

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

З В І Т

**Лабораторна робота №10
з дисципліни
«Комп’ютерні системи
штучного інтелекту»**

Виконавець:

студент групи КІ-22м

Косей М.П.

Керівник:

викладач

Саяпін В.Г.

2023

Лабораторна робота №9

Тема: Основні методи попередньої обробки сигналів і зображень.

Мета: Одержані знання роботи методів за допомогою яких можна розпізнавати графічні об'єкти.

XІД РОБОТИ

1) Ознайомитись з теоретичними відомостями до лабораторної роботи

Раніше для розв'язання задач розпізнавання образів використовувалися повністю зв'язані мережі.

Проте, коли ми працюємо з зображеннями, повністю зв'язані мережі можуть бути неефективними.

Зображення мають великі розміри: одне зображення розміром 256×256 (типова роздільна здатність для класифікації) пікселів має $256 \times 256 \times 3$ вхідних значень (3 кольори для кожного пікселя). Якщо використовувати модель з одним прихованим шаром, який містить 1000 нейронів, цей шар буде мати практично 200 мільйонів параметрів.

Оскільки для успішної класифікації зображень потрібно використовувати кілька шарів, використовуючи повністю зв'язані шари, ми отримаємо мільярди параметрів.

З такою великою кількістю параметрів стане майже неможливим уникнути перенавчання моделі, це означає, що мережа не здатна узагальнювати, а просто запам'ятовує результати.

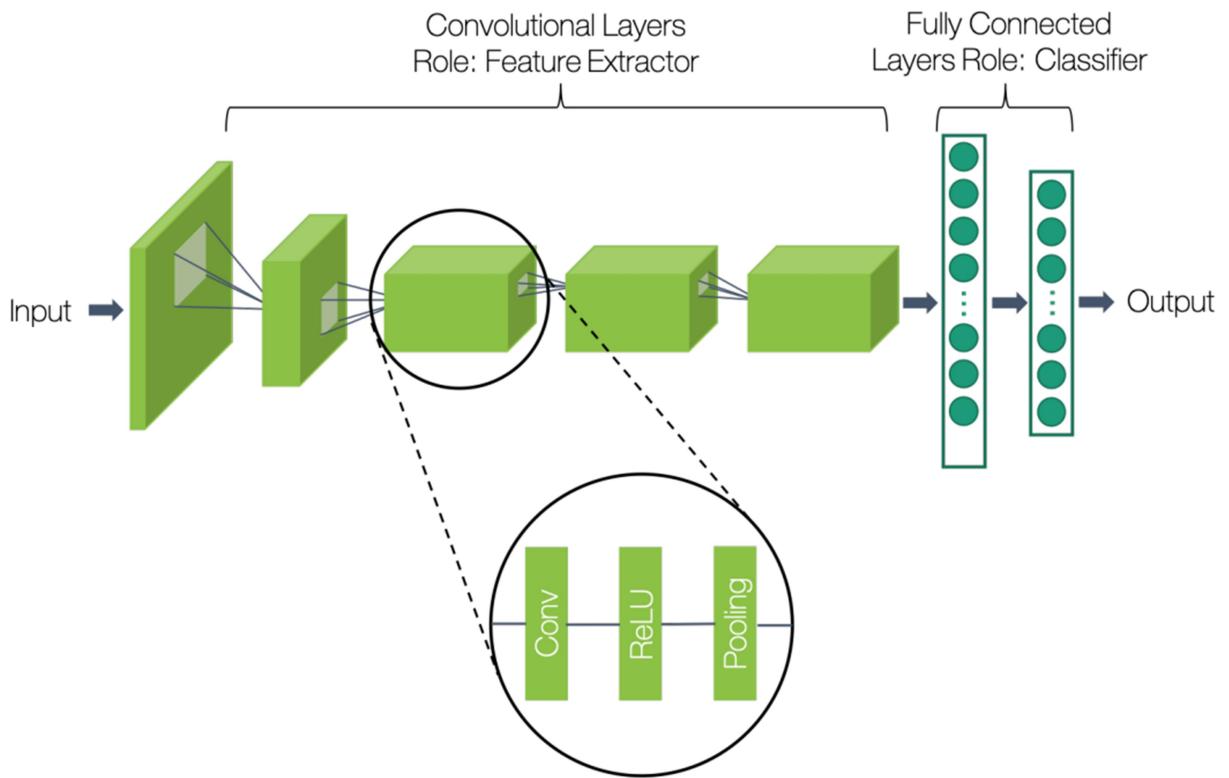


Рис. 1 Структура згорткової нейронні мережі.

Згорткові нейронні мережі (**Convolutional Neural Networks** дивіться рисунок 1) надають нам можливість навчати класифікатори зображень, які

перевершують можливості людини, використовуючи значно меншу кількість параметрів.

Вони досягають цього шляхом імітації способу, яким тварини та люди бачать.

Це досягається шляхом моделювання способу сприйняття зображень тваринами та людьми.

У 1981 році нейробіологи Торстен Віцель та Девід Габель досліджували зорову кору мозку кота і виявили, що існують так звані прості клітини, які особливо сильно реагують на прямі лінії під різними кутами, а також складні клітини, які реагують на рух ліній в одному напрямку.

Пізніше французький науковець Ян Лекун запропонував використовувати так звані згорткові нейронні мережі як аналог зорової кори мозку для розпізнавання зображень.

Згорткові нейронні мережі (**Convolutional Neural Networks** дивіться рисунок 1) складається з двох основних частин:

1. **Вилучення ознак (Feature Extraction):** Ця частина мережі використовує згорткові шари для виявлення різних візуальних ознак у вхідних зображеннях. Кожен згортковий шар використовує фільтри для згортки по зображеню і видачі карт ознак, що відображають виявлені характеристики. Цей процес дозволяє мережі автоматично вчитися розпізнавати різні шаблони із зображень, поступово підвищуючи абстракцію і складність ознак.

2. **Класифікація (Classification):** Після вилучення ознак, використовуються повнозв'язні шари (**fully connected layers**) для класифікації зображень. Ці шари отримують вектор ознак з останнього згорткового шару та використовують його для класифікації категорії зображення. Цей шар зв'язує вилучені ознаки з конкретними класами, навчаючи мережу розпізнавати і класифікувати об'єкти або образи.

Основною операцією у згортковій нейронній мережі (**CNN**) є згортка (**Convolution** див. рисунок 2).

Замість того, щоб застосовувати функцію до всього вхідного зображення, згортка сканує невелике вікно зображення по одному кроku.

На кожній позиції вікна застосовується ядро(**kernel**) (зазвичай це множення матриць, а потім застосування функції активації, так само як у повністю зв'язаній мережі).

Окремі ядра часто називають фільтрами. Результатом застосування ядра до всього зображення є нове, можливо, менше зображення. Наприклад, поширенний розмір фільтра - (3, 3). Якщо ми застосуємо 32 таких фільтри до вхідного зображення, нам знадобиться $3 * 3 * 3$ (кількість кольорів у вхідному зображені) * 32 = 864 параметри - це велика економія порівняно з повністю зв'язаною мережею!

