|  |  |
| --- | --- |
| **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 №5**

**По курсу**

«**Разработка нейронных сетей**»  
**По теме «Разработка автоэнкодеров»**

Выполнил:  
Студент группы ИУ5-22МБибиков П.А.  
24.04.2025

Проверил:

**Канев А.И.**

2025 г.

# Задание

Для набора данных с помощью автоэнкодера получить эмбеддинг изображений и его визуализировать. Загрузить собственную аудиозапись и использовать автоэнкодер для удаления шума из аудиозаписи.

1. Проведисти обучение модели по варианту.
2. Изменить гиперпараметры обучения для понижения ошибки модели: количество эпох, размер батча, скорость обучения,
3. Изменить количество слоев/нейронов, чтобы уменьшить и увеличить размерность скрытого пространства. Сравнить полученные результаты
4. Обучить модель автоэнкодера для удаления шума из аудио.
5. Изменить гиперпараметры обучения для улучшения коэффициента детерминации: количество эпох, размер батча, скорость обучения
6. Изменить количество каналов, чтобы уменьшить и увеличить размерность скрытого пространства. Сравните полученные результаты
7. Указать, какие действия помогли улучшить метрики ваших моделей и объясните почему.

Для задания нужно сформировать свою подвыборку CIFAR100 по варианту. Вы должны использовать следующие классы из CIFAR100:

1. Номер группы + 15
2. Номер варианта + 56
3. ИУ5 (Номер варианта + 21); ГУИМЦ (80); Иностранцы (90)

Полученные варианты:

1. 22 + 15 = 37 (дома)
2. 3 + 56 = 59 (сосны)
3. 3 + 21 = 24 (тараканы)

# Часть 1. Получение эмбединга изображений.

По варианту возьмем классы – дома, сосны, тараканы. Прогоним исходную модель автоэнкодера. Посмотрим на полученные результаты, представленные на рисунках.

Изображение выглядит как График, линия, текст, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 - Графики для исходной модели автоэнкодера.

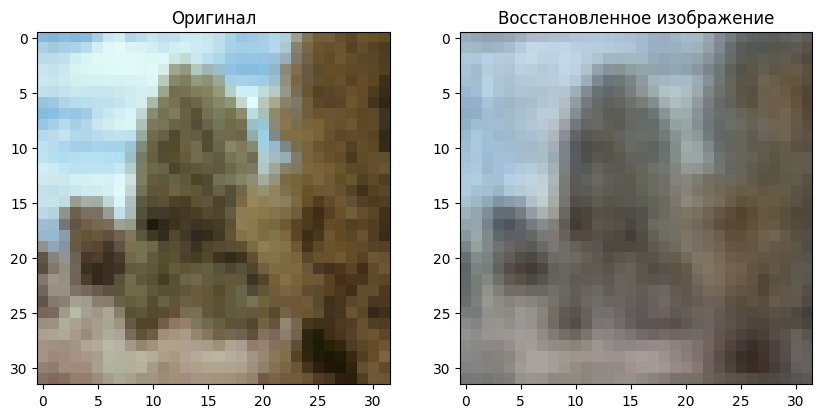


Рисунок 2 - Сравнение исходных изображений и автоэнкодера.

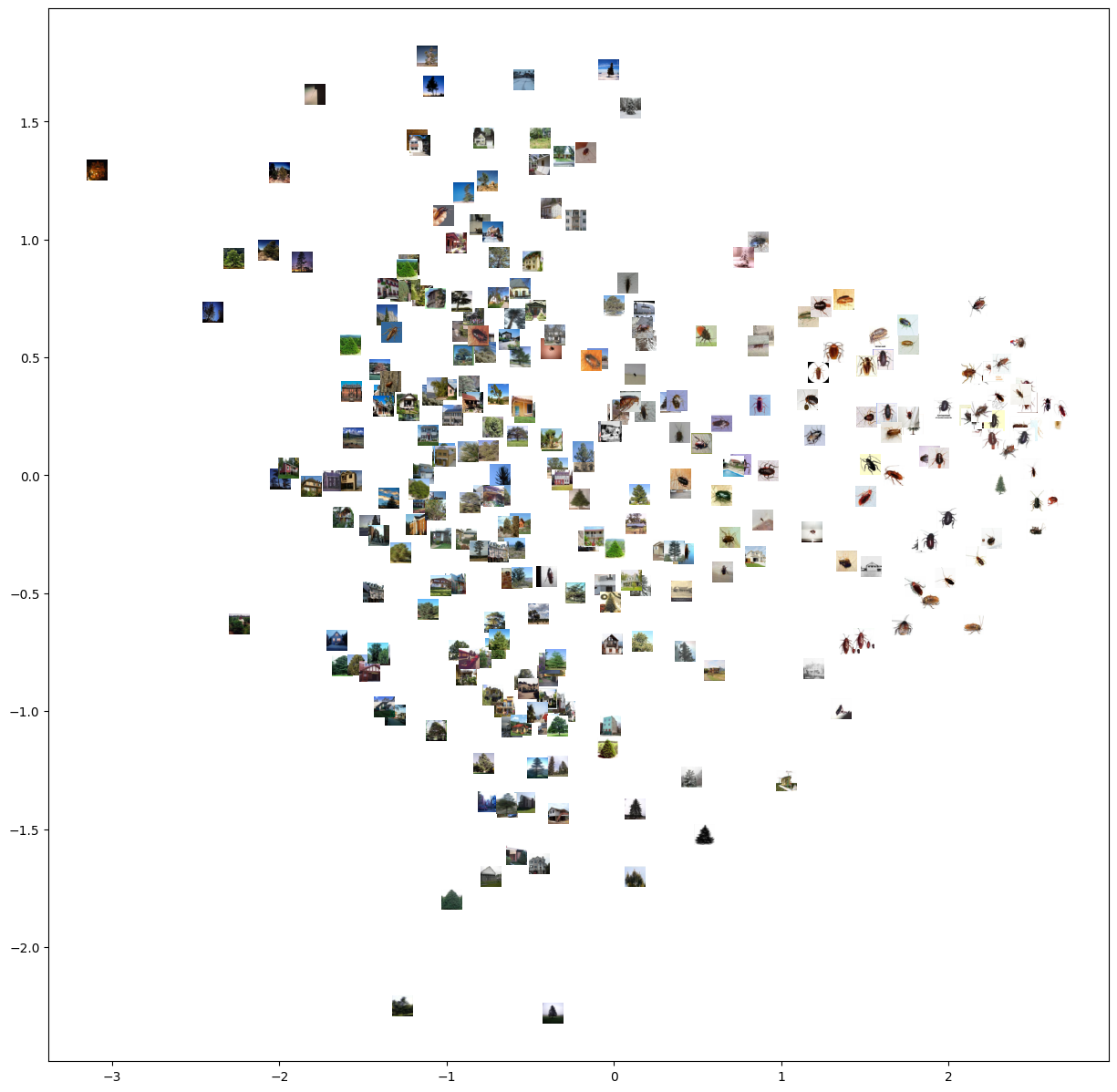


Рисунок 3 – Проекция эмбединга изображений.

Проведем настройку гиперпараметров и количества нейронов в скрытых слоях, отметим результаты в финальной таблице. Изображение выглядит как линия, График, диаграмма, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Лучший вариант.

# Часть 2. Работа с аудиозаписями.

Попробуем очистить музыку от шума. В качестве исходной композиции возьмём композицию T-Fest – Улети. В качестве шума возьмём звук ночи. Запустим на исходной конфигурации. Результаты представлены на рисунке 4.

