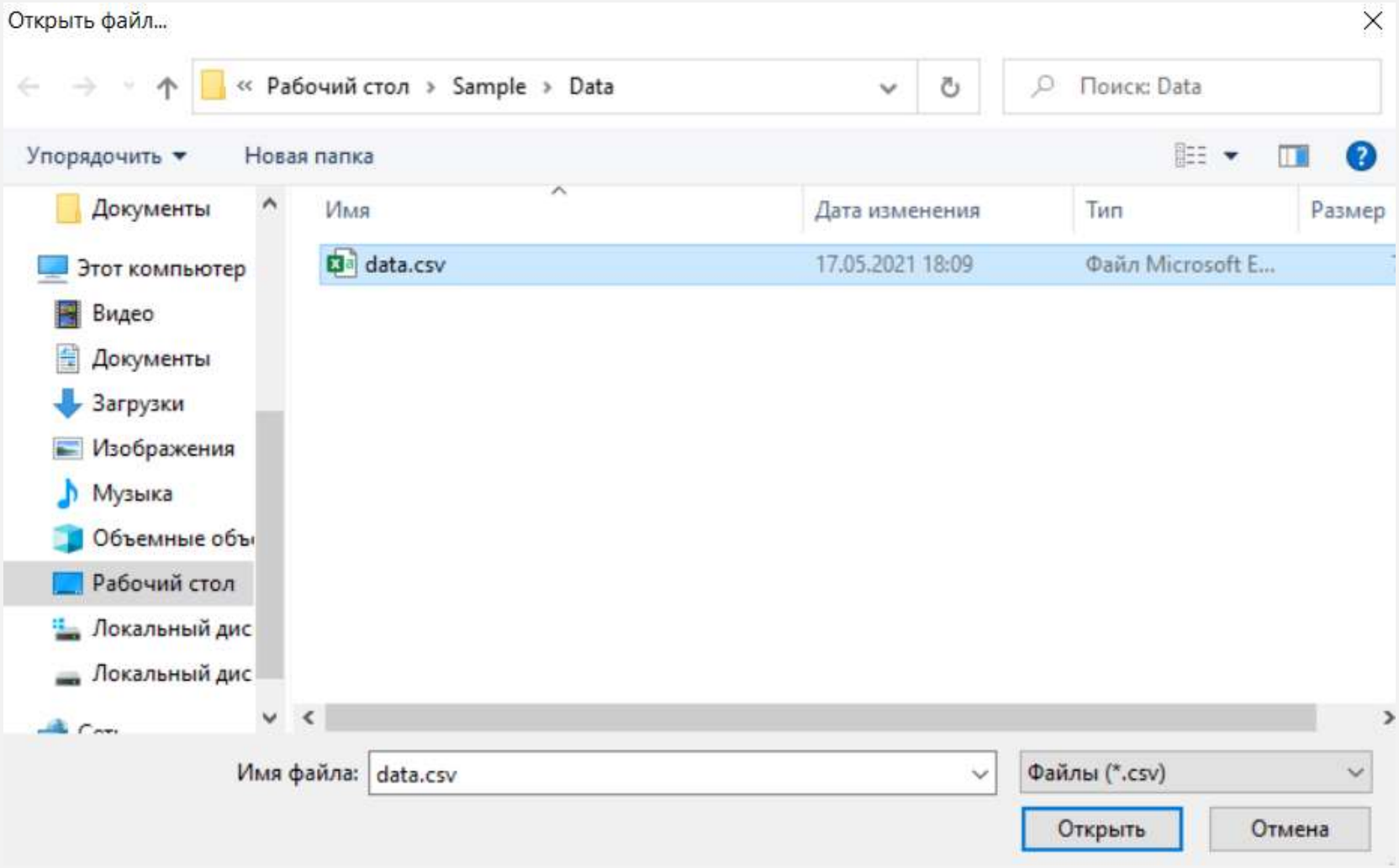
File

```
        plt.figure(1, figsize=(8, 5))
        plt.plot(loss, 'b', label='Train')
        plt.plot(v_loss, 'r', label='Validation')
        plt.title('Loss')
        plt.ylabel('Loss')
        plt.xlabel('Epochs')
        plt.legend()
        plt.show()
        plt.clf()

def plot_acc(acc, val_acc):
        plt.plot(acc, 'b', label='Train')
        plt.plot(val_acc, 'r', label='Validation')
        plt.title('Accuracy')
        plt.ylabel('Accuracy')
        plt.xlabel('Epochs')
        plt.legend()
        plt.show()
        plt.clf()

dataframe = pandas.read_csv("C:\Users\baheroff\Desktop\Sample\Data\data.csv", header=None)
dataset = dataframe.values
row_length = dataset.shape[0]
train_input_data = dataset[:row_length/2:][:,0]
test_input_data = dataset[row_length/2::][:,0]
train_output_data = dataset[:row_length/2:][:,1]
test_output_data = dataset[row_length/2::][:,1]

x0 = Input(shape=(20,))

x1 = Dense(15, activation=relu)(x0)

x2 = Dense(1, activation=softmax)(x1)

x3 = Dense(1, activation=softmax)(x1)

model = Model(inputs=[x0], outputs=[x2, x3])

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

H = model.fit(train_input_data, train_output_data, batch_size=10, epochs=100, validation_data=(test_input_data, test_output_data))

model.save('saved_model/my_model')

plot_loss(H.history['loss'], H.history['val_loss'])
plot_acc(H.history['accuracy'], H.history['val_accuracy'])
```

