МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
СТАРООСКОЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ ИМ. А.А. УГАРОВА
(филиал) федерального государственного автономного образовательного учреждения высшего образования
«Национальный исследовательский технологический университет «МИСиС»
ОСКОЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

ОТЧЕТ
по практической работе №5
по МДК 06.04 «Интеллектуальные системы и технологии»

Тема «Моделирование самообучающихся интеллектуальных систем»

выполнил:
студент группы ИСП-?-?
ИМЯ

проверил:
преподаватель ИМЯ.

Старый Оскол, 20?? год

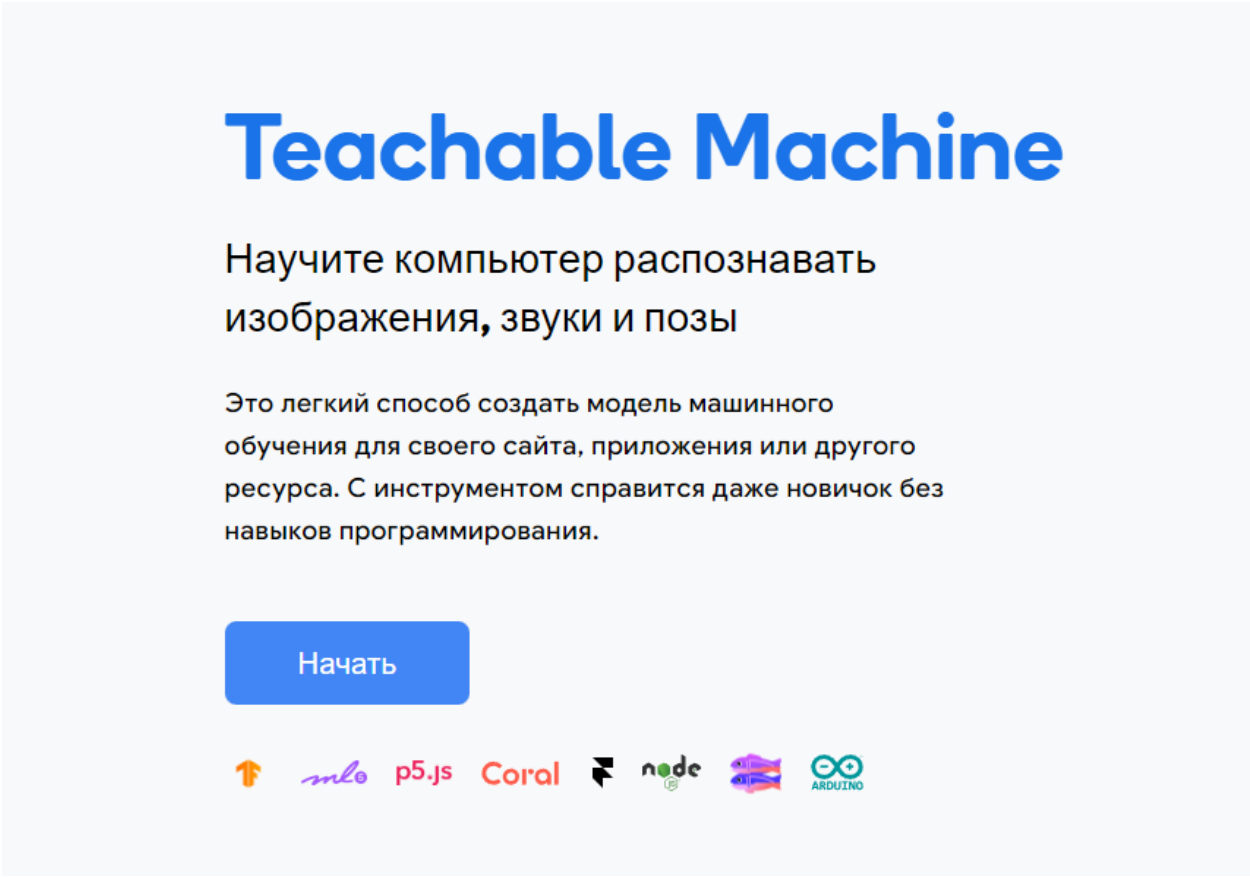

Рисунок 1 – Начало работы

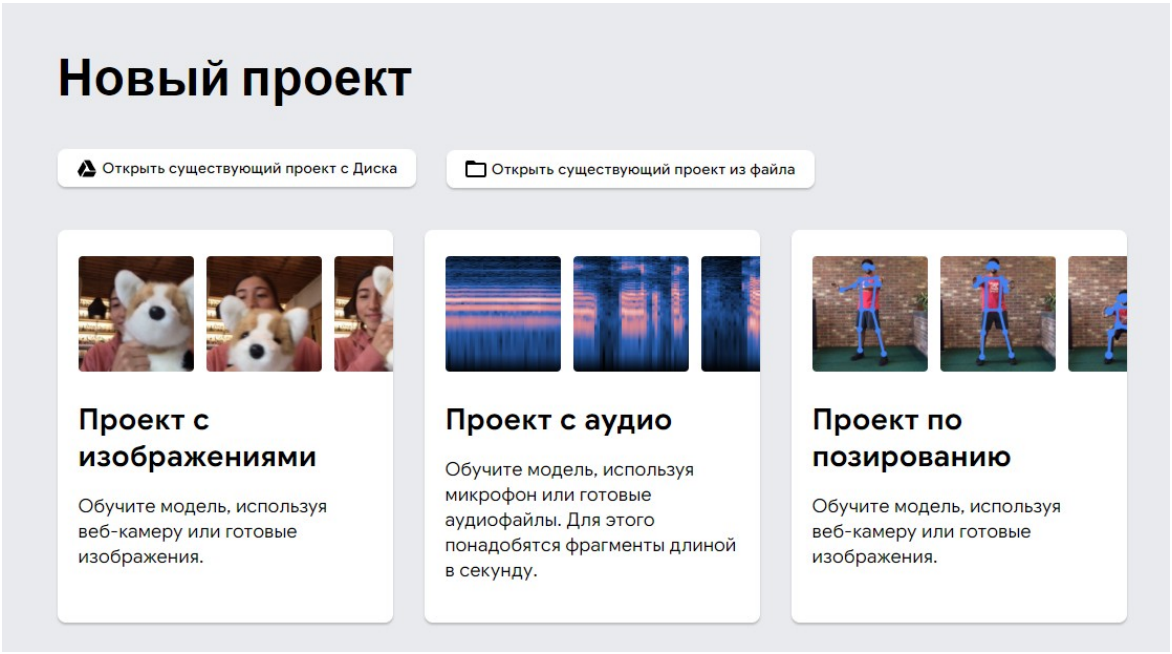

Рисунок 2 – Создание нового проекта с изображениями

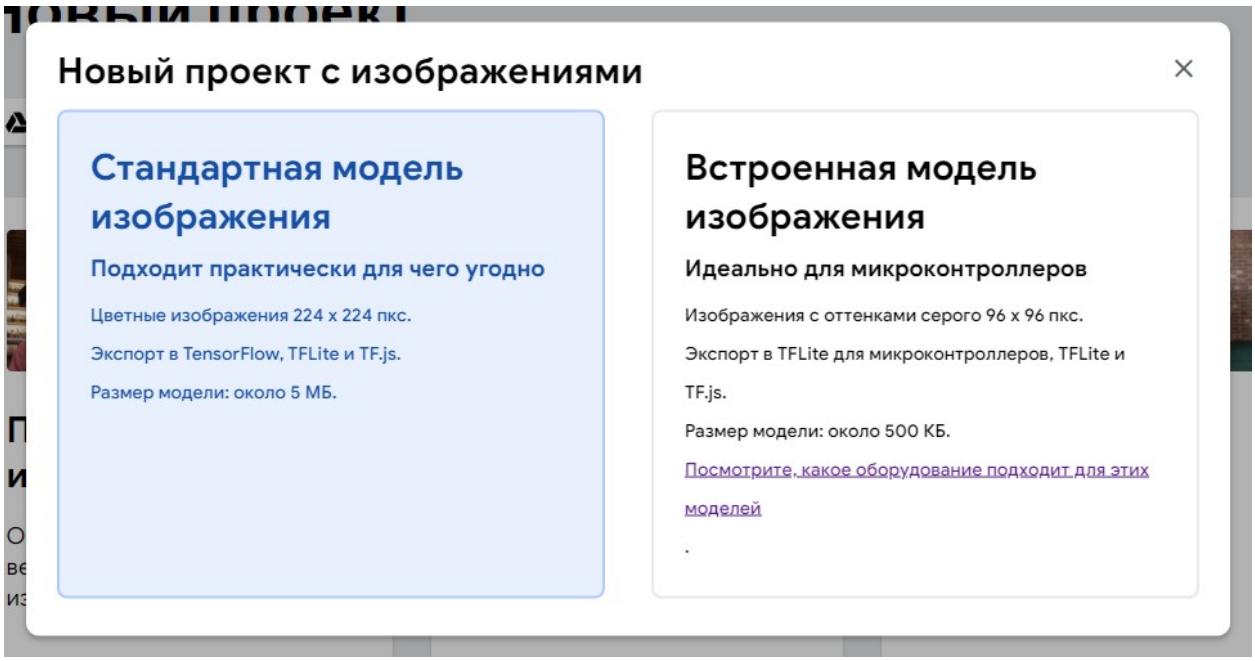

Рисунок 3 – Создание нового проекта с изображениями

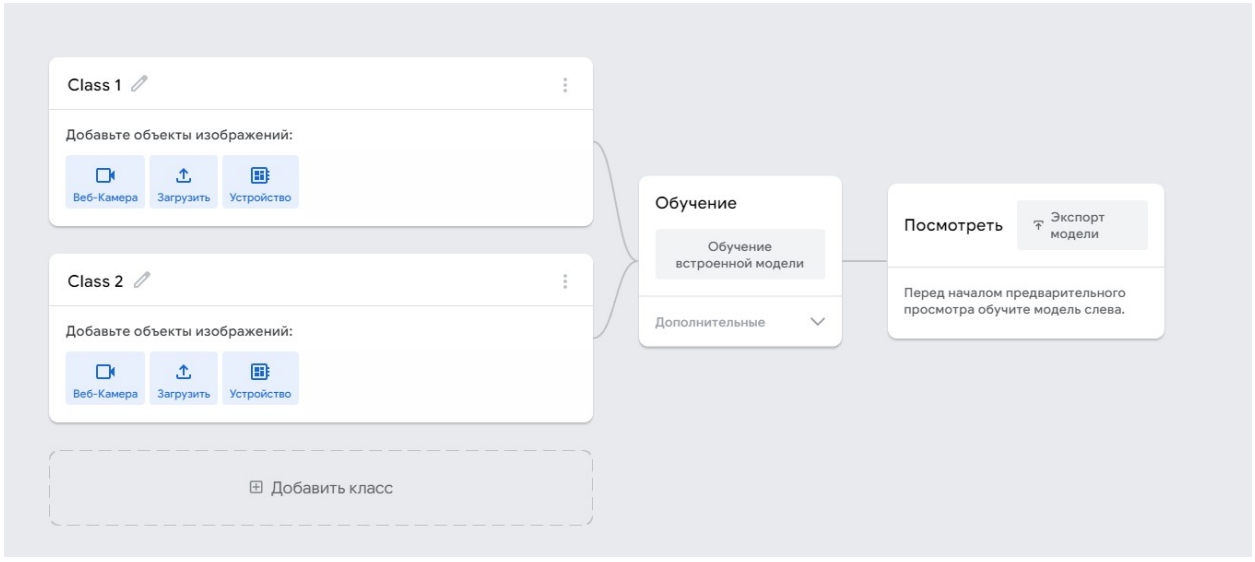

