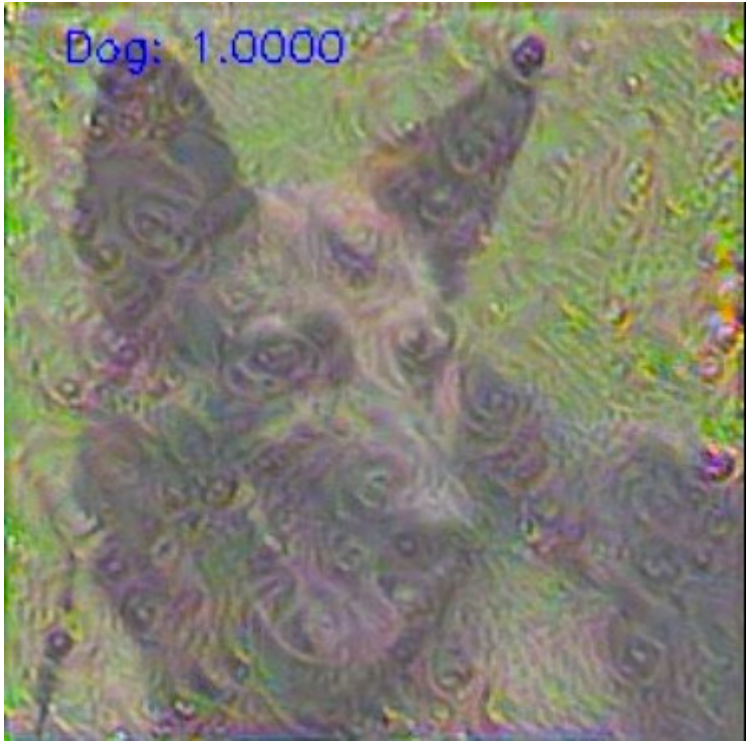
Пошук “першопричин”



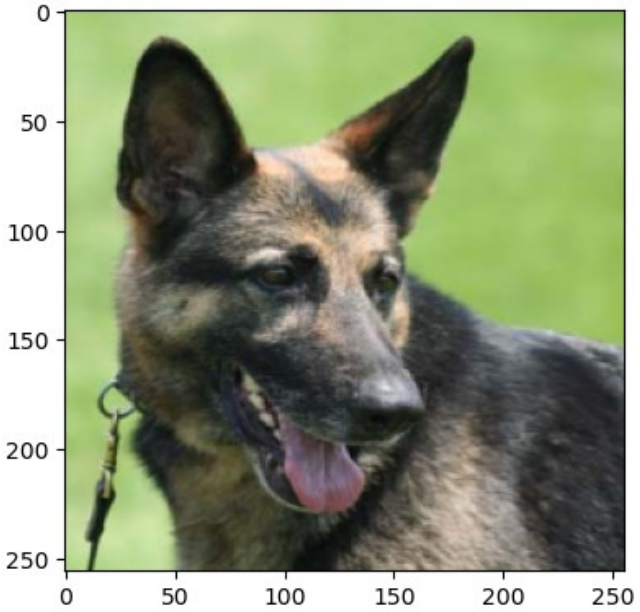
Дрєєв Олександр Миколайович

Мета

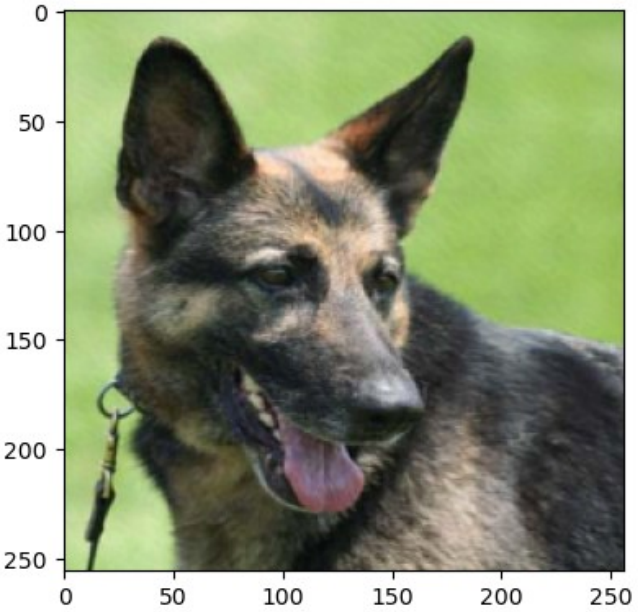
Мета: Виділити елементи на зображенні, які враховувалися як позитивна ознака в класифікації нейронною мережею.