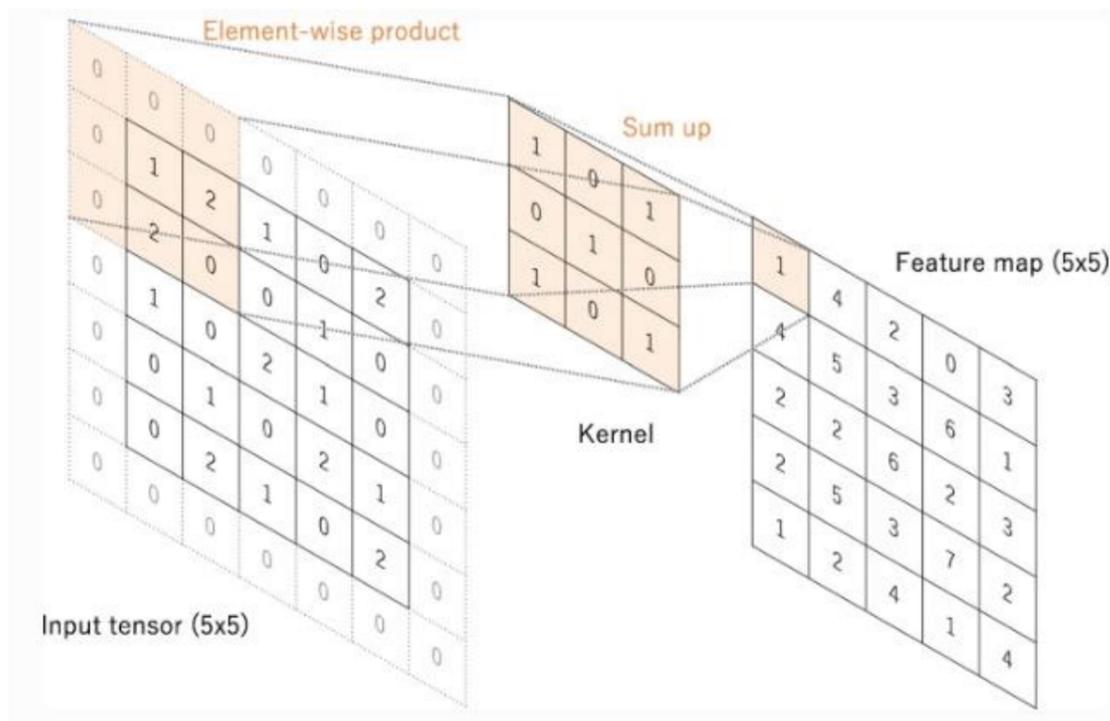


Рис. 2 Операція згортки (**Convolution**).

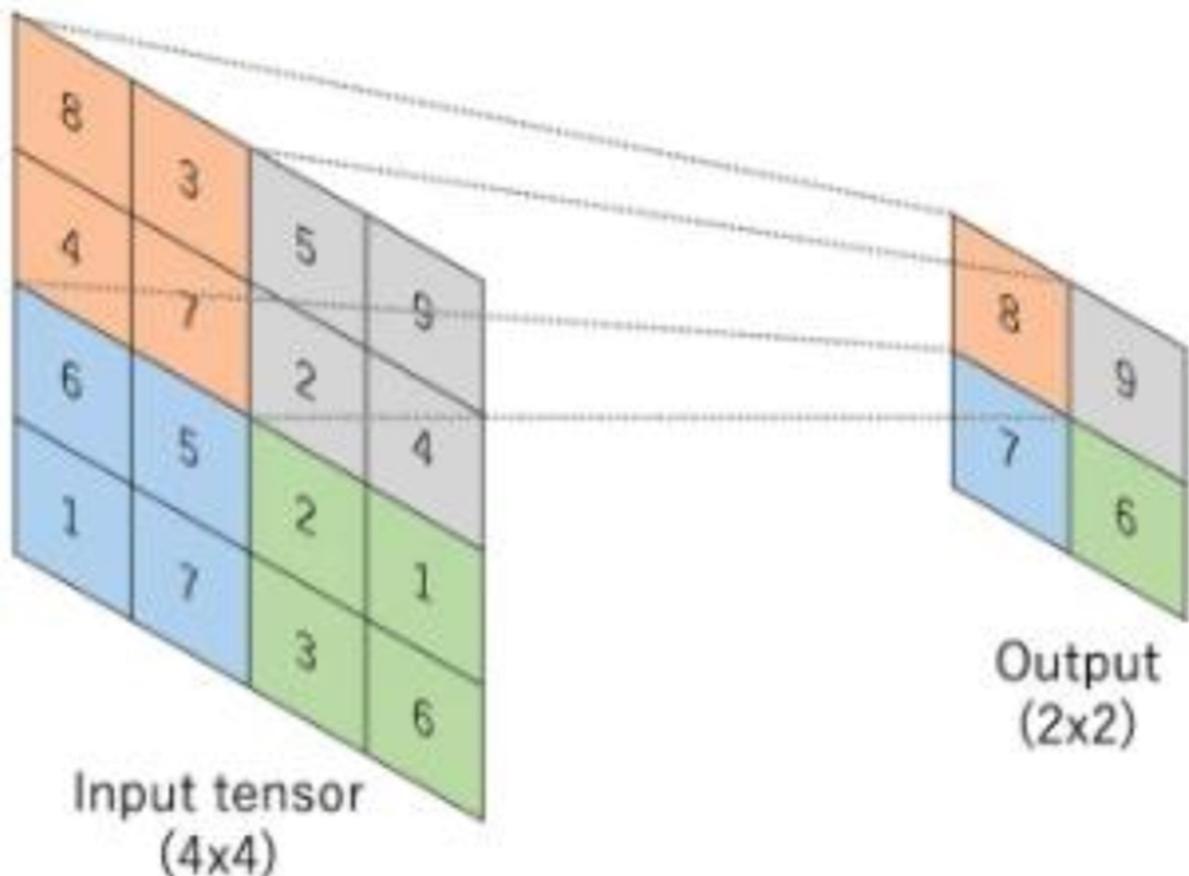


Рис. 3 Операція піддискретизації – пулінг (**Pooling**).

Згортка дозволяє зменшити кількість параметрів, але виникає інша проблема. Кожен шар у мережі може "дивитись" лише на 3×3 шар зображення одночасно: як можемо розпізнати об'єкти, які займають всю картинку?

Для вирішення цього проблеми типова згорткова мережа використовує

піддискретизацію (**subsampling**) для зменшення розміру зображення при проходженні через мережу.

Один з механізмів піддискретизації є пулінг (**Pooling**).

Шар пулінгу насправді є ще одним видом згортки, але замість множення на матрицю ми застосовуємо оператор пулінгу.

Зазвичай для пулінгу використовується оператор максимуму або середнє значення.

Максимальний пулінг вибирає найбільше значення з кожного каналу (кольору) в заданій області.

Середнє значення пулінгу обчислює середнє значення всіх значень в області.

Піддискретизацію можна розглядати як спосіб підвищення абстрактності того, що робить мережа.

На найнижчому рівні згортки виявляють невеликі, локальні ознаки.

Є багато таких ознак, які не є дуже глибокими.

З кожним кроком пулінга ми підвищуюмо рівень абстракції: кількість ознак зменшується, але глибина кожної ознаки збільшується.

Цей процес триває до того часу, поки ми не отримаємо дуже небагато ознак з високим рівнем абстракції, які можна використовувати для розпізнавання образів.

Точність CNN також може бути покращена за допомогою додаткових технік, таких як переднє навчання (**pre-training**) на великих наборах даних, після чого навчання на конкретній задачі (**fine-tuning**), а також використання архітектур, які отримали успіх у конкурсах розпізнавання образів, наприклад, **ResNet**, **Inception** і **VGG** (див. рисунок 4).