Рисунок 4 – Добавление объектов изображений

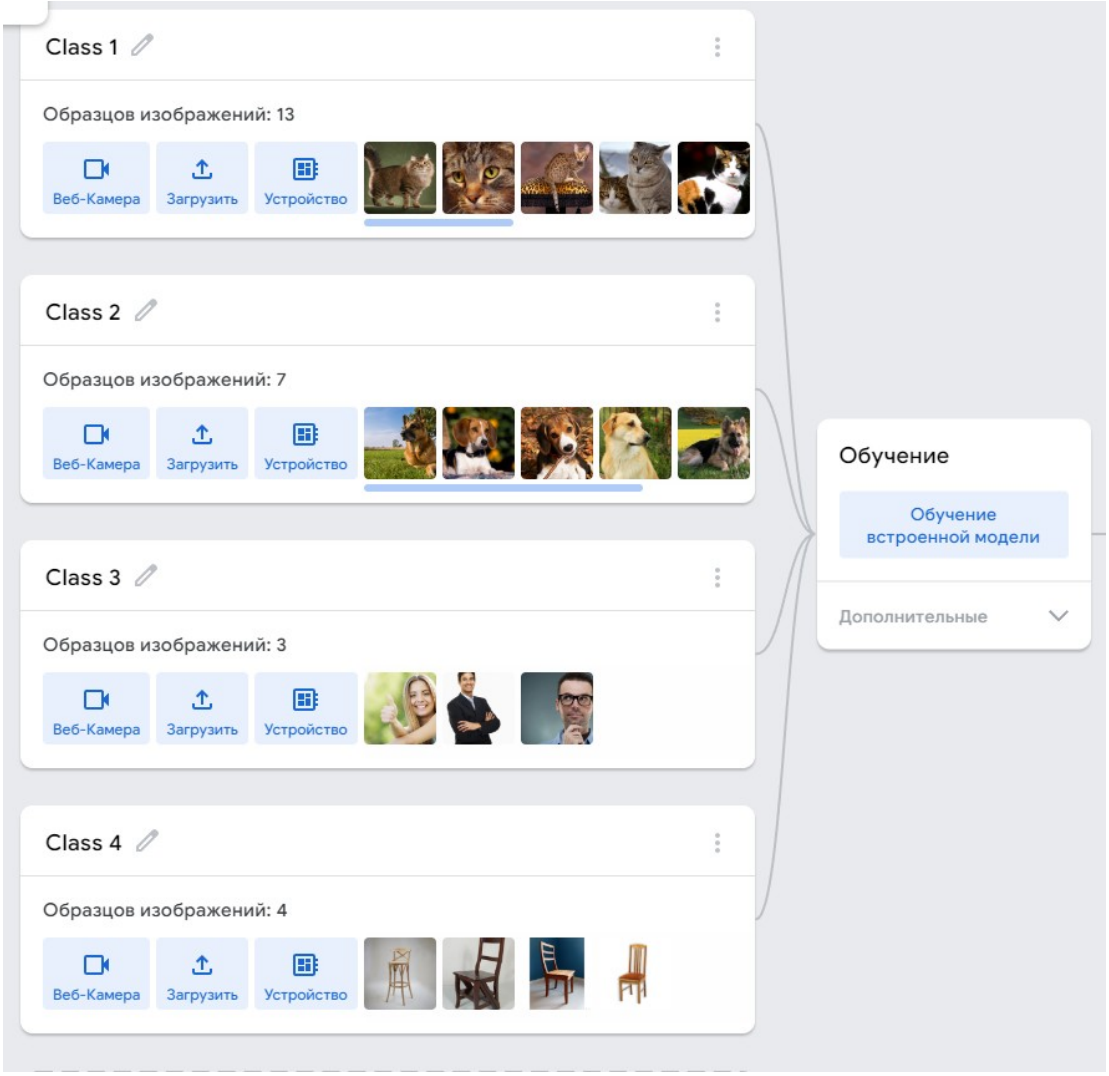

Рисунок 5 – Начало обучения образцов изображений

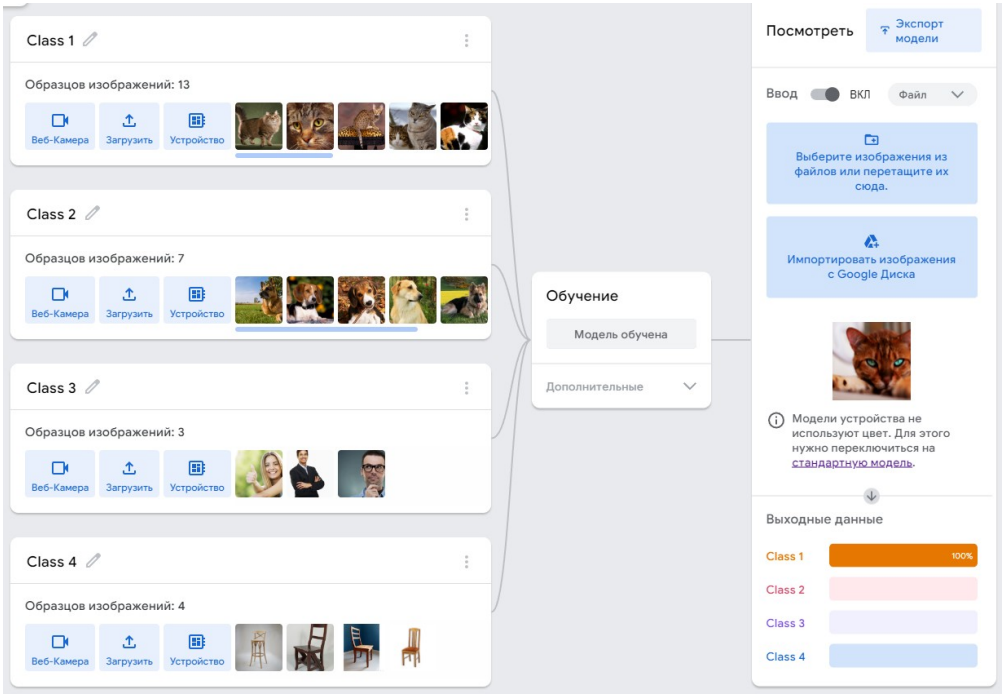

Рисунок 6 – обучение модели класса class1

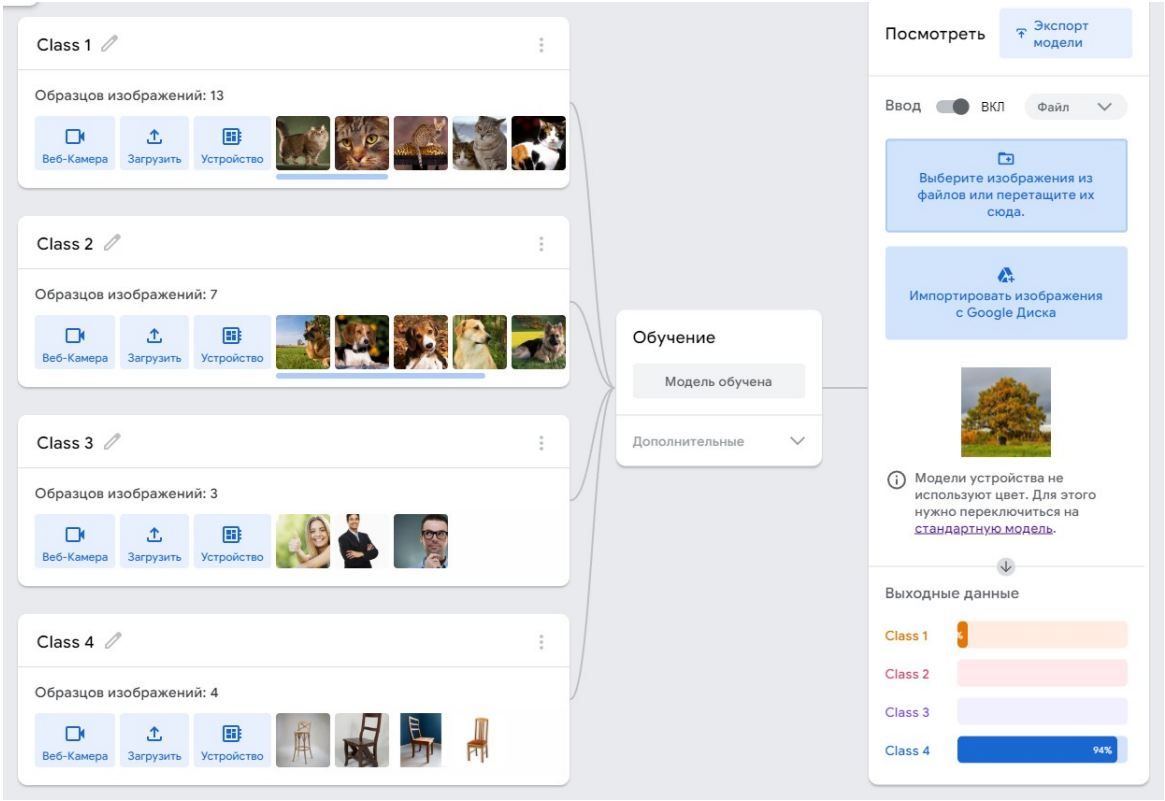

Рисунок 7 – обучение модели «Неопознанно»

