НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ**​** ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**​**

Факультет прикладної математики

Кафедра прикладної математики

Звіт

Лабораторна робота №4

із дисципліни «Методи штучного інтелекту»

Тема: « Розпізнавання двовимірних кольорових об’єктів за допомогою згорткової нейронної мережі»

Виконала:

студентка групи КМ-81

Верзун Поліна Валеріївна

Керівник:

Терейковська Л.О.

Київ **​**—**​**2021

Зміст

[Вступ 2](#_Toc57684266)

[Теоретичні відомості 2](#_Toc57684267)

[Хід роботи 6](#_Toc57684268)

[Висновки 9](#_Toc57684269)

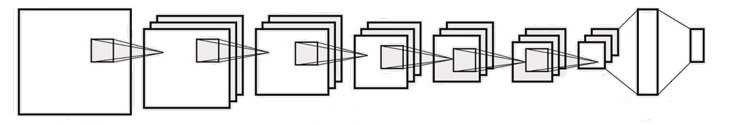
[Лістинг 10](#_Toc57684270)

# Вступ

*Мета роботи*: Отримання практичних навичок з розробки програмного забезпечення для реалізації згорткової нейронної мережі, призначеної для розпізнавання двовимірних кольорових об’єктів.

# Теоретичні відомості

Згорткові нейронні мережі (convolutional neural network, CNN, ConvNet) в машинному навчанні — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень. ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання мінімального обсягу попередньої обробки. Вони відомі також як інваріа́нтні відно́сно зсу́ву (англ. shift invariant) або просторо́во інваріа́нтні шту́чні нейро́нні мере́жі (англ. space invariant artificial neural networks, SIANN), виходячи з їхньої архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення.

