**Федеральное агентство по образованию**Государственное образовательное учреждение высшего профессионального   
образования **«Тихоокеанский Государственный университет»**

Факультет компьютерных и фундаментальных наук

Кафедра ПОВТАС

**Лабораторная работа №3**

по дисциплине: «Архитектура систем ИИ»

на тему: «Задача распознавания образов. Используя ИНС»  
Вариант №4

Выполнил: студент группы ПИИ(м)-21

Забавин А.С.

Проверил: ст. преп. кафедры ПОВТАС

Тормозов В.С.

# Постановка задачи

Для распознавания предъявляется циферблат электронных часов, который состоит из четырех позиций: две позиции для вывода часов и две позиции для вывода минут (чч:мм). Каждая цифра представляется набором пикселов из 9 строк по 6 пикселов в каждой строке. Каждый пиксел может принимать одно из двух значений символов «\*» или «.». Сочетания «\*» и «.» задают одну из 10 цифр.

В идеале на циферблате часов показывается реальное время. Однако в системе часов произошел сбой и некоторые пикселы приняли «неправильные» значения.

Требуется определить, какое же время выводят часы или вывести сообщение о невозможности определения времени.

**Исходные данные**

В текстовом файле “etalon.txt” приведены образцы цифр в виде 10 групп 6 х 8 символов «\*» или «.». Во втором текстовом файле “times.txt” записаны показания с циферблата часов приведены образцы цифр в виде 4 групп 6 х 8 символов «\*» или «.», как с точными значениями, так и поврежденными. При запуске – программа попросит ввести время из файла times.txt, затем сама разобьет ее на соответствующие часам и минутам значения и попытается распознать цифры используя ИНС.

**Выходные данные**

На экран выводится сообщение о времени в форме «чч:мм» или сообщение «[Невозможно определить]» если средняя ошибка распознавания ИНС превышает 50%.

# Краткая теория

Под распознаванием образа понимаются процессы восприятия и познания, связанные с поиском, выделением, идентификацией и классификацией образов на основе имеющихся данных.

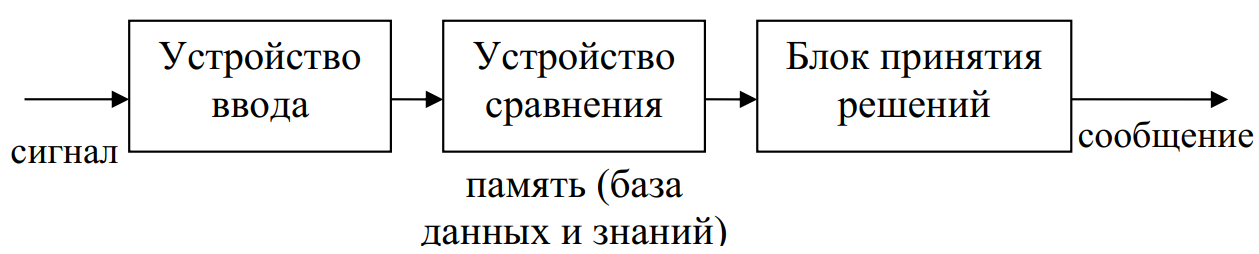


Рис. . Система распознавания

В общем виде систему, проводящую распознавание образов, можно представить как совокупность устройств ввода информации–образа, сравнения с накопленными ранее знаниями и устройства принятия решений (Рис. 1).

Известны следующие типы задач распознавания.

1. Отнесение предъявляемого объекта по его формализованному описанию (образу) к одному из заданных классов.
2. Разбиение множества объектов по их формализованным описаниям к системе непересекающихся классов.
3. Определение набора признаков для построения формализованного описания объекта распознавания; оценка информативности отдельных признаков и их сочетаний.
4. Построение формализованного описания объекта распознавания.

В приведенной на рисунке 1 схеме нашли отражение задачи первого и второго типов данной классификации. С учетом всех четырех типов общая структура системы распознавания выглядит так, как изображено на Рис. 2.