Також широко використовується трансферне навчання (**Transfer learning**, див. мал. 5) - це підхід в глибокому навчанні, коли знання, набуті під час тренування моделей на одному завданні, використовуються для поліпшення результатів на іншому схожому завданні.

Замість початкового навчання моделі з нуля, ми використовуємо вже навчену модель, яка має загальні знання про певні візуальні ознаки, і тільки тренуємо або налаштовуємо її для нового завдання. Це забезпечує ефективніше використання обмежених ресурсів та швидше створення високоякісних моделей глибокого навчання.

Layer (type:depth-idx)	Kernel Shape	Output Shape	Param #
VGG			
└ Sequential: 1-1	input:	[1, 3, 224, 224]	
└ Conv2d: 2-1	[3, 3]	[1, 64, 224, 224]	1,792
└ ReLU: 2-2	--	[1, 64, 224, 224]	--
└ Conv2d: 2-3	[3, 3]	[1, 64, 224, 224]	36,928
└ ReLU: 2-4	--	[1, 64, 224, 224]	--
└ MaxPool2d: 2-5	2	[1, 64, 112, 112]	--
└ Conv2d: 2-6	[3, 3]	[1, 128, 112, 112]	73,856
└ ReLU: 2-7	--	[1, 128, 112, 112]	--
└ Conv2d: 2-8	[3, 3]	[1, 128, 112, 112]	147,584
└ ReLU: 2-9	--	[1, 128, 112, 112]	--
└ MaxPool2d: 2-10	2	[1, 128, 56, 56]	--
└ Conv2d: 2-11	[3, 3]	[1, 256, 56, 56]	295,168
└ ReLU: 2-12	--	[1, 256, 56, 56]	--
└ Conv2d: 2-13	[3, 3]	[1, 256, 56, 56]	590,080
└ ReLU: 2-14	--	[1, 256, 56, 56]	--
└ Conv2d: 2-15	[3, 3]	[1, 256, 56, 56]	590,080
└ ReLU: 2-16	--	[1, 256, 56, 56]	--
└ MaxPool2d: 2-17	2	[1, 256, 28, 28]	--
└ Conv2d: 2-18	[3, 3]	[1, 512, 28, 28]	1,180,160
└ ReLU: 2-19	--	[1, 512, 28, 28]	--
└ Conv2d: 2-20	[3, 3]	[1, 512, 28, 28]	2,359,808
└ ReLU: 2-21	--	[1, 512, 28, 28]	--
└ Conv2d: 2-22	[3, 3]	[1, 512, 28, 28]	2,359,808
└ ReLU: 2-23	--	[1, 512, 28, 28]	--
└ MaxPool2d: 2-24	2	[1, 512, 14, 14]	--
└ Conv2d: 2-25	[3, 3]	[1, 512, 14, 14]	2,359,808
└ ReLU: 2-26	--	[1, 512, 14, 14]	--
└ Conv2d: 2-27	[3, 3]	[1, 512, 14, 14]	2,359,808
└ ReLU: 2-28	--	[1, 512, 14, 14]	--
└ Conv2d: 2-29	[3, 3]	[1, 512, 14, 14]	2,359,808
└ ReLU: 2-30	--	[1, 512, 14, 14]	--
└ MaxPool2d: 2-31	2	[1, 512, 7, 7]	--
└ AdaptiveAvgPool2d: 1-2	--	[1, 512, 7, 7]	--

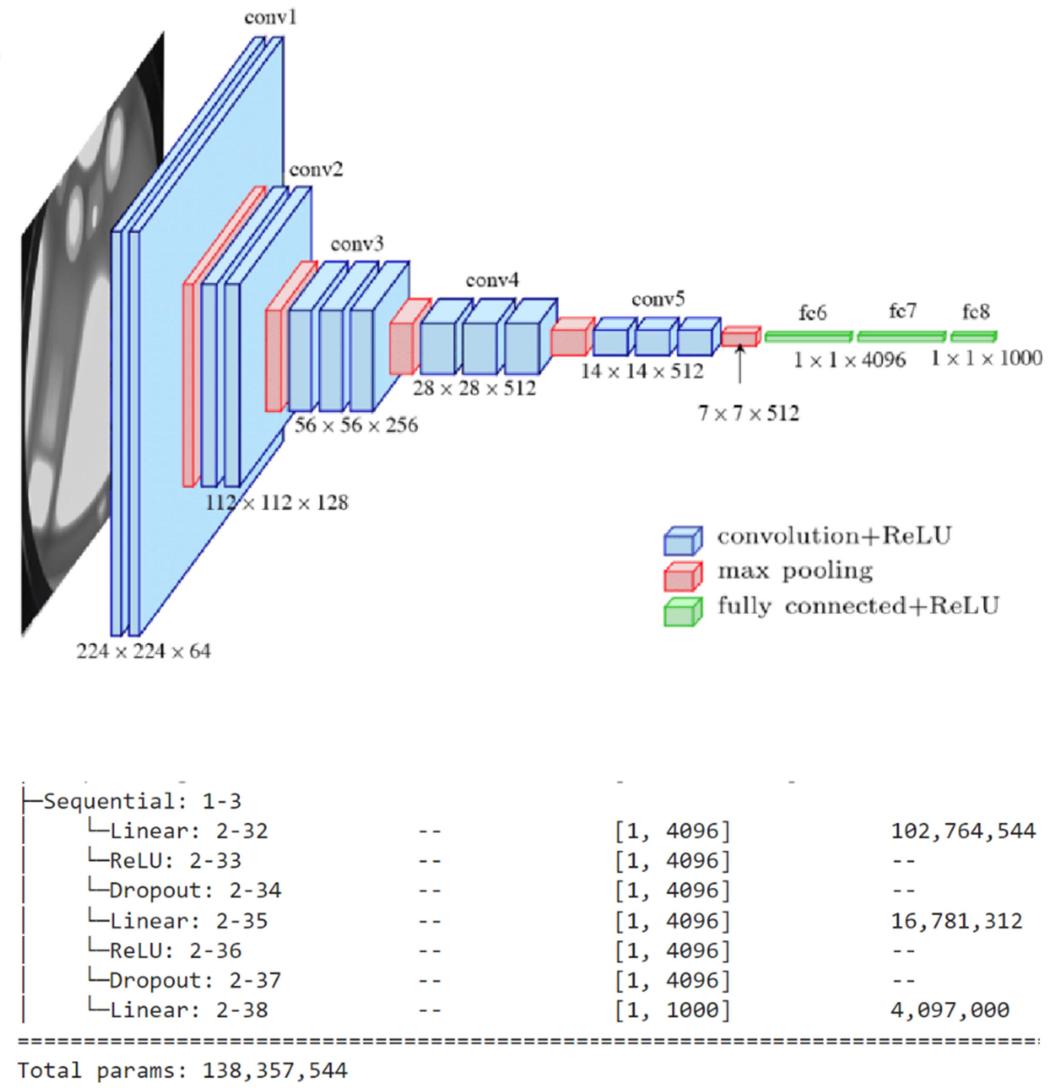


Рис. 4 Структура згорткової мережі **VGG16** яка була навчена розпізнавати кольорові зображення розміром 224×224 пікселів з тисячами класів (таких як автомобілі, коти та інші об'єкти з набору даних **ImageNet**).

