ДОДАТОК А
ГРАФІЧНИЙ МАТЕРІАЛ КВАЛІКАЦІЙНОЙ РОБОТИ

Актуальність

В сучасному світі, де все швидко розвивається і наука досягла значних успіхів, необхідність в автоматизації процесів стала дедалі вищою. Через це з'являється потреба в нових інноваційних технологіях, які можуть спростити і полегшити роботу людей та організацій.

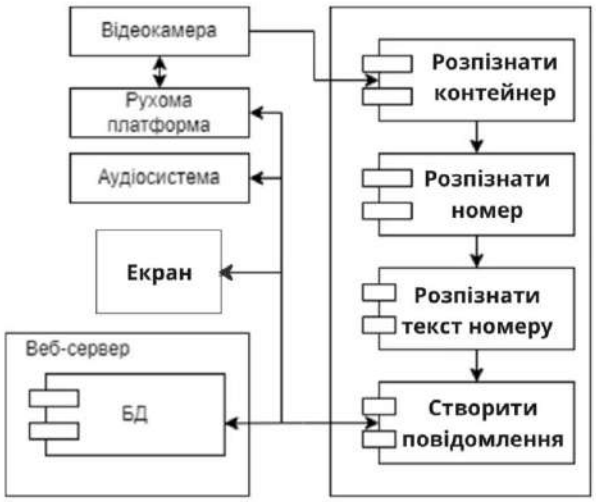
Причини актуальність роботи:

- Автоматизація логістичних операцій
- Зменшення помилок(людський фактор)
- Підвищення безпеки
- Вартісна ефективність

Можливі реалізації

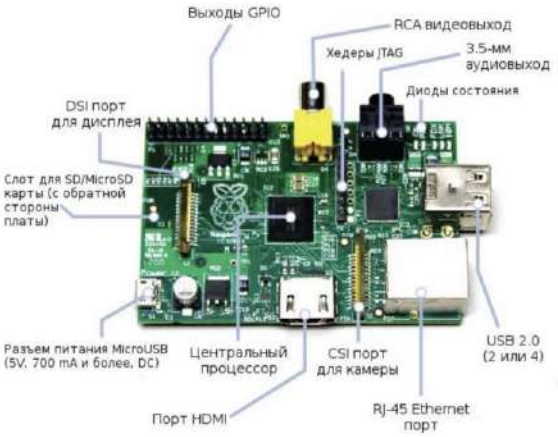
- Метод шаблонів
- Метод на основі машинного навчання
- Метод на основі штучних нейронних мереж

Фізична модель системи



5

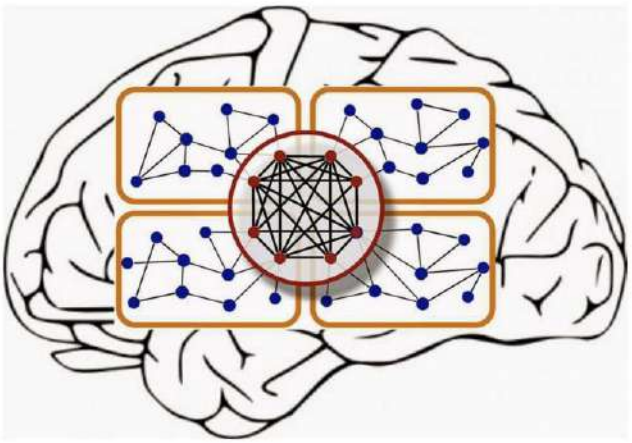
Вибір робота



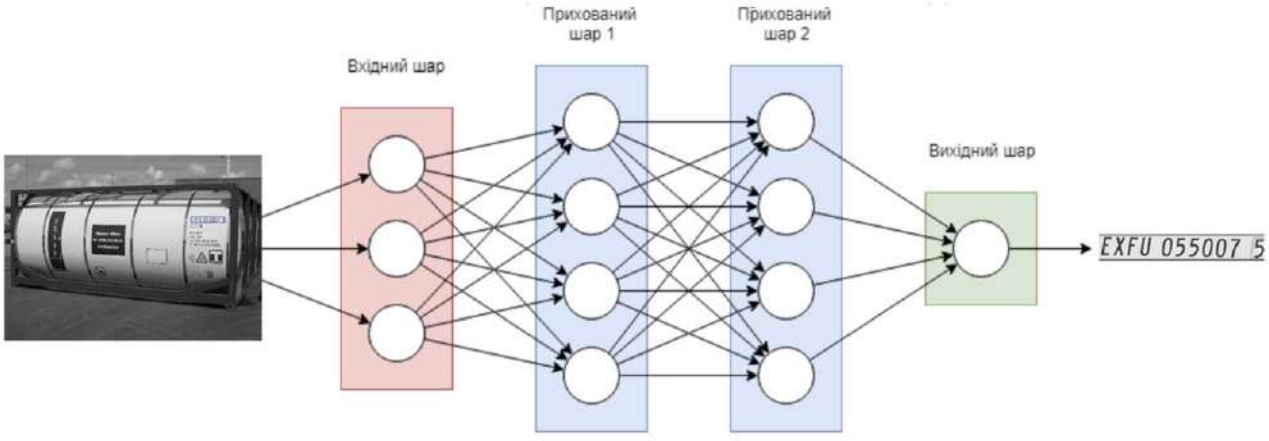
6

Штучна нейронна мережа

ШНМ - це математична, програмна та апаратна модель, побудована за принципом функціонування біологічних нейронних мереж клітин живого організму. Біологічні нейрони передають одне одному хімічні сигнали за допомогою електричних імпульсів.