Шлях до результату: створити схему оптимізатора, який зменшує значення функції втрат для зображення з більш впевненою класифікацією.

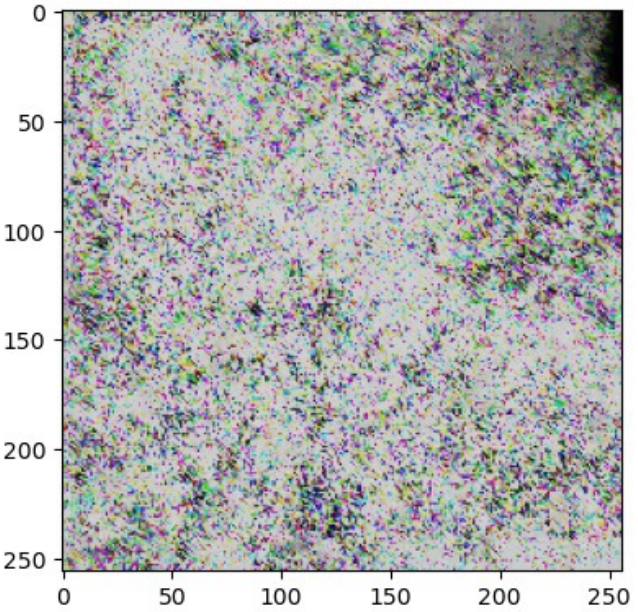
Малі зміни — колосальний результат



[[0.056; 0.944]]

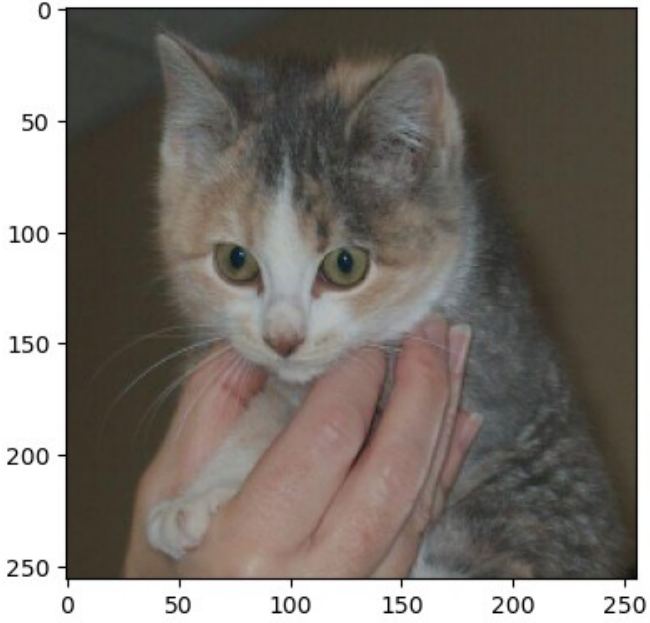


[[0.999997; 2.947e-06]]

