****

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА  
(САМАРСКИЙ УНИВЕРСИТЕТ)»

ИНСТИТУТ ИНФОРМАТИКИ И КИБЕРНЕТИКИ

Кафедра программных систем

Дисциплина

Нейронные сети

ОТЧЕТ

по лабораторной работе №5

Студент: Соколова А.Д.

Группа: 6301-020302D

Проверил:

профессор Тюгашев А.А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Самара  
2025

СОДЕРЖАНИЕ

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc197905845)

[1 Исходный текст программы 4](#_Toc197905846)

[2 Протокол исполнения 7](#_Toc197905847)

[ЗАКЛЮЧЕНИЕ 9](#_Toc197905848)

# ПОСТАНОВКА ЗАДАЧИ

Цель работы ознакомиться с принципами построения сверточных нейронных сетей (CNN) для анализа изображений. Реализуется классификация рукописных японских символов катакана с помощью CNN и распознавание синтетических геометрических фигур.

В рамках выполнения работы предполагается выполнение следующих задач:

1. реализация сверточной нейронной сети на языке Python с использованием библиотеки Keras;
2. загрузка и предварительная обработка изображений из датасета Kuzushiji-MNIST;
3. обучение модели и оценка её точности на тестовых данных;
4. визуализация результатов и интерпретация поведения модели в процессе обучения.
5. Исходный текст программы

import numpy as np

import matplotlib.pyplot as plt

import cv2

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.utils import to\_categorical

data\_path = "C:/Users/llcky/Documents/datasets/kmnist/"

x\_train = np.load(data\_path + "kmnist-train-imgs.npz")['arr\_0']

y\_train = np.load(data\_path + "kmnist-train-labels.npz")['arr\_0']

x\_test = np.load(data\_path + "kmnist-test-imgs.npz")['arr\_0']

y\_test = np.load(data\_path + "kmnist-test-labels.npz")['arr\_0']

# предобработка

x\_train = x\_train.astype("float32") / 255.0

x\_test = x\_test.astype("float32") / 255.0

x\_train = np.expand\_dims(x\_train, -1) # (60000, 28, 28, 1)

x\_test = np.expand\_dims(x\_test, -1) # (10000, 28, 28, 1)

# One-hot кодирование меток

y\_train\_cat = to\_categorical(y\_train, 10)

y\_test\_cat = to\_categorical(y\_test, 10)

class\_names = [f"Класс {i}" for i in range(10)]

# создание модели CNN

model = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

MaxPooling2D(2, 2),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D(2, 2),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.3),

Dense(10, activation='softmax')

])

# компиляция и обучение модели

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(x\_train, y\_train\_cat, epochs=10, batch\_size=64, validation\_split=0.2)

# оценка точности на тесте

test\_loss, test\_acc = model.evaluate(x\_test, y\_test\_cat)

print(f"Точность на тестовой выборке: {test\_acc:.4f}")

# график точности

plt.figure()

plt.plot(history.history['accuracy'], label='Train')

plt.plot(history.history['val\_accuracy'], label='Validation')

plt.title('KMNIST: Точность на обучении и валидации')

plt.xlabel('Эпохи')

plt.ylabel('Точность')

plt.legend()

plt.grid(True)

plt.savefig("kmnist\_accuracy.png")

# визуализация предсказаний

np.random.seed(42)

indices = np.random.choice(len(x\_test), 10, replace=False)

sample\_images = x\_test[indices]

true\_labels = np.argmax(y\_test\_cat[indices], axis=1)

pred\_labels = np.argmax(model.predict(sample\_images), axis=1)

fig, axes = plt.subplots(2, 5, figsize=(12, 5))

for i, ax in enumerate(axes.ravel()):

ax.imshow(sample\_images[i].squeeze(), cmap='gray')

ax.set\_title(f"True: {class\_names[true\_labels[i]]}\nPred: {class\_names[pred\_labels[i]]}", fontsize=10)

ax.axis('off')

plt.tight\_layout()

plt.savefig("kmnist\_predictions.png")