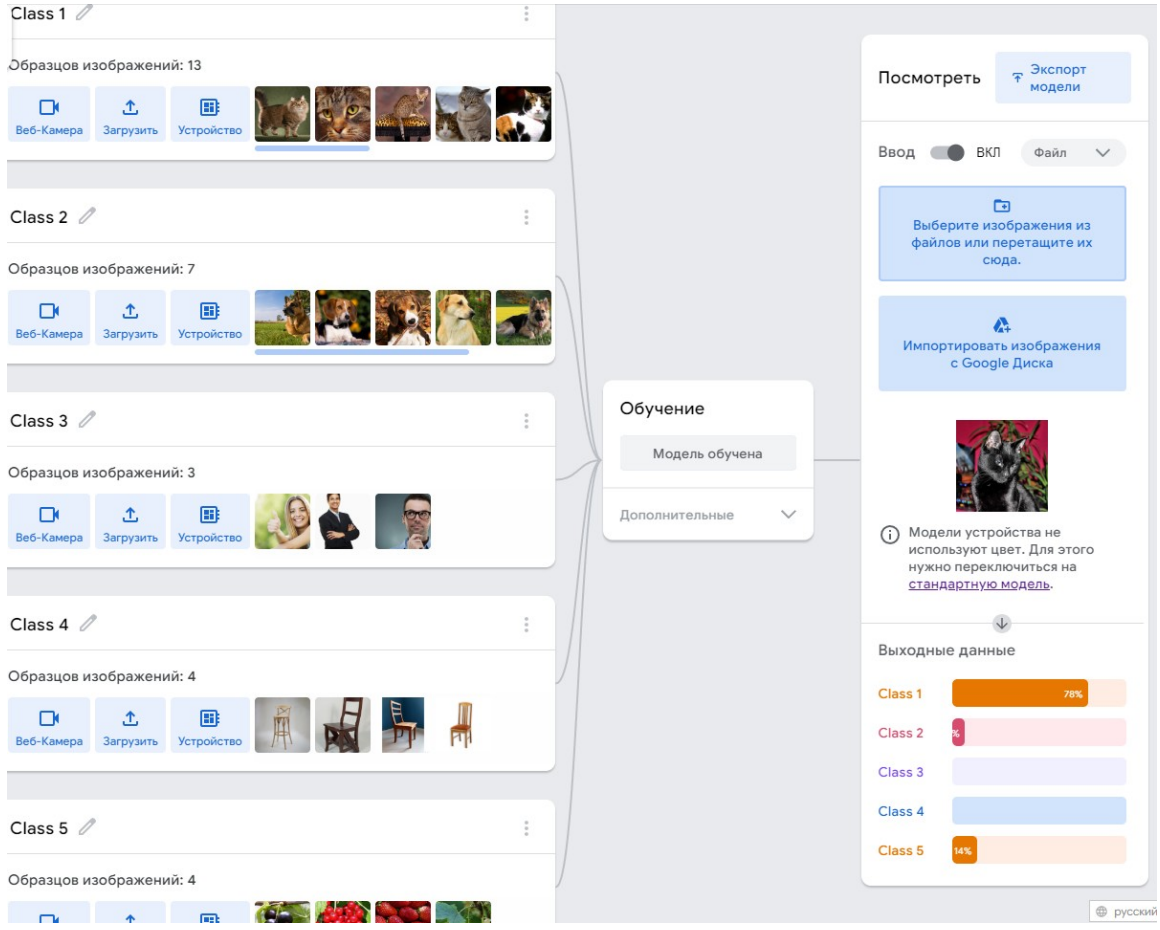

Рисунок 8 – Неопознанная модель класса

Задание 2

```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

  def feedforward(self, inputs):

    total = np.dot(self.weights, inputs)
+ self.bias
    return sigmoid(total)

weights = np.array([0, 1])
bias = 4
n = Neuron(weights, bias)

x = np.array([2, 3])
print(n.feedforward(x))
```

```
0.9990889488055994
>
```

Рисунок 9 - Код с результатом

```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

class Neuron:
  def __init__(self, weights, bias):
    self.weights = weights
    self.bias = bias

  def feedforward(self, inputs):
    total = np.dot(self.weights, inputs) + self.bias
    return sigmoid(total)

class OurNeuralNetwork:
  def __init__(self):
    weights = np.array([0, 1])
    bias = 0

    self.h1 = Neuron(weights, bias)
    self.h2 = Neuron(weights, bias)
    self.o1 = Neuron(weights, bias)

  def feedforward(self, x):
    out_h1 = self.h1.feedforward(x)
    out_h2 = self.h2.feedforward(x)

    out_o1 = self.o1.feedforward(np.array([out_h1,
out_h2]))

    return out_o1

network = OurNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x))
```

```
0.7216325609518421
>
```

