|  |  |
| --- | --- |
| **Изображение выглядит как эмблема, герб, символ, нашивка  Контент, сгенерированный ИИ, может содержать ошибки.** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ                 Информатика и системы управления

КАФЕДРА                     Системы обработки информации и управления

**Лабораторная работа №6**

**По курсу**

«**Разработка нейронных сетей**»  
**По теме «Классификация LiDAR»**

Выполнил:  
Студент группы ИУ5-22МБибиков П.А.  
24.05.2025

Проверил:

**Канев А.И.**

2025 г.

# Задание

Обучить модель нейронной сети для классификации пород деревьев, используя файлы облака точек. Указать файл по варианту и изменить список классов, которые относятся к варианту.

Обучить модель. Изменить гиперпараметры обучения для улучшения модели: количество эпох, размер батча, скорость обучения. Указать, какие действия помогли улучшить метрики ваших моделей и объяснить почему.

Номер в группе нечетный, поэтому вариант – v1.h5. Деревья – ели, сосны, березы.

# Часть 1. Обучение исходной нейронной сети.

По варианту возьмём модель v1.h5 - она содержит следующие деревья: Рябина, Ель, Сосна. На рисунке ниже представлен пример облака точек.

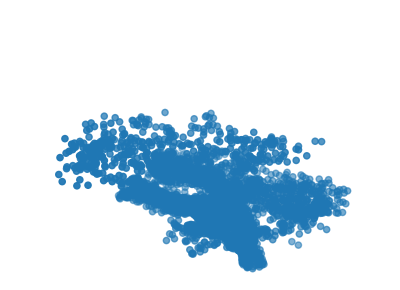


Рисунок 1. Облако точек.

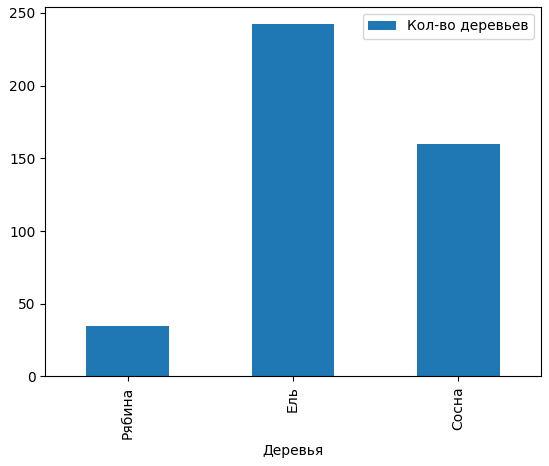


Рисунок 2. Количество деревьев по классам.

Прогоним исходную модель, посмотрим на получившиеся результаты. Ниже на рисунках показаны график точности, а также матрица ошибок.

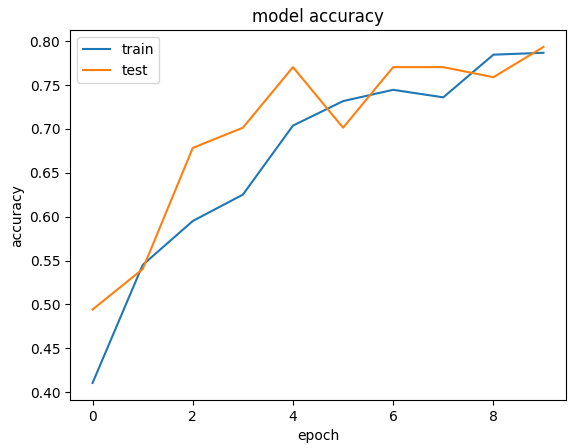


Рисунок 3. Точность исходной модели.

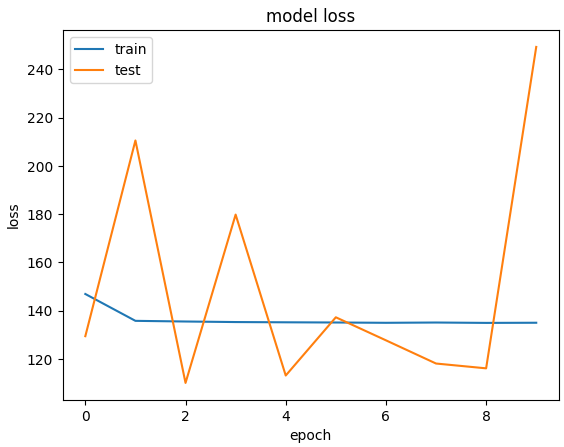


Рисунок 4. Потери исходной модели.



Рисунок 5. Результаты предсказания исходной модели для облаков точек.

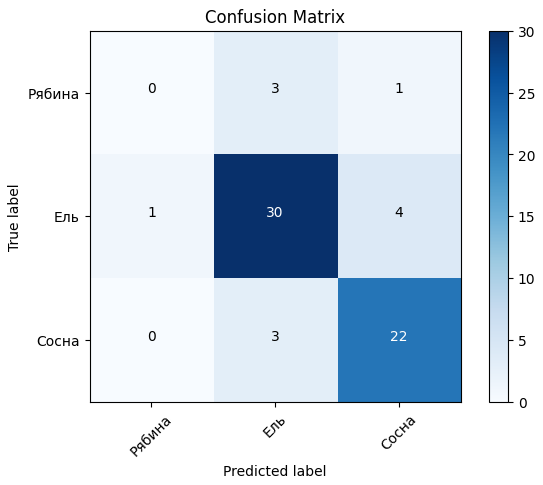


Рисунок 6. Матрица ошибок исходной модели.

# Часть 2. Улучшение модели.

Переобучение произошло на 4 эпохе, поэтому сократим число эпох до 4 и обучим модель заново. Результаты представлены на рисунках ниже.