# синтетика

def generate\_synthetic\_data(num\_samples=1000, img\_size=32):

images, labels = [], []

for \_ in range(num\_samples):

img = np.zeros((img\_size, img\_size), dtype=np.float32)

center = (img\_size // 2, img\_size // 2)

radius = np.random.randint(10, 14)

cv2.circle(img, center, radius, 1.0, thickness=-1)

label = np.random.choice([0, 1]) # 0 - годное, 1 - брак

if label == 1:

if np.random.rand() < 0.5:

p1 = (np.random.randint(0, img\_size), np.random.randint(0, img\_size))

p2 = (np.random.randint(0, img\_size), np.random.randint(0, img\_size))

cv2.line(img, p1, p2, 0.0, 1)

if np.random.rand() < 0.5:

noise = np.random.rand(img\_size, img\_size) < 0.05

img[noise] = 0.0

if np.random.rand() < 0.5:

pts = [(np.random.randint(0, img\_size), np.random.randint(0, img\_size)) for \_ in range(3)]

for i in range(2):

cv2.line(img, pts[i], pts[i + 1], 0.0, 1)

images.append(img)

labels.append(label)

return np.array(images), np.array(labels)

train\_imgs, train\_lbls = generate\_synthetic\_data(1000)

test\_imgs, test\_lbls = generate\_synthetic\_data(200)

train\_imgs = np.expand\_dims(train\_imgs, -1)

test\_imgs = np.expand\_dims(test\_imgs, -1)

train\_lbls\_cat = to\_categorical(train\_lbls, 2)

test\_lbls\_cat = to\_categorical(test\_lbls, 2)

model\_synth = Sequential([

Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 1)),

MaxPooling2D((2, 2)),

Conv2D(64, (3, 3), activation='relu'),

MaxPooling2D((2, 2)),

Flatten(),

Dense(64, activation='relu'),

Dropout(0.5),

Dense(2, activation='softmax')

])

model\_synth.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

history\_synth = model\_synth.fit(train\_imgs, train\_lbls\_cat, epochs=10, batch\_size=32, validation\_split=0.2)

test\_loss\_synth, test\_acc\_synth = model\_synth.evaluate(test\_imgs, test\_lbls\_cat)

print(f"Точность на тестовой выборке (синтетика): {test\_acc\_synth:.2f}")

# график точности (синтетика)

plt.figure()

plt.plot(history\_synth.history['accuracy'], label='Train')

plt.plot(history\_synth.history['val\_accuracy'], label='Validation')

plt.title('Синтетика: Точность на обучении и валидации')

plt.xlabel('Эпохи')

plt.ylabel('Точность')

plt.legend()

plt.grid(True)

plt.savefig("synthetic\_accuracy.png")

# визуализация предсказаний

new\_imgs, new\_lbls = generate\_synthetic\_data(10)

new\_imgs\_exp = np.expand\_dims(new\_imgs, -1)

new\_preds = np.argmax(model\_synth.predict(new\_imgs\_exp), axis=1)

fig, axes = plt.subplots(2, 5, figsize=(10, 4))

for i, ax in enumerate(axes.flat):

ax.imshow(new\_imgs[i], cmap='gray')

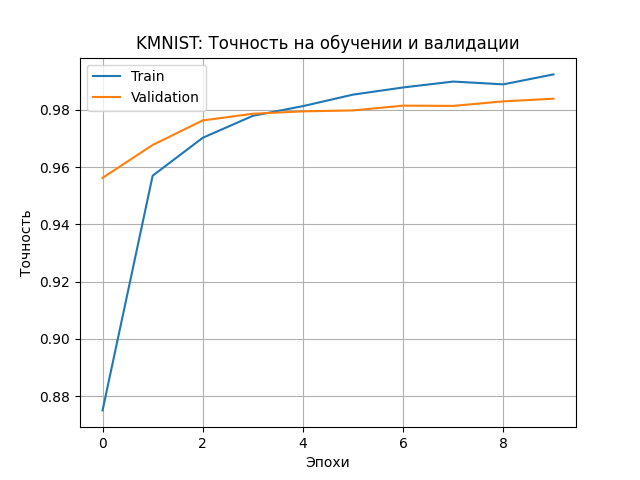
ax.set\_title(f"True: {new\_lbls[i]}, Pred: {new\_preds[i]}")

