



Министерство науки и высшего образования Российской  
Федерации  
Калужский филиал федерального государственного автономного  
образовательного учреждения высшего образования  
«Московский государственный технический университет  
имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

---

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК5 «Системы обработки информации»

## ЛАБОРАТОРНАЯ РАБОТА №1.1

**«Общая информация о библиотеках Keras и TensorFlow.  
Глубокое обучение сверточных нейронных сетей»**

**по дисциплине: «Методы глубокого обучения»**

Выполнил: студент группы ИУК5-21М

(Подпись)

А. Э. Дармограй

(И.О. Фамилия)

Проверил:

(Подпись)

Ю. С. Белов

(И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2025

**Цель:** получение практических навыков построения сверточных нейронных сетей с применением глубокого обучения.

**Задачи:**

- Ознакомление с методами глубоких сверточных нейронных сетей (визуализация промежуточных активаций, фильтров сверточных нейронных сетей, тепловых карт активации класса);
- Реализация модели глубокой сверточной нейронной сети.

**Задание:**

- 1) Продемонстрировать работу метода визуализации тепловых карт на следующих изображениях. Произвольно выбрать объект для распознавания – рыба (Вариант 2).
- 2) Визуализируйте карту признаков для изображений из задания 1. Количество каналов выбираются произвольно.
- 3) Реализовать градиентный спуск для изображений из задания 1

**Вариант 2: Рыба**

## Выполнение работы

Код доступен в репозитории GitHub:

[https://github.com/Dariarty/Deep\\_Learning\\_Methods](https://github.com/Dariarty/Deep_Learning_Methods)

Данную лабораторную работы выполнял на Python версии 3.12.7 и Tensorflow версии 2.19

**Задание 1.** Продемонстрировать работу метода визуализации тепловых карт на следующих изображениях. Произвольно выбрать объект для распознавания – рыба.

[https://github.com/Dariarty/Deep\\_Learning\\_Methods/blob/main/src/LAB1/heat\\_map.ipynb](https://github.com/Dariarty/Deep_Learning_Methods/blob/main/src/LAB1/heat_map.ipynb)

В данной работе визуализируем тепловую карту изображения

Начинаем с подключения библиотек и задания пути к изображению. Выведем исходную картинку.

```
#В данной работе использую Python 3.12.7 и tensorflow 2.19

#Имя входного файла
input_file_name = 'images/fish.jpg'

import os
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'

#Импорты необходимых модулей
import numpy as np
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from IPython.display import Image, display #Для вывода исходной картинки
from pathlib import Path

# Загружаем изображение
img_path = Path(input_file_name).resolve()
print("Исходное изображение - фото золотой рыбки")
display(Image(filename = img_path))
```

Исходное изображение - фото золотой рыбки



## Получение предсказаний при помощи модели VGG16

Выполняем препроцессинг изображения, загружаем модель VGG16. Выводим структуру модели и первые 5 предсказаний. Находим класс предсказанного изображения.

```
#Загружаем изображение и производим препроцессинг
img = keras.preprocessing.image.load_img(img_path, target_size=(224, 224))
img = keras.preprocessing.image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img = keras.applications.vgg16.preprocess_input(img)

# Загружаем модель VGG16
model = keras.applications.VGG16(weights='imagenet', include_top=True)

#Получаем предсказания
preds = model.predict(img)
decoded_preds = keras.applications.vgg16.decode_predictions(preds, top=5)[0];

# Вывод структуры модели
model.summary()

print('Топ 5 предсказаний:')
for pred in decoded_preds:
    print('%s - %.4f%%' % (pred[1], pred[2]*100))

#Предсказанный класс изображения
predicted_class = np.argmax(preds[0])
```

Топ 5 предсказаний:  
goldfish - 99.9944%  
anemone\_fish - 0.0031%  
sea\_slug - 0.0012%  
puffer - 0.0006%  
sea\_anemone - 0.0004%

