Управление образования города Пензы

МКУ «Центр комплексного обслуживания и методологического обеспечения учреждений образования» г. Пензы

МБОУ «Лицей современных технологий управления № 2» г. Пензы

XXVI научно-практическая конференция школьников г. Пензы
«Я исследую мир»

«Создание арифметического тренажёра DigiStarter с использованием нейросети для детей дошкольного возраста»

Выполнил: Годов Дмитрий Алексеевич,
ученик 11д класса МБОУ ЛСТУ №2 г. Пензы,

Руководители: Адамский Сергей Сергеевич,
учитель информатики высшей категории

Голикова Ирина Александровна,
учитель информатики первой категории.

Пенза, 2021 год

✉- **440008, г. Пенза, ул. Бакунина, 115**
☎- **телефон /841-2/ 54-20-44; e-mail: school02@guoedu.ru**
http://www.lstu2.ru

Оглавление

Введение

В наше время искусственный интеллект (ИИ) используется повсеместно: анализ запросов в интернете для подбора рекламы, системы навигации для беспилотных аппаратов, программы-боты, способные обыгрывать людей в различных играх, прогнозирование и многое другое. Современные компании используют ИИ для ускорения и автоматизации своих процессов.

ИИ помогает справляться со множеством проблем, которые приходится решать современному человечеству. К таким проблемам можно отнести освоение космоса, прогнозирование природных катаклизмов и антропогенного воздействия на окружающую среду. Из-за больших объёмов данных, окружающих человека ИИ актуален как никогда, ведь он обрабатывает информацию намного быстрее и точнее человека.

Нейронные сети – это одно из направлений в разработке ИИ. Названы они так, потому что принцип их работы напоминает функционирование нервной системы. Типичные задачи для нейросетей – предсказание, классификация и распознавание. Так, их можно использовать в образовательных целях. Например, для обучения детей дошкольного возраста написанию различных символов. Игровая форма обучения позволяет заинтересовать ребёнка и удержать его внимание на более длительное время. Каждый ребёнок индивидуально обучается в своём темпе.

Цель работы: реализация программы-тренажёра с использованием нейросети, способной распознавать рукописный ввод символов ребёнка.

Задачи:

- Разработать концепт игры-тренажёра.
- Изучить возможность использования нейросетей.
- Создать программу, способную в игровой форме обучать детей написанию символов.
- Апробировать разработку на занятиях школы раннего развития детей.

Конечным результатом проекта является продукт – приложение, использующее нейросеть для распознавания рукописного цифрового ввода, способное сообщать результат распознавания в игровой форме.

Общие сведения

Искусственный интеллект(ИИ) – это система или машина, которая способна имитировать человеческое поведение для выполнения различных задач и постепенного обучаться, используя собираемую информацию. Благодаря этому компьютеры можно «научить» с помощью обработки большого объема данных и выявления в них закономерностей. Целью ИИ является расширение человеческих умений и возможностей.

Нейронные сети - математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей. Они не программируются в привычном смысле этого слова, а обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что в случае успешного обучения сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

Алгоритм работы нейронных сетей:

1. поступление данных на входной слой нейронов
2. информация передаётся с помощью связей между нейронами (по - научному синапсы) следующему слою, причём каждый синапс имеет собственный коэффициент веса, а любой следующий нейрон способен иметь несколько входящих синапсов
3. данные, полученные следующим нейроном, — это сумма всех данных для нейронных сетей, которые перемножены на коэффициенты весов (каждый на свой)
4. полученное в итоге значение подставляется в функцию активации, в результате чего происходит формирование выходной информации
5. информация передаётся дальше до тех пор, пока не дойдёт до конечного выхода

Как говорил Альберт Эйнштейн – «*Игра — высшая форма исследования»*. С ним согласна и современная педагогика. Игровые методы обучения – одни из самых увлекательных и эффективных, особенно, когда речь идёт об обучении детей дошкольного возраста.

