# ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ «САРТСНА»

**Класифікація:**

- введення тексту, цифр, символів;
- вибір із певної кількості картинок;
- рішення математичної задачі;
- reCAPTCHA - підтвердження дії.
-

**Навіщо потрібна CAPTCHA:**

- захист від спаму;
- захист від DDoS-атак;
- захист від брутфорсингу, або підбору логінів та паролів;
- захист від перехоплення товарів у інтернет-магазинах;
- захист від перехоплення товарів у інтернет-магазинах;

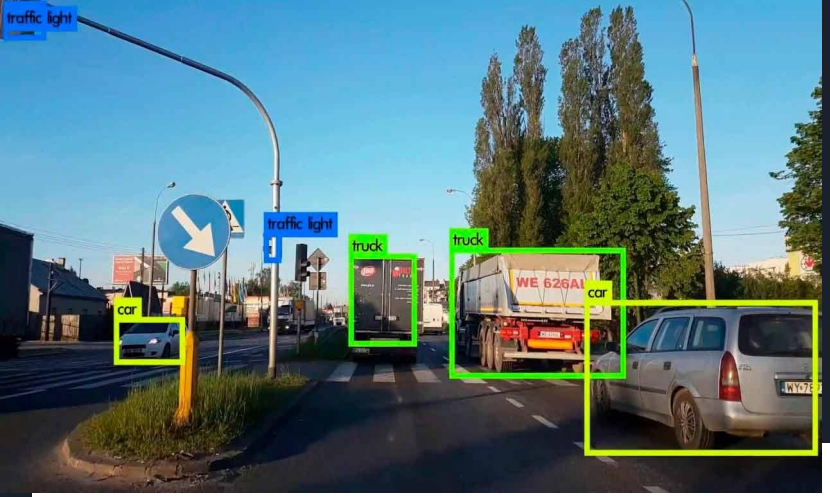
## Мета розробки нейронної мережі для зчитування данних із зображень (CAPTCHA)

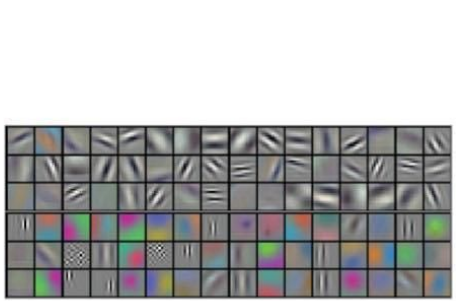
CAPTCHA призначена для запобігання спаму, злочинної діяльності та автоматизованого доступу до ресурсів, але в той же час вона може бути складною для людей, які зустрічаються з нею. Таким чином, розробка ефективної нейронної мережі для зчитування даних з зображень CAPTCHA може допомогти у зниженні труднощів, пов'язаних з використанням CAPTCHA для користувачів та допомогти в забезпеченні безпеки ресурсу від автоматизованого доступу.

**Деякі з найбільш відомих готових нейронних мереж для розпізнавання капчі включають:**

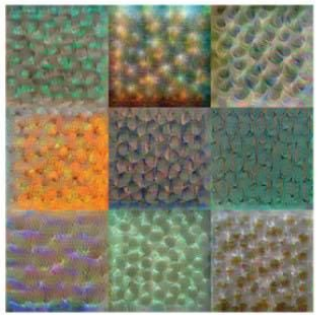

- CNN + RNN
- Google's reCAPTCHA
- Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks
- Deep Learning for Captcha Recognition
- Deep Captcha