## Grad CAM

Выполняем визуализацию карты активации слоя при помощи процедуры Grad CAM. Вычисляем градиенты, создаем карту активации и накладываем ее на исходное изображение

```
target_layer_name = "block5_conv3" # Название слоя, для которого выводим карту
активации
target_layer_filters = 512 # Количество фильтров в слое

# Вычисляем градиенты с помощью GradientTape
with tf.GradientTape() as tape:
    target_layer = model.get_layer(target_layer_name)
    iterate = tf.keras.Model([model.input], [target_layer.output,
model.output])
    conv_output, predictions = iterate([img])
    loss = predictions[:, predicted_class] # Выбираем значение предсказанного
класса
    grads = tape.gradient(loss, conv_output) # Вычисляем градиенты
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2)) # Среднее значение
градиентов

# Корректируем выход свёрточного слоя
conv_output = conv_output.numpy()[0]
pooled_grads = pooled_grads.numpy()

for i in range(target_layer_filters):
    conv_output[:, :, i] *= pooled_grads[i]

# Создаём карту активации
heatmap = np.mean(conv_output, axis=-1)
heatmap = np.maximum(heatmap, 0)
heatmap /= np.max(heatmap)

# Отображаем карту активации
print('Карта активации')
plt.matshow(heatmap)
plt.show()

# Наложение карты на изображение
img = cv2.imread(img_path)
heatmap = cv2.resize(heatmap, (img.shape[1], img.shape[0]))
heatmap = np.uint8(255 * heatmap)
heatmap = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)
```

```

superimposed_img = heatmap * 0.4 + img
superimposed_img = np.clip(superimposed_img, 0, 255).astype(np.uint8)

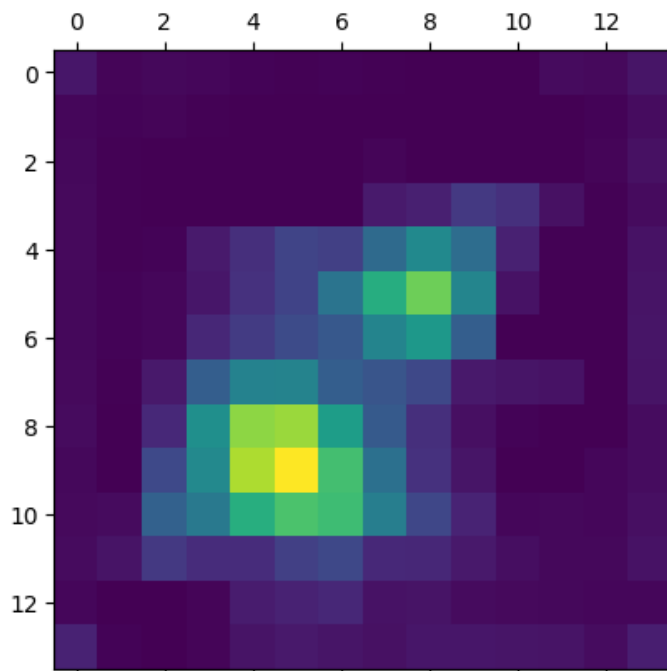
# Сохраняем изображение с наложенной тепловой картой
print('Тепловая карта, наложенная на изображение')
plt.imshow(superimposed_img)
plt.show()

#Сохранение тепловой карты и изображения с наложенной тепловой картой в файлы
write_success = cv2.imwrite('images/superimposed_image.jpg', superimposed_img)
write_success = write_success and cv2.imwrite('images/heat_map.jpg', heatmap)

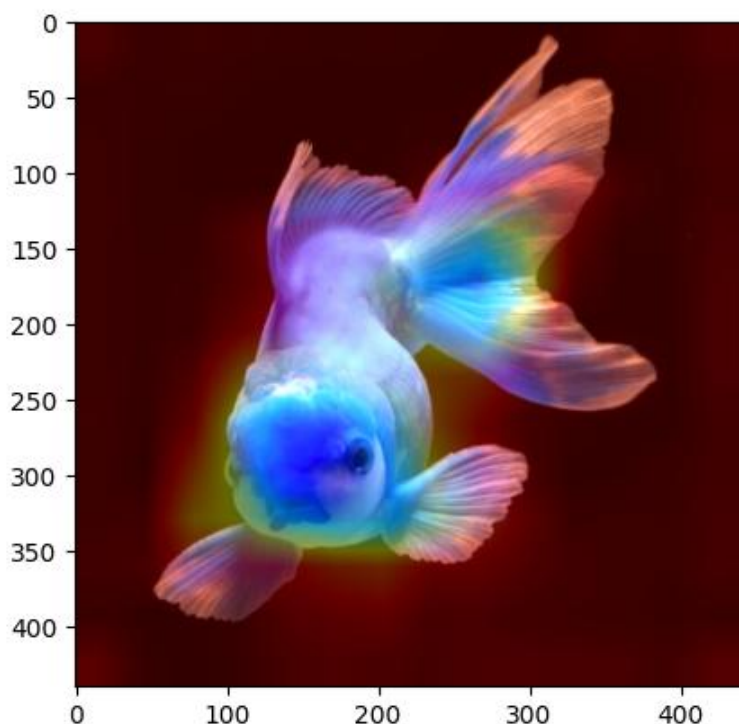
print("Изображения успешно сохранены" if write_success else "Ошибка сохранения изображений")

```

Карта активации



Тепловая карта, наложенная на изображение



Изображения успешно сохранены

**Задание 2.** Визуализируйте карту признаков для изображений из задания 1. Количество каналов выбираются произвольно.

[https://github.com/Dariarty/Deep\\_Learning\\_Methods/blob/main/src/LAB1/feature\\_map.ipynb](https://github.com/Dariarty/Deep_Learning_Methods/blob/main/src/LAB1/feature_map.ipynb)

В данной работе визуализируем карты признаков слоев модели VGG16

Начинаем с подключения библиотек и задания пути к изображению. Выведем исходную картинку.

```
#В данной работе использую Python 3.12.7 и tensorflow 2.19
```

```
#Имя входного файла
```

```
input_file_name = 'images/fish.jpg'
```

```
import os
```

```
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
```

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'
```

```
#Импорты необходимых модулей
```

```
import numpy as np
```

```
import cv2
```

```
import matplotlib.pyplot as plt
```

```
import tensorflow as tf
```

```
from tensorflow import keras
```

```
from IPython.display import Image, display
```

```
from pathlib import Path
```

```
# Загружаем изображение
```

```
img_path = Path(input_file_name).resolve()
```

```
print("Исходное изображение - фото золотой рыбки")
```

```
display(Image(filename = img_path))
```

Исходное изображение - фото золотой рыбки



## Препроцессинг изображения

`tf.keras.applications.vgg16.preprocess_input()` изменяет диапазон значений пикселей, вычитая средние значения каналов RGB: [123.68, 116.779, 103.939]

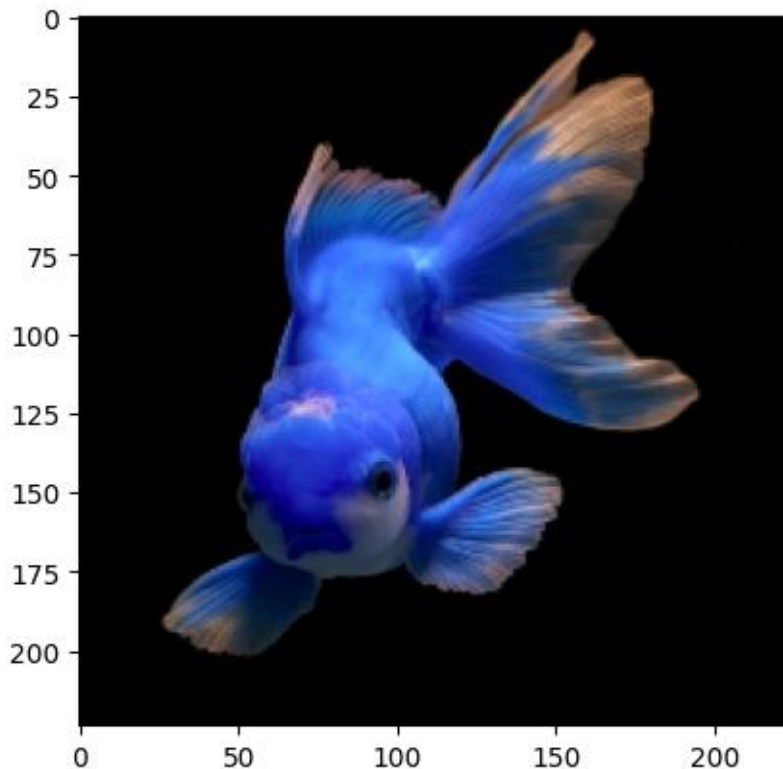
Для корректного отображения изображения перед вызовом `plt.imshow()` денормализуем значения в диапазон [0,1], добавляя соответствующие средние значения каналов, с учетом использования tensorflow модели BGR вместо RGB