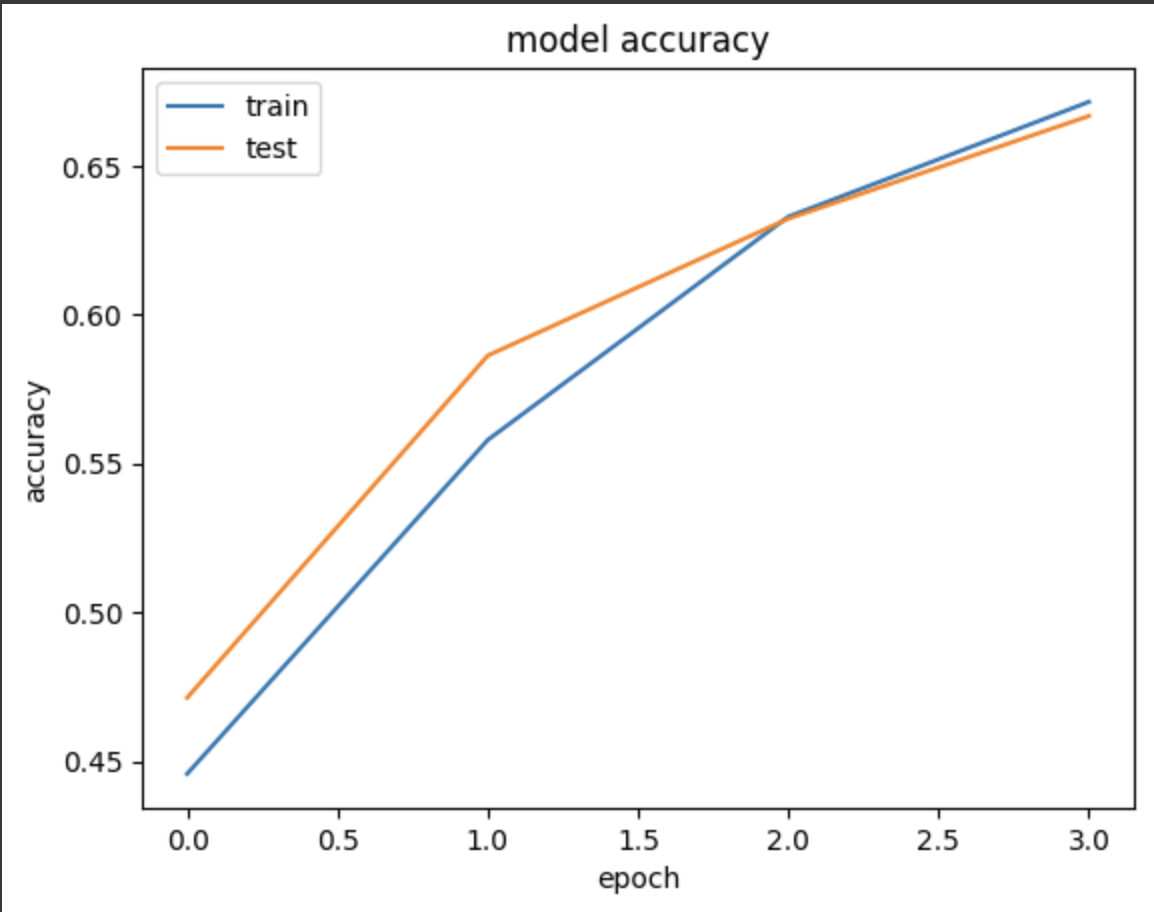


Рисунок 7. Точность модели.

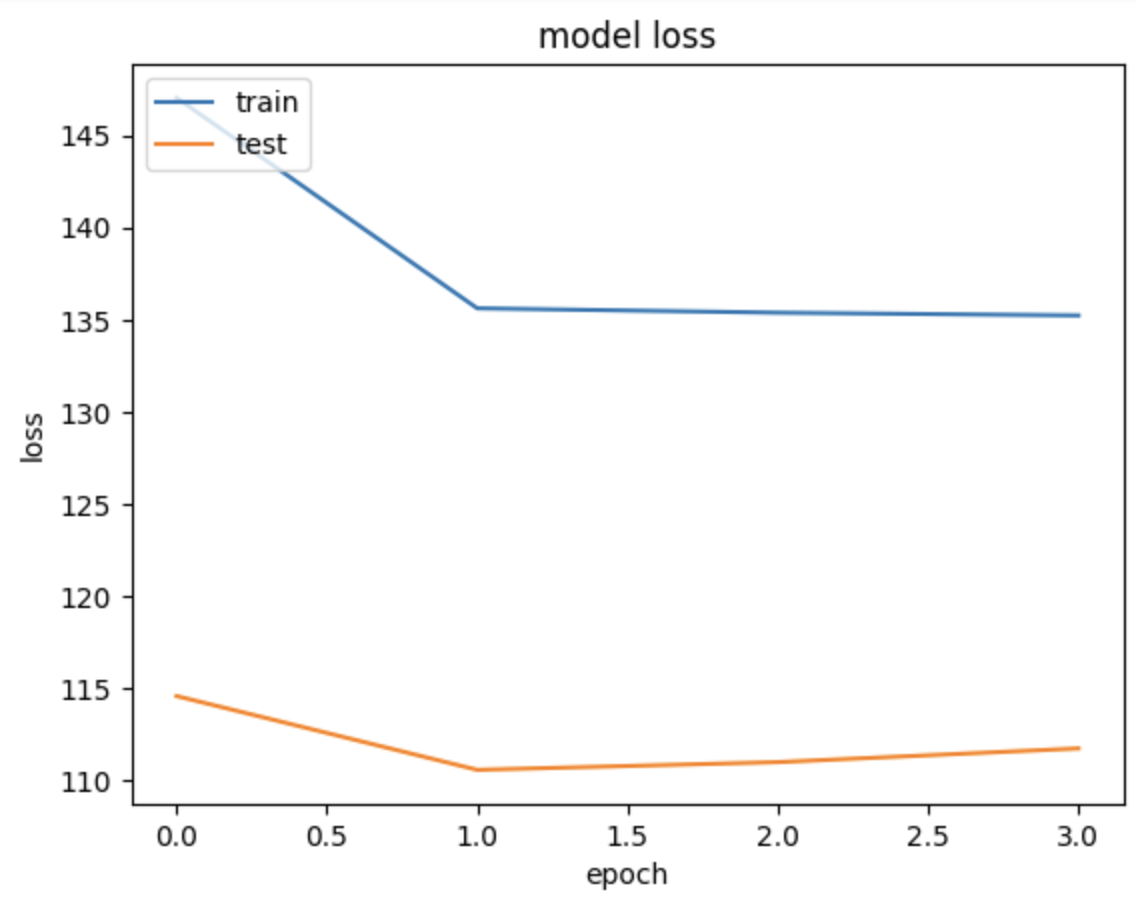


Рисунок 8. Потери модели.

Увеличим dropout до 0.4. Результаты представлены ниже.