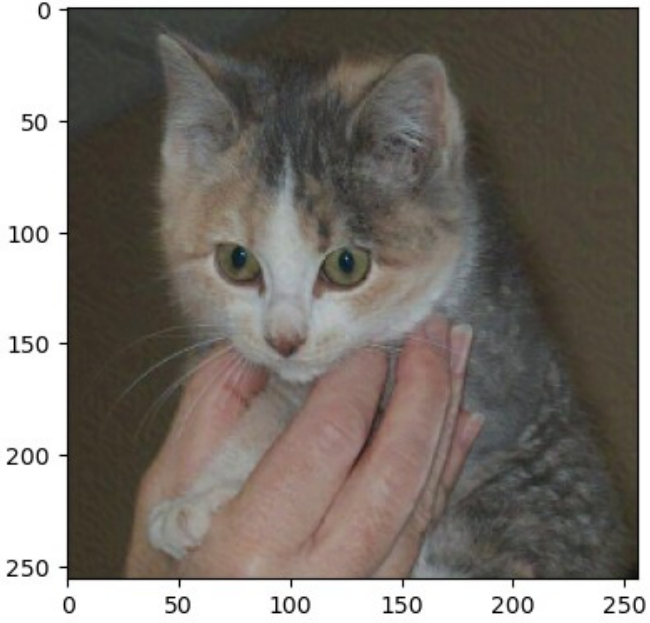


Different

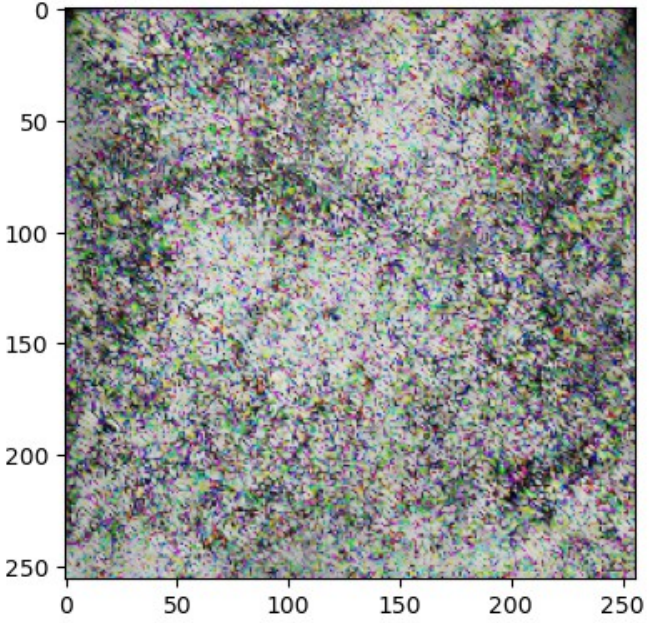
Малі зміни — колосальний результат



[[0.999; 3.38e-05]]