Рисунок 10 –Код с результатом

Рисунок 11 –Код с результатом

```
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

def deriv_sigmoid(x):
  fx = sigmoid(x)
  return fx * (1 - fx)

def mse_loss(y_true, y_pred):
  return ((y_true - y_pred) ** 2).mean()

class OurNeuralNetwork:
  def __init__(self):
    #веса
    self.w1 = np.random.normal()
    self.w2 = np.random.normal()
    self.w3 = np.random.normal()
    self.w4 = np.random.normal()
    self.w5 = np.random.normal()
    self.w6 = np.random.normal()

    #пороги
    self.b1 = np.random.normal()
    self.b2 = np.random.normal()
    self.b3 = np.random.normal()

  def feedforward(self, x):
    h1 = sigmoid(self.w1 * x[0] + self.w2 * x[1] +
self.b1)
    h2 = sigmoid(self.w3 * x[0] + self.w4 * x[1] +
self.b2)
    o1 = sigmoid(self.w5 * h1 + self.w6 * h2 + self.b3)
    return o1
```

Рисунок 12 – Программный код

```
65          d_h2_d_b2 = deriv_sigmoid(sum_h2)
66
67          #обновляем веса и пороги
68          #нейрон h1
69          self.w1 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h1 * d_h1_d_w1
70          self.w2 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h1 * d_h1_d_w2
71          self.b1 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h1 * d_h1_d_b1
72
73          #нейрон h2
74          self.w3 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h2 * d_h2_d_w3
75          self.w4 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h2 * d_h2_d_w4
76          self.b2 -= learn_rate * d_L_d_ypred *
    d_ypred_d_h2 * d_h2_d_b2
77
78          #нейрон o1
79          self.w5 -= learn_rate * d_L_d_ypred *
    d_ypred_d_w5
80          self.w6 -= learn_rate * d_L_d_ypred *
    d_ypred_d_w6
81          self.b3 -= learn_rate * d_L_d_ypred *
    d_ypred_d_b3
82
83        #считаем полные потери в конце каждой эпохи
84        if epoch % 10 == 0:
85          y_preds = np.apply_along_axis(self.feedforward,
    1, data)
86          loss = mse_loss(all_y_trues, y_preds)
87          print("Epoch %d loss: %.3f" % (epoch, loss))
88
89  #определим набор данных
```

Рисунок 13 – Программный код

```
    d_ypred_d_h2 * d_h2_d_b2
77
78          #нейрон o1
79          self.w5 -= learn_rate * d_L_d_ypred *
    d_ypred_d_w5
80          self.w6 -= learn_rate * d_L_d_ypred *
    d_ypred_d_w6
81          self.b3 -= learn_rate * d_L_d_ypred *
    d_ypred_d_b3
82
83        #считаем полные потери в конце каждой эпохи
84        if epoch % 10 == 0:
85          y_preds = np.apply_along_axis(self.feedforward,
    1, data)
86          loss = mse_loss(all_y_trues, y_preds)
87          print("Epoch %d loss: %.3f" % (epoch, loss))
88
89  #определим набор данных
90  data = np.array([
91    [-2, -1],  #Алиса
92    [25, 6],   #Боб
93    [17, 4],   #Чарли
94    [-15, -6], #Диана
95  ])
96  all_y_trues = np.array([
97    1, #Алиса
98    0, #Боб
99    0, #Чарли
100   1, #Диана
101 ])
102
103 #обучаем нашу нейронную сеть
104 network = OurNeuralNetwork()
105 network.train(data, all_y_trues)
106
```

```
Epoch 590 loss: 0.003
Epoch 600 loss: 0.003
Epoch 610 loss: 0.003
Epoch 620 loss: 0.003
Epoch 630 loss: 0.003
Epoch 640 loss: 0.003
Epoch 650 loss: 0.002
Epoch 660 loss: 0.002
Epoch 670 loss: 0.002
Epoch 680 loss: 0.002
Epoch 690 loss: 0.002
Epoch 700 loss: 0.002
Epoch 710 loss: 0.002
Epoch 720 loss: 0.002
Epoch 730 loss: 0.002
Epoch 740 loss: 0.002
Epoch 750 loss: 0.002
Epoch 760 loss: 0.002
Epoch 770 loss: 0.002
Epoch 780 loss: 0.002
Epoch 790 loss: 0.002
Epoch 800 loss: 0.002
Epoch 810 loss: 0.002
Epoch 820 loss: 0.002
Epoch 830 loss: 0.002
Epoch 840 loss: 0.002
Epoch 850 loss: 0.002
Epoch 860 loss: 0.002
Epoch 870 loss: 0.002
Epoch 880 loss: 0.002
Epoch 890 loss: 0.002
Epoch 900 loss: 0.002
Epoch 910 loss: 0.002
Epoch 920 loss: 0.002
Epoch 930 loss: 0.002
Epoch 940 loss: 0.002
Epoch 950 loss: 0.002
Epoch 960 loss: 0.002
Epoch 970 loss: 0.002
Epoch 980 loss: 0.002
Epoch 990 loss: 0.002
>
```

Рисунок 14 – Программный код№1

```
102
103 #обучаем нашу нейронную сеть
104 network = OurNeuralNetwork()
105 network.train(data, all_y_trues)
106 #предсказания
107 emily = np.array([-7, -3])
108 frank = np.array([20, 2])
109 print("Эмили: %.3f" % network.feedforward(emily))
110 print("Фрэнк: %.3f" % network.feedforward(frank))
```

```
Epoch 930 loss: 0.002
Epoch 940 loss: 0.002
Epoch 950 loss: 0.002
Epoch 960 loss: 0.002
Epoch 970 loss: 0.002
Epoch 980 loss: 0.001
Epoch 990 loss: 0.001
Эмили: 0.968
Фрэнк: 0.038
>
```

Рисунок 14 – Программный код№2

Контрольные вопросы

1. Что такое самообучающаяся система

Самообучающаяся система – это ИИС, которая на основе примеров реальной практики автоматически формирует единицы знаний.