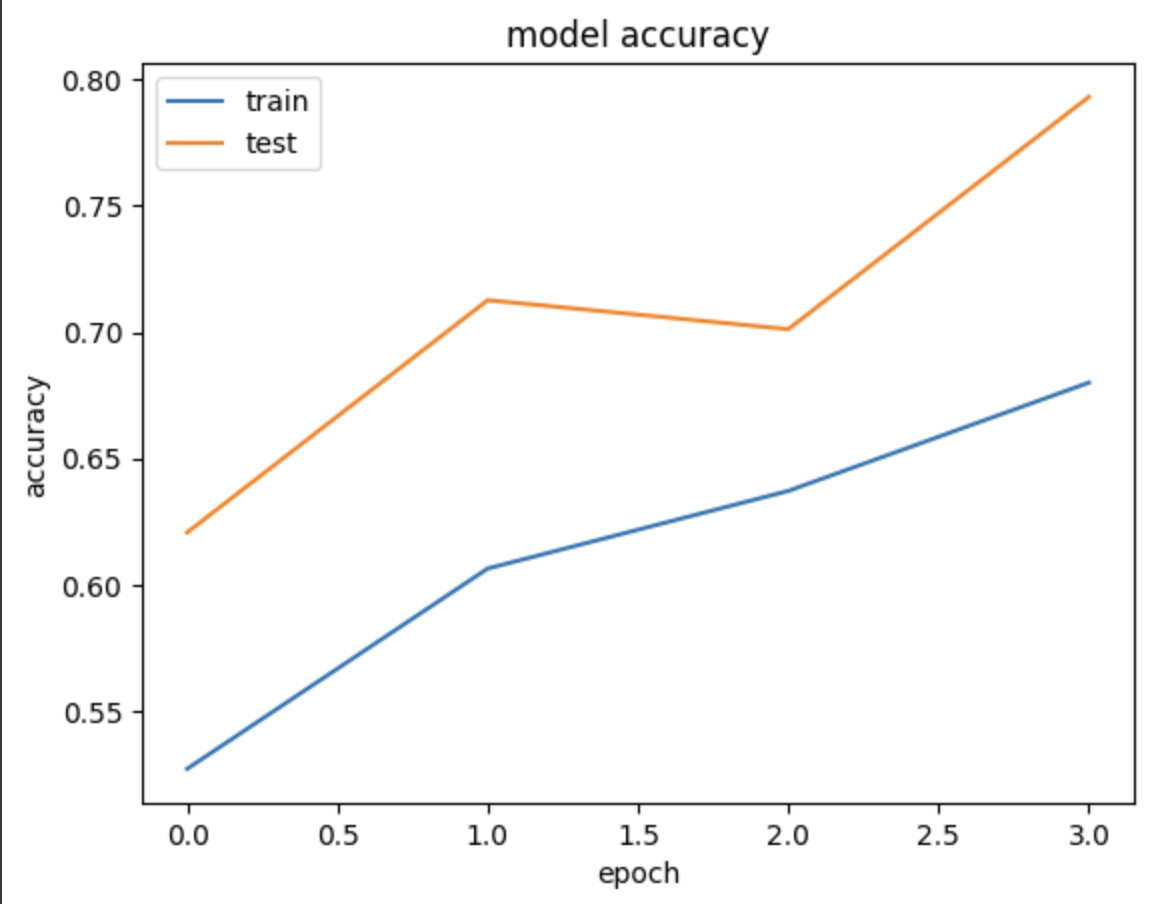


Рисунок 9. Точность модели.

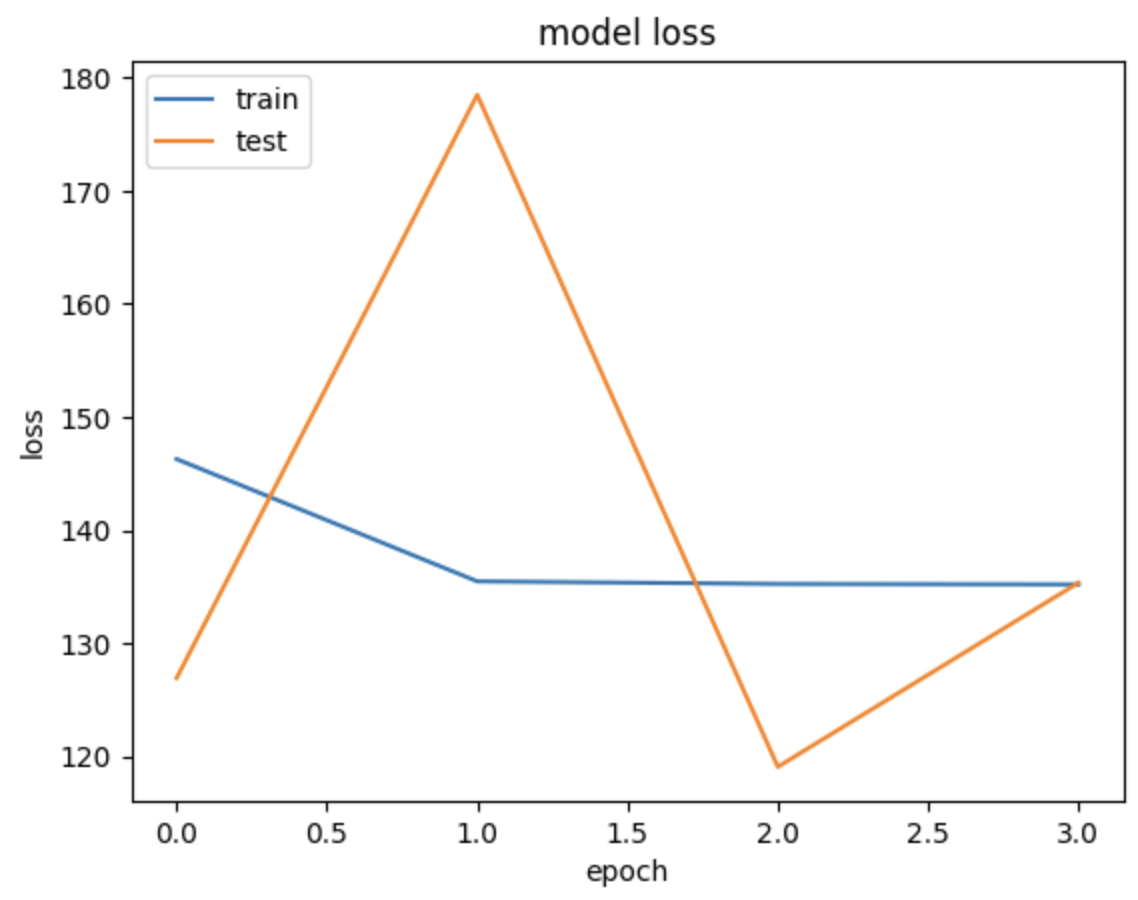


Рисунок 10. Потери модели.