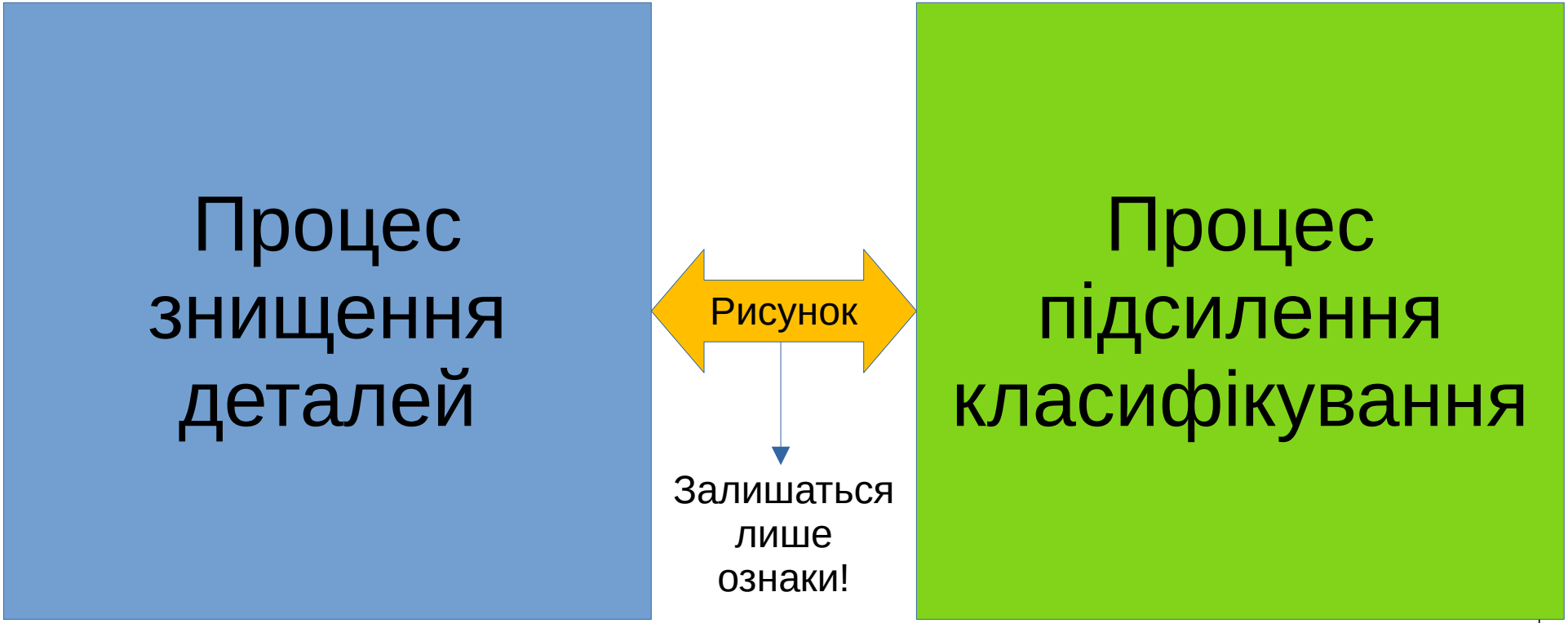


[[1.69e-05; 0.999]]