Концепт игры

На просторах сети Интернет были найдены арифметические тренажёры, вот наиболее удачные из них:

- Тренажёр https://kids-smart.ru/exercises/1-class/reshi-primer генерирует задания, озвучивает их, но ответ просит вводить в виде текста при помощи клавиатуры.
- Тренажёр https://matematika.club/app/#11100 предлагает генерируемые задания с возможностью повышения сложности, но не озвучивает задания и не предлагает рукописный ввод.

Ни один из тренажёров не предлагает рукописного ввода цифр, хотя отлично помогают изучить арифметические операции. Следовательно, функция распознавания символов цифр в подобном проекте будет являться отличительной возможность и его новизной.

Геймплей игры основывается не механике распознавании нейросетью нарисованного в рабочем поле программы изображения. Для реализации были выбраны несколько режимов игры:

1. Нарисуй цифру
2. Реши арифметический пример и нарисуй ответ

Первый режим предлагает нарисовать цифру на чистом холсте. Второй режим предлагает простой пример на сложение или вычитание, в результате которого должно получиться решение в виде одной цифры, которую и нужно нарисовать.

Все задания и результаты озвучиваются:

- «Нарисуй цифру – четыре!»
- «Попробуй ещё раз!»
- «Молодец!»
- «Напиши, сколько будет – пять минус два?»

Цифровой холст для рисования обладает функцией очистки на случай, если что-то будет нарисовано ошибочно.

Реализация

Для использования возможностей нейросетей в Python использована библиотека TensorFlow. TensorFlow — открытая программная библиотека для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия. Реализованный фрагмент кода для распознавания цифр представлен в приложении 1.

Нейросеть была обучена на библиотеке с изображениями цифр, после чего была получена модель обучения в формате json, которую можно использовать при инициализации программы.

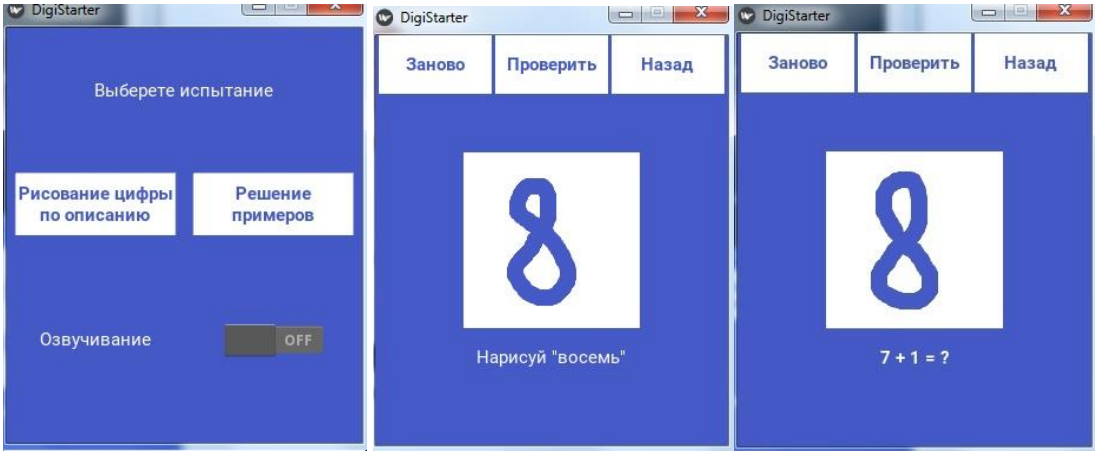
Для работы с изображениями использована библиотека Pillow. Для воспроизведения звуковых оповещений используется winsound – это модуль в Python, который используется для доступа к основному механизму воспроизведения звука операционной системы Windows.

```
f_n = randrange(1, 9)
s_n = randrange(1, 10 - f_n)
MathScreen.c = f_n + s_n
winsound.PlaySound(os.path.join('Math', 'how.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
sleep(1.2)
winsound.PlaySound(os.path.join('Numbers', a[f_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
sleep(0.8)
winsound.PlaySound(os.path.join('Math', '+.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
sleep(0.7)
winsound.PlaySound(os.path.join('Numbers', a[s_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
sleep(0.8)
```

В приведённом выше фрагменте кода генерируется озвучивание фразы «Сколько будет 4 плюс 2?» из заранее записанных фрагментов. Числа генерируется случайным образом так, чтобы ответ был в виде одной цифры.