```
#Загружаем изображение и производим препроцессинг
img = keras.preprocessing.image.load_img(img_path, target_size=(224,
224))
img_tensor = keras.preprocessing.image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor = tf.keras.applications.vgg16.preprocess_input(img_tensor)

print("Изображение после препроцессинга")
plt.imshow((img_tensor[0] + [103.939, 116.779, 123.68]).astype(np.uint8))
plt.show()
```

Изображение после препроцессинга



Продemonстрируем визуализацию каналов активации слоя  
block1\_conv1

```
model = tf.keras.applications.VGG16(weights='imagenet', include_top=True)
layer_outputs = [layer.output for layer in model.layers[1:19]]
activation_model = keras.models.Model(inputs=model.input,
outputs=layer_outputs)
activations = activation_model.predict(img_tensor)

first_layer_activation = activations[0]
print("Форма слоя", model.layers[1].name)
print(first_layer_activation.shape)
print()

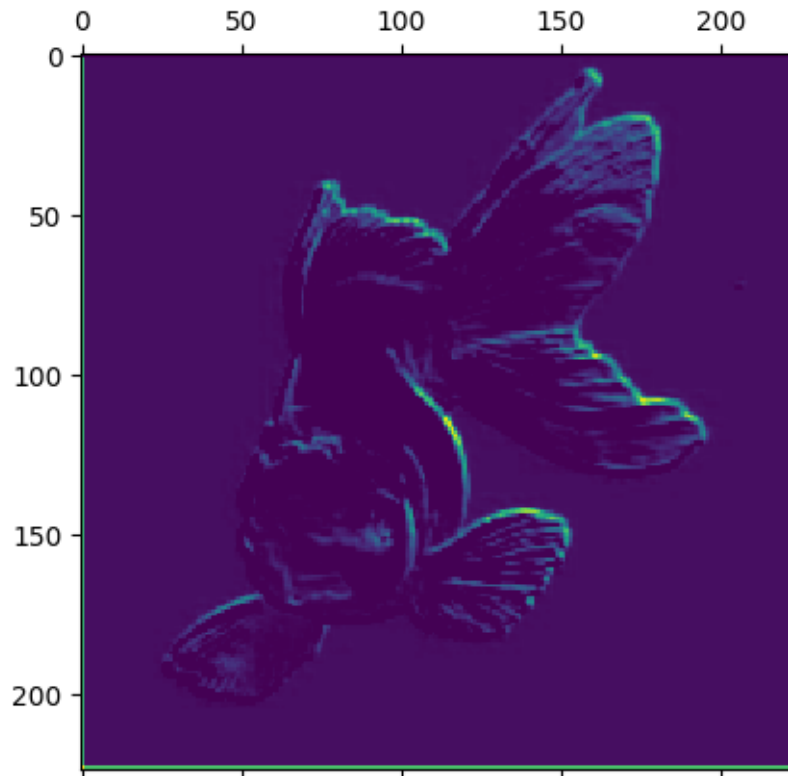
print("54 канал активации первого слоя")
plt.matshow(first_layer_activation[0, :, :, 54], cmap='viridis')
plt.show()
```

1/1 ————— 0s 182ms/step

Форма слоя block1\_conv1

(1, 224, 224, 64)

54 канал активации первого слоя



## Составим карту признаков

Составим карты признаков всех слоев свертки и пулинга модели VGG16. Начинаем со слоя block\_conv1, заканчиваем слоем block5\_pool.