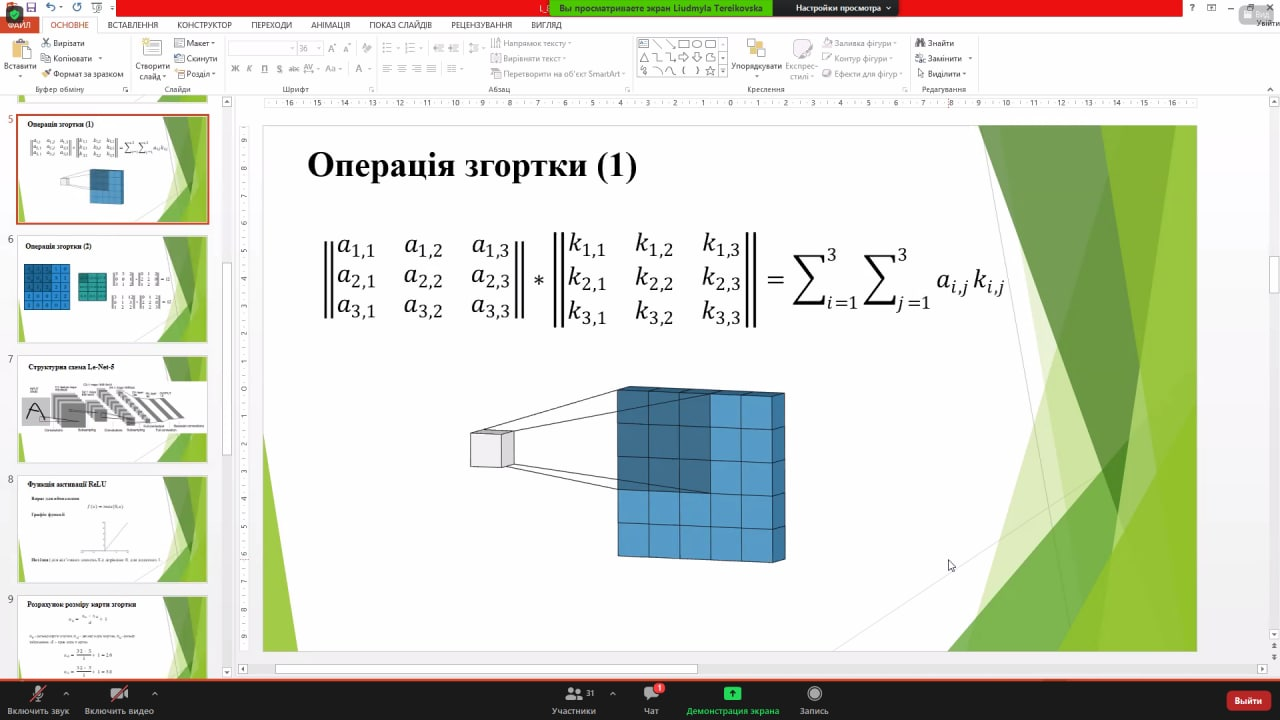


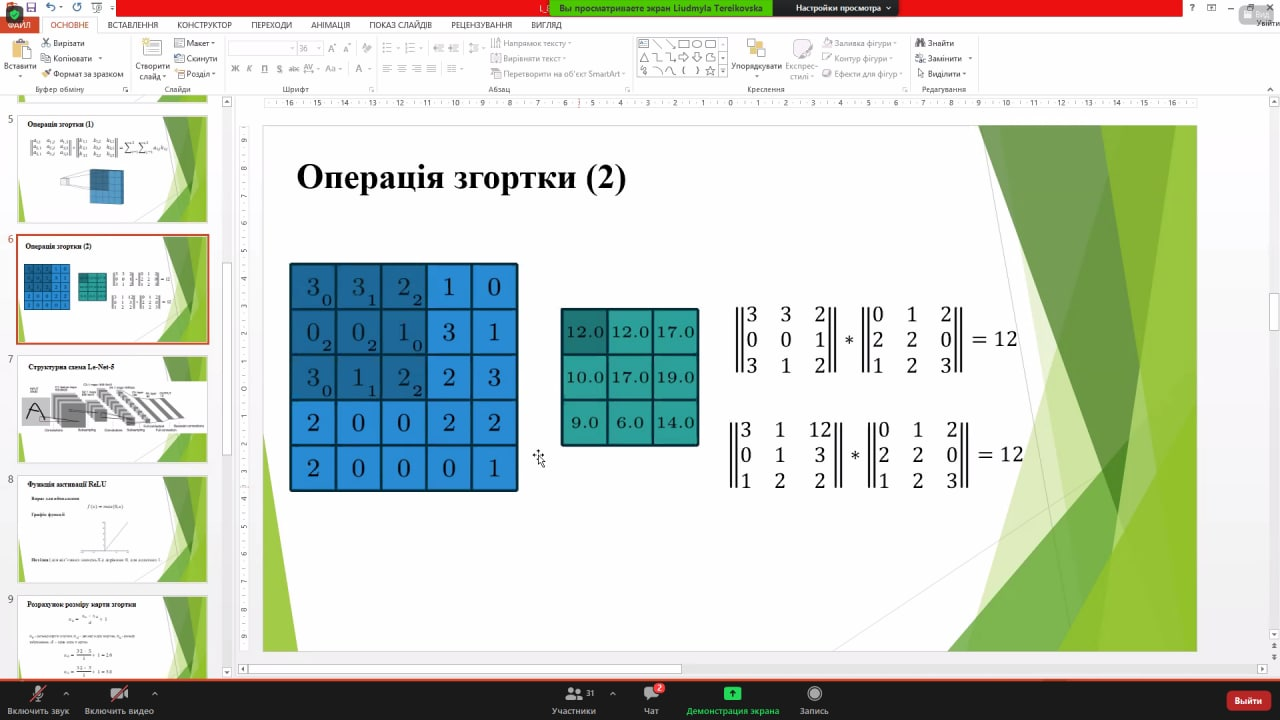
Структура використаної згорткової нейронної мережі

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації. Цей процес описують в нейронних мережах як згортку за домовленістю. З математичної точки зору він є радше взаємною кореляцією, ніж згорткою. Це має значення лише для індексів у матриці, й відтак які ваги на якому індексі розташовуються.

***Згорткові шари***:

Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул. Кожен згортковий нейрон обробляє дані лише для свого рецептивного поля.

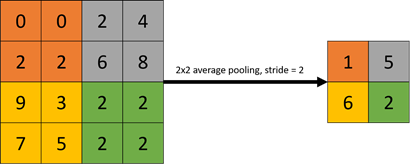




***Агрегувальні шари***:

Згорткові мережі можуть включати шари локального або глобального агрегування, які об'єднують виходи кластерів нейронів одного шару до одного нейрону наступного шару. Наприклад, максимізаційне агрегування (max pooling) використовує максимальне значення з кожного з кластерів нейронів попереднього шару. Іншим прикладом є усереднювальне агрегування (average pooling), що використовує усереднене значення з кожного з кластерів нейронів попереднього шару.