Для использования графических возможностей операционной системы используется библиотека kivy.app. Пример реализации одной из версий приложения представлен в приложении 2.

Пример реализации такого приложения представлен на рисунке 1, 2, 3.



Рисунки 1, 2, пример реализации проекта

Заключение

Данный проект был апробирован на занятиях школы раннего развития в МБОУ ЛСТУ №2 г. Пензы. Детям дошкольного возраста было предложено при помощи стилуса на планшете нарисовать цифры по заданиям. Дети с восторгом выполняли задания, проверяя свои знания и умения. Позитивно было воспринято и звуковое оформление. Все дети смогли справиться с заданиями.

В итоге была получена работающая нейронная сеть, корректно определяющая рукописные цифры. Все задачи были выполнены, а цель – достигнута. Создан продукт – тренажёр для детей дошкольного возраста, который выполняет корректно свои функции.

В перспективе планируется добавить поддержку распознавания букв, расширив образовательные возможности приложения.

План реализации проекта

Мероприятие	**Сроки**
Изучение отзывов о приложении	Зима 2022
Доработка функций и возможностей	Весна 2022
Распространение разработки через сеть Интернет	Весна 2022

Использованные источники

1. Википедия: Нейронная сеть [Электронный ресурс], URL: https://ru.wikipedia.org/wiki/Нейронная_сеть (Дата обращения: 28.12.2021)
2. Государство. Бизнес. Технологии: Нейросети. [Электронный ресурс], URL: https://www.tadviser.ru/index.php/Статья:Нейросети_(нейронный_сети) (Дата обращения: 28.12.2021)
3. Otus. Онлайн-образование: Типы нейросетей. [Электронный ресурс], URL: https://otus.ru/nest/post/1263/ (Дата обращения: 28.12.2021)
4. Хабр: нечувствительные к весам нейронные сети(WANN). [Электронный ресурс], URL: https://habr.com/ru/post/465369/ (Дата обращения: 28.12.2021)
5. PythonRu: Руководство по использованию NUMPHY для работы смассивами [Электронный ресурс], URL: https://pythonru.com/biblioteki/rukovodstvo-po-ispolzovaniju-python-biblioteki-numpy (Дата обращения: 28.12.2021)
6. TesorFlow: Последовательная модель. [Электронный ресурс], URL: https://www.tensorflow.org/guide/keras/sequential_model (Дата обращения: 28.12.2021)
7. Python 3: Pillow обработка изображений в Python на примерах. [Электронный ресурс], URL: https://python-scripts.com/pillow (Дата обращения: 28.12.2021)