2. Виды самообучающихся систем

Различают следующие виды самообучающихся систем.

1. Система с индуктивным выводом – это самообучающаяся ИИС, работа которой основана на правилах индуктивного вывода с помощью классификации примеров по значимым признакам.

2. Нейронные сети – это самообучающиеся ИИС, которые на основе обучения по реальным примерам строят ассоциативную сеть понятий (нейронов) для параллельного поиска решений.

3. Системы, основанные на прецедентах – это самообучающиеся ИИС, которые в качестве единиц знаний хранят прецеденты решений и позволяют по запросу подбирать и адаптировать наиболее похожие прецеденты. Для поиска решения задачи используется алгоритм поиска по аналогии, который включает в себя следующие этапы:

1) получение подробной информации о текущей проблеме;

2) сопоставление полученной информации со значениями признаков прецедентов из базы знаний;

3) выбор прецедента из базы знаний, наиболее близкого к рассматриваемой проблеме;

4) выполнение адаптации выбранного прецедента к текущей проблеме;

5) проверка корректности каждого полученного решения;

6) занесение детальной информации о полученном решении в базу знаний.

4. Информационные хранилища – это самообучающиеся ИИС, которые позволяют извлекать знания из баз данных и создавать специально-организованные базы знаний. Информационные хранилища представляют собой хранилища значимой информации, регулярно извлекаемой из

оперативных баз данных и предназначенной для оперативного анализа данных.

3. Понятие и принцип работы нейронной сети

За основу создания ИНС взят человеческий мозг, где в процессе сложного взаимодействия между нейронами, соединенными между собой синаптической связью, обеспечивается выполнение огромного количества разных функций организма. Роль нейронов в искусственных устройствах выполняют простейшие процессоры, которые собраны в крупную сеть и поэтому способны решать довольно сложные задачи.

4. Достоинства и недостатки нейронных сетей.

Плюсы и минусы нейронных сетей

Перечислим главные достоинства ИНС:

– Способность игнорировать постороннюю информацию.

– Возможность сохранять работоспособность в случае утраты отдельных элементов.

– Высокая скорость работы

Тем не менее, полностью полагаться на нейросети нельзя. Их можно использовать как эффективное дополнение к другим методам, но не как единственный вариант достижения цели. Причин несколько:

– Предлагаемый ИНС ответ не будет абсолютно точным, только примерным.

– Каждый искусственный нейрон действует независимо от соседних, он не соотносит свое поведение с другими микропроцессорами. Задачи и области применения нейронных сетей

– Классификация.

– Прогнозирование.

– Распознавание. На данный момент эта функция применяется чаще остальных. Поиск по фото в Яндексе или Google, возможность отметить лица друзей на фото в социальных сетях и другие современные

возможности обеспечены именно умением ИНС выделять объект среди множества подобных.

Перечисленными сферами использование нейросетей не ограничивается, есть и другие существующие и перспективные способы задействовать их для решения различных задач:

– Машинное обучение является одной из разновидностей искусственного интеллекта. Google, Яндекс, Бинг, Байду активно применяют machinelearning для повышения релевантность результатов запросам пользователей. Алгоритмы самообучаются, опираясь на миллионы однотипных фраз, вводимых в поисковую строку.

– Для нормального функционирования роботов необходимо разрабатывать множество алгоритмов, и здесь не обойтись без нейросетей.

– Возможности ИНС используются архитекторами компьютерных сетей, чтобы справиться с проблемой параллельных вычислений.

– В математике нейронные сети позволяют быстрее решать сложные задачи.

5. Задачи и области применения нейронных сетей

Применение нейронных сетей позволяет решать задачи следующих типов:

- Классификация. Например, когда нужно определить соответствует ли человек категории населения, которой положены льготы.

- Предсказание. Например, чтобы спрогнозировать стоимость акций компании.

- Распознавание. Например, когда нужно определить, кто изображён на фотографии — мужчина или женщина.

- Решение задач без учителя. Например, выбор аудитории для таргетированной рекламы.

В каждой предметной области при ближайшем рассмотрении можно найти постановки задач для нейронных сетей. Вот список отдельных

областей, где решение такого рода задач имеет практическое значение уже сейчас.