Побудова нейронної мережі



8

Тестування

Вхідний набір

Результат роботи системи

9

Висновки

У результаті роботи була розроблена система розпізнавання номерів контейнеру за допомогою веб-камери, встановленої на мобільному колісному роботі, яка може реалізовувати такі функції:

- Знаходження контейнеру;
- Зчитування номеру;
- Обробка результатів.
- Збереження даних або вивід на екран;

10

ДОДАТОК Б

Структура нейромережі для знаходженням номеру контейнеру на зображені

```
def unet_carnum(num_classes, input_shape):
    img_input = Input(input_shape)

    # Block 1
    x = Conv2D(64, (4, 4), padding='same', name='block1_conv1')(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(64, (4, 4), padding='same', name='block1_conv2')(x)
    x = BatchNormalization()(x)
    block_1_out = Activation('relu')(x)

    x = MaxPooling2D()(block_1_out)

    # Block 2
    x = Conv2D(128, (4, 4), padding='same', name='block2_conv1')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(128, (4, 4), padding='same', name='block2_conv2')(x)
    x = BatchNormalization()(x)
    block_2_out = Activation('relu')(x)

    x = MaxPooling2D()(block_2_out)

    # Block 3
    x = Conv2D(256, (4, 4), padding='same', name='block3_conv1')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(256, (4, 4), padding='same', name='block3_conv2')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)

    x = Conv2D(256, (4, 4), padding='same', name='block3_conv3')(x)
    x = BatchNormalization()(x)
    block_3_out = Activation('relu')(x)

    x = MaxPooling2D()(block_3_out)

    # Block 4
    x = Conv2D(512, (3, 3), padding='same', name='block4_conv1')(x)x =
    BatchNormalization()(x)  x = Activation('relu')(x)
```

```
x = Conv2D(512, (3, 3), padding='same', name='block4_conv2')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = Conv2D(512, (3, 3), padding='same', name='block4_conv3')(x)
x = BatchNormalization()(x)
block_4_out = Activation('relu')(x)
x = block_4_out


# UP 2
x = Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = concatenate([x, block_3_out])
x = Conv2D(256, (3, 3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = Conv2D(256, (3, 3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


# UP 3
x = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = concatenate([x, block_2_out])
x = Conv2D(128, (4, 4), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = Conv2D(128, (4, 4), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


# UP 4
x = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)


x = concatenate([x, block_1_out])
```

```
    x = Conv2D(64, (4, 4), padding='same')(x)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)


    x = Conv2D(64, (4, 4), padding='same')(x)
    x =
    BatchNormalization()(x)

    x = Activation('relu')(x)

    x = Conv2D(num_classes, (3, 3), activation='softmax', padding='same')(
x)
```

ДОДАТОК В

Структура нейромережі розпізнавання символів номер контейнера

```
def signs(input_shape, cls):
  inp = Input (shape=input_shape, name="inp")
  x0 = Conv2D (16, 4, strides=2, padding="same",
       activation="linear", name="x0") (inp)
  x1 = Conv2DTranspose 16, 4, strides=1, padding="same",
       activation="linear", name="x1") (x0)
  x2 = Conv2DTranspose (16, 4, strides=1, padding="same",
       activation="tanh", name="x2") (x1)
  x3 = Conv2D (16, 4, strides=1, padding="same",
       activation="linear", name="x3") (x2)
  x4 = Concatenate (name="x4") ([x3, x2, x1])
  x5 = MaxPooling2D (4, strides=1, padding="same", name="x5" ) (x4)
  x6 = Conv2DTranspose (32, 4, strides=1, padding="same",
       activation="tanh", name="x6") (x5)
  x7 = Conv2DTranspose (16, 4, strides=1, padding="same",
       activation="relu", name="x7") (x6)
  x8 = BatchNormalization (name="x8") (x7)
  x_last = Conv2DTranspose (16, 4, strides=1, padding="same",
           activation="relu", name="x_last") (x8)
  fin = Reshape ((20, 13, -1), name="fin") (x_last)
  fin = BatchNormalization (name="fin_bn") (fin)
  fin_fl = Flatten (name="fin_fl") (fin)
  last = Dense (cls, activation="softmax", name="last") (fin_fl)
  out = Reshape ((cls,), name="out") (last)
```