# CNN (Згорткові нейронні мережі)



Edge+Blob



Texture

Numerical

Data-driven

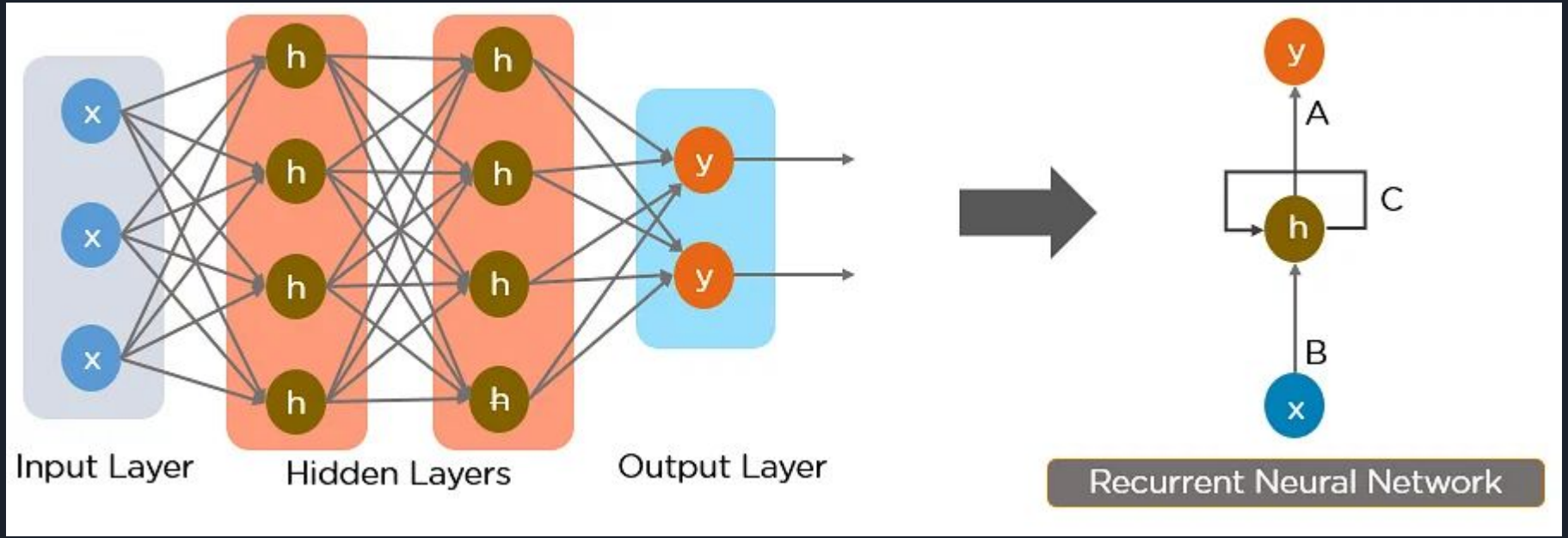
Object Parts


Object Classes

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Завантаження даних
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

# Нормалізація даних
x_train = x_train.astype("float32") / 255.0
x_test = x_test.astype("float32") / 255.0

# Додавання каналу кольору
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)

# Створення моделі
model = keras.Sequential(
    [
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu", input_shape=(28, 28, 1)),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(10, activation="softmax"),
    ]
)
```

```python
# Компіляція моделі
model.compile(loss="sparse_categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

# Навчання моделі
model.fit(x_train, y_train, batch_size=128, epochs=15, validation_split=0.1)

# Оцінка точності на тестових даних
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy:", test_acc)
```

# RNN (рекурентні нейронні мережі)



```python
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D, LSTM, Dense, Bidirectional

# Визначення архітектури згорткової нейронної мережі
def create_cnn_model(input_shape):
    model = tf.keras.Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(256, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    return model

# Визначення архітектури рекурентної нейронної мережі
def create_rnn_model(input_shape, num_classes):
    model = tf.keras.Sequential()
    model.add(Bidirectional(LSTM(128, return_sequences=True), input_shape=input_shape))
    model.add(Bidirectional(LSTM(64)))
    model.add(Dense(num_classes, activation='softmax'))
    return model
```

```
# Завантаження та підготовка даних
(X_train, y_train), (X_test, y_test) = load_captcha_data()
X_train = X_train / 255.0
X_test = X_test / 255.0
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)

# Визначення розмірності вхідних даних
input_shape = X_train.shape[1:]

# Створення та компіляція згорткової нейронної мережі
cnn_model = create_cnn_model(input_shape)
cnn_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# Навчання згорткової нейронної мережі
cnn_model.fit(X_train, y_train, batch_size=32, epochs=10, validation_data=(X_test, y_test))

# Відокремлення символів на зображенні з допомогою згорткової нейронної мережі
captcha_image = load_captcha_image('captcha.png')
captcha_chars = cnn_model.predict(captcha_image)

#Визначення кількості символів у капчі та кількості можливих значень для кожного символу
num_chars = captcha_chars.shape[0]
num_classes = captcha_chars.shape[1]
```

```python
#Створення та компіляція рекурентної нейронної мережі
rnn_model = create_rnn_model((num_chars, num_classes), num_classes)
rnn_model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

#Навчання рекурентної нейронної мережі
rnn_model.fit(captcha_chars, y_train, batch_size=32, epochs=10)

#Розпізнавання символів на капчі з допомогою рекурентної нейронної мережі
predicted_chars = rnn_model.predict(captcha_chars)

# Перетворення прогнозованих значень у текст
predicted_text = ''
for i in range(num_chars):
predicted_index = np.argmax(predicted_chars[i])
predicted_text += char_set[predicted_index]

print(predicted_text)
```