Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет систем управления и робототехники

Отчет по практической работе №1

«КЛАССИФИКАЦИЯ ОБЪЕКТОВ С ПОМОЩЬЮ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ»

по дисциплине «Глубокое обучение»

Выполнил: студент гр. R4124c
Абрамов М. В.

Преподаватель: Евстафьев О. А.

Санкт-Петербург
2025

1. Цель работы
   – Обучить модель для задачи классификации объектов на изображении

2. Подготовка набора данных

```
class MyDataset(Dataset):  4 usages  Max
    def __init__(self, path, transform=None, target_transform=None, labels={}):  Max
        self.__labels = labels
        self.__img_labels = []
        self.__img_paths = []
        for i, sub_dir in enumerate(os.listdir(path)):
            if sub_dir not in self.__labels:
              self.__labels[sub_dir] = len(self.__labels)
            sub_dir_path = os.path.join(path, sub_dir)
            for image_path in os.listdir(sub_dir_path):
                self.__img_labels.append(i)
                self.__img_paths.append(os.path.join(sub_dir_path, image_path))
        self.__transform = transform
        self.__target_transform = target_transform

    def __len__(self):  Max
        return len(self.__img_labels)

    def __getitem__(self, idx):  Max
        image = read_image(self.__img_paths[idx])
        label = self.__img_labels[idx]
        if self.__transform:
            image = self.__transform(image)
        if self.__target_transform:
            label = self.__target_transform(label)
        return image, label

    def get_labels(self):  1 usage  Max
        return self.__labels
```

Код 1 – Свой датасет

```
train_data = MyDataset(os.path.join(data_path, "train"),
                       transform=transform)
labels_names = train_data.get_labels()
valid_data = MyDataset(os.path.join(data_path, "validation"),
                       transform=transform,
                       labels=labels_names)
test_data = MyDataset(os.path.join(data_path, "test"),
                      transform=transform,
                      labels=labels_names)

train_dataloader = DataLoader(train_data, batch_size=batch_size, shuffle=True, num_workers=NUM_WORKERS,
                              persistent_workers=True)
valid_dataloader = DataLoader(valid_data, batch_size=batch_size, shuffle=False, num_workers=NUM_WORKERS,
                              persistent_workers=True)
test_dataloader = DataLoader(test_data, batch_size=batch_size, shuffle=True, num_workers=NUM_WORKERS,
                             persistent_workers=True)
```