Приложение 1, фрагмент кода с использованием Tensorflow

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import numpy as np
from tensorflow import keras
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.datasets import mnist
from PIL import Image
#загрузка выборок из базы mnist
#x_train, x_test - тренировочные и тестовые входные значения для нейронной сети
#y_train, y_test - тренировочные и тестовые выходные значения
(x_train, y_train), (x_test, y_test) = mnist.load_data()
#преобразование обучающих и тестовых данных к числовым значениям в диапазоне [0, 1]
x_train = x_train/255
x_test = x_test/255
#преобразование выходных значений в вектора
y_train_vectors = keras.utils.to_categorical(y_train, 10)
y_test_vectors = keras.utils.to_categorical(y_test, 10)
#создание модели нейронной сети
model = keras.Sequential([Flatten(input_shape = (28, 28, 1)),#первый слой, содержащий изображение цифры 28x28 пикселей
                  Dense(196, activation = 'relu'),#второй слой со 128 нейронами
                  Dense(10, activation = 'softmax')#выходной слой с 10 нейронами
])
#компиляция нейронной сети
model.compile(optimizer = 'adam',#оптимизация
          loss = 'categorical_crossentropy',#критерий качества
          metrics = (['accuracy'])#процент правильно определенных цифр
          )
#процесс обучения нейронной сети
model.fit(x_train, y_train_vectors, epochs = 10, batch_size = 25, validation_split = 0.2)
def test_model(file_name):
          #тестирование модели
          img = Image.open(os.path.join('Pictures', file_name+'.png')).convert('L').resize((28, 28))
          result = model.predict(np.expand_dims(255 - np.array(img), axis = 0))
          img.close()
          return 'Определённая цифра: ' + str(np.argmax(result))
while True:
          if input('Хотите протестировать нейросеть?(y/n) ').lower() == 'y':
                    print(test_model(input('Введите имя файла: ')))
          else:
                    print('Окей, до связи')
                    break
```

Приложение 2, фрагмент кода с примером реализации графического интерфейса.

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
import numpy as np
from PIL import Image
from random import randrange, choice
from time import sleep
import winsound
from tensorflow.keras.models import model_from_json
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.graphics import (Color, Ellipse, Line, Rectangle)
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.switch import Switch
from kivy.uix.screenmanager import ScreenManager, Screen
from kivy.config import Config
Config.set('graphics', 'resizable', '0')
Config.set('graphics', 'width', '300')
Config.set('graphics', 'height', '350')
from kivy.core.window import Window
Window.clearcolor = (.27, .35, .78, 1)
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights("model_weights.h5")
def test_model():
          img = Image.open('number.png').convert('L').resize((28, 28))
          result = model.predict(np.expand_dims(255 - np.array(img), axis = 0))
          img.close()
          os.remove('number.png')
          return str(np.argmax(result))
class ScreenManagement(ScreenManager):
    def __init__(self, **kwargs):
        super(ScreenManagement, self).__init__(**kwargs)
```

```
class PainterWidget(Widget):
	def on_touch_down(self, touch):
		with self.canvas:
			Color(.27, .35, .78, 1)
			rad = 7
			Ellipse(pos = (touch.x - rad/2, touch.y - rad/2), size = (rad, rad))
			touch.ud['line'] = Line(points = (touch.x, touch.y), width = 7)
	def on_touch_move(self, touch):
		touch.ud['line'].points += (touch.x, touch.y)
class MenuScreen(Screen):
	mod = 'text'
	def __init__(self, **kwargs):
		super(MenuScreen, self).__init__(**kwargs)
		self.add_widget(Label(text = 'Выберете испытание', size_hint = (1, .3), pos_hint = {'x': 0, 'y': .7}))
		self.add_widget(Button(text = 'Рисование цифры\nпо описанию', on_press = self.text_transition, size_hint = (.45, .15), pos_hint = {'x':
.025, 'y': .5}, halign = 'center', background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True))
		self.add_widget(Button(text = 'Решение\nпримеров', on_press = self.math_transition, size_hint = (.45, .15), pos_hint = {'x': .525, 'y': .5},
halign = 'center', background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True))
		self.add_widget(Label(text = 'Озвучивание', size_hint = (.5, .3), pos_hint = {'x': 0, 'y': .1}))
		self.voice = Switch(size_hint = (.5, .3), pos_hint = {'x': .5, 'y': .1})
		self.voice.bind(active = self.change_mod)
		self.add_widget(self.voice)
	def text_transition(self, *args):
		if MenuScreen.mod == 'text':
			self.manager.current = 'text'
		else:
			self.manager.current = 'voice'
			VoiceScreen.generate_voice(self)

	def math_transition(self, *args):
		if MenuScreen.mod == 'text':
			self.manager.current = 'math'
		else:
			VoiceMathScreen.generate_expression(self)
			self.manager.current = 'voice_math'
	def change_mod(self, instance, value):
		if self.voice.active == True:
			MenuScreen.mod = 'voice'
		else:
			MenuScreen.mod = 'text'