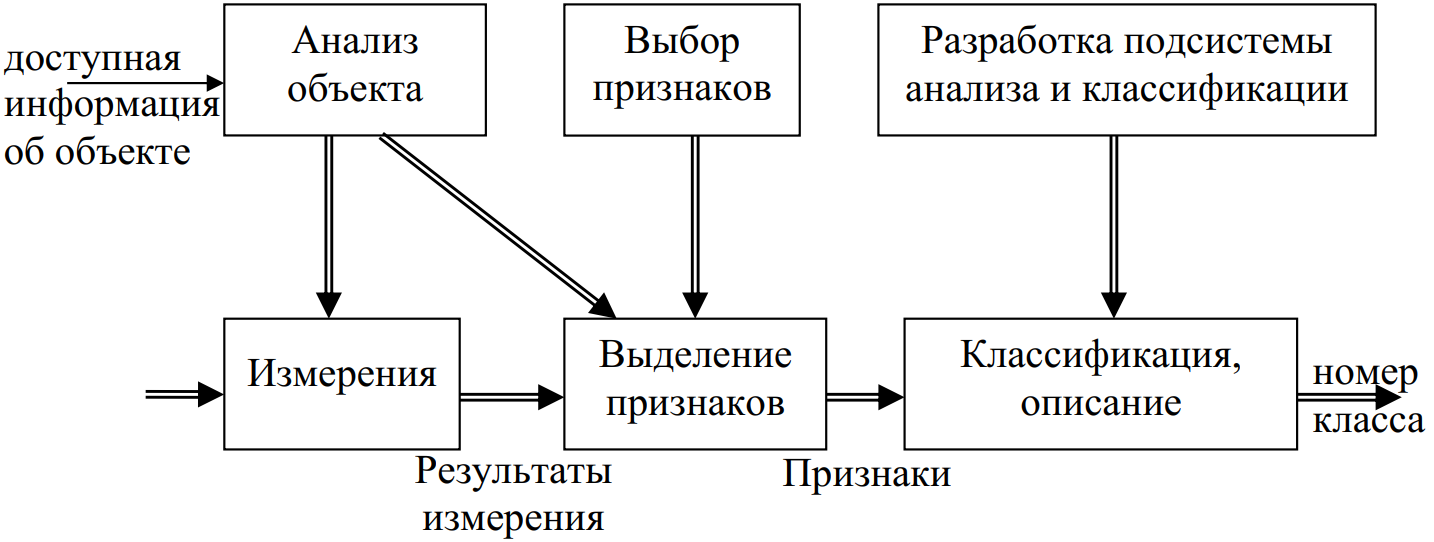


Рис. . Общая структура системы распознавания

В настоящее время разработано большое количество методов (моделей) решения задачи распознавания. Перечислим некоторые из них.

*Р-модель* (*модель разделения*) характеризуется тем, что проводиться граница между классами в пространстве Rn размерности n. При построении информационного вектора исследуется положение объекта относительно данной границы.

*Статистические модели* основаны на использовании аппарата математической статистики. Применяются тогда, когда могут быть определены вероятностные характеристики классов, например, функции распределения.

*П-модели* используют понятие потенциала, определенного для каждой точки пространства. Для определения принадлежности объекта классу вычисляются значения потенциальной функции.

В *моделях, основанных на вычислении оценок*, вычисление близости объектов между собой определяется совпадением их частичных описаний. Вычисляется обобщенная близость объекта классам, по которой и устанавливаются значения для информационного вектора .

На аппарате алгебры логики основаны *Л-модели*.

# Результаты работы

Первым делом, согласно поставленной задачи, было создано 2 файла, отвечающих за ввод «эталонных» и входных «неправильных» значений (см. Листинг 1, Листинг 1).

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*....\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

.....\*

...\*.\*

..\*..\*

.\*...\*

\*....\*

.....\*

.....\*

.....\*

.....\*

..\*\*\*.

.\*...\*

\*...\*.

...\*..

..\*...

.\*....

\*.....

\*.....

\*\*\*\*\*\*

\*\*\*\*\*\*

.....\*

.....\*

.....\*

\*\*\*\*\*\*

.....\*

.....\*

.....\*

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

.....\*

.....\*

.....\*

.....\*

\*\*\*\*\*\*

\*.....

\*.....

\*.....

\*\*\*\*..

....\*.

.....\*

....\*.

\*\*\*\*..

\*\*\*\*\*\*

\*.....

\*.....

\*.....

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

\*\*\*\*\*\*

.....\*

.....\*

.....\*

..\*\*\*\*

.....\*

.....\*

.....\*

.....\*

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

\*\*\*\*\*\*

\*....\*

\*....\*

\*....\*

\*\*\*\*\*\*

.....\*

.....\*

.....\*

.....\*

Листинг . Эталонные значения (etalon.txt)

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

\*....\* \*....\* \*....\* \*....\*

.....\* \*....\* \*....\* \*....\*

\*....\* \*....\* + \*....\* \*.....