Код 2 – Загрузка данных в даталоадер

3. EfficiencyNet

3.1 Архитектура

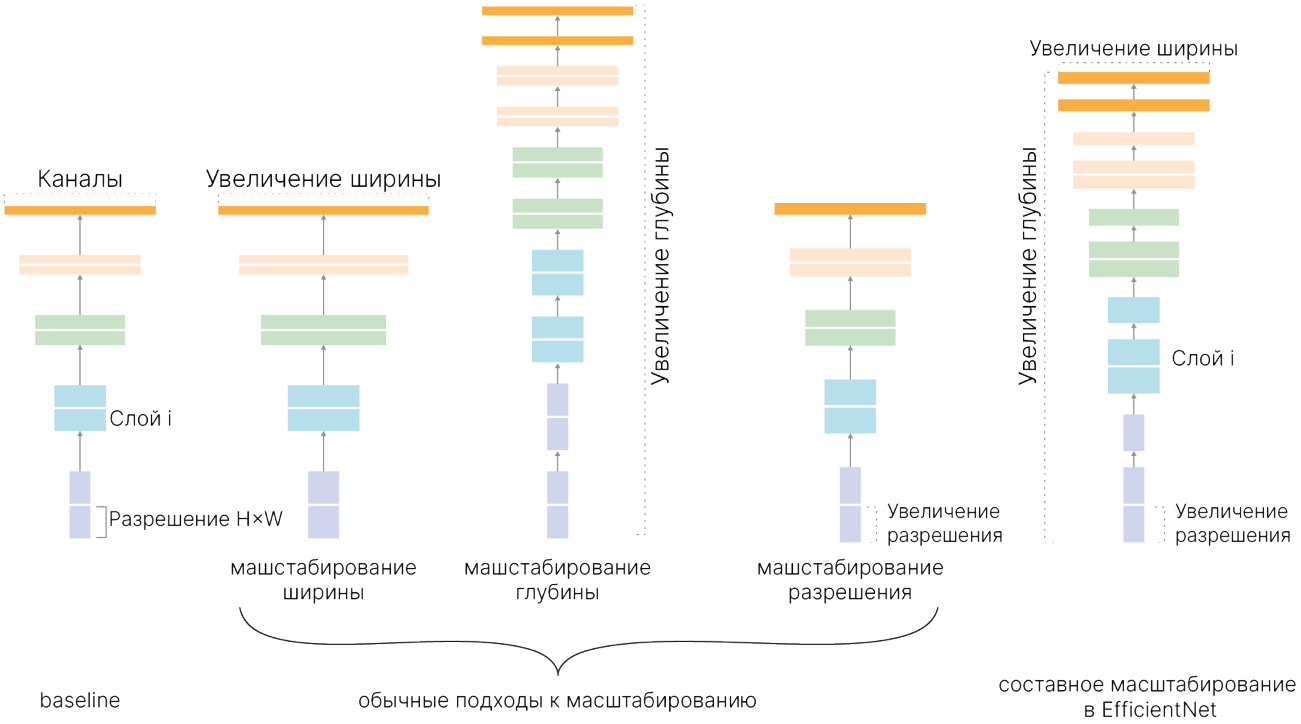

Рисунок 1 – Основная идея архитектуры EfficientNet

3.2 Обучение модели

Перед началом обучения необходимо изменить выходные слои модели для решения поставленной задачи классификации на 15 классов.

```python
def my_efficient_net(num_classes, weights):  2 usages  Max *
    model = efficientnet_b0(weights = weights)
    num_features = model.classifier[1].in_features
    model.classifier[1] = nn.Linear(num_features, num_classes)
    model.classifier.append(nn.Softmax(dim=1))
    return model
```

Код 3 – Модификация модели

Для обучения, валидации и тестирования модели модели использовался фреймворк lightning.

```python
model = model(len(labels_names), weights=weights)

logger = TensorBoardLogger(save_dir="logs", name=model.__class__.__name__)
lit_model = LModel(model, labels_names, device, learning_rate=learning_rate)
trainer = L.Trainer(max_epochs=epochs, logger=logger)

trainer.fit(model=lit_model, train_dataloaders=train_dataloader, val_dataloaders=valid_dataloader)
trainer.test(model=lit_model, dataloaders=test_dataloader)
```

