Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Южно-Уральский государственный университет»
(национальный исследовательский университет)
Институт естественных и точных наук
Кафедра прикладной математики и программирования

Отчет по лабораторной работе № 6
по дисциплине «Современные нейросетевые технологии»

Авторы работы
Студент группы ЕТ-122
_____/ Матвеева А.В.
«____» _____2025 г.

Руководитель работы,
_____/ Кичеев Д.М.
«____» _____2025 г.

Челябинск 2025

Задание:

Применение переноса обучения для решения задачи, поставленной во второй лабораторной работе

Цель настоящей работы состоит в том, чтобы исследовать возможности переноса обучения для решения целевой задачи, выбранной изначально для выполнения практических работ.

Выполнение практической работы предполагает решение следующих задач:

1. Поиск исходной задачи (близкой по смыслу к целевой задаче) и поиск натренированной модели для решения исходной задачи.
2. Выполнение трех типов экспериментов по переносу знаний (типы экспериментов описаны в лекции).
3. Сбор результатов экспериментов.

1. В ходе данного проекта я освоила базовые навыки работы с библиотекой глубокого обучения PyTorch на примере переноса весов. Я использовала модель VGG16, так как она широко используется, имеет предобученные веса в torchvision, и её полносвязная часть легко адаптируется под прошлые эксперименты.

2. Были использованы данные подмножество набора данных Food-101. Боссард, Лукас, Матье Гийомен и Люк Ван Гул. «Food-101 – Mining Discriminative Components with Random Forests». Изображения разделены на две папки: pizza и not_pizza, имеющее равное количество изображений (по 983 шт.). Все изображения были масштабированы таким образом, чтобы максимальная длина стороны составляла 512 пикселей. Содержание папок соответствует их названию.

   Данные разделены:

   Количество обучающих изображений: 1376

   Количество валидационных изображений: 294

   Количество тестовых изображений: 296

3. Я использую метрику качества Accuracy, которая определяется следующим образом:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

где:

- *TP* (True Positives) — количество истинных положительных предсказаний (правильно классифицированные объекты класса 1).

- *TN* (True Negatives) — количество истинных отрицательных предсказаний (правильно классифицированные объекты класса 0).
- *FP* (False Positives) — количество ложных положительных предсказаний (объекты класса 0, ошибочно классифицированные как класс 1).
- *FN* (False Negatives) — количество ложных отрицательных предсказаний (объекты класса 1, ошибочно классифицированные как класс 0).

4. Данные мной были скачаны с сайта kaggle, хранятся локально в двух папках файлами в формате . Jpg.
5. Я использовала создание датасета с помощью встроенной функции torch.utils.data.Dataset,, в нее подавались пути к изображениям и изображения, а на выходе получили Кортеж (features, label): где features — это тензор с признаками, а label — это тензор с меткой класса.

   После этого я использовала загрузчик данных DataLoder. В него подается полученный на предыдущем этапе датасет, а в итоге получаем пакеты данных (batch), состоящие из кортежей (features, labels) для каждого образца в пакете.