- Экономика и бизнес: прогнозирование временных рядов (курсов валют, цен на сырьё, спроса, объемов продаж,..), автоматический трейдинг (торговля на валютной, фондовой или товарной бирже), оценка рисков невозврата кредитов, предсказание банкротств, оценка стоимости недвижимости, выявление переоцененных и недооцененных компаний, рейтингование, оптимизация товарных и денежных потоков, считывание и распознавание чеков и документов, безопасность транзакций по пластиковым картам.
- Медицина и здравоохранение: постановка диагноза больному (диагностика заболеваний), обработка и распознавание медицинских изображений (рентгеновских снимков, томограмм и т.д.), очистка показаний приборов от шумов, мониторинг состояния пациента, прогнозирование результатов применения разных методов лечения, анализ эффективности проведённого лечения.
- Авионика: обучаемые автопилоты, распознавание сигналов радаров, адаптивное пилотирование сильно поврежденного самолета, беспилотные летательные аппараты (дроны), распознавание/детекция объектов на фото/видеосъёмке с дрона.
- Связь: сжатие видеоинформации, быстрое кодирование-декодирование, оптимизация сотовых сетей и схем маршрутизации пакетов.
- Интернет: ассоциативный поиск информации, электронные секретари и автономные агенты в интернете, фильтрация и блокировка спама, автоматическая рубрикация сообщений из новостевых лент, адресные (персонализированные) реклама и маркетинг для электронной торговли, чат-боты, автоматизация распознавания captcha.
- Автоматизация производства: оптимизация режимов производственного процесса, контроль качества продукции, мониторинг и

визуализация многомерной диспетчерской информации, предупреждение аварийных ситуаций.

- Робототехника: распознавание сцены, объектов и препятствий перед роботом, прокладка маршрута, управление манипуляторами (например, решение обратной задачи кинематики), поддержание равновесия.
- Политологические и социологические исследования: предсказание результатов выборов, анализ опросов, предсказание динамики рейтингов, выявление значимых факторов, кластеризация электората, изучение и визуализация социальной динамики населения.
- Безопасность, охранные системы: распознавание лиц; идентификация личности по отпечаткам пальцев, голосу или подписи; распознавание автомобильных номеров; мониторинг пакетов информации и информационных потоков в компьютерной сети для обнаружения вторжений; обнаружение подделок; анализ данных с видеокамер и разнообразных сенсоров; анализ аэрокосмических снимков (обнаружение лесных пожаров, незаконных вырубок леса и т.д.).
- Ввод и обработка информации: распознавание рукописных текстов, отсканированных почтовых, платежных, финансовых и бухгалтерских документов; распознавание речевых команд, речевой ввод текста в компьютер.
- Геологоразведка: анализ сейсмических данных, ассоциативные методики поиска полезных ископаемых, оценка ресурсов месторождений.
- Компьютерные и настольные игры

6. Сбор данных для обучения нейронной сети

Как правило, в качестве объекта анализа ИНС предлагаются числовые данные в пределах ограниченного диапазона. Если предстоит работа с информацией другого формата, могут возникнуть проблемы.

Нечисловые данные в целом считаются для нейронной сети более сложным вариантом решения поставленной задачи. В качестве примера можно привести номинальные переменные типа Пол = {Муж, Жен}.

Чтобы упростить ИНС работу по анализу информации используется присвоение числовых значений. Например, нейросеть создается для оценки объектов недвижимости конкретного города, в котором каждый микрорайон имеет собственное название. На первый взгляд, проще всего добавить переменные с соответствующими словесными обозначениями. Однако это серьезно затруднит процесс обучения сети и приведет к большому проценту ошибок на выходе.

Оптимальным решением в этом случае представляется присвоение отдельным районам рейтинговых баллов, основанных не результатах экспертной оценки стоимости жилья в каждом из них.

7. Проблемы функционирования нейронных сетей

3 проблемы функционирования нейронных сетей

– Переобучение

Суть этого явления заключается в неспособности ИНС уловить закономерность, по которой можно сделать правильный вывод. Вместо этого сеть просто фиксирует ответы, что приводит к ее переобучению и низкой эффективность выдаваемых результатов. Для решения этой проблемы предложены такие инструменты, как регуляризация, нормализация батчей, наращивание данных и т. д.

– «Забывчивость»

Следующая проблема состоит в необходимости создавать новую ИНС под каждую конкретную задачу, поскольку нейронная сеть плохо подходят для работы в постоянно меняющихся условиях. Предположим, перед ней стоит задача по прогнозированию поведения акций на фондовом рынке. Возможно, какое-то время она будет довольно успешно справляться со своей функцией, но со временем достоверность предлагаемых нейронной сетью вариантов может значительно понизиться.

Чтобы обойти эту проблему, разработчики тестируют различные архитектуры ИНС в попытках найти оптимальную, которая сможет подстраиваться под меняющиеся параметры, а также создают динамические

нейронные сети, способные отслеживать появление новых условий во внешней среде и вносить коррективы в свою архитектуру. В этом случае применяется MSO (multi-swarmoptimization) и аналогичные ему алгоритмы.

– Закрытость и непредсказуемость

Нейронную сеть сравнивают с непрозрачным ящиком, в который закладывается набор переменных, а на выходе получается некий результат. Ни процесс принятия решений, ни промежуточная статистика, ни принцип работы ИНС наблюдателю не доступны. Исключением являются только сверточные нейронные сети, используемые для распознавания. Здесь есть возможность отследить возбуждение отдельных микропроцессоров, поскольку некоторые внутренние слои имеют смысл карт признаков.

В качестве предлагаемых исследователями способов решения этой проблемы можно назвать работу над алгоритмами изъятия правил (rule-extractionalgorithms), нацеленную на повышение прозрачности архитектур. В результате использования таких алгоритмов удается извлечь информацию из ИНС в виде символьной логики, математических выражений или деревьев решений.