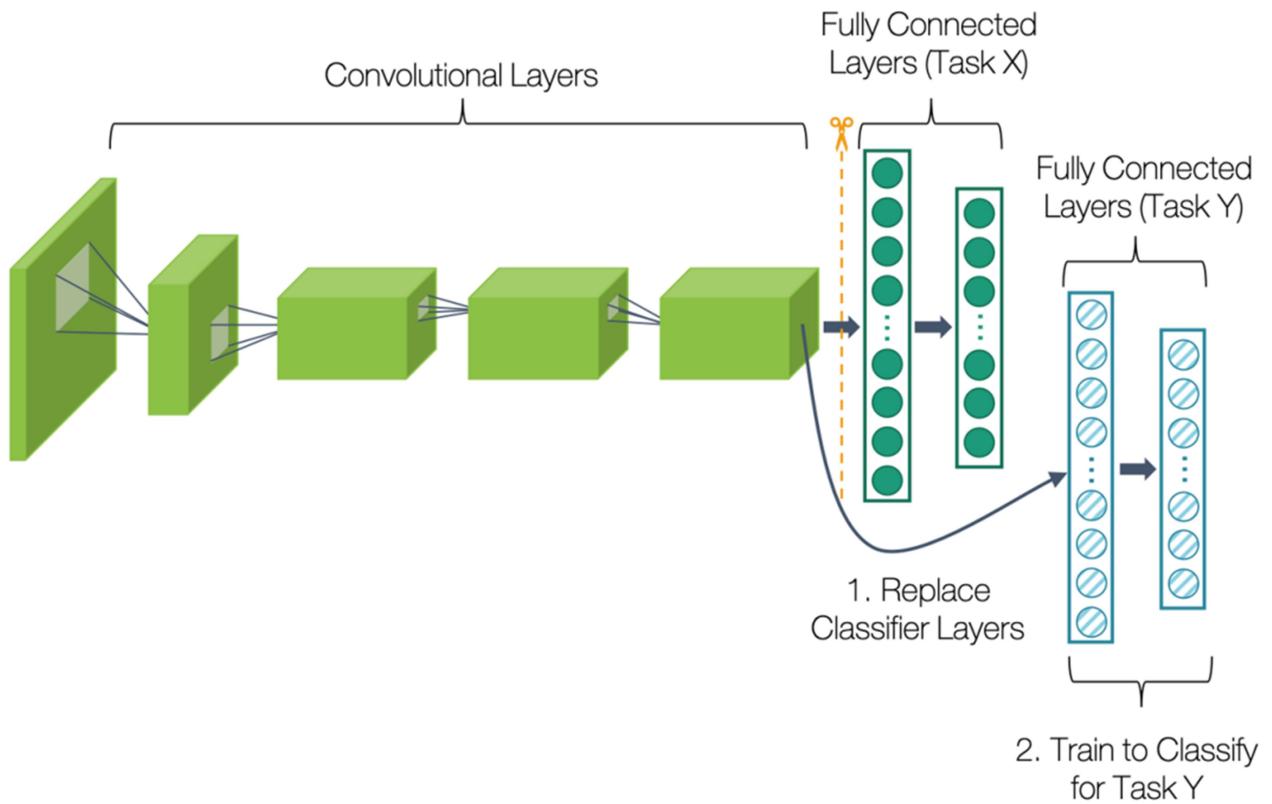


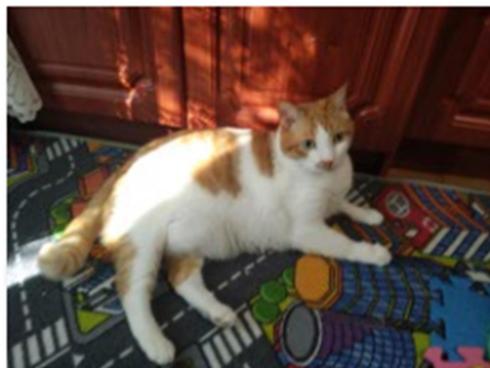
Рис. 5 Принцип трансферного навчання (**Transfer learning**).

2) Використати згорткові мережі для розпізнавання образів.

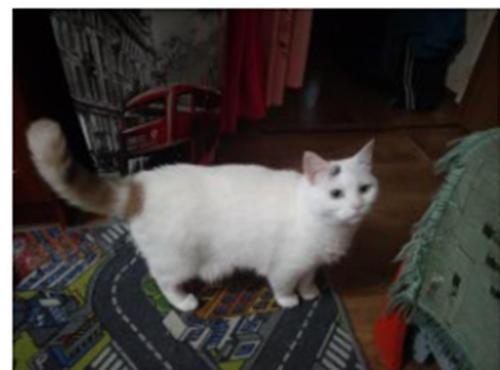
Використовуємо мову програмування **Python** та проект [**JupyterLab**](#).



Робимо фотографії наших домашніх тварин та підготуємо зображення для розпізнання.



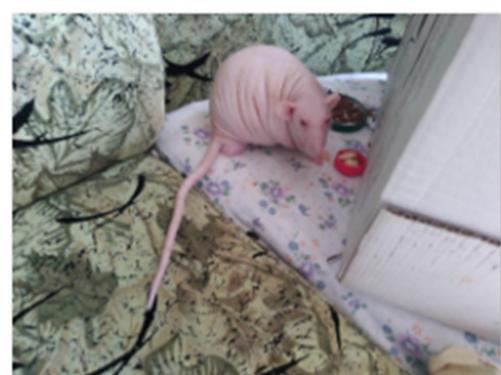
Cat1.jpg



Cat2.jpg



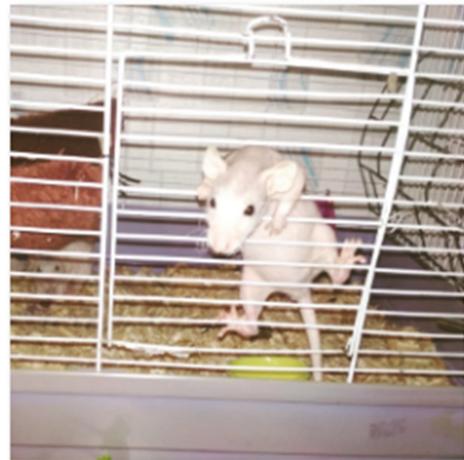
Rat2.jpg



Rat3.jpg



Rat2.jpg



Rat3.jpg

Імпортуємо необхідні бібліотеки.

```
%matplotlib inline

from keras.applications.vgg16 import VGG16
from keras.applications.inception_v3 import InceptionV3
from keras import applications
from keras.preprocessing import image
from keras.applications.vgg19 import preprocess_input

from scipy import ndimage
from matplotlib.pyplot import imshow
from PIL import Image
from keras.applications.imagenet_utils import decode_predictions
from keras.applications.imagenet_utils import preprocess_input
import numpy as np
```

Using TensorFlow backend.