Уменьшим dropout до 0.15

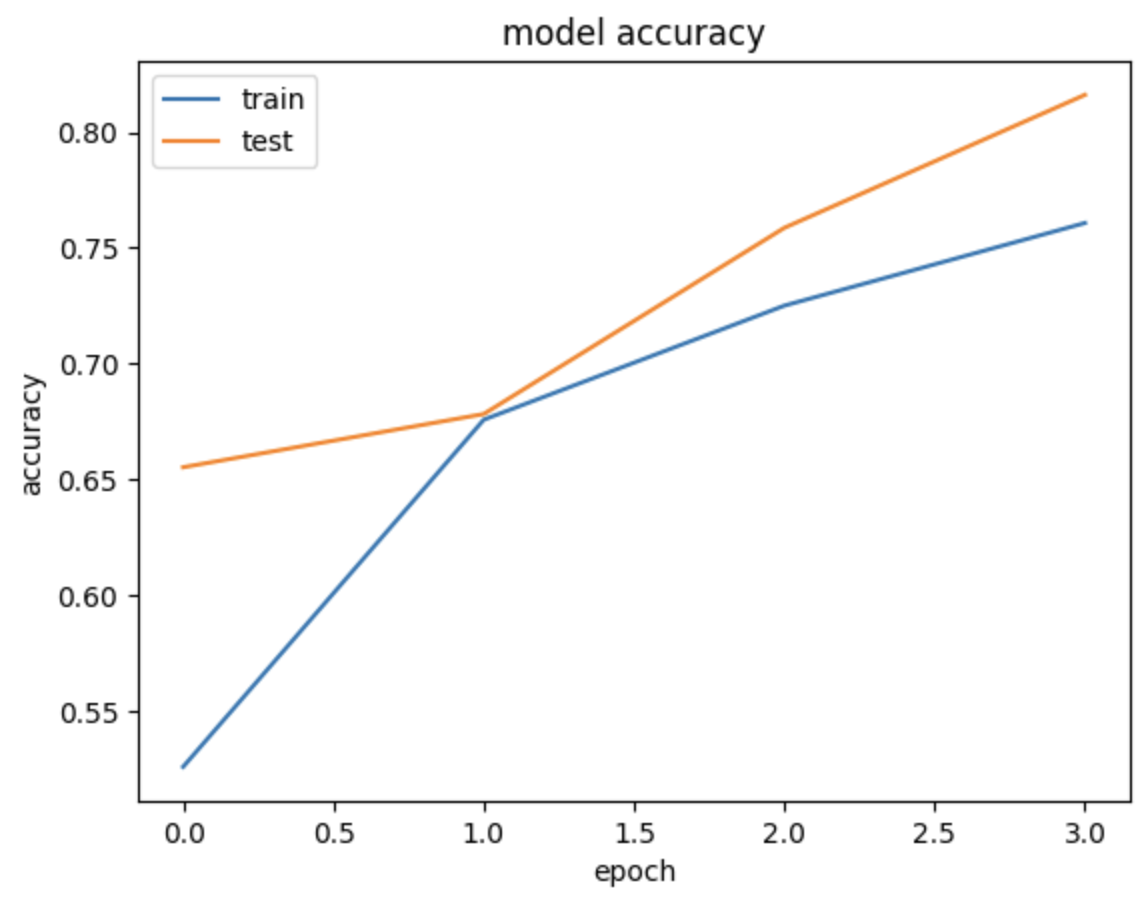


Рисунок 11. Точность модели.

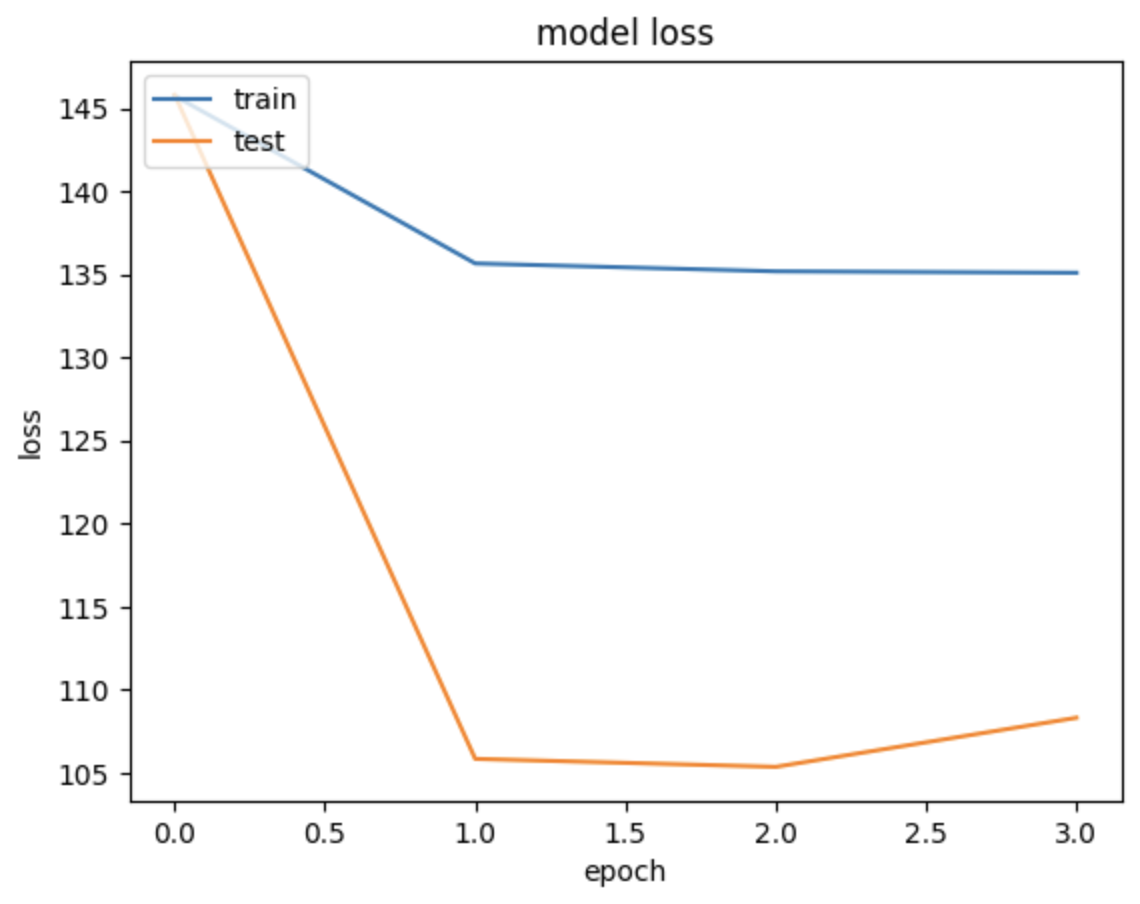


Рисунок 12. Потери модели.