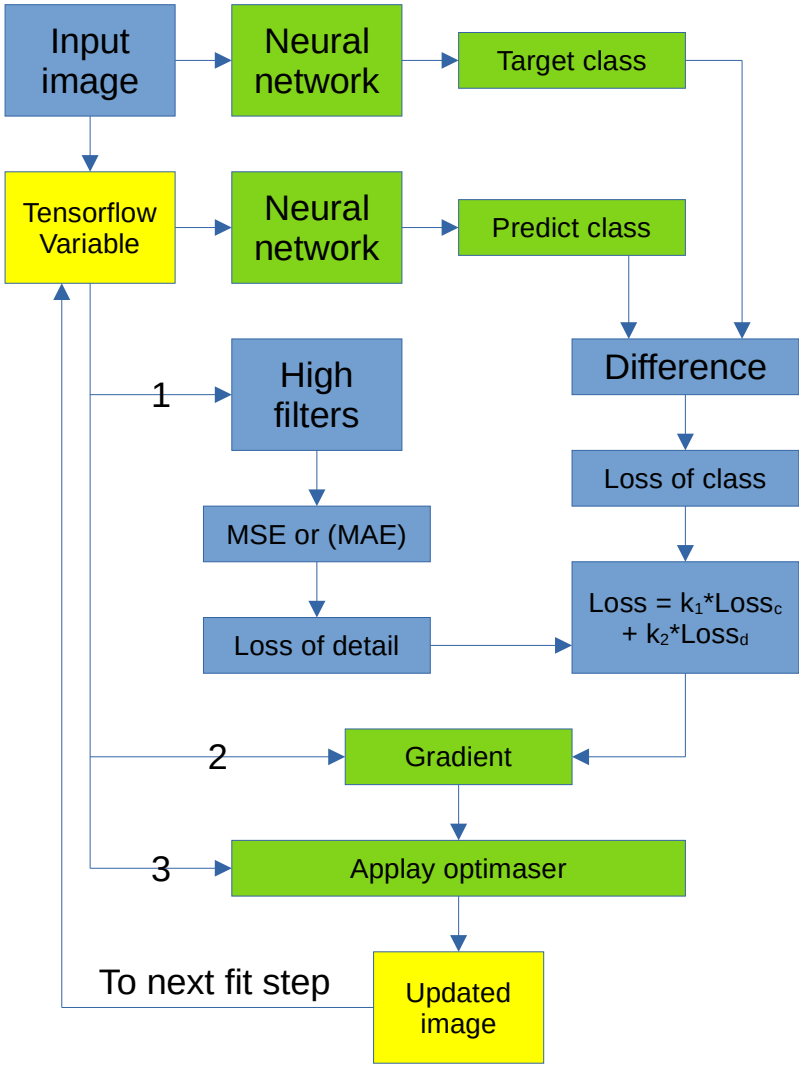


Different

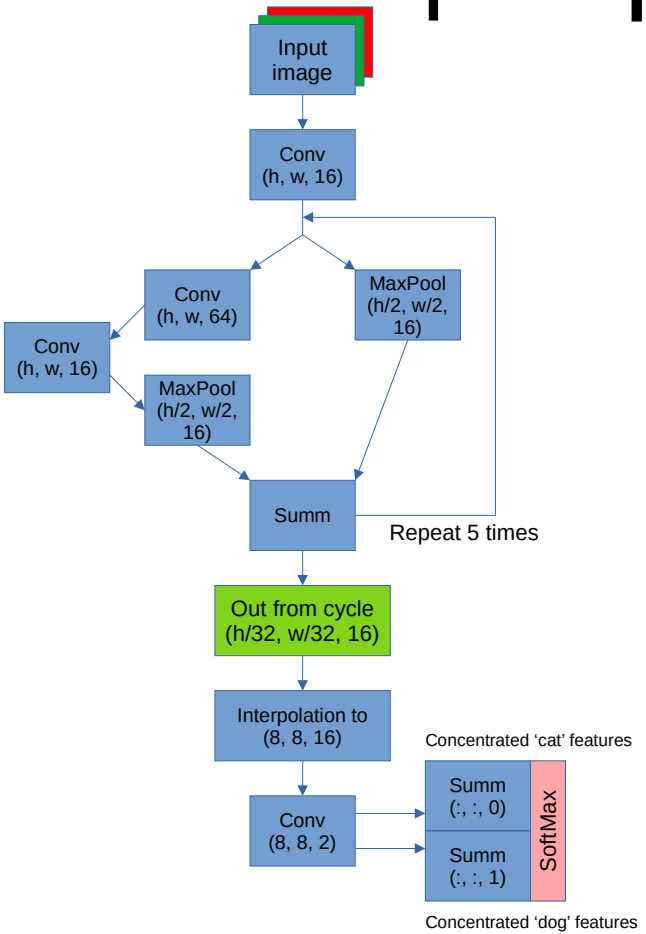
Ідея пошуку ознак для прийняття рішення



Деталізована схема процесу



Архітектура мережі з концентрацією особливостей



Недолік активації ***softmax***

Input	Output
52.25	~1
0.001	~0

Input	Output
0.001	~1
0.000	~0

Архітектура мережі з концентрацією особливостей

Відмінність запропонованої архітектури:

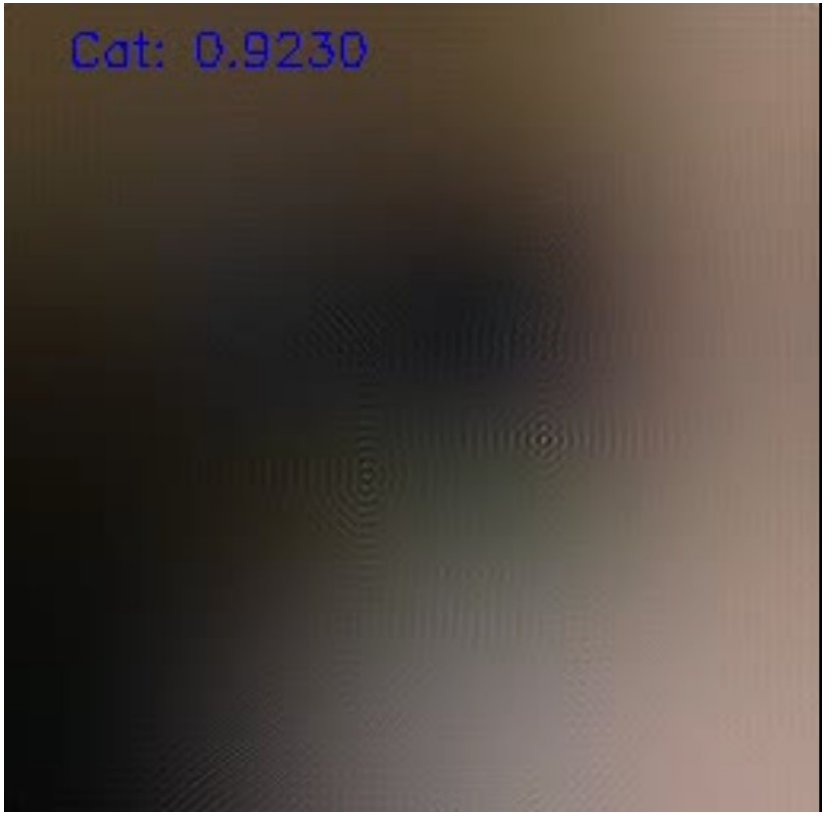
1) Можна вилучати ознаки протилежного класу із зображення для посилення впевненості класифікатора. Після цього вилучення ознак поточного класу, крім одного, не погіршить впевненість класифікатора.

2) Є можливість класифікувати зображення різних розмірів

Результат розмиття кота (mse)

Результат на інших мережах (mse)