a = ['ноль', 'один', 'два', 'три', 'четыре', 'пять', 'шесть', 'семь', 'восемь', 'девять']
b = {a[i]: i for i in range(10)}
class TextScreen(Screen):
	chosen_number = ''
	text_numbers = a.copy()
	def generate_number(self):
		TextScreen.chosen_number = choice(TextScreen.text_numbers)
		return 'Нарисуй ' + '"' + TextScreen.chosen_number + '"'
	def __init__(self, **kwargs):
		super(TextScreen, self).__init__(**kwargs)
		main = Widget()
		self.painter = PainterWidget()
		self.painter.canvas.add(Color(1, 1, 1, 1))
		self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
		main.add_widget(self.painter)
		main.add_widget(Button(text = 'Заново', on_press = self.clear_canvas, size = (96, 50), pos = (3, 300), background_color = (1, 1, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
		self.out = Label(text = '', size = (100, 50), pos = (100, 0), halign = 'center')
		self.answer = Label(text = self.generate_number(), size = (100, 50), pos = (100, 50), halign = 'center')
		main.add_widget(self.answer)
		main.add_widget(self.out)
		self.check_btn = Button(text = 'Проверить', on_press = self.check_canvas, size = (96, 50), pos = (102, 300), disabled = False,
background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True)
		main.add_widget(self.check_btn)
		main.add_widget(Button(text = 'Назад', on_press = self.menu_transition, size = (96, 50), pos = (201, 300), background_color = (1, 1, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
		self.add_widget(main)
	def menu_transition(self, *args):
		self.check_btn.disabled = False
		TextScreen.text_numbers = a.copy()
		self.answer.text = self.generate_number()
		self.out.text = ''
		self.manager.current = 'menu'
	def clear_canvas(self, instance):
		self.check_btn.disabled = False
		self.painter.canvas.clear()
		self.out.text = ''
		self.painter.canvas.add(Color(.27, .35, .78, 1))
		self.painter.canvas.add(Rectangle(size = (300, 350)))
```

```
                        self.painter.canvas.add(Color(1, 1, 1, 1))
                        self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                        TextScreen.text_numbers = a.copy()
                        self.answer.text = self.generate_number()
        def check_canvas(self, instance):
                        self.painter.size = (150, 150)
                        self.painter.pos = (75, 100)
                        self.painter.export_to_png('number.png')
                        return_number = test_model()
                        if len(TextScreen.text_numbers) == 1:
                                self.out.text = 'Вы завершили это испытание!'
                                self.answer.text = ''
                                self.check_btn.disabled = True
                        elif b[TextScreen.chosen_number] == int(return_number):
                                self.out.text = choice(['Молодец!', 'Отлично!', 'Супер!', 'Умница!'])
                                TextScreen.text_numbers.remove(TextScreen.chosen_number)
                                self.answer.text = self.generate_number()
                        else:
                                self.out.text = choice(['Попробуй ещё раз', 'Немного неверно', 'Ещё разок', 'Почти получилось'])
                        self.painter.canvas.clear()
                        self.painter.canvas.add(Color(.27, .35, .78, 1))
                        self.painter.canvas.add(Rectangle(size = (300, 350)))
                        self.painter.canvas.add(Color(1, 1, 1, 1))
                        self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
class MathScreen(Screen):
        c = 0
        def generate_expression(self):
                        sign = choice(['+', '-'])
                        if sign == '+':
                                f_n = randrange(1, 9)
                                s_n = randrange(1, 10 - f_n)
                                MathScreen.c = f_n + s_n
                                return str(f_n) + ' + ' + str(s_n) + ' = ?'
                        else:
                                f_n = randrange(9, 0, -1)
                                s_n = randrange(f_n, 0, -1)
                                MathScreen.c = f_n - s_n
                                return str(f_n) + ' - ' + str(s_n) + ' = ?'
        def __init__(self, **kwargs):
                        super(MathScreen, self).__init__(**kwargs)
                        main = Widget()
                        self.painter = PainterWidget()
                        self.painter.canvas.add(Color(.27, .35, .78, 1))
                        self.painter.canvas.add(Rectangle(size = (300, 350)))
                        self.painter.canvas.add(Color(1, 1, 1, 1))
                        self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                        main.add_widget(self.painter)
                        main.add_widget(Button(text = 'Заново', on_press = self.clear_canvas, size = (96, 50), pos = (3, 300), background_color = (1, 11, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                        self.out = Label(text = '', size = (100, 50), pos = (100, 0), halign = 'center')