```
layer_names = []
for layer in model.layers[1:19]:
    layer_names.append(layer.name)
images_per_row = 16
for layer_name, layer_activation in zip(layer_names, activations):
    n_features = layer_activation.shape[-1]
    size = layer_activation.shape[1]
    n_cols = n_features // images_per_row
    display_grid = np.zeros((size * n_cols, images_per_row * size))
    for col in range(n_cols):
        for row in range(images_per_row):
            channel_image = layer_activation[0, :, :, col *
images_per_row + row]
            channel_image -= channel_image.mean()
            channel_image /= (channel_image.std() + 1e-5)
            channel_image *= 64
            channel_image += 128
            channel_image = np.clip(channel_image, 0,
255).astype('uint8')
```

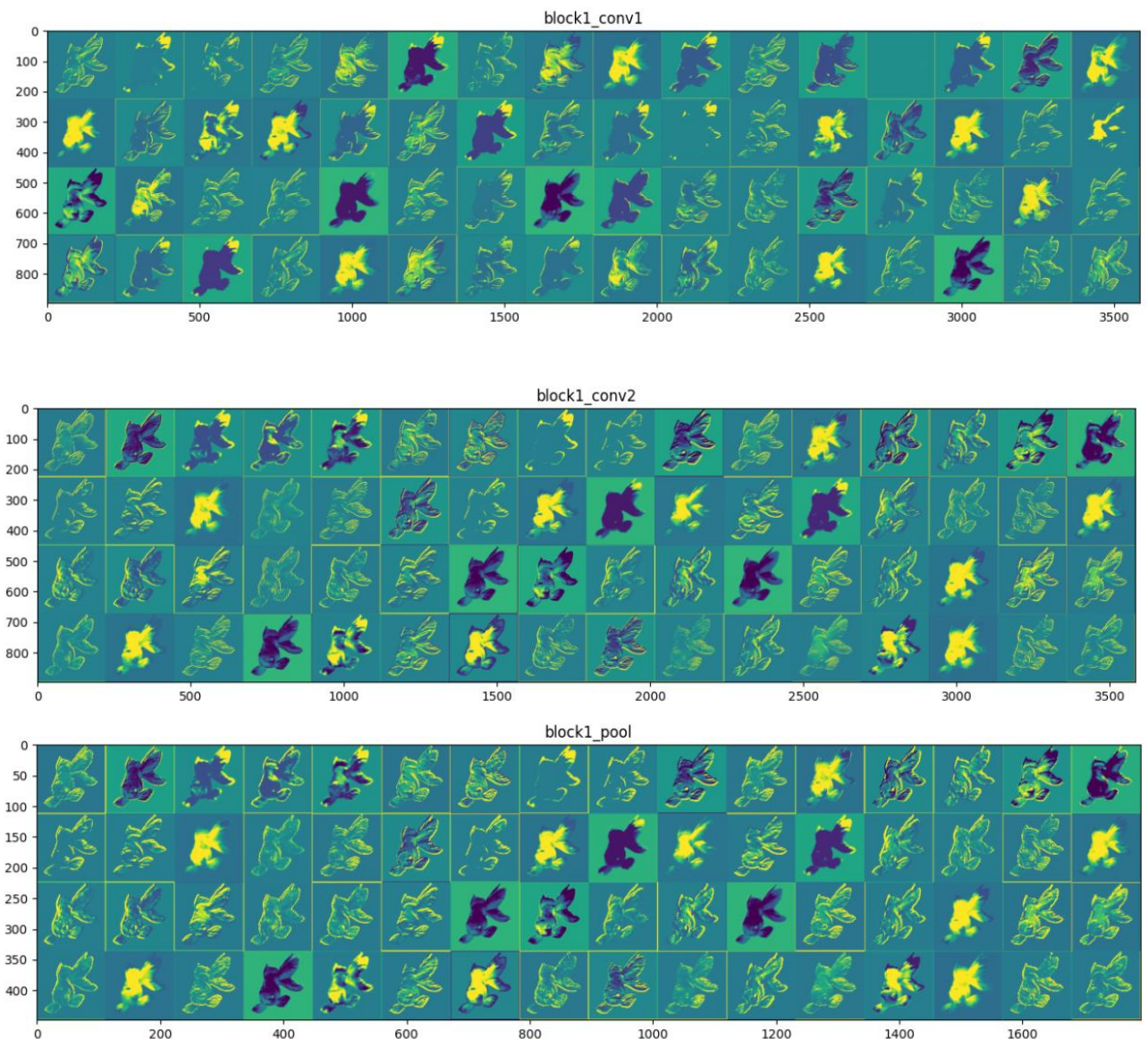
```

        display_grid[col * size : (col + 1) * size, row * size : (row
+ 1) * size] = channel_image
        scale = 1. / size
        plt.figure(figsize=(scale * display_grid.shape[1], scale *
display_grid.shape[0]))
        plt.title(layer_name)
        plt.grid(False)
        plt.imshow(display_grid, aspect='auto', cmap='viridis')
        save_path = 'images/feature_maps/' + layer_name + '.png'
        print('Карта признаков слоя', layer_name, 'сохранена в файл', save_path)
        plt.savefig(Path(save_path).resolve())
        plt.close()