Код 4 – Обучение, валидация и тестирование модели

3.3 Метрики

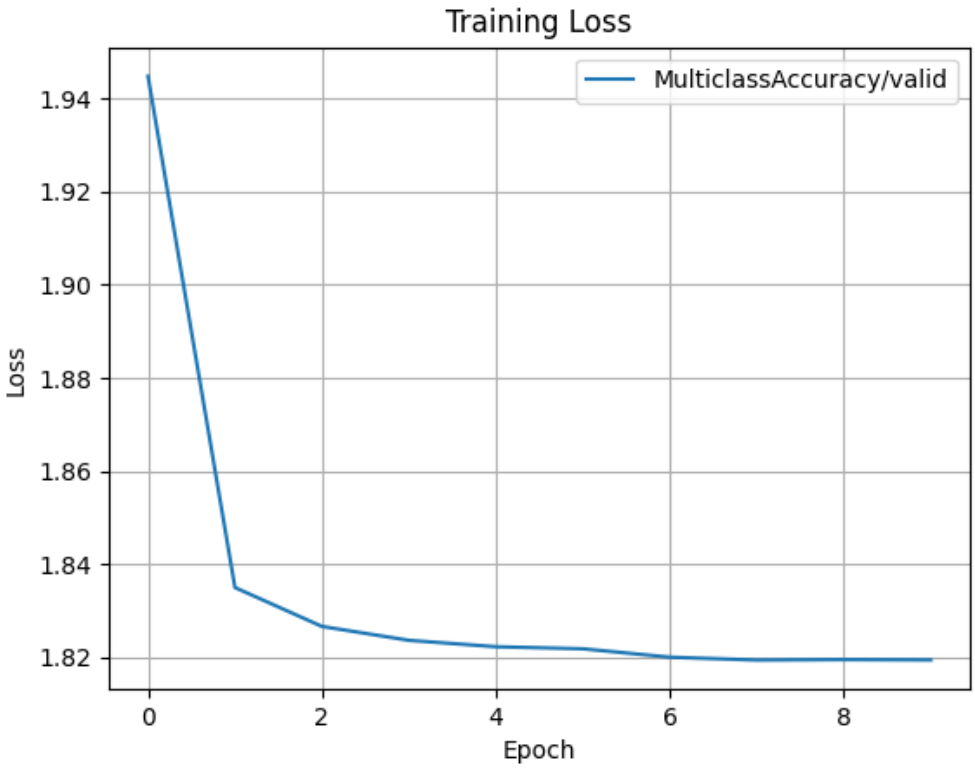

Рисунок 2 – График функции потерь во время обучения

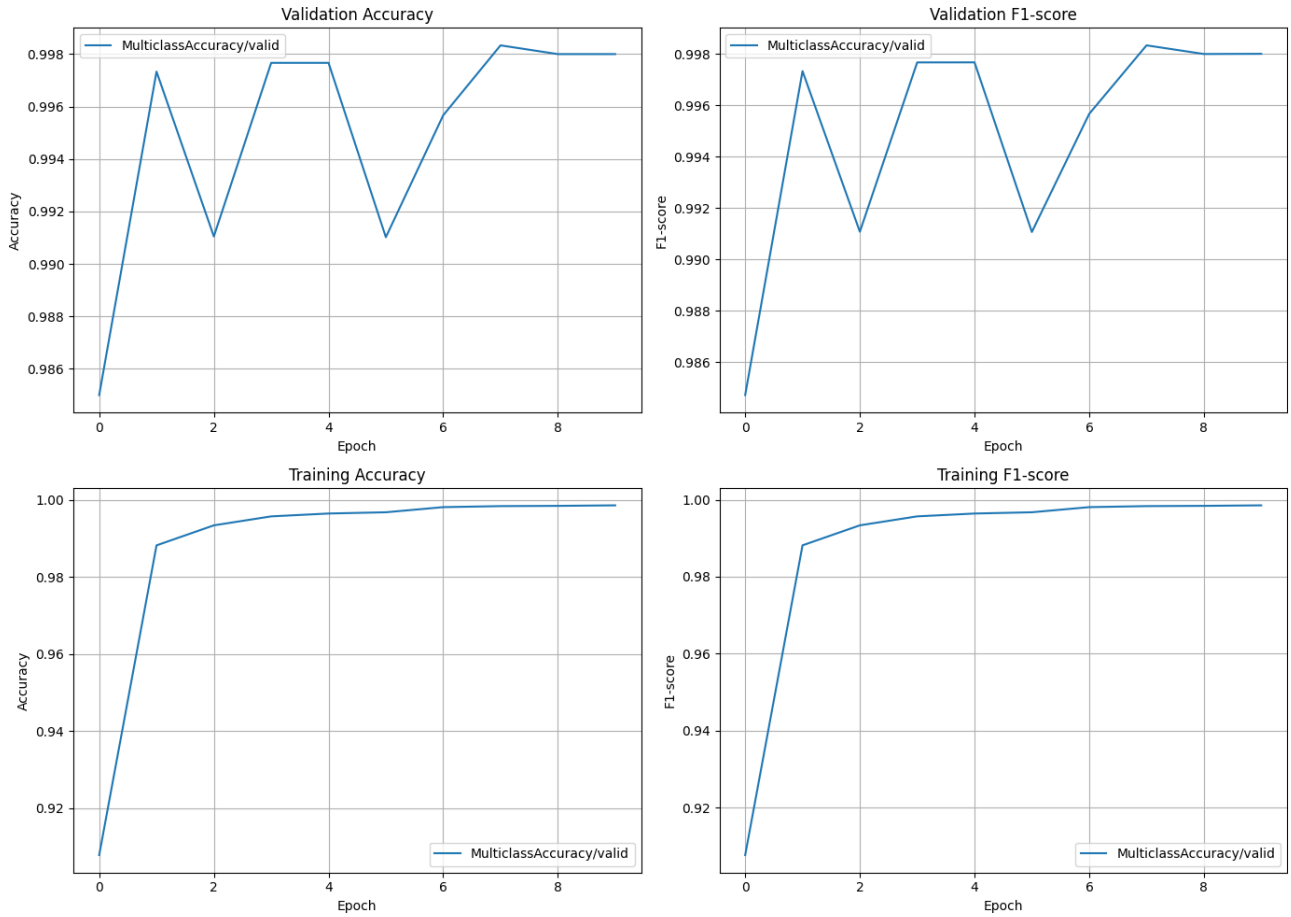

Рисунок 3 – Метрики точности модели

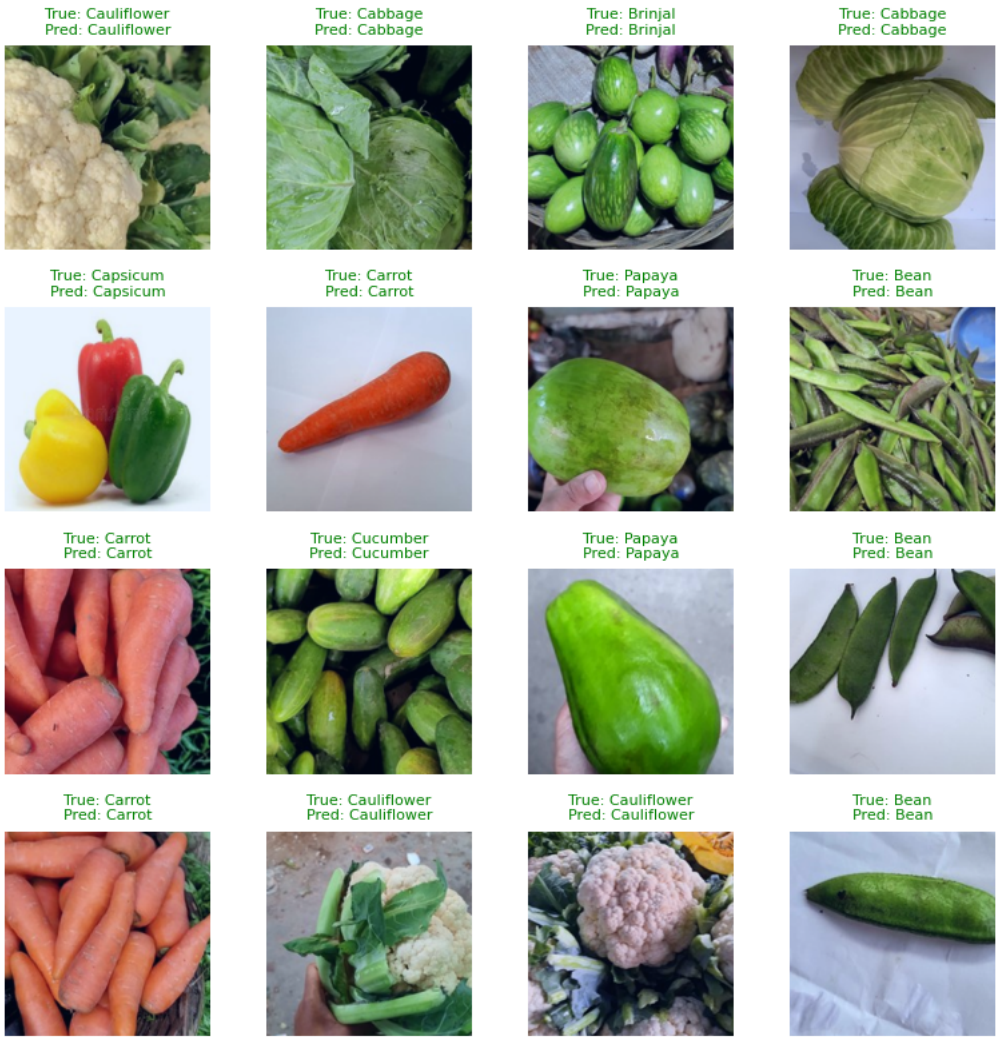

Рисунок 4 – Пример работы на тестовых изображениях