                        self.answer = Label(text = self.generate_expression(), size = (100, 50), pos = (100, 50), halign = 'center')
                        main.add_widget(self.answer)
                        main.add_widget(self.out)
                        self.check_btn = Button(text = 'Проверить', on_press = self.check_canvas, size = (96, 50), pos = (102, 300), disabled = False,
background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True)
                        main.add_widget(self.check_btn)
                        main.add_widget(Button(text = 'Назад', on_press = self.menu_transition, size = (96, 50), pos = (201, 300), background_color = (1, 1, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                        self.add_widget(main)
        def menu_transition(self, *args):
                        self.answer.text = self.generate_expression()
                        self.out.text = ''
                        self.manager.current = 'menu'
        def clear_canvas(self, instance):
                        self.check_btn.disabled = False
                        self.painter.canvas.clear()
                        self.out.text = ''
                        self.painter.canvas.add(Color(.27, .35, .78, 1))
                        self.painter.canvas.add(Rectangle(size = (300, 350)))
                        self.painter.canvas.add(Color(1, 1, 1, 1))
                        self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                        self.answer.text = self.generate_expression()
        def check_canvas(self, instance):
                        self.painter.size = (150, 150)
                        self.painter.pos = (75, 100)
                        self.painter.export_to_png('number.png')
                        return_number = test_model()
                        if MathScreen.c == int(return_number):
                                self.out.text = 'Правильно'
```

```
                                self.answer.text = self.generate_expression()
                        else:
                                self.out.text = 'Попробуй ещё раз'
                        self.painter.canvas.clear()
                        self.painter.canvas.add(Color(.27, .35, .78, 1))
                        self.painter.canvas.add(Rectangle(size = (300, 350)))
                        self.painter.canvas.add(Color(1, 1, 1, 1))
                        self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
class VoiceScreen(Screen):
        chosen_number = ''
        text_numbers = a.copy()
        def generate_voice(self):
                VoiceScreen.chosen_number = choice(VoiceScreen.text_numbers)
                winsound.PlaySound(os.path.join('Records', 'нарисуй.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                sleep(1.32)
                winsound.PlaySound(os.path.join('Numbers', VoiceScreen.chosen_number + '.wav'), winsound.SND_ASYNC |
winsound.SND_FILENAME)
        def __init__(self, **kwargs):
                super(VoiceScreen, self).__init__(**kwargs)
                main = Widget()
                self.painter = PainterWidget()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                main.add_widget(self.painter)
                main.add_widget(Button(text = 'Заново', on_press = self.clear_canvas, size = (96, 50), pos = (3, 300), background_color = (1, 11, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                self.check_btn = Button(text = 'Проверить', on_press = self.check_canvas, size = (96, 50), pos = (102, 300), disabled = False,
background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True)
                main.add_widget(self.check_btn)
                main.add_widget(Button(text = 'Назад', on_press = self.menu_transition, size = (96, 50), pos = (201, 300), background_color = (1, 1, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                self.add_widget(main)
        def menu_transition(self, *args):
                self.check_btn.disabled = False
                VoiceScreen.text_numbers = a.copy()
                self.manager.current = 'menu'
        def clear_canvas(self, instance):
                self.check_btn.disabled = False
                self.painter.canvas.clear()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                VoiceScreen.text_numbers = a.copy()
                self.generate_voice()
        def check_canvas(self, instance):
                self.painter.size = (150, 150)
                self.painter.pos = (75, 100)
                self.painter.export_to_png('number.png')
                return_number = test_model()
                if len(VoiceScreen.text_numbers) == 1:
                        winsound.PlaySound(os.path.join('Records', 'congrats.wav'), winsound.SND_FILENAME)
                        self.check_btn.disabled = True
                elif b[VoiceScreen.chosen_number] == int(return_number):
                        winsound.PlaySound(os.path.join('Records', choice(['молодец', 'отлично', 'супер', 'умница']) + '.wav'),