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* + .....\* \*....\*

\*....\* \*....\* \*....\* \*....\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

.....\* .....\* .....\* .....\*

...\*.\* ...\*.\* ...\*.\* ...\*.\*

..\*..\* ..\*..\* ..\*..\* ..\*..\*

.\*...\* .\*...\* + .\*...\* .\*...\*

\*....\* \*....\* \*....\* \*....\*

.....\* .....\* .....\* .....\*

.....\* .....\* + .....\* .....\*

.....\* .....\* .....\* .....\*

.....\* .....\* .....\* .....\*

..\*.\*. ...... ...... ......

.\*...\* .\*...\* .\*...\* .\*...\*

\*...\*. \*...\*. \*...\*. \*...\*.

...\*.. ...\*.. + ...\*.. ...\*..

...... ...... ..\*... ..\*...

.\*.... ...\*.. .\*.... ......

...... \*.\*... + \*..... \*..\*..

\*..... \*..... \*..... \*.....

\*\*.\*.\* .\*\*.\*\* \*\*.\*.\* \*\*.\*.\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

.....\* .....\* .....\* .....\*

.....\* .....\* .....\* .....\*

.....\* .....\* + .....\* .....\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

.....\* .....\* .....\* .....\*

.....\* .....\* + .....\* .....\*

.....\* .....\* .....\* .....\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* + \*....\* \*....\*

\*\*\*\*\*\* \*\*.\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

.....\* .....\* .....\* .....\*

.....\* .....\* + .....\* .....\*

.....\* .....\* .....\* .....\*

.....\* .....\* .....\* .....\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

\*..... \*..... \*..... \*.....

\*..... \*..... \*..... \*.....

\*..... \*..... + \*..... \*.....

\*\*\*\*.. \*\*\*\*.. \*\*\*\*.. \*\*\*\*..

....\*. ....\*. ....\*. ....\*.

.....\* .....\* + .....\* .....\*

....\*. ....\*. ....\*. ....\*.

\*\*\*\*.. \*\*\*\*.. \*\*\*\*.. \*\*\*\*..

\*\*\*\*\*\* \*\*\*\*\*\* .\*\*.\*\* \*\*\*\*\*\*

\*..... \*..... \*..... \*.....

\*..... \*..... \*..... \*.....

\*..... \*..... + \*..... \*.....

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

\*....\* \*....\* \*....\* \*....\*

\*....\* \*....\* + \*....\* \*....\*

\*....\* \*....\* \*....\* \*....\*

\*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\* \*\*\*\*\*\*

Листинг 2. Поврежденные значения показаний времени (times.txt)

Средствами языка python были написана программа, код программы приведен на Листинге 3 (некоторые части сокращены «. . . . . . .», полный код программы в приложенном файле main.py).

class **NNDigitRecogniteModel**:

*"""*

*Класс Нейронной сети для распознавания*

*"""*

def **\_\_init\_\_**(*self*, digits\_in\_line: int=1, nn\_key: str=*'main'*):

*"""*

*Модель с определенным алиасом должна создаваться один раз*

*"""*

*self*.nn\_key = nn\_key

*self*.digits\_in\_line = digits\_in\_line

*self*.encoder = DEFAULT\_OH\_ENCODER

if os.path.isfile(*f'keras\_train\_model\_{self.nn\_key}{self.digits\_in\_line}.bin'*):

*self*.model = keras.models.load\_model(*f'keras\_train\_model\_{self.nn\_key}{self.digits\_in\_line}.bin'*)

else:

*self*.model = keras.Sequential()

*self*.model.add(keras.layers.Dense(input\_shape=(SIZE\_OF\_ONE\_IMAGE \* digits\_in\_line,), units=128, activation=*'relu'*))

*self*.model.add(keras.layers.Dense(10 \* digits\_in\_line, activation=*'softmax'*))

*self*.model.summary()

*self*.model.compile(optimizer=*'sgd'*,

loss=*'categorical\_crossentropy'*,

metrics=[*'accuracy'*])

*self*.labels\_vect\_dataset = []

*self*.images\_vect\_dataset = []