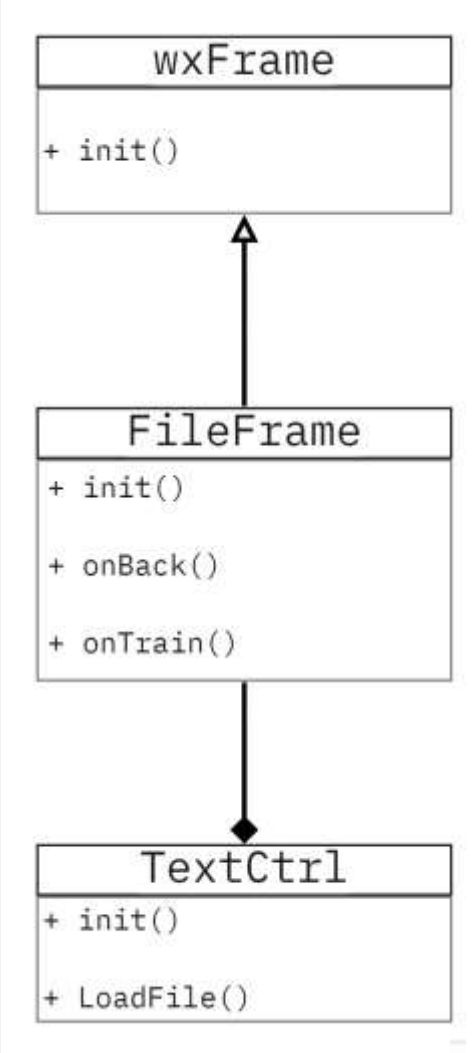
Назад

Начать обучение

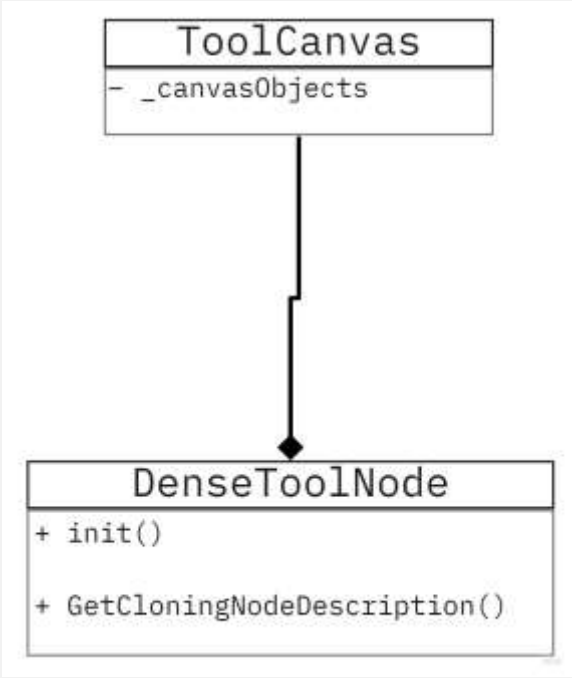
ЗАГРУЗКА ОБУЧАЮЩИХ ДАННЫХ



UML-ДИАГРАММЫ



Uml-диаграмма окна со сгенерированным файлом



Uml-диаграмма панели инструментов