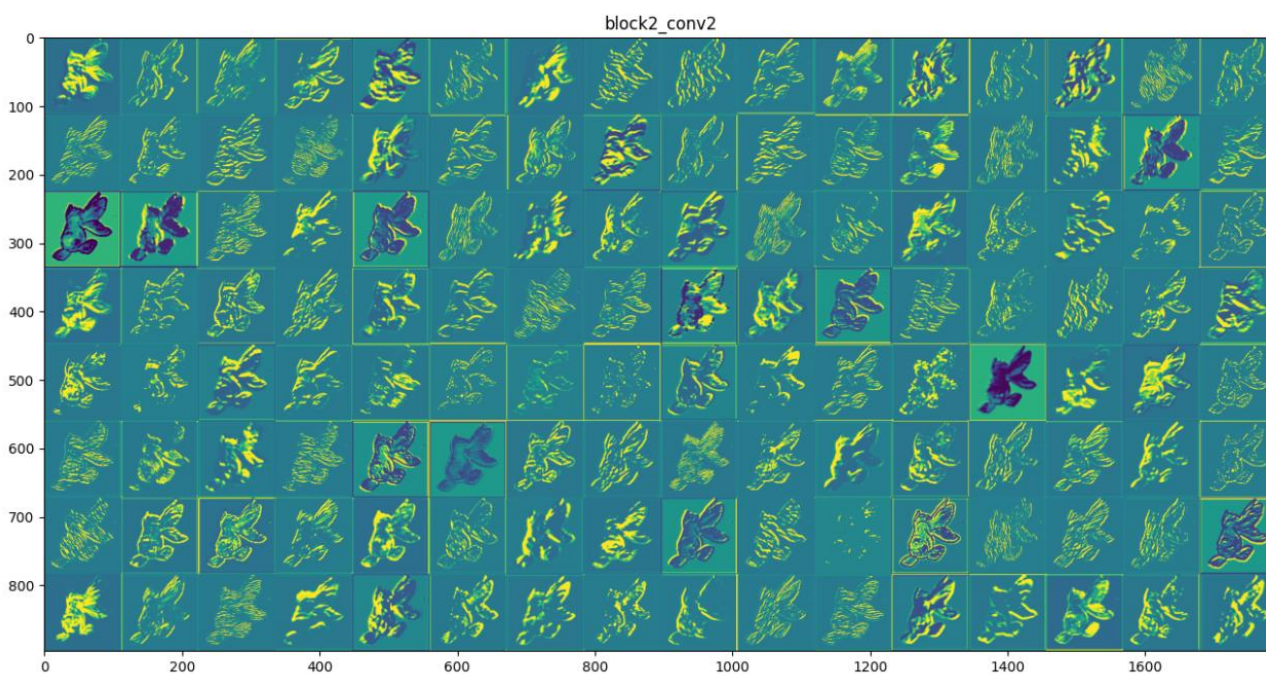
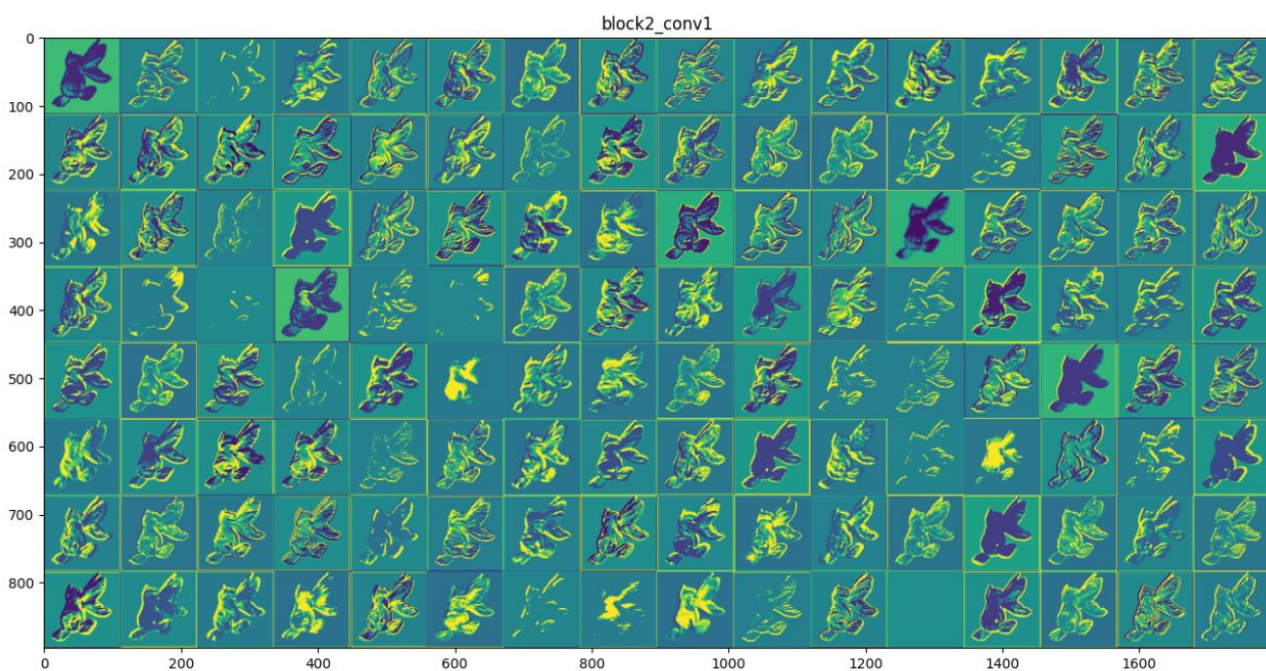
```

Приведу примеры сохраненных изображений, изображения для всех полученных слоев доступны в репозитории

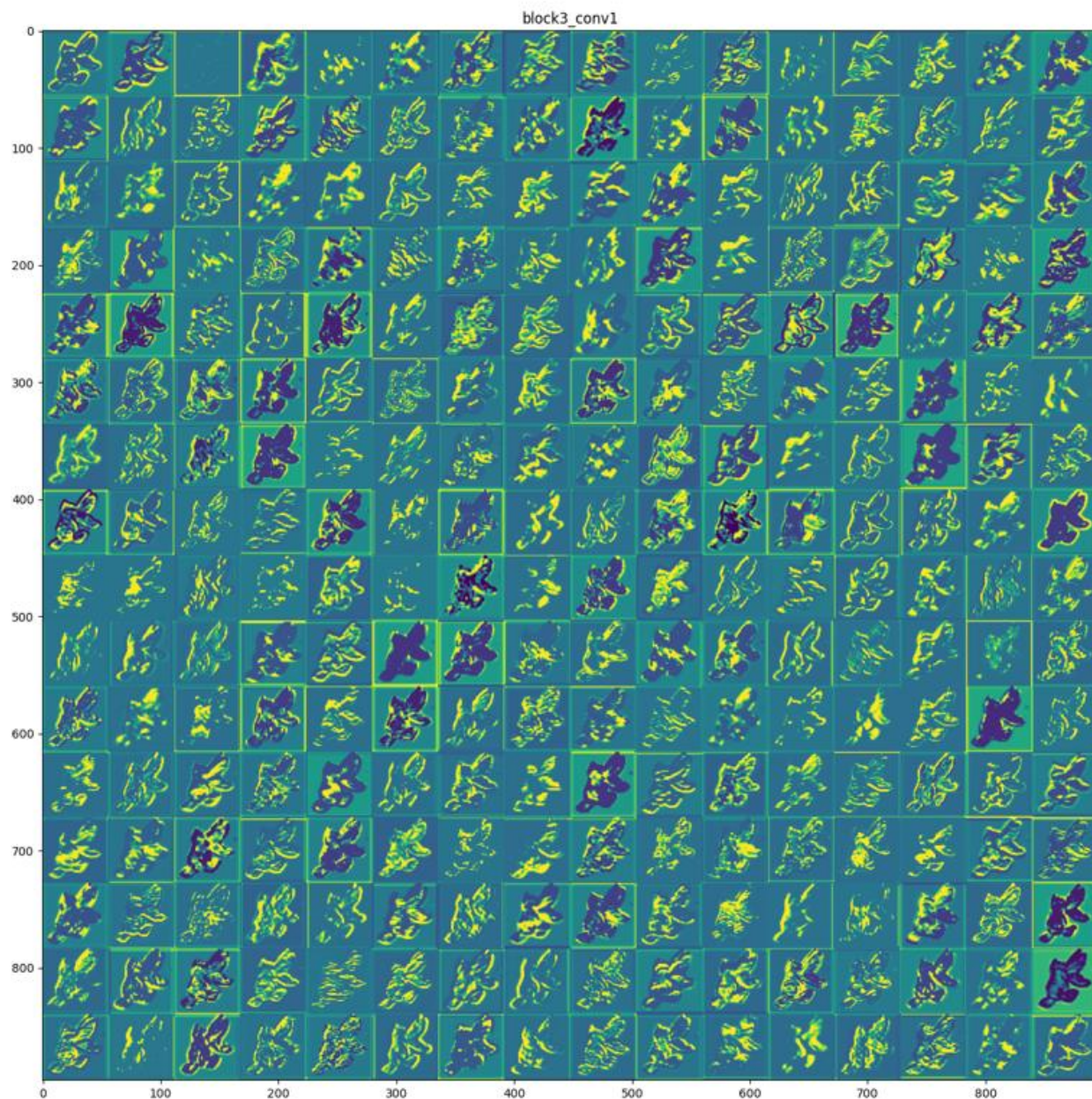
[https://github.com/Dariarty/Deep\\_Learning\\_Methods/tree/main/src/LAB1/images/feature\\_maps](https://github.com/Dariarty/Deep_Learning_Methods/tree/main/src/LAB1/images/feature_maps)

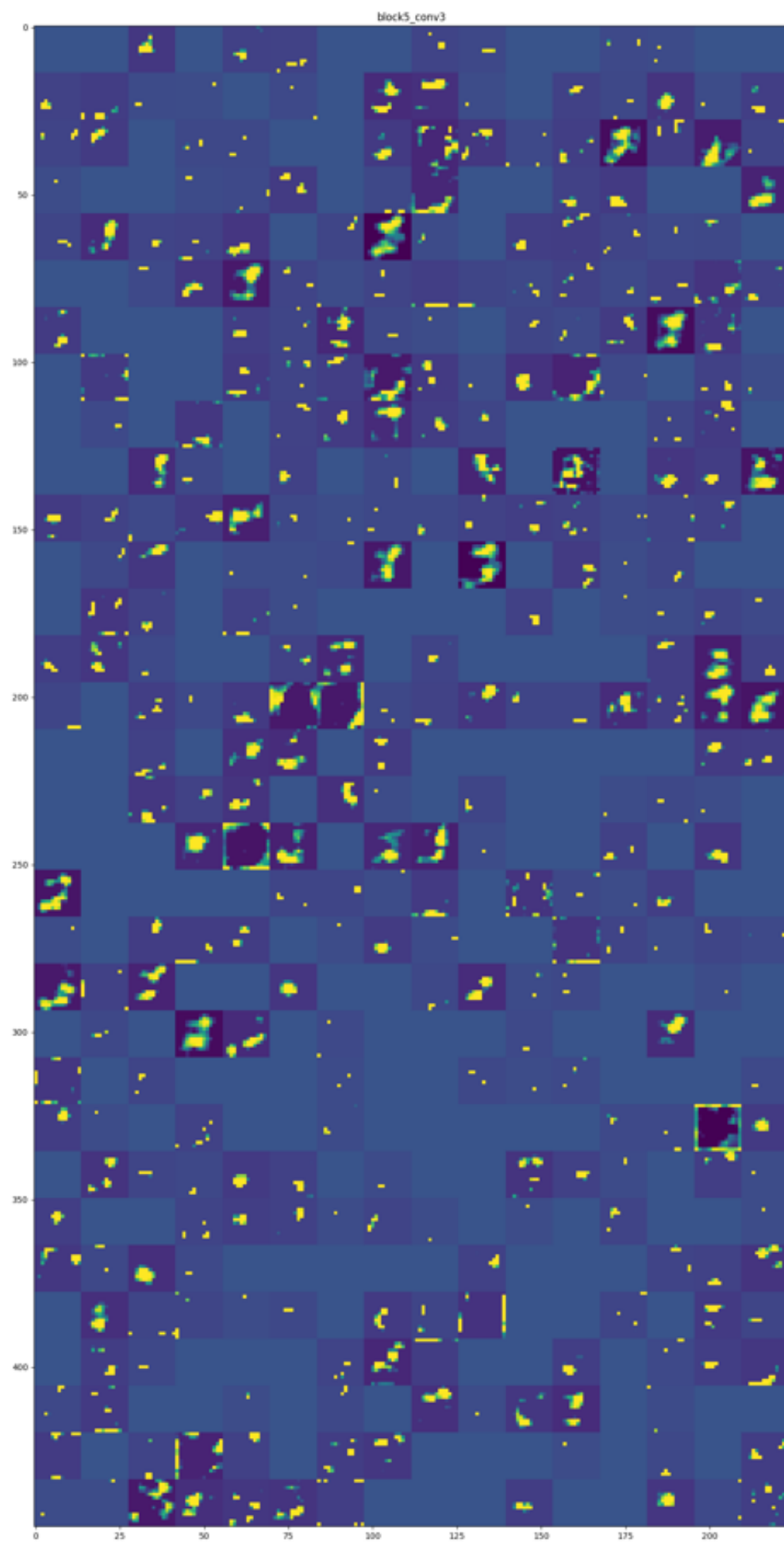












### Задание 3. Градиентный спуск.

[https://github.com/Dariarty/Deep\\_Learning\\_Methods/blob/main/src/LAB1/gradient\\_descent.ipynb](https://github.com/Dariarty/Deep_Learning_Methods/blob/main/src/LAB1/gradient_descent.ipynb)

В данной работе реализуем градиентный спуск для вывода шаблонов фильтров слоев модели VGG16

Выводим первые 64 фильтра слоев block1\_conv1, block2\_conv1, block3\_conv1, block4\_conv1. Изображения выводятся в размере 64x64 в сетке 8x8.