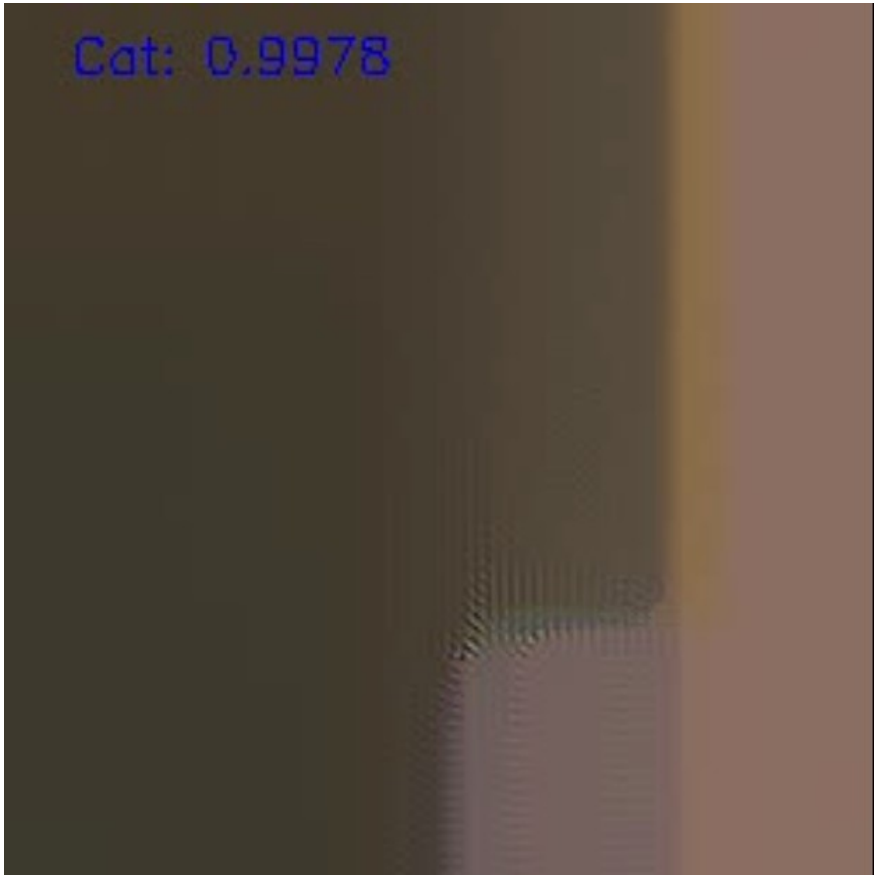
CNN



MIXER

https://github.com/Nguyendat-bit/MLP-Mixer/blob/main/model.py

Результат розмиття собаки (mae)



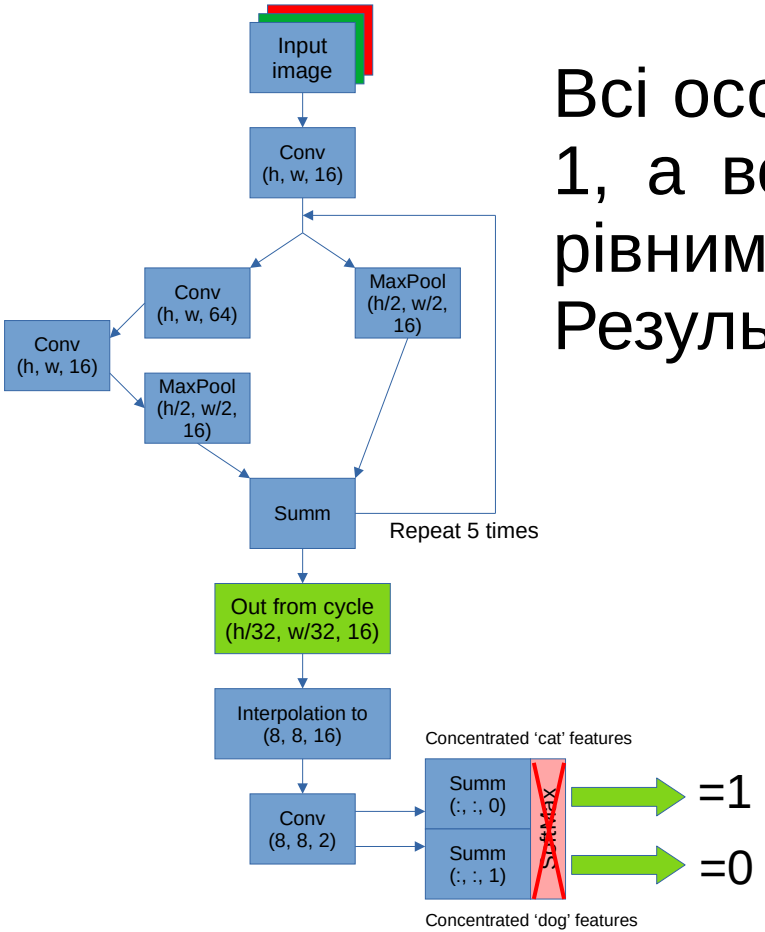
Результат на інших мережах (mae)