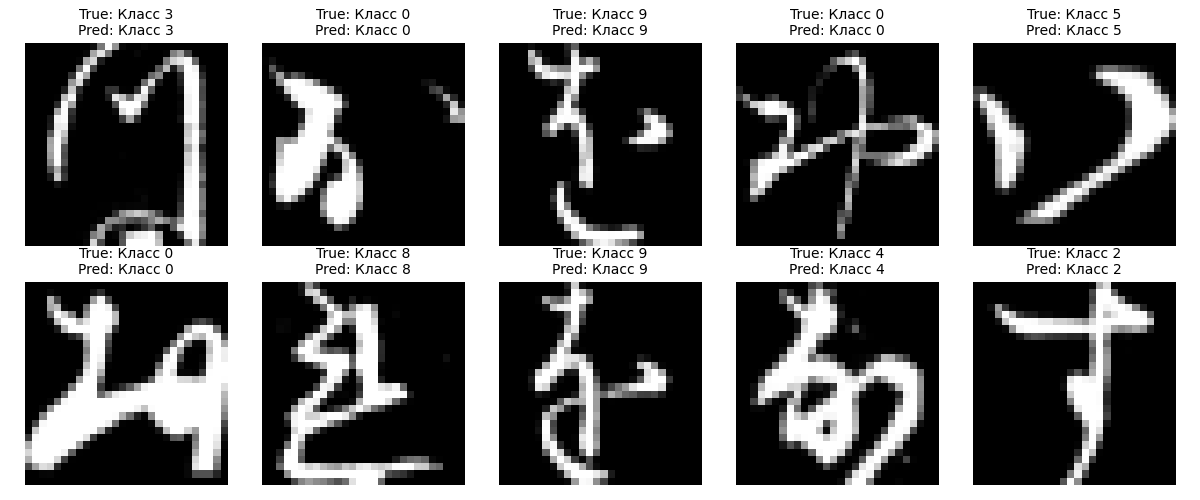
ax.axis('off')

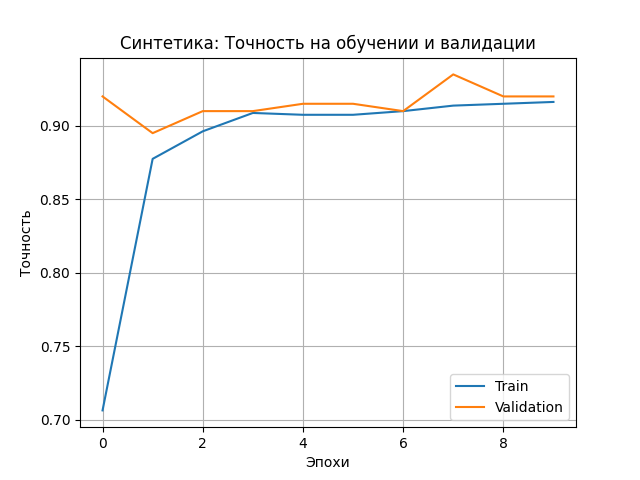
plt.tight\_layout()

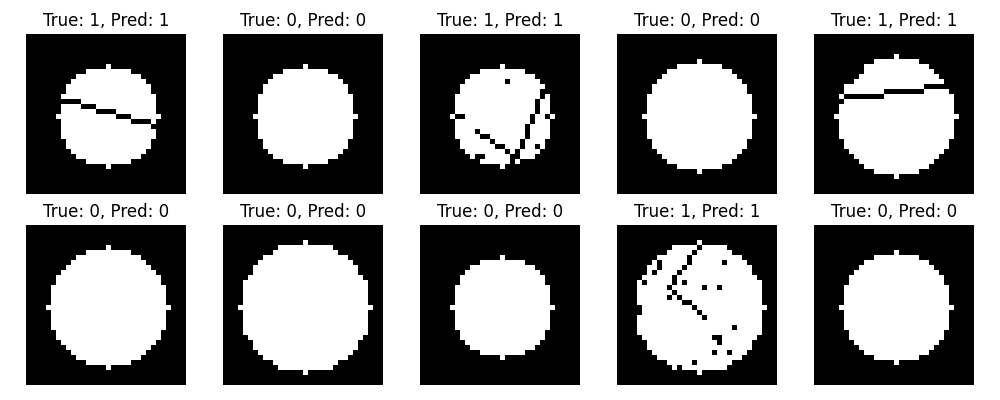
plt.savefig("synthetic\_predictions.png")

1. Протокол исполнения

  
Рисунок 1 – График точности на KMNIST

  
Рисунок 2 – Примеры предсказаний KMNIST

  
Рисунок 3 – График точности на синтетических данных

  
Рисунок 4 – Примеры предсказаний на синтетических изделиях

# ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы были успешно изучены и реализованы сверточные нейронные сети на практике. Модель показала высокую точность при распознавании изображений японских иероглифов из базы Kuzushiji-MNIST. Были применены стандартные приёмы регуляризации, такие как Dropout, что позволило снизить риск переобучения. Полученные результаты подтверждают эффективность использования сверточных сетей в задачах классификации изображений и демонстрируют потенциал их применения в сфере компьютерного зрения.