Завантажимо модель **VGG16** яка була навчена на зображеннях з набору **ImageNet**.

```
model = VGG16(weights='imagenet', include_top=True)
model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 [=====] - 49s 0us/step
Model: "vgg16"
```

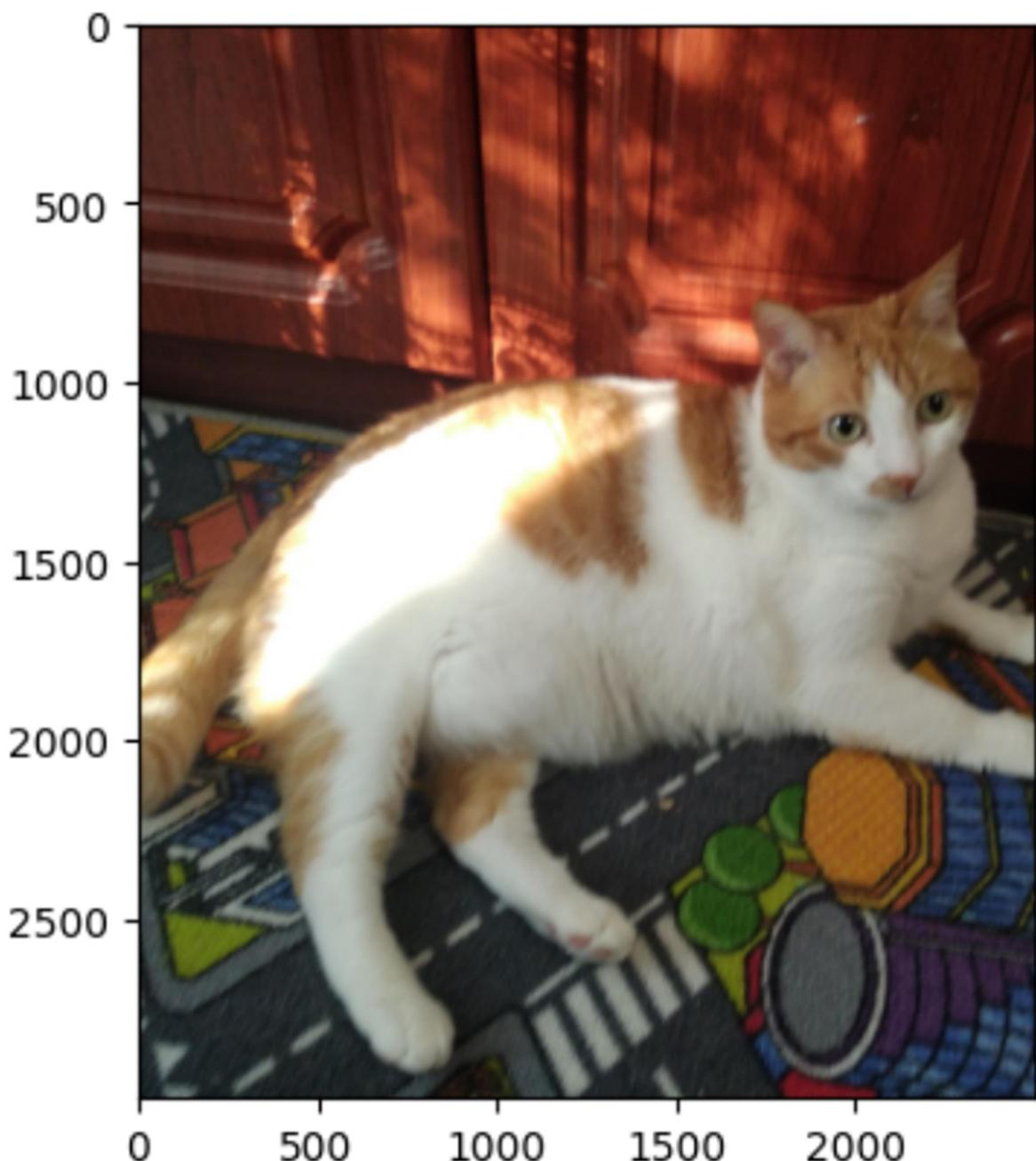
Layer (type)	Output Shape	Param #
<hr/>		
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584

block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000
<hr/>		
Total params: 138,357,544		
Trainable params: 138,357,544		
Non-trainable params: 0		

Змінimo форму зображення на квадратну.

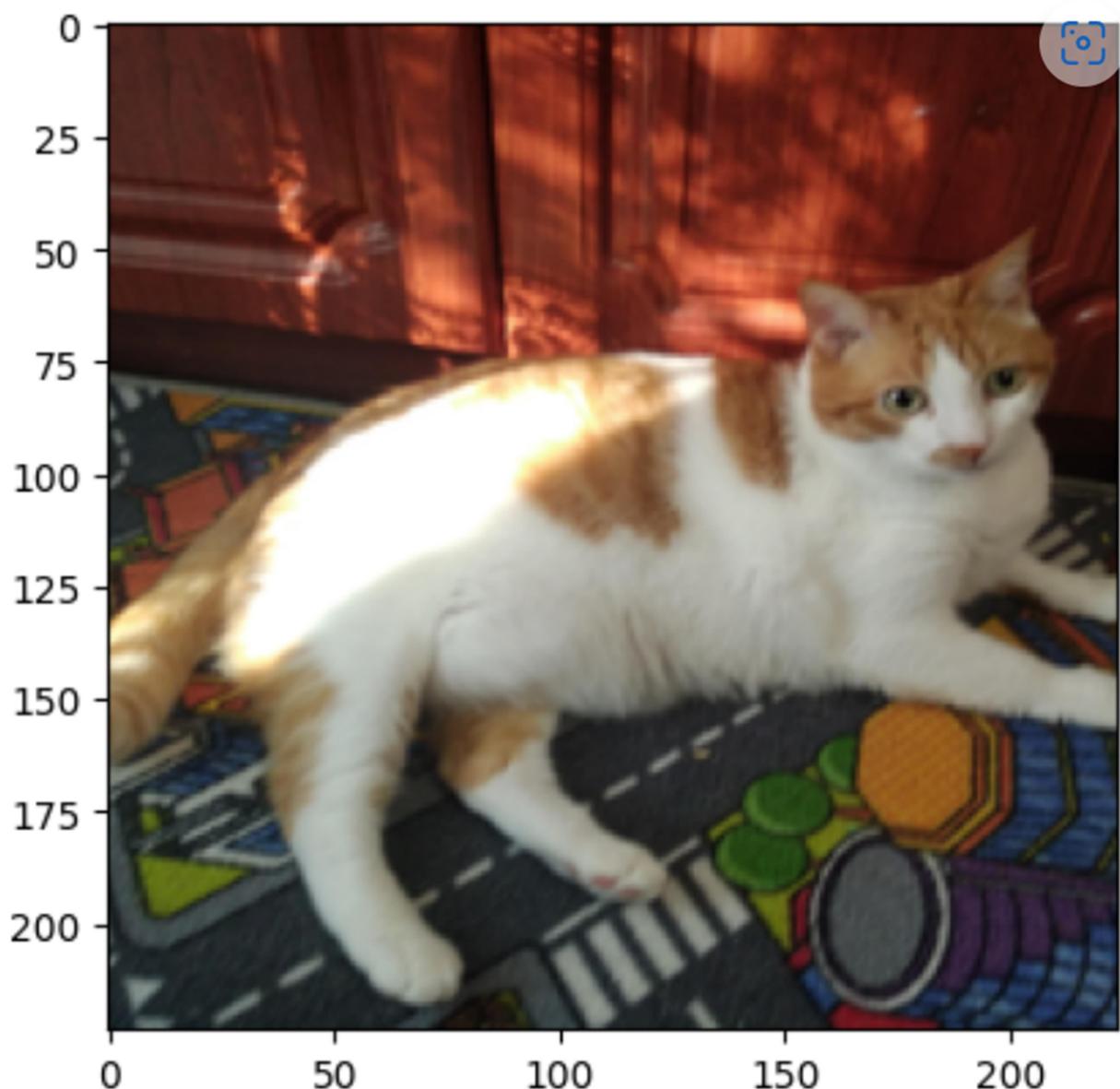
```
img = Image.open('data/cat1.jpg')
w, h = img.size
s = min(w, h)
y = (h - s) // 2
x = (w - s) // 2
img = img.crop((x, y, s, s))
imshow(np.asarray(img))
```