6. Мной были проведены четыре эксперимента по переносу весов модели

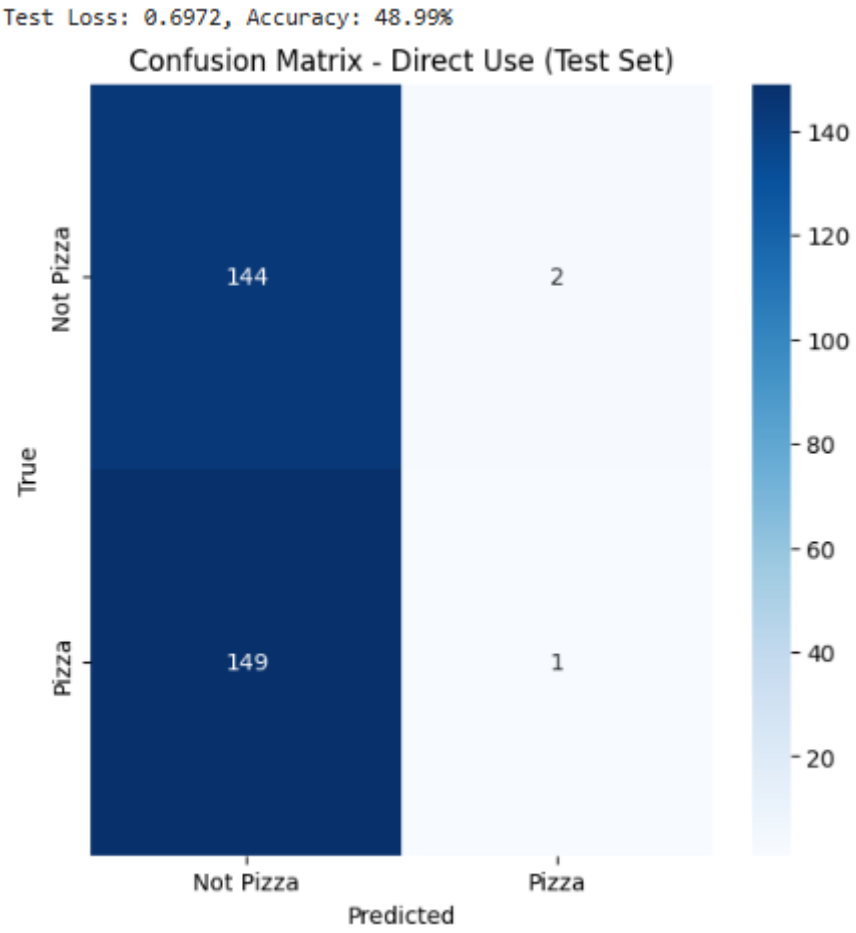
Для обучения я использовала модель VGG16/ на была адаптирована и в нее добавлены полносвязные слои как в лаб №2.

```python
# загружаем модель, модифицируем конец модели
class VGG16Adapted(nn.Module):
    def __init__(self):
        super(VGG16Adapted, self).__init__()
        vgg16 = models.vgg16(pretrained=True)
        self.features = vgg16.features
        self.avgpool = vgg16.avgpool
        self.classifier = nn.Sequential(
            nn.Linear(512 * 7 * 7, 512),
            nn.BatchNorm1d(512),
            nn.LeakyReLU(negative_slope=0.01),
            nn.Dropout(p=0.5),
            nn.Linear(512, 256),
            nn.BatchNorm1d(256),
            nn.LeakyReLU(negative_slope=0.01),
            nn.Dropout(p=0.5),
            nn.Linear(256, 128),
            nn.BatchNorm1d(128),
            nn.LeakyReLU(negative_slope=0.01),
            nn.Linear(128, 1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