```
#В данной работе использую Python 3.12.7 и tensorflow 2.19

import os
os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'

import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras

# Загружаем модель VGG16
model = keras.applications.VGG16(weights='imagenet', include_top=False)

def deprocess_image(x):
    x -= x.mean()
    x /= (x.std() + 1e-5)
    x *= 0.1
    x += 0.5
    x = np.clip(x, 0, 1)
    x *= 255
    x = np.clip(x, 0, 255).astype('uint8')
    return x

def generate_pattern(layer_name, filter_index, size=64):
    input_img_data = tf.Variable(np.random.random((1, size, size, 3)) * 20 + 128., dtype=tf.float32)
    step = 1.0
    activation_model = keras.Model(inputs=model.input,
    outputs=model.get_layer(layer_name).output)
    for i in range(40):
        with tf.GradientTape() as tape:
            tape.watch(input_img_data)
            activation = activation_model(input_img_data)
            loss = tf.reduce_mean(activation[:, :, :, filter_index])
            grads = tape.gradient(loss, input_img_data)
            grads /= (tf.sqrt(tf.reduce_mean(tf.square(grads))) + 1e-5)
            input_img_data.assign_add(grads * step)
    img = input_img_data.numpy()[0]
    return deprocess_image(img)
```



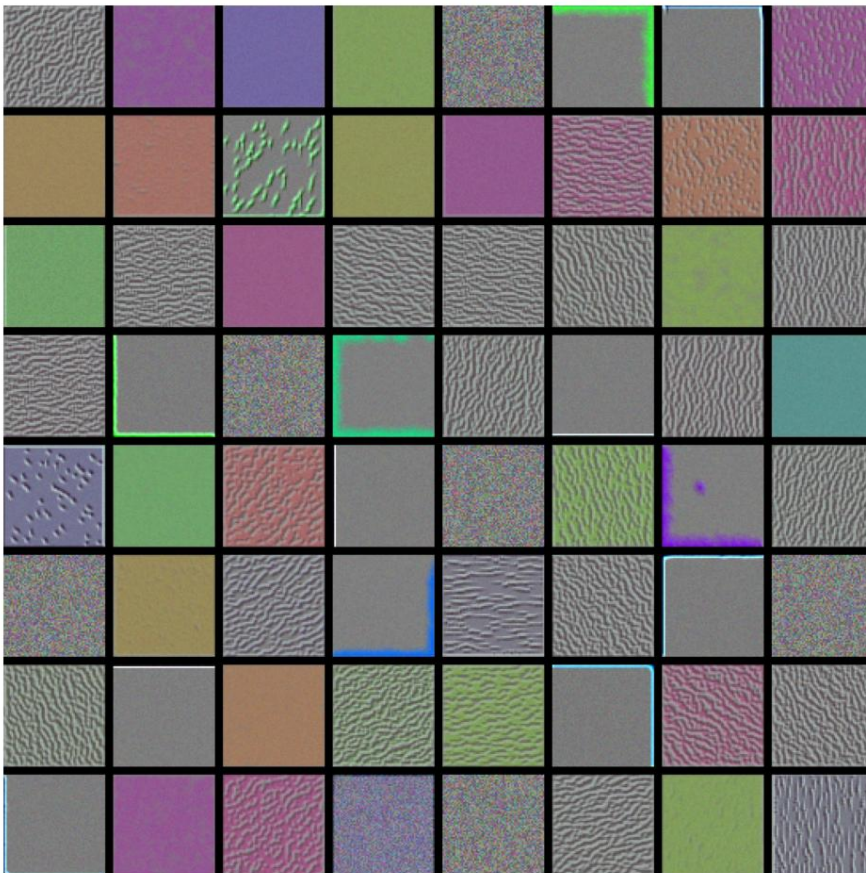
```

def display_results(layer_name):
    # Параметры визуализации
    size = 64
    margin = 5
    print("Шаблоны фильтров из слоя", layer_name)
    results = np.zeros((8 * size + 7 * margin, 8 * size + 7 * margin, 3),
dtype=np.uint8)
    for i in range(8):
        for j in range(8):
            filter_img = generate_pattern(layer_name, i + (j * 8), size=size)
            horizontal_start = i * size + i * margin
            horizontal_end = horizontal_start + size
            vertical_start = j * size + j * margin
            vertical_end = vertical_start + size
            results[horizontal_start:horizontal_end,
vertical_start:vertical_end, :] = filter_img
    plt.figure(figsize=(12, 12))
    plt.imshow(results)
    plt.axis('off')
    plt.show()

display_results('block1_conv1')
display_results('block2_conv1')
display_results('block3_conv1')
display_results('block4_conv1')