Точность модели на тестовом датасете: 99,73%

4. Своя модель

4.1 Архитектура

```
class MyModel(nn.Module):  1 usage  Max +1
    def __init__(self, num_classes, weights):  Max +1
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(in_channels=3, out_channels=32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Linear(128 * 6 * 6, out_features: 512),
            nn.ReLU(),
            nn.Dropout(0.5),
            nn.Linear( in_features: 512, num_classes)
        )

    def forward(self, x):  Maksim +1
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```

4.2 Обучение модели

Код тот же, что и при обучении EfficientNet.

4.3 Метрики

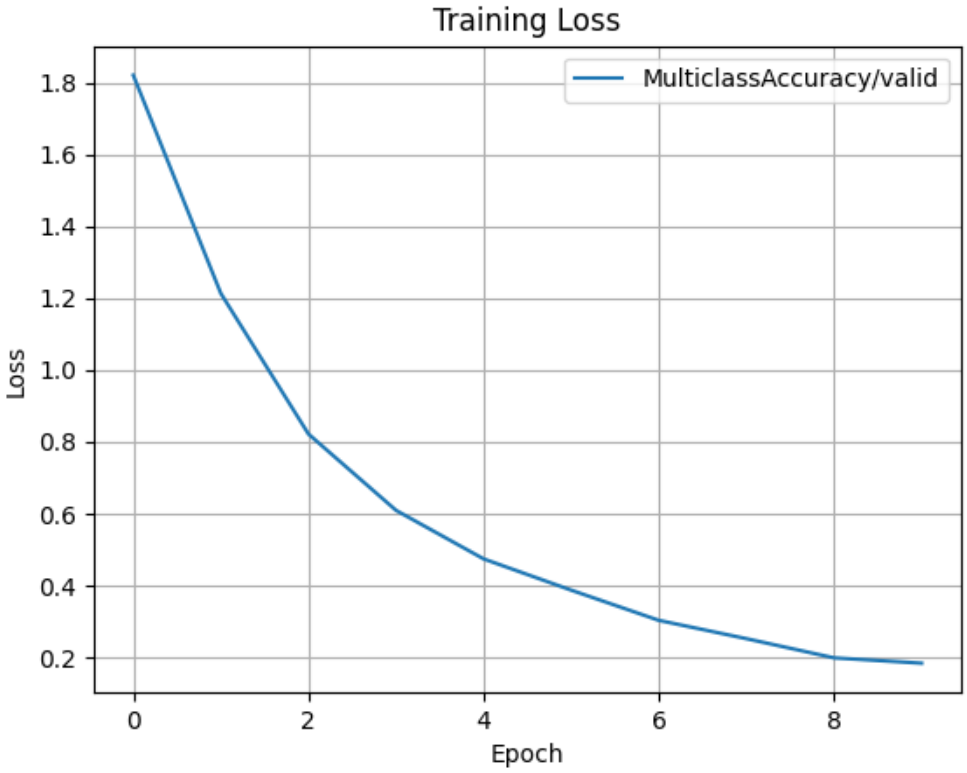