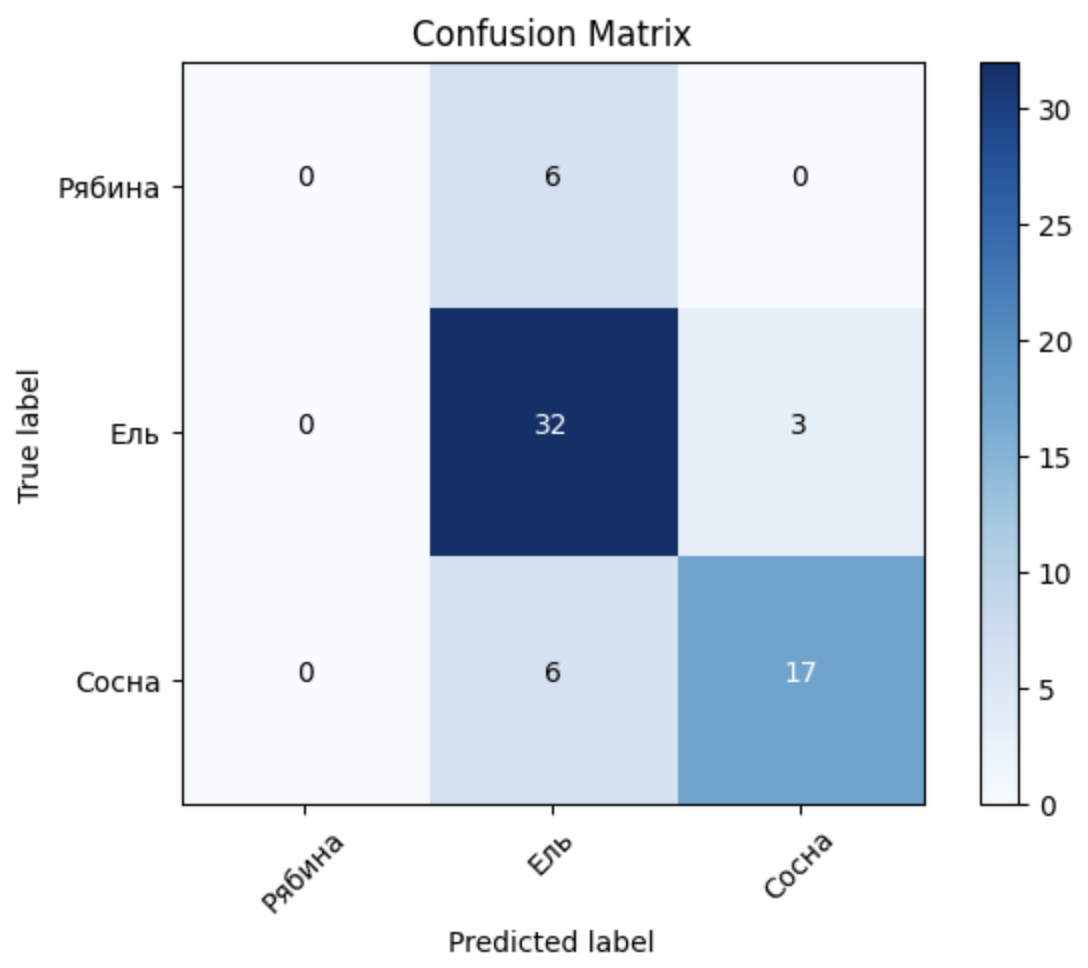


Рисунок 13. Матрица ошибок

Уменьшим скорость обучения в 2 раза и, соответственно, увеличим число эпох в 2 раза.

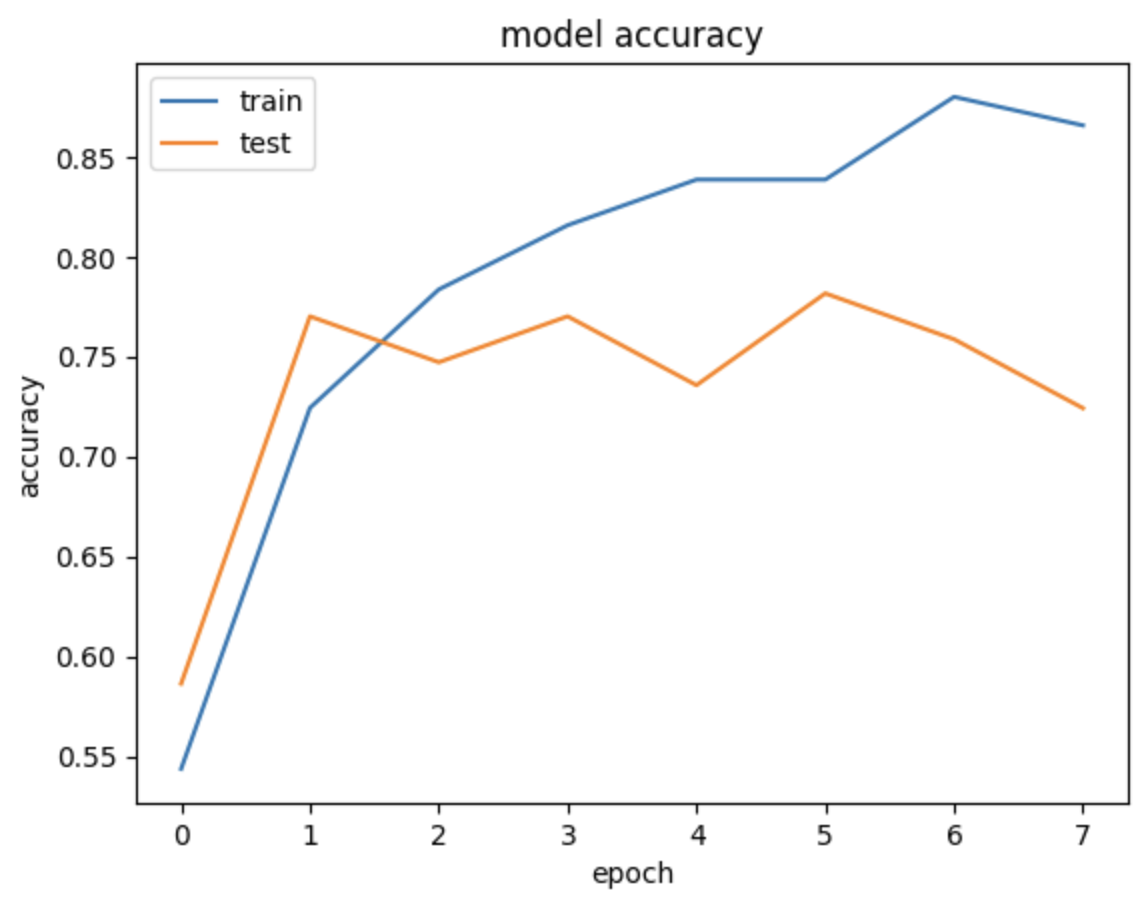


Рисунок 14. Точность модели.