- Прямое использование модели (Parameter Transfer)

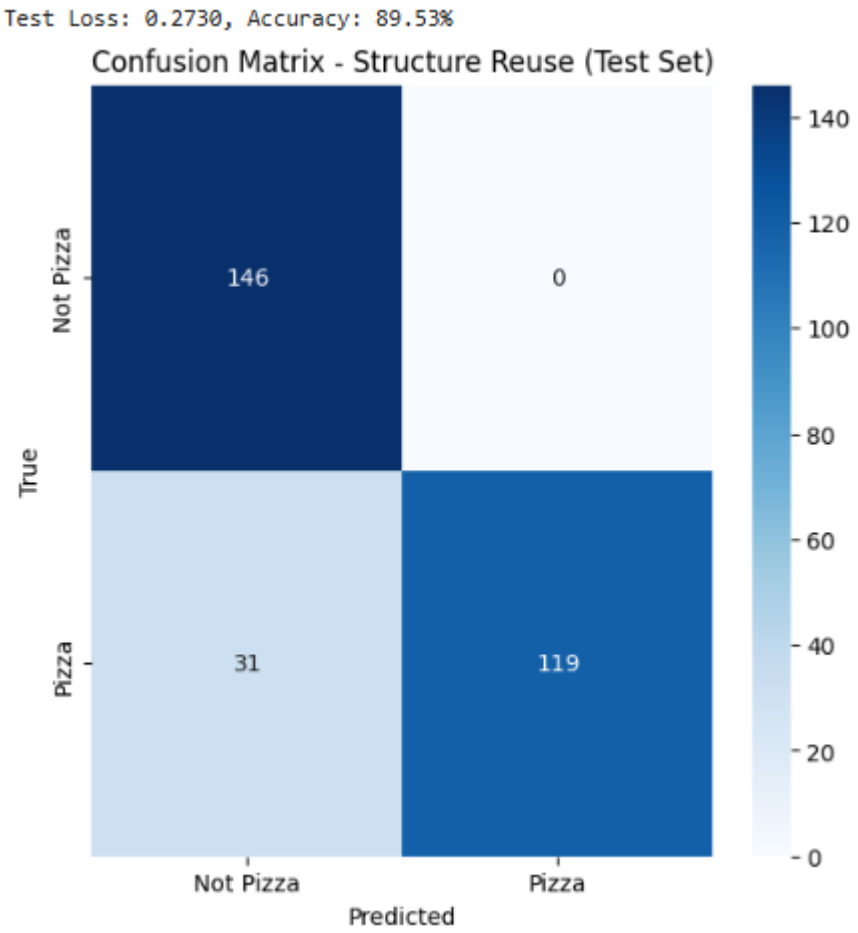
Использовала VGG16, адаптированную под бинарную классификацию, с весами, дообученными на ImageNet, и тестируем её на данных «пицца/не пицца» без дополнительного обучения. Это проверяет, насколько предобученная модель обобщает на вашу задачу. Итог:



Модель не классифицирует. Совсем.

- Использование структуры модели с обучением на новых данных (Transfer Learning for Relational Domains)

Использую архитектура VGG16, но веса инициализируются случайно, и модель обучается с нуля на данных «пицца/не пицца». Это проверяет, насколько структура модели подходит для задачи. Итог:

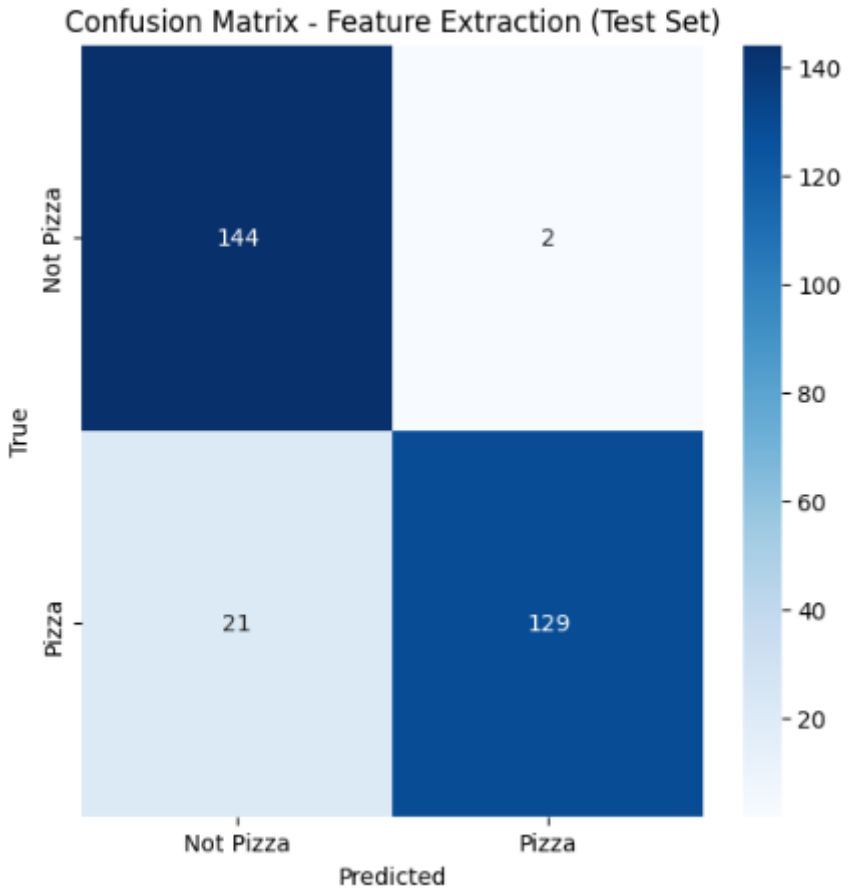


Хорошая классификация, но признаки переобучения.

- Использование модели как фиксированного метода извлечения признаков (Feature Representation Transfer)

Свёрточная часть VGG16 (предобученная на ImageNet) используется для извлечения признаков, а полносвязная часть(аналигичная модели в лаб №2) обучается на данных «пицца/не пицца». Свёрточные слои замораживаются. Итог:

```
Test Loss: 0.2238, Accuracy: 92.23%
```

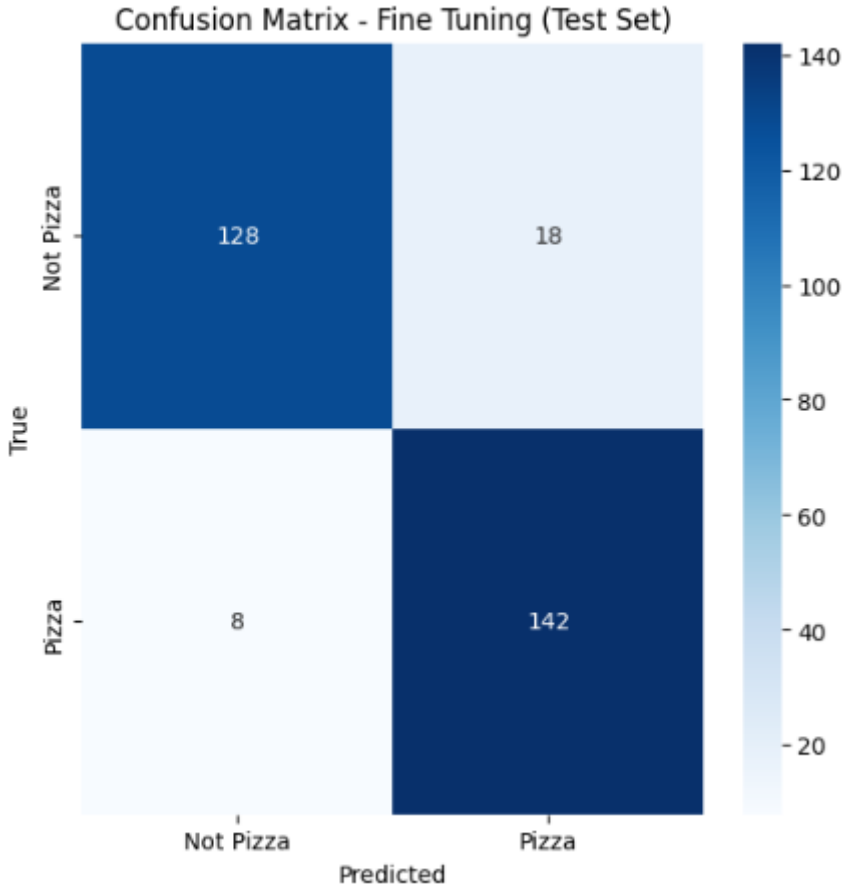


Лучшая метрика, модель быстро и хорошо обучается

- Тонкая настройка параметров (Instance-based Transfer Learning)

Используем предобученную VGG16, заменяем полносвязную часть на структуру, схожую с моделью из лаб №2, и дообучаем всю модель на данных «пицца/не пицца» с меньшим learning rate.Итог:

```
Test Loss: 0.1988, Accuracy: 91.22%
```



7. Результаты:

Вид переноса	Accuracy
Parameter Transfer	48,99 %
Transfer Learning for Relational Domains	89,53 %
Feature Representation Transfer	92,23 %
Instance-based Transfer Learning	91,22 %