*self*.X\_train, *self*.X\_test, *self*.y\_train, *self*.y\_test = None, None, None, None,

def **\_dataset\_to\_ndarray**(*self*):

if len(*self*.labels\_vect\_dataset) and isinstance(*self*.labels\_vect\_dataset, list):

*self*.labels\_vect\_dataset = np.array(*self*.labels\_vect\_dataset)

if len(*self*.images\_vect\_dataset) and isinstance(*self*.images\_vect\_dataset, list):

*self*.images\_vect\_dataset = np.array(*self*.images\_vect\_dataset)

def **\_dataset\_to\_traintest**(*self*):

*self*.\_dataset\_to\_ndarray()

*self*.X\_train, *self*.X\_test, *self*.y\_train, *self*.y\_test = train\_test\_split(*self*.images\_vect\_dataset, *self*.labels\_vect\_dataset)

def **nn\_fit**(*self*):

*self*.\_dataset\_to\_traintest()

*self*.model.fit(*self*.X\_train, *self*.y\_train, epochs=60, batch\_size=128)

*self*.model.save(*f'keras\_train\_model\_{self.nn\_key}{self.digits\_in\_line}.bin'*)

*self*.model.evaluate(*self*.X\_test, *self*.y\_test)

def **nn\_predict**(*self*, image\_vectors=[]):

image\_vectors\_ = image\_vectors

if not isinstance(image\_vectors, (list, tuple, )):

image\_vectors\_ = [image\_vectors]

images\_vect = np.array(list(itertools.chain.from\_iterable(image\_vectors\_)))

if len(images\_vect) != *self*.digits\_in\_line \* SIZE\_OF\_ONE\_IMAGE:

raise ValueError

return *self*.model.predict(images\_vect.reshape((1, -1)))

def **nn\_predicted\_digit\_and\_prob**(*self*, image\_vectors=[]):

ret = *self*.nn\_predict(image\_vectors)

ret = ret.tolist()[0]

label = int(np.argmax(ret))

prob = ret[label]

return label, prob

def **add\_to\_dataset**(*self*, labels=[], image\_vectors=[]):

if not isinstance(labels, (list, tuple, )):

labels = [labels]

if not isinstance(image\_vectors, (list, tuple, )):

image\_vectors = [image\_vectors]

if len(labels) != *self*.digits\_in\_line or len(image\_vectors) != *self*.digits\_in\_line:

raise ValueError

none\_labels\_vect\_data = len(LABELS\_ONE\_HOT\_VECTORS[0])

none\_labels\_vect\_data = list(itertools.repeat(0, none\_labels\_vect\_data))

none\_labels\_vect\_data = np.array(none\_labels\_vect\_data)

def **get\_lab\_vect**(label):

if label is None:

return none\_labels\_vect\_data

return LABELS\_ONE\_HOT\_VECTORS[label]

labels\_vect = list(itertools.chain.from\_iterable(map(get\_lab\_vect, labels)))

images\_vect = list(itertools.chain.from\_iterable(image\_vectors))

*self*.labels\_vect\_dataset.append(labels\_vect)

*self*.images\_vect\_dataset.append(images\_vect)

NEURAL\_MODEL\_REGISTRY = {}

class **NNPixelDigit**:

*"""*

*Класс цифры*

*"""*

def **\_\_init\_\_**(*self*, string\_with\_lb: str, label: int=None, nn\_key: str=*'main'*):

. . . . . . .

*self*.image\_vect = *self*.bin\_vectorize\_block()

*self*.label = label

*self*.label\_vect = *self*.bin\_vectorize\_label()

def **bin\_vectorize\_block**(*self*):

*"""*

*С помощью numpy предстваить список "пикселей" как бинарный вектор*

*"""*

return np.array(list(map(lambda x: 1 if x == *'\*'* else 0, itertools.chain.from\_iterable(*self*.\_pixel\_data))))

def **bin\_vectorize\_label**(*self*):

*"""*

*С помощью sklearn предстваить метку как бинарный вектор, используем One-Hot кодирование (датасет для различий - все эталоны)*

*"""*

if not *self*.label:

return None

return LABELS\_ONE\_HOT\_VECTORS[*self*.label]

def **plot\_image**(*self*):

*"""*

*Нарисовать картинку из вектора изображения*

*"""*

plt.imshow(*self*.image\_vect.reshape((MATRIX\_MAX\_Y, MATRIX\_MAX\_X)), cmap=*'gray'*)

plt.show()

class **Clock**:

*"""*

*Класс Показаний часов*

*"""*

def **devide\_digits**(*self*, string\_time: str):

. . . . . . .

return time\_digits[0], time\_digits[1], time\_digits[2], time\_digits[3]

def **\_\_init\_\_**(*self*, string\_time: str, nn\_key: str=*'main'*):

. . . . . . .