CNN

MIXER

https://github.com/Nguyendat-bit/MLP-Mixer/blob/main/model.py

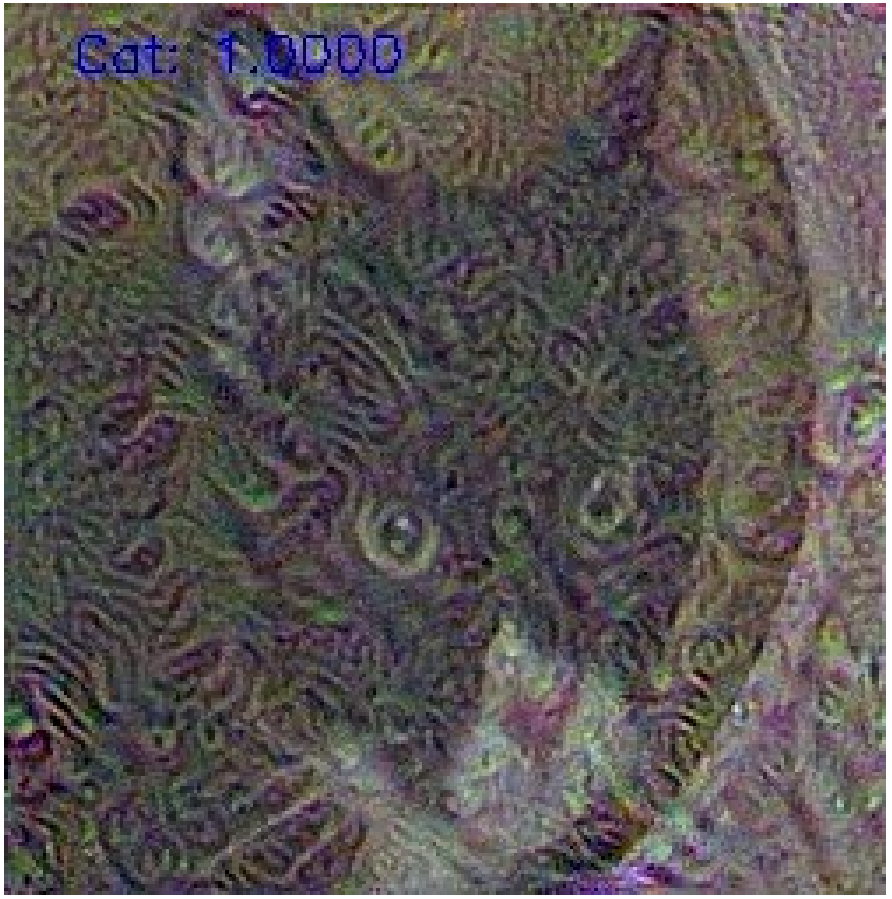
Результати без врахування активації ***sofmax***



Всі особливості кота робимо рівними 1, а всі особливості собаки робимо рівними нулеві.
Результат з “чистого листа”:

Супер-кіт

Результат з фото кота:



Deep-dream

Генерація котів різними мережами

Greed

CNN

Mixer

Використаний код для пошуку особливостей

```
err_type = tf.square #err_type = tf.abs
image_init = tf.Variable(image_init, name='image_data')
while True:
    for _ in range(10):
        with tf.GradientTape() as t:
            f = model(image_init)
            sh = tf.nn.conv2d(image_init, filters=sobel_h, strides=(1,1), padding='VALID')
            sw = tf.nn.conv2d(image_init, filters=sobel_w, strides=(1,1), padding='VALID')
            sh = err_type(sh)
            sw = err_type(sw)
            sh = tf.reduce_mean(sh)
            sw = tf.reduce_mean(sw)
            loss_markers = tf.reduce_mean(tf.square(target-f))*weight_class
            loss_sharpness = (sh+sw)*weight_blur
            loss = loss_markers*k1 + loss_sharpness*k2
        gradients = t.gradient(loss, image_init)
        optimaser.apply_gradients(zip([gradients], [image_init]))
        image_init = tf.subtract(image_init, tf.reduce_min(image_init))
        image_init = image_init/tf.reduce_max(image_init)
        image_init = tf.Variable(image_init, name='image_data')
```

Висновки

- Різної архітектури нейронні мережі мають відмінні по формі та розміру ознаки, які враховуються в класифікуванні.
- Різної архітектури мережі вважають за більш важливими зони по краю або по центру зображення.
- Жодна нейронна мережа не показала використання загальної форми об’єкту в класифікації.
- MIXER є архітектурою, яка враховує ознаки різного розміру, проте її навчання є найважчим.

Дякую за увагу!

https://www.csail.mit.edu/news/fooling-googles-image-recognition-ai-1000x-faster