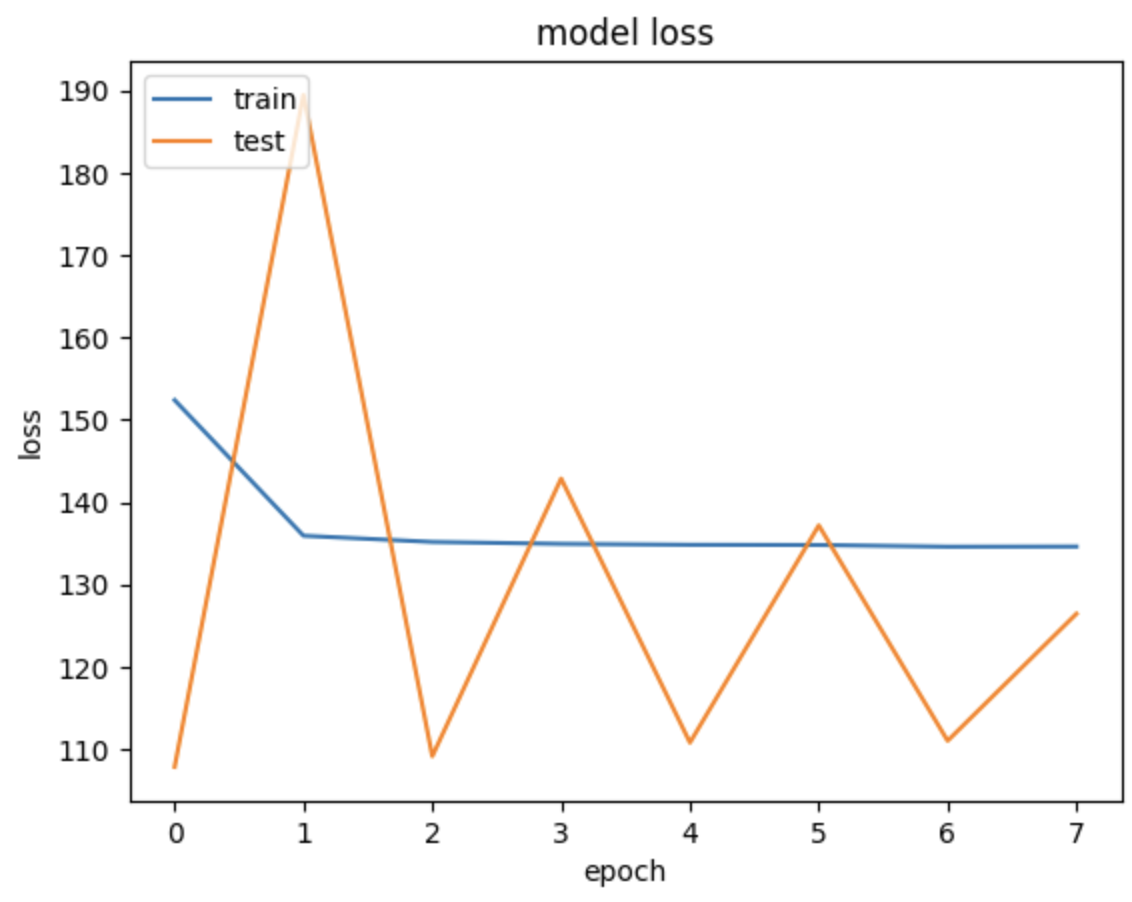


Рисунок 15. Потери модели.

Переобучение происходит на 3 эпохе, сократим число эпох до 3.

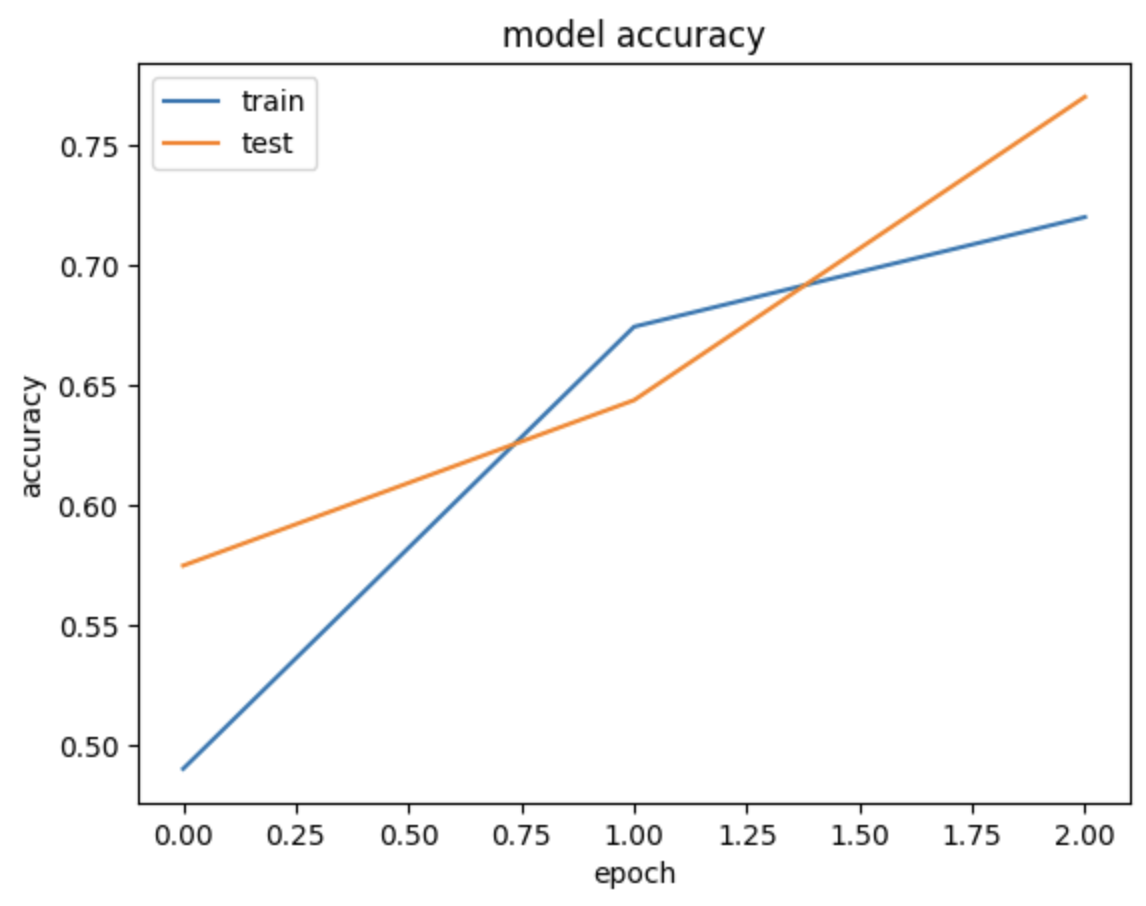


Рисунок 16. Точность модели.