*@property*

def **mid\_prob\_loss**(*self*):

digit\_nn = *self*.first\_digit.nn

if *self*.first\_digit.label is None and *self*.first\_prob == 0:

*self*.first\_digit.label, *self*.first\_prob = digit\_nn.nn\_predicted\_digit\_and\_prob(*self*.first\_digit.image\_vect)

if *self*.second\_digit.label is None and *self*.second\_prob == 0:

*self*.second\_digit.label, *self*.second\_prob = digit\_nn.nn\_predicted\_digit\_and\_prob(*self*.second\_digit.image\_vect)

if *self*.third\_digit.label is None and *self*.third\_prob == 0:

*self*.third\_digit.label, *self*.third\_prob = digit\_nn.nn\_predicted\_digit\_and\_prob(*self*.third\_digit.image\_vect)

if *self*.four\_digit.label is None and *self*.four\_prob == 0:

*self*.four\_digit.label, *self*.four\_prob = digit\_nn.nn\_predicted\_digit\_and\_prob(*self*.four\_digit.image\_vect)

mid\_prob\_loss = 1.0 - ((*self*.first\_prob +

*self*.second\_prob +

*self*.third\_prob +

*self*.four\_prob) / 4)

return mid\_prob\_loss

def **add\_to\_dataset**(*self*, labels: list=None):

. . . . . . .

def **\_\_str\_\_**(*self*):

if *self*.mid\_prob\_loss > 0.5:

return *'[Невозможно определить]'*

# Поправки для контекста "Часы и минуты"

if *self*.first\_digit.label > 2:

*self*.first\_digit.label = None

else:

if *self*.first\_digit.label == 2 and *self*.second\_digit.label > 4:

*self*.second\_digit.label = None

if *self*.third\_digit.label > 5:

*self*.third\_digit.label = None

if *self*.first\_digit.label is not None and *self*.second\_digit.label is not None and *self*.third\_digit.label is not None and *self*.four\_digit.label is not None:

return *f'{self.first\_digit.label}{self.second\_digit.label}:{self.third\_digit.label}{self.four\_digit.label}'*

return *'[Невозможно определить]'*

time = property(\_\_str\_\_)

if \_\_name\_\_ == *"\_\_main\_\_"*:

# Если в папке найдены образцы для обучения цифр

answer = *''*

training\_samples = glob.glob(os.path.join(os.path.dirname(\_\_file\_\_), *'train\_dataset'*, *'\*.txt'*))

if len(training\_samples):

while not answer:

print(*'В папке программы найдены образцы для тренировки сети, обучить сеть по ним? (y/n): '*)

answer = input()

answer = answer.strip()

if answer.lower() == *'y'*:

nn\_digit = NNDigitRecogniteModel(1, *'main'*)

NEURAL\_MODEL\_REGISTRY[*'main1'*] = nn\_digit

samples\_list = []

for sample in training\_samples:

training\_etalons = defaultdict(list)

\_\_pixels\_load(sample, training\_etalons)

for k, pixels in training\_etalons.items():

digit = NNPixelDigit(*'\n'*.join(pixels), label=k)

samples\_list.append((k, digit))

random.shuffle(samples\_list)

for k, dig in samples\_list:

nn\_digit.add\_to\_dataset(k, dig.bin\_vectorize\_block())

nn\_digit.nn\_fit()

# Код програмы

last\_clock = None

stop\_while = False

while not stop\_while:

print(*'Введите строку для распознавания: '*)

contents = []

while True:

try:

line = input()

except EOFError:

break

if not notempty(line):

break

contents.append(line)

txt = *'\n'*.join(contents)

rec\_time = Clock(txt)

print(*'''Введите сотвествующее время (для обучения нейронной сети) ,*

*восклицательный знак вначале говорит о несуществующем времени),*

*введите "recognite" или пустую строку для распознавания: '''*)

time\_str = input()

time\_str = time\_str.strip()

if time\_str == *''* or time\_str == *'recognite'*:

digit\_nn = rec\_time.first\_digit.nn

if len(digit\_nn.labels\_vect\_dataset) != 0 and len(digit\_nn.labels\_vect\_dataset) == len(digit\_nn.images\_vect\_dataset):

print(*'Дообучим сеть одиночных цифр'*)

digit\_nn.nn\_fit()

if not len(rec\_time.first\_digit.image\_vect):

rec\_time = last\_clock

print(*'Распознано время:'*)

print(rec\_time)

print(*f'Вероятность ошибки: {rec\_time.mid\_prob\_loss}'*)

stop\_while = True

continue

negate = False

if not time\_str or time\_str.startswith(*'!'*):

negate = True

time\_str = time\_str[1:].strip()

if time\_str:

hours, \_, minutes = time\_str.partition(*':'*)

if hours:

hours = hours.zfill(2)

hours = [int(hours[0]), int(hours[1])]

if minutes:

minutes = minutes.zfill(2)

minutes = [int(minutes[0]), int(minutes[1])]

if not negate:

rec\_time.add\_to\_dataset([hours[0], hours[1], minutes[0], minutes[1]])

else:

rec\_time.add\_to\_dataset([None, None, None, None])

elif negate:

rec\_time.add\_to\_dataset([None, None, None, None])

last\_clock = rec\_time

Вначале опишем класс **NNPixelDigit** который преобразует ввод цифры, закодированной в строке (пользовательский ввод) в нормализованный вектор который потом мы будем подавать на вход ИНС. Определим класс **NNDigitRecogniteModel** для работы с ИНС, для построения ИНС используется пакет tensorflow.keras. В классе **NNDigitRecogniteModel** определяем устройство (модель) ИНС. Определим последовательную модель с двумя полно-связными слоями, первый принимающий на вход 6х9=54 входа и имеющий 128 нейронов, на выходном слое 10 выходов. Класс **Clock** принимает пользовательский ввод из 4 цифр, разбивает его на экземпляры **NNPixelDigit** и при запросе показания времени обращается к методу **nn\_predicted\_digit\_and\_prob()** класса **NNDigitRecogniteModel** для вычисления веса наиболее подходящего выхода обученной нейронной сети, выходы пронумерованы 0,1, … ,9 в соответствии с распознанными цифрами.

Программа при каждом запуске может спросить пользователя обучить/дообучить ИНС, если в папке программы «/train\_dataset» имеются файлы изображений цифр, каждый отдельный файл должен содержать цифры строго в порядке определенном 0,1, … ,9, в ходе обучения образцы перемешиваются случайно во всем множестве папки train\_dataset. Результат обучения сохраняется в файле keras\_train\_model\_main1.bin. ИНС уменьшает ошибку с большим обучением, что можно пронаблюдать, вводя один и тот-же образец для распознания, предварительно дообучив программу из train\_dataset или в ходе исполнения программы.

Результаты работы программы представлены на рисунке 3.

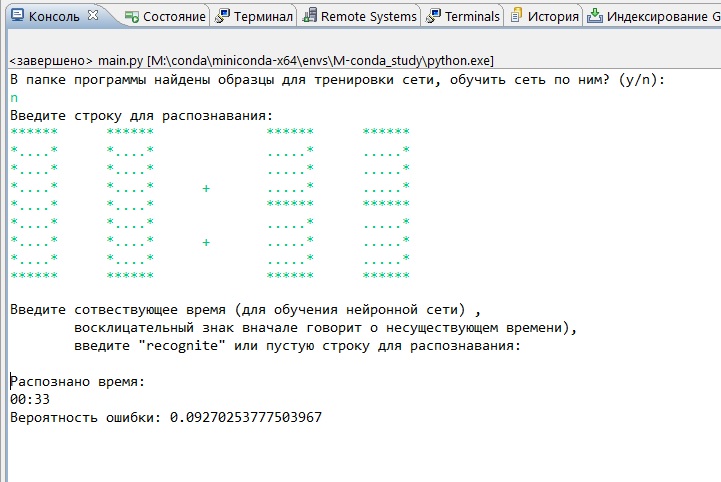


Рисунок 3. Результаты работы программы

# Вывод

В ходе лабораторной работы были изучены общие характеристики задачи распознавания и реализовано распознавание «битых» значений циферблата часов с использованием ИНС.

Материалы доступны на https://github.com/PaiNt-git/study.arch\_ai/blob/main/ src/lab3/main.py