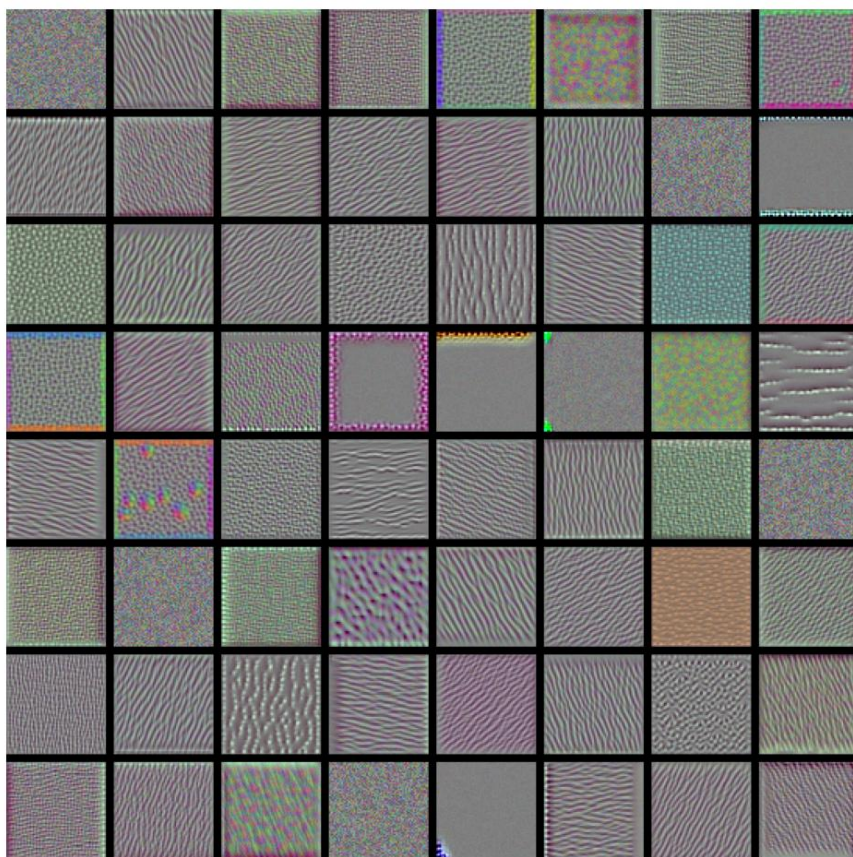
```

Шаблоны фильтров из слоя block1\_conv1

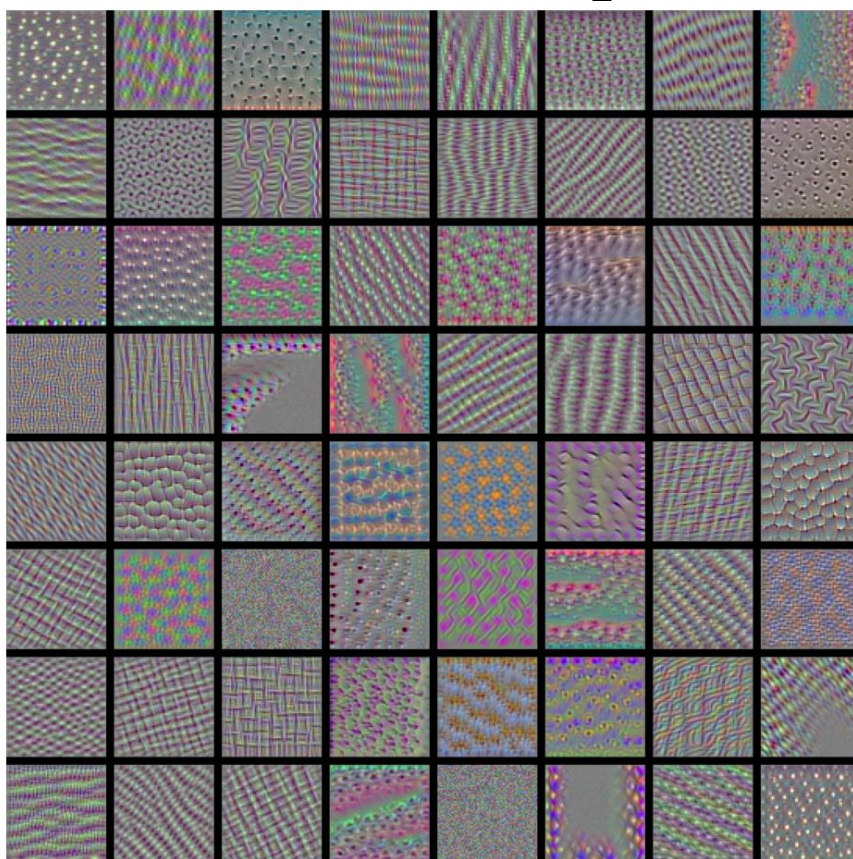




Шаблоны фильтров из слоя block2\_conv1

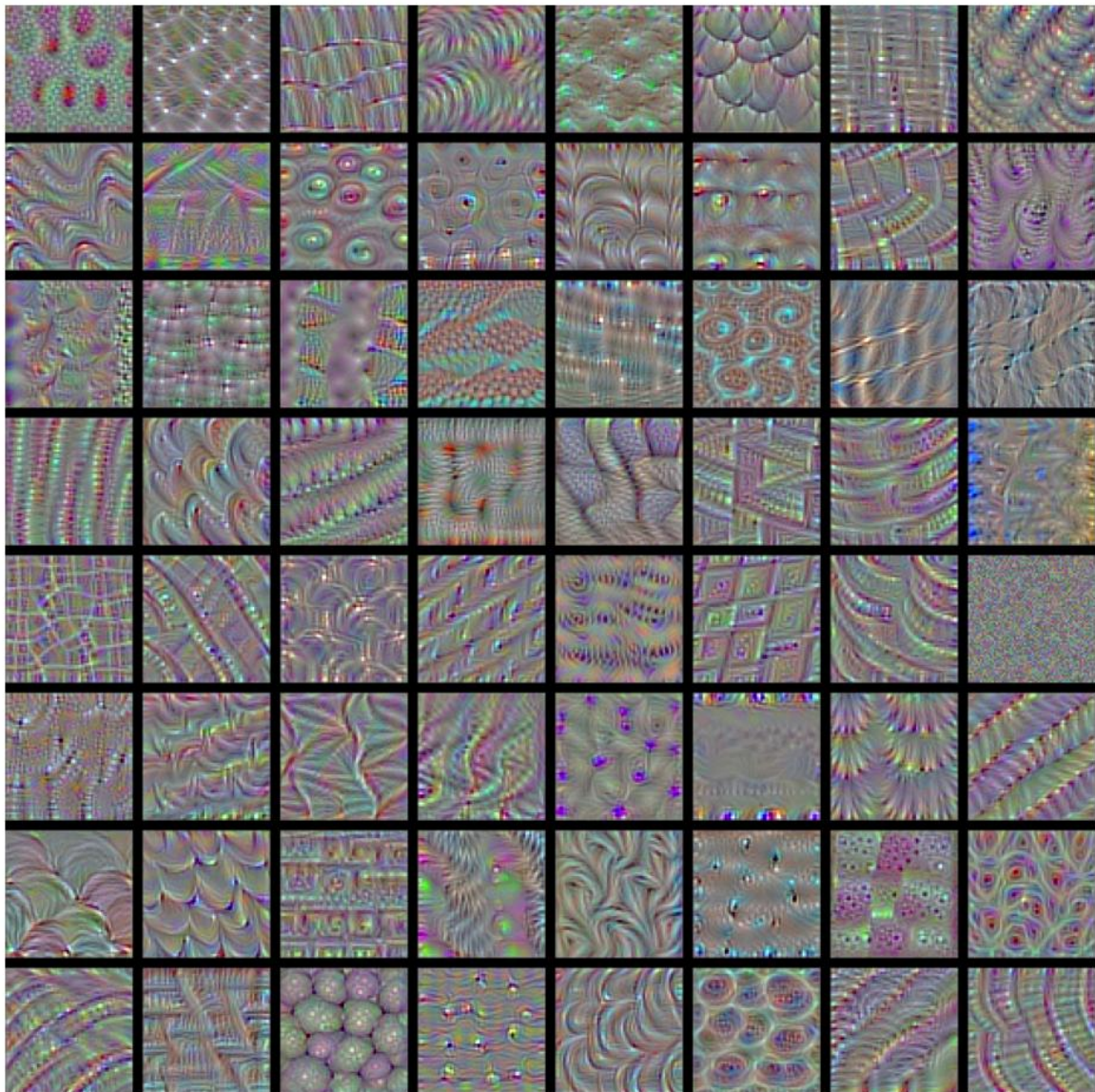


Шаблоны фильтров из слоя block3\_conv1





Шаблоны фильтров из слоя block4\_conv1



**Вывод:** в ходе выполнения лабораторной работы была продемонстрирована работа тепловых карт, карт признаков и был реализован градиентный спуск согласно варианту задания.