Рисунок 5 – График функции потерь во время обучения

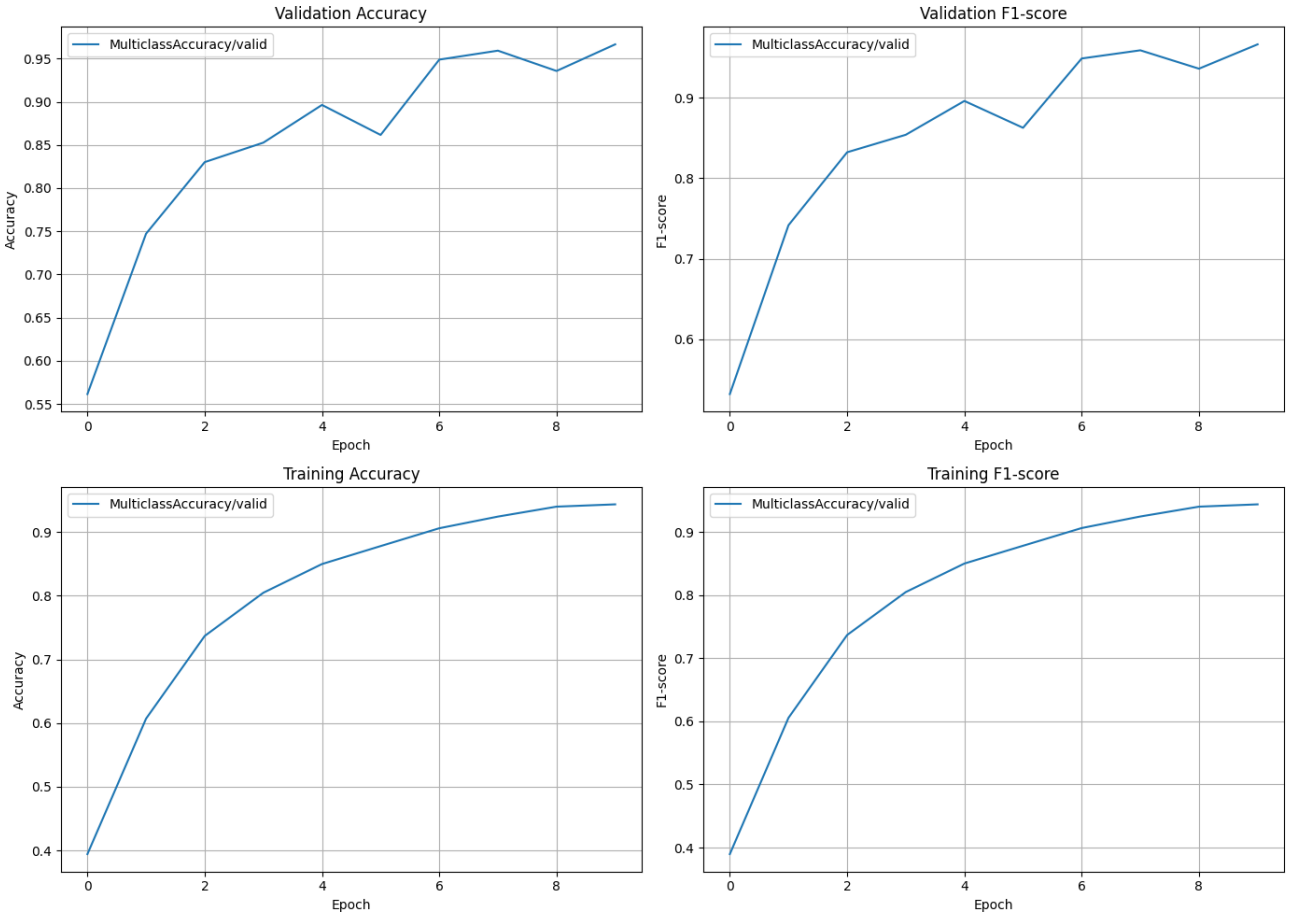

Рисунок 6 – Метрики точности модели

Рисунок 7 – Пример работы на тестовых изображениях

Точность модели на тестовом датасете: 96,93%

5. Вывод

В ходе лабораторной работы были обучены две модели сверточных нейронных сетей для решения задачи классификации изображений.