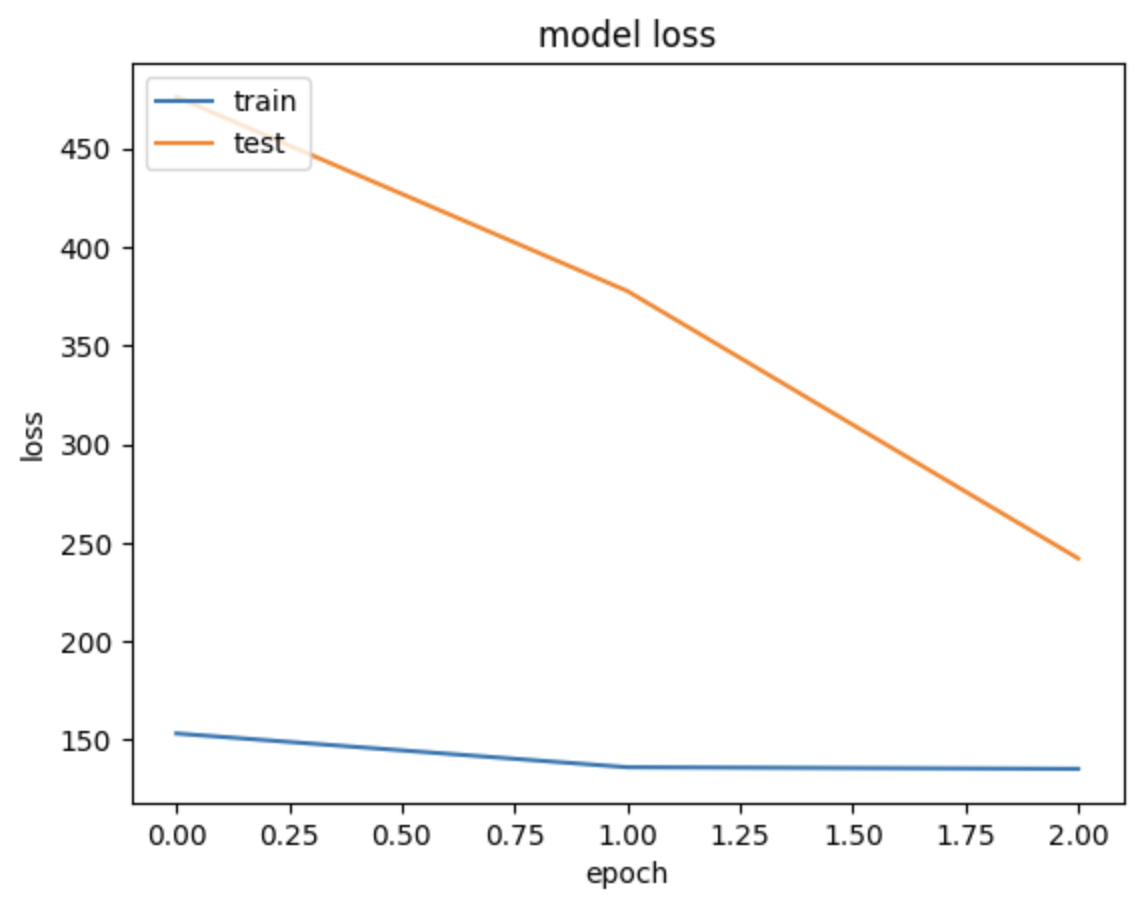


Рисунок 17. Потери модели.

Попробуем увеличить размер батча в 2 раза и, соответственно, число эпох.

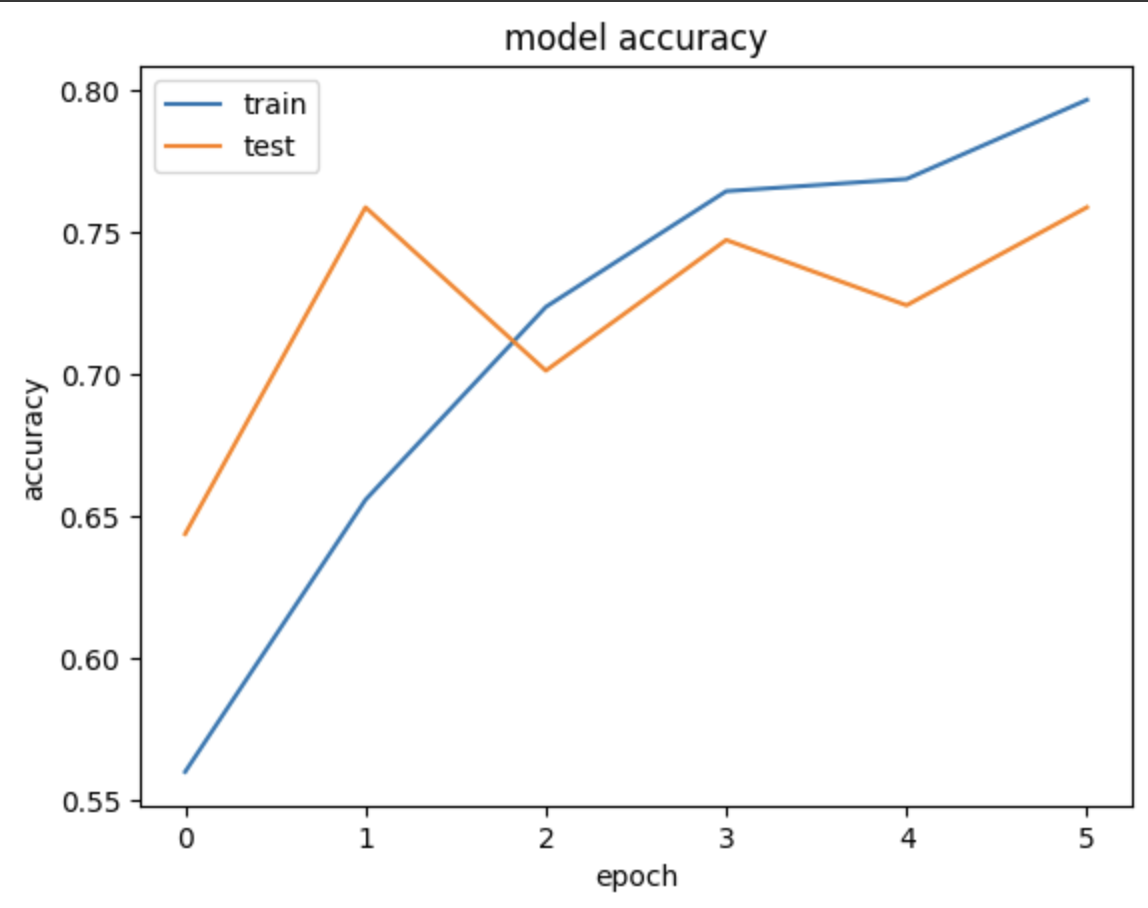


Рисунок 18. Точность модели.