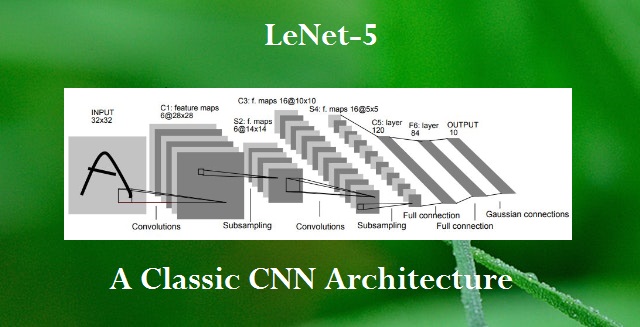


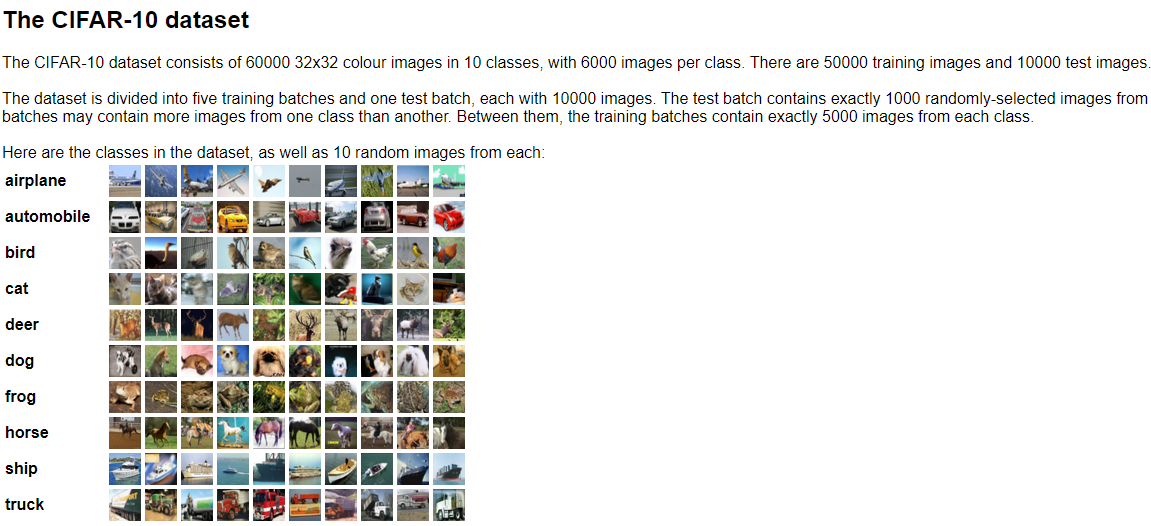
***Повноз’єнані шари***:

Повноз'єднані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Це є тим же, що й традиційна нейронна мережа багатошарового перцептрону.

***LeNet-5***

LeNet-5, новаторську 7-рівневу згорткову мережу від ЛеКуна 1998 року, яка класифікує цифри, було застосовано декількома банками для розпізнавання рукописних цифр на чеках, оцифровуваних у зображення 32×32 пікселів. Здатність оброблювати зображення вищої роздільності вимагає більших та згортковіших шарів, тож цю методику обмежено наявністю обчислювальних ресурсів.





# Хід роботи

Імпортуємо необхідні бібліотеки:

import numpy

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Flatten, Activation

from keras.layers import Dropout

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.utils import np\_utils

from keras.optimizers import SGD

Завантажуємо дані

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

Дані мають структуру

X\_train.shape

(50000, 32, 32, 3)

50тис зразків (зображень), зразок складається з масиву 32×32 (пікселів), кожен елемент масива масив розмірністю 3 (кількість каналів кольору).