<matplotlib.image.AxesImage at 0x2a4087afad0>



Змінimo роздільну здатність зображення на 224x224 пікселів.

```
img = img.resize((224, 224), Image.ANTIALIAS)  
imshow(np.asarray(img))
```



```
np_img = np.array(img)
img_batch = np.expand_dims(np_img, axis=0)
pre_processed = preprocess_input(img_batch)
pre_processed.shape
```

```
(1, 224, 224, 3)
```

```
features = model.predict(pre_processed)
features.shape
```

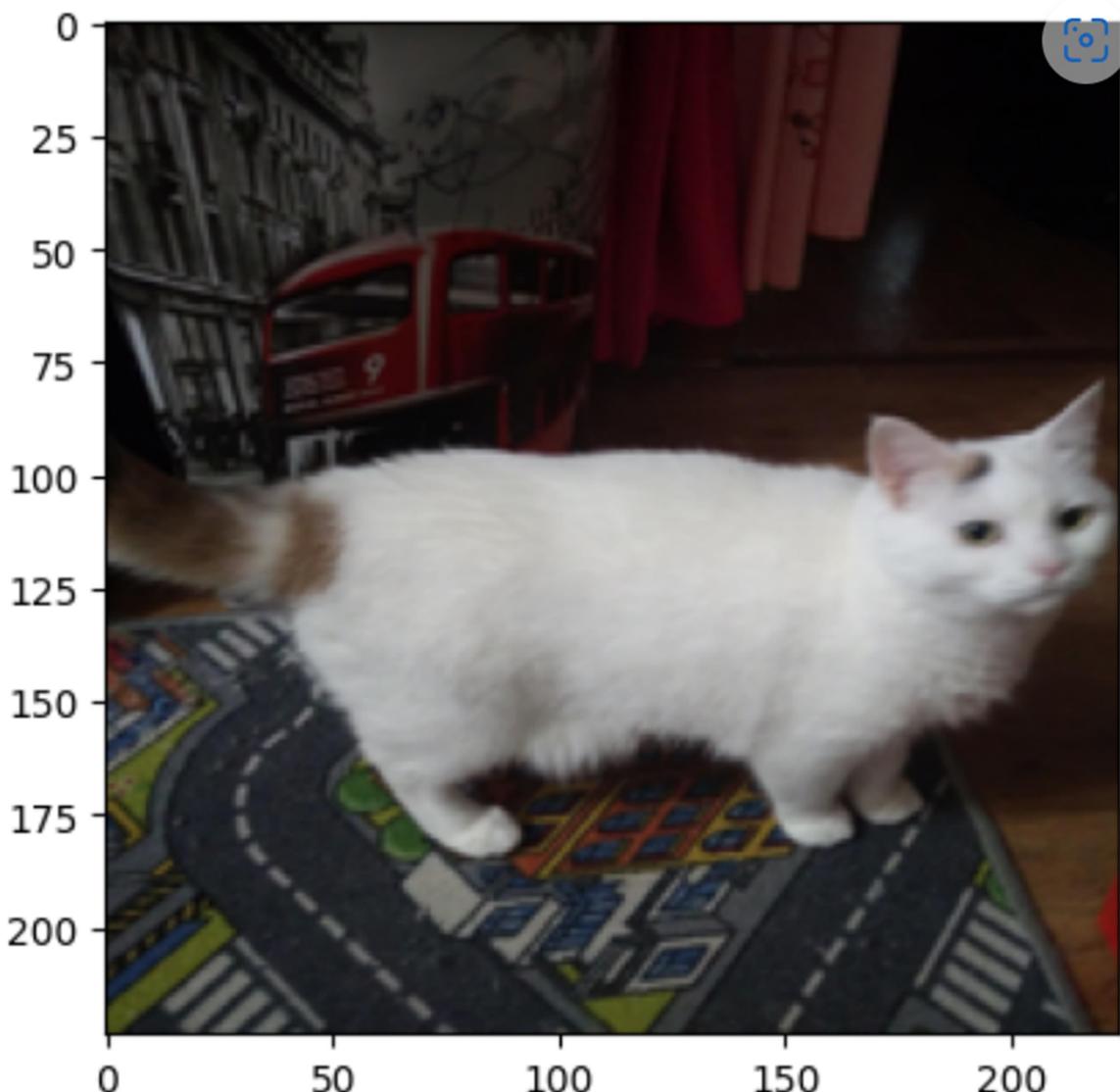
```
1/1 [=====] - 2s 2s/step
(1, 1000)
```

```
decode_predictions(features, top=5)
```

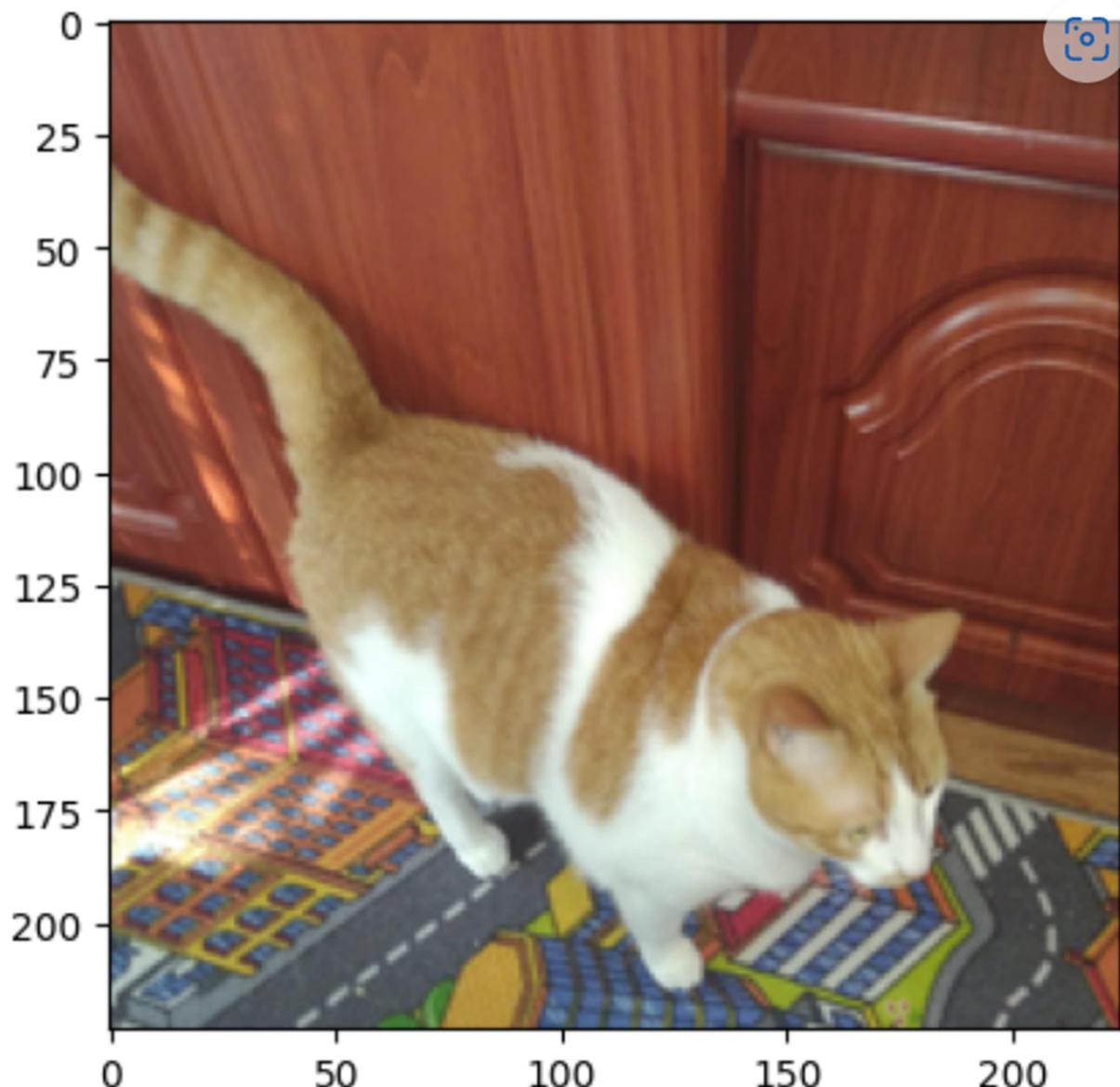
```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [=====] - 0s 1us/step
[[('n04265275', 'space_heater', 0.1853548),
 ('n02123159', 'tiger_cat', 0.117584415),
 ('n02124075', 'Egyptian_cat', 0.06389175),
 ('n04004767', 'printer', 0.05814121),
 ('n03642806', 'laptop', 0.045682933)]]
```