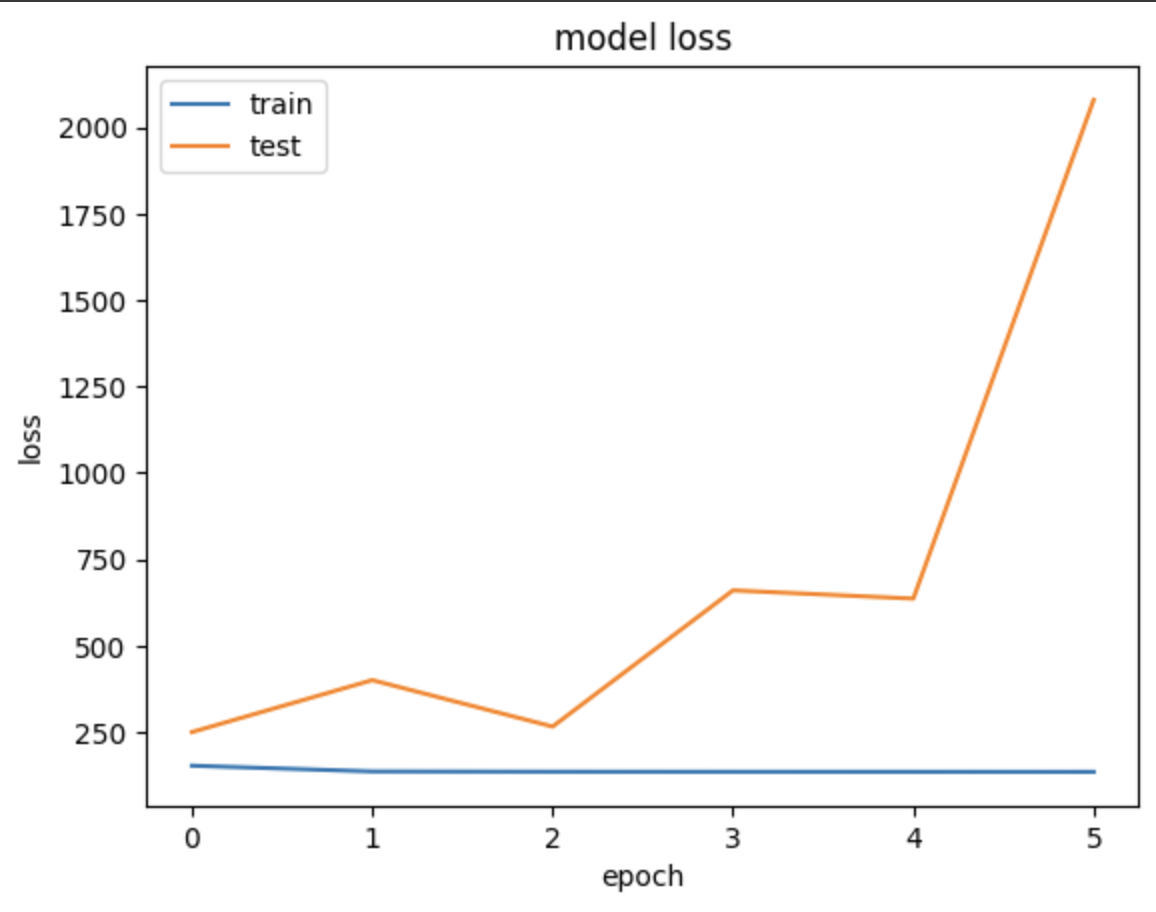


Рисунок 19. Потери модели.

# Таблица 1 - Итоговая таблица.

| **Конфигурация сети** | **Гиперпараметры** | **Точность** | **Комментарии** |
| --- | --- | --- | --- |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.3)(x)  dense\_bn(x, 128)  layers.Dropout(0.3)(x) | lr = 0.001,  batch\_size = 64,  epochs = 10  optimizator = SGD  augmentation=0.005 | test =79%,  train = 78% | Базовый вариант |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.3)(x)  dense\_bn(x, 128)  layers.Dropout(0.3)(x) | lr = 0.001,  batch\_size = 64,  epochs = 4  optimizator = SGD  augmentation=0.005 | test = 67%,  train = 68% | Переобучение произошло после 4 эпохи, уменьшили число эпох до 4 |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.4)(x)  dense\_bn(x, 128)  layers.Dropout(0.4)(x) | lr = 0.001,  batch\_size = 64,  epochs = 4  optimizator = SGD  augmentation=0.005 | test = 79%,  train = 71% | Увеличили значение dropout с 0.3 до 0.4. |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.15)(x)  dense\_bn(x, 128)  layers.Dropout(0.15)(x) | lr = 0.001,  batch\_size = 64,  epochs = 4  optimizator = SGD  augmentation=0.005 | test = 80%,  train = 74% | Уменьшили значение dropout с 0.3 до 0.15. |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.15)(x)  dense\_bn(x, 128)  layers.Dropout(0.15)(x) | lr = 0.0005,  batch\_size = 64,  epochs = 8  optimizator = SGD  augmentation=0.005 | test = 74%,  train = 85% | Уменьшили скорость обучения вдвое, вдвое увеличили число эпох |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.3)(x)  dense\_bn(x, 128)  layers.Dropout(0.3)(x) | lr = 0.0005,  batch\_size = 64,  epochs = 3  optimizator = SGD  augmentation=0.005 | test = 77%,  train = 72% | Переобучение произошло после 3 эпохи, уменьшили число эпох до 3 |
| x = tnet(inputs, 3)  conv\_bn(x, 32)  conv\_bn(x, 32)  tnet(x, 32)  conv\_bn(x, 32)  conv\_bn(x, 64)  conv\_bn(x, 512)  layers.GlobalMaxPooling1D(x)  dense\_bn(x, 256)  layers.Dropout(0.3)(x)  dense\_bn(x, 128)  layers.Dropout(0.3)(x) | lr = 0.0005,  batch\_size = 128,  epochs = 6  optimizator = SGD  augmentation=0.005 | test = 74%,  train = 81% | Увеличили размер батча в 2 раза, увеличили число эпох в 2 раза. |

**Вывод:**

В ходе лабораторной работы была обучена модель нейронной сети для классификации пород деревьев с использованием облаков точек. Проводились эксперименты с гиперпараметрами обучения для улучшения модели. Были изменены количество эпох, размер батча и скорость обучения, а также значения dropout. Наилучшие результаты получились при скорости обучения 0.001, числе эпох 4, размере батча 64 и значения dropout = 0.15.

В ходе выполнения лабораторной работы была изучена архитектура PointNet и возможности по её использованию для обучения модели на данных, полученных при помощи LiDAR.

В итоге, была получена модель нейронной сети, способная классифицировать породы деревьев с точностью около 70%. Данный результат не является хорошим, с точки зрения промышленного использования нейронной сети. Такая относительно низкая точность связана с тем, что сеть работает с трёхмерным пространством точек, к которому нельзя применить термин “упорядоченность”.

Таким образом, для решения задачи классификации пород деревьев на основе данных, полученных при помощи LiDAR, требуется либо изменение способа подачи данных на вход нейронной сети, либо разработка другой архитектуры нейронной сети.