Задаємо параметри:

Розмір міні-вибірки – кількість навчальних прикладів за одну ітерацію. Чим більше batch size, тим більше місця буде необхідно

batch\_size = 32

# Кількість класів зображень

nb\_classes = 10

# Кількість епох навчання

nb\_epoch = 25

# Розмір зображення

img\_rows, img\_cols = 32, 32

# Кількість каналів: RGB

img\_channels = 3

Нормалізуємо дані:

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

Використаємо наступний метод для унітарного кодування:

Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

Створюємо нейромережеву модель:

model = Sequential()

Перший та другий шари згортки:

model.add(Conv2D(32, (3, 3), padding='same',

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

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

де 32 – кількість фільтрів (застосувань ядра),(3, 3) -розмір ядра

Перший шар субдискретизації:

model.add(MaxPooling2D(pool\_size=(2, 2)))

Перший шар Dropout (запобігання перенавчанню, деактивується частина нейронів з ймовірністю, 0.25 – доля нейронів, які відкидаються)

model.add(Dropout(0.25))

Третій та четвертий шари згортки

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), activation='relu'))

Другий шар субдискретизації та дропаут:

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.25))

Шар перетворення вхідних даних (розкладання n вимірного масиву в одну строчку)

model.add(Flatten())

Повнозв’язний шар. Це означає, що активність всіх вузлів в одному шарі переходить на кожен вузол в наступному. В такому випадку шари будуть повністю пов'язаними.

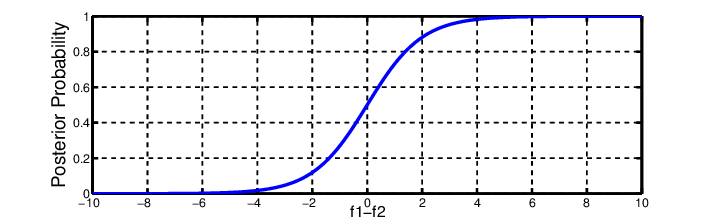
model.add(Dense(512, activation='relu'))

Третій шар дропаут та вихідний шар:

model.add(Dropout(0.5))

model.add(Dense(nb\_classes, activation='softmax'))

Використані функції активації ReLu та softmax:





Параметри оптимізації:

Для оновлення ваг в моделі використовуються оптимізатор SGD (Stochastic Gradient Descent).

За допомогою функції втрати (loss function) ви можете вирахувати помилку в конкретної частини навчання. Це середнє значення функції для навчання - 'categorical\_crossentropy' - для мультікласової логарифмічної.

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd,

metrics=['accuracy'])

Навчання моделі:

model.fit(X\_train, Y\_train,

batch\_size=batch\_size,

epochs=nb\_epoch,

validation\_split=0.1,

shuffle=True,

verbose=2)

Оцінка якості навчання на тестових даних:

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("Accuracy on test data: %.2f%%" % (scores[1]\*100))

Режим розпізнавання зображень:

model.predict(x)

# Висновки

В ході виконання лабораторної роботи виконана розробка програмного забезпечення для реалізації згорткової нейронної мережі, призначеної для розпізнавання двовимірних кольорових об’єктів.

Досягнена точність на тестових даних:

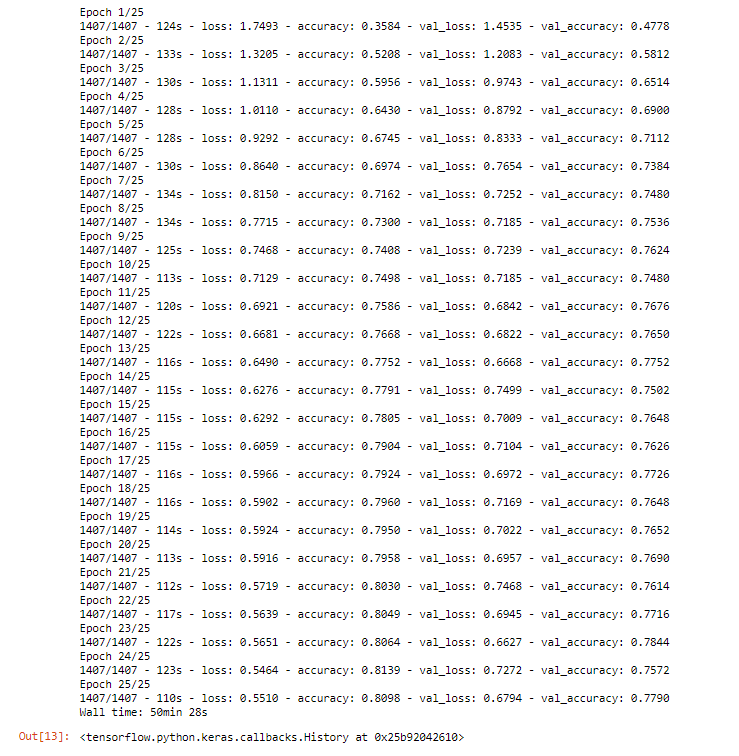
Accuracy on test data: 76.66%

Розпізнавання відбувається у вигляді:

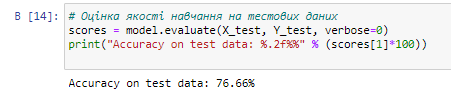
Розпізнавання картинки «кіт»:



Навчання за 25 епох:



Оцінка точності на тестових даних:



# Лістинг

import numpy

from keras.datasets import cifar10

from keras.models import Sequential

from keras.layers import Dense, Flatten, Activation

from keras.layers import Dropout

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.utils import np\_utils

from keras.optimizers import SGD

(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()

# Розмір міні-вибірки

batch\_size = 32

# Кількість класів зображень

nb\_classes = 10

# Кількість епох навчання

nb\_epoch = 25

# Розмір зображення

img\_rows, img\_cols = 32, 32

# Кількість каналів: RGB

img\_channels = 3

# Нормалізація даних

X\_train = X\_train.astype('float32')

X\_test = X\_test.astype('float32')

X\_train /= 255

X\_test /= 255

Y\_train = np\_utils.to\_categorical(y\_train, nb\_classes)

Y\_test = np\_utils.to\_categorical(y\_test, nb\_classes)

# Створення нейромережевої моделі

model = Sequential()

# Перший шар згортки

model.add(Conv2D(32, (3, 3), padding='same',

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

# Друний шар згортки

model.add(Conv2D(32, (3, 3), activation='relu', padding='same'))

# Перший шар субдискретизаії

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Перший шар Dropout

model.add(Dropout(0.25))

# Третій шар згортки

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

# Четвертий шар згортки

model.add(Conv2D(64, (3, 3), activation='relu'))

# Другий шар субдисктеризації

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Другий шар Dropout

model.add(Dropout(0.25))

# Шар перетворення вхідних даних

model.add(Flatten())

# Повнозв’язний шар

model.add(Dense(512, activation='relu'))

# Третій шар Dropout

model.add(Dropout(0.5))

# Вихідний шар

model.add(Dense(nb\_classes, activation='softmax'))

# Параметри оптимізації

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd,

metrics=['accuracy'])

%%time

# Навчання моделі

model.fit(X\_train, Y\_train,

batch\_size=batch\_size,

epochs=nb\_epoch,

validation\_split=0.1,

shuffle=True,

verbose=2)

# Оцінка якості навчання на тестових даних

scores = model.evaluate(X\_test, Y\_test, verbose=0)

print("Accuracy on test data: %.2f%%" % (scores[1]\*100))

# Збереження моделі

model.save('my\_model.h5')

numpy.savez('test\_data.npz', X\_test=X\_test, y\_test=y\_test)

import numpy as np

from keras.utils import np\_utils

from keras.models import load\_model

%matplotlib inline

import matplotlib.pyplot as plt

data = np.load('test\_data.npz')

X\_test = data['X\_test']

y\_test = data['y\_test']

model = load\_model('my\_model.h5')

for i in range(9):

plt.subplot(350 + 1 + i) #\

plt.imshow(X\_test[i], interpolation='bicubic')

plt.show()

plt.imshow(X\_test[1], interpolation='bicubic')

plt.show()

X\_test = X\_test.astype('float32')

X\_test /= 255

Y\_test = np\_utils.to\_categorical(y\_test, 10)

labels = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

counter\_labels = dict(enumerate(labels, 0))

print(counter\_labels)

pred = model.predict(X\_test)

y = np.argmax(pred, axis=-1)

y[:20]

plt.imshow(X\_test[8], interpolation='bicubic')

plt.show()

print('Recognized object:', counter\_labels[y[index]])