winsound.SND_FILENAME)
                        VoiceScreen.text_numbers.remove(VoiceScreen.chosen_number)
                        self.generate_voice()
                else:
                        winsound.PlaySound(os.path.join('Records', choice(['попробуй ещё раз', 'немного неверно', 'ещё разок', 'почти получилось'])
+ '.wav'), winsound.SND_FILENAME)
                self.painter.canvas.clear()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
class VoiceMathScreen(Screen):
        c = 0
        def generate_expression(self):
                sign = choice(['+', '-'])
                if sign == '+':
                        f_n = randrange(1, 9)
                        s_n = randrange(1, 10 - f_n)
                        MathScreen.c = f_n + s_n
                        winsound.PlaySound(os.path.join('Math', 'how.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                        sleep(1.2)
                        winsound.PlaySound(os.path.join('Numbers', a[f_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
```

```
                                sleep(0.8)
                                winsound.PlaySound(os.path.join('Math', '+.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(0.7)
                                winsound.PlaySound(os.path.join('Numbers', a[s_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(0.8)
                        else:
                                f_n = randrange(9, 0, -1)
                                s_n = randrange(f_n, 0, -1)
                                MathScreen.c = f_n - s_n
                                winsound.PlaySound(os.path.join('Math', 'how.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(1.2)
                                winsound.PlaySound(os.path.join('Numbers', a[f_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(0.8)
                                winsound.PlaySound(os.path.join('Math', '-.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(0.7)
                                winsound.PlaySound(os.path.join('Numbers', a[s_n] + '.wav'), winsound.SND_ASYNC | winsound.SND_FILENAME)
                                sleep(0.8)
        def __init__(self, **kwargs):
                super(VoiceMathScreen, self).__init__(**kwargs)
                main = Widget()
                self.painter = PainterWidget()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                main.add_widget(self.painter)
                main.add_widget(Button(text = 'Заново', on_press = self.clear_canvas, size = (96, 50), pos = (3, 300), background_color = (1, 11, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                self.check_btn = Button(text = 'Проверить', on_press = self.check_canvas, size = (96, 50), pos = (102, 300), disabled = False,
background_color = (1, 1, 1, 1), background_normal = '', color = (.27, .35, .78, 1), bold = True)
                main.add_widget(self.check_btn)
                main.add_widget(Button(text = 'Назад', on_press = self.menu_transition, size = (96, 50), pos = (201, 300), background_color = (1, 1, 1, 1),
background_normal = '', color = (.27, .35, .78, 1), bold = True))
                self.add_widget(main)
        def menu_transition(self, *args):
                self.manager.current = 'menu'
        def clear_canvas(self, instance):
                self.check_btn.disabled = False
                self.painter.canvas.clear()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
                self.generate_expression()
        def check_canvas(self, instance):
                self.painter.size = (150, 150)
                self.painter.pos = (75, 100)
                self.painter.export_to_png('number.png')
                return_number = test_model()
                if MathScreen.c == int(return_number):
                        winsound.PlaySound(os.path.join('Records', choice(['молодец', 'отлично', 'супер', 'умница']) + '.wav'),
winsound.SND_FILENAME)
                        self.generate_expression()
                else:
                        winsound.PlaySound(os.path.join('Records', choice(['попробуй ещё раз', 'немного неверно', 'ещё разок', 'почти получилось'])
+ '.wav'), winsound.SND_FILENAME)
                self.painter.canvas.clear()
                self.painter.canvas.add(Color(.27, .35, .78, 1))
                self.painter.canvas.add(Rectangle(size = (300, 350)))
                self.painter.canvas.add(Color(1, 1, 1, 1))
                self.painter.canvas.add(Rectangle(size = (150, 150), pos = (75, 100)))
class DigiStarterApp(App):
        def build(self):
                sm = ScreenManager()
                sm.add_widget(MenuScreen(name = 'menu'))
                sm.add_widget(TextScreen(name = 'text'))
                sm.add_widget(VoiceScreen(name = 'voice'))
                sm.add_widget(MathScreen(name = 'math'))
                sm.add_widget(VoiceMathScreen(name = 'voice_math'))
                return sm
if __name__ == "__main__":
        DigiStarterApp().run()*
```