Для інших зображень:

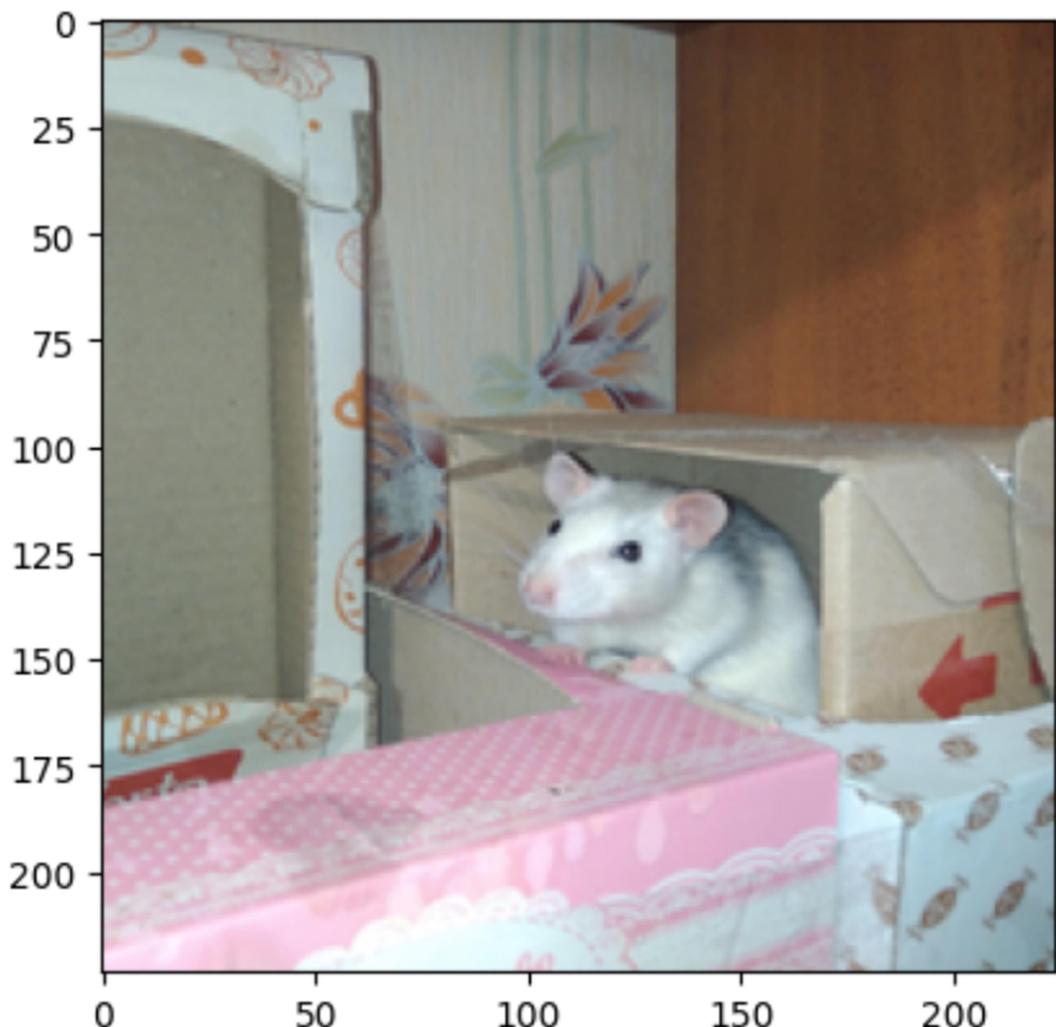
```
: [[('n02123597', 'Siamese_cat', 0.24995397),  
 ('n04265275', 'space_heater', 0.1546431),  
 ('n02441942', 'weasel', 0.053610407),  
 ('n04074963', 'remote_control', 0.036818985),  
 ('n02328150', 'Angora', 0.035866503)]]
```



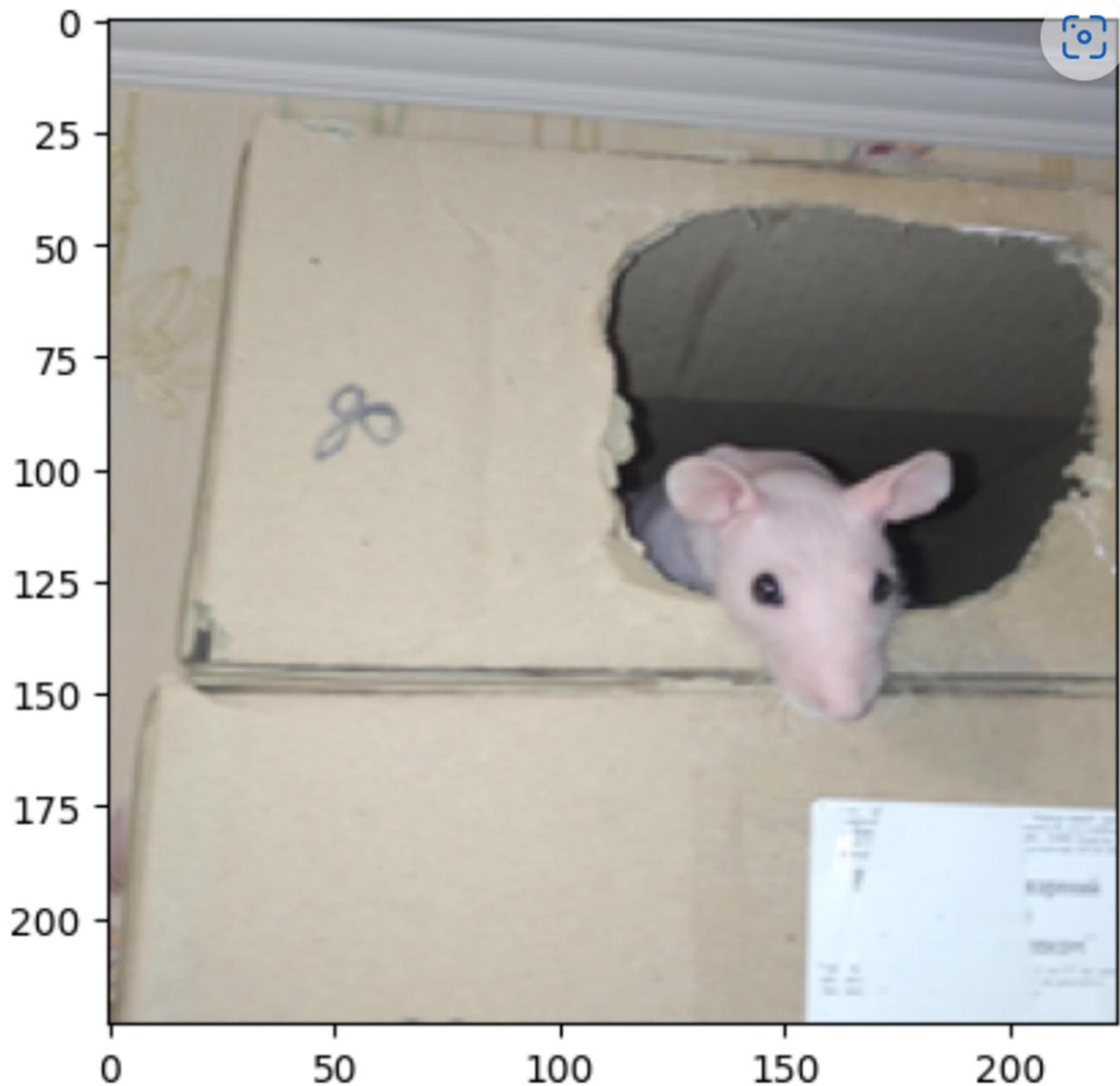
```
[[('n02113023', 'Pembroke', 0.107525565),  
 ('n03207941', 'dishwasher', 0.082160704),  
 ('n02123159', 'tiger_cat', 0.06442944),  
 ('n02110806', 'basenji', 0.04480375),  
 ('n04517823', 'vacuum', 0.034878306)]]
```



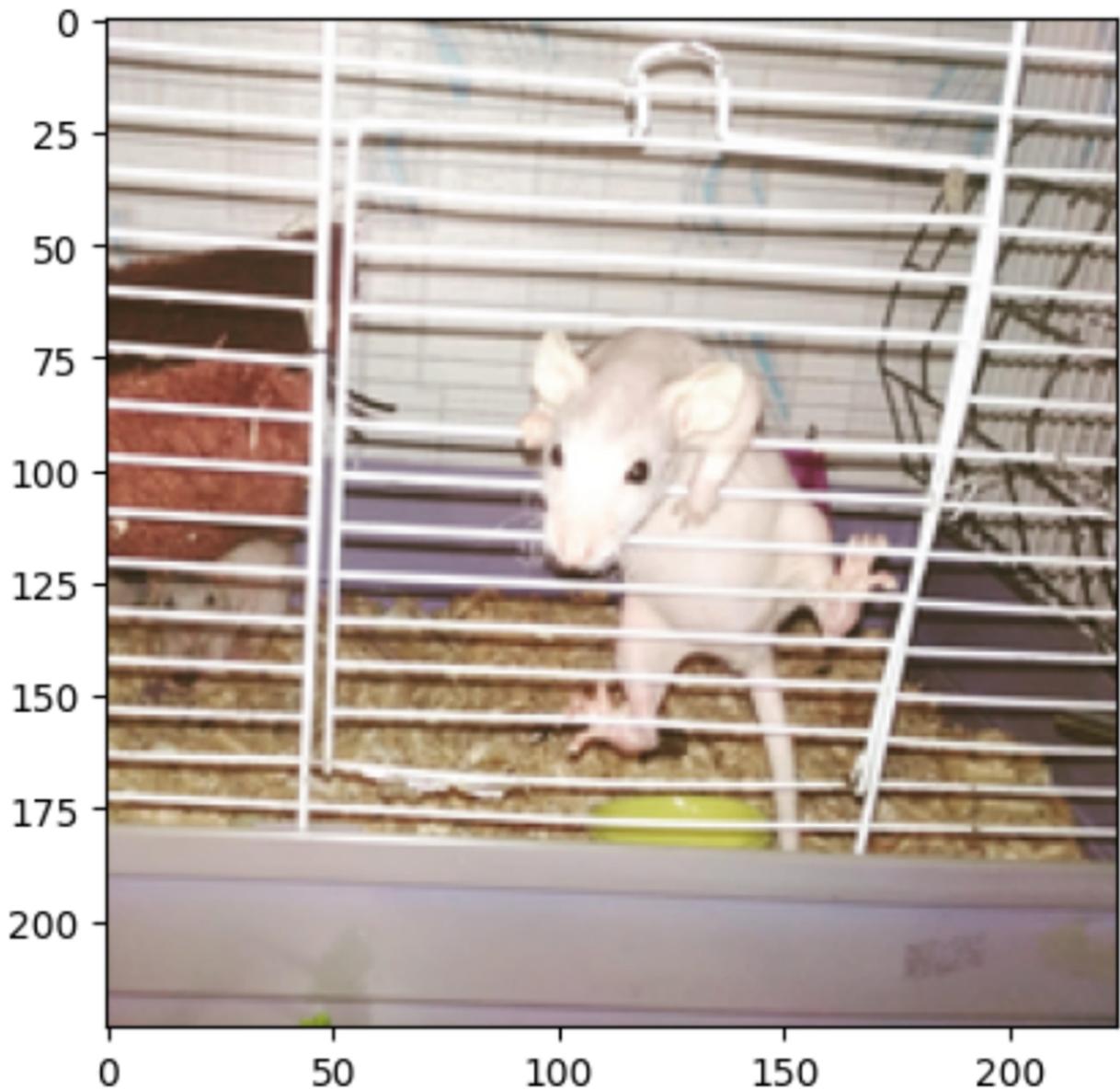
```
[28]: [[('n02971356', 'carton', 0.40700147),  
        ('n02098286', 'West_Highland_white_terrier', 0.06693309),  
        ('n03482405', 'hamper', 0.06114097),  
        ('n02342885', 'hamster', 0.05821179),  
        ('n02111889', 'Samoyed', 0.05439526)]]
```



```
[[('n02971356', 'carton', 0.41733077),  
 ('n03794056', 'mousetrap', 0.05195534),  
 ('n03223299', 'doormat', 0.038847473),  
 ('n03291819', 'envelope', 0.03645412),  
 ('n03642806', 'laptop', 0.032872707)]]
```



```
[[('n02443484', 'black-footed_ferret', 0.404192),  
 ('n02441942', 'weasel', 0.1873888),  
 ('n02342885', 'hamster', 0.11341722),  
 ('n03207941', 'dishwasher', 0.051470824),  
 ('n02325366', 'wood_rabbit', 0.051396903)]]
```



ВИСНОВКИ

В результаті виконаної лабораторної роботи одержати знання роботи методів за допомогою яких можна розпізнавати графічні об'єкти.

Усі матеріали викладені у репозіторії GitHub, за посиланням <https://github.com/Max11mus/Artifition-Intelect-Lab9.git>.

ЛІТЕРАТУРА:

1.[Strengthening Deep Neural Networks](#) by Katy Warr Released July 2019 Publisher(s): O'Reilly Media, Inc. ISBN: 9781492044956

2.[Practical Deep Learning for Cloud, Mobile, and Edge](#) by Anirudh Koul, Siddha Ganju, Meher Kasam Released October 2019 Publisher(s): O'Reilly Media, Inc. ISBN: 1492034865

3.[Deep Learning Cookbook](#) by Douwe Osinga Released June 2018 Publisher(s): O'Reilly Media, Inc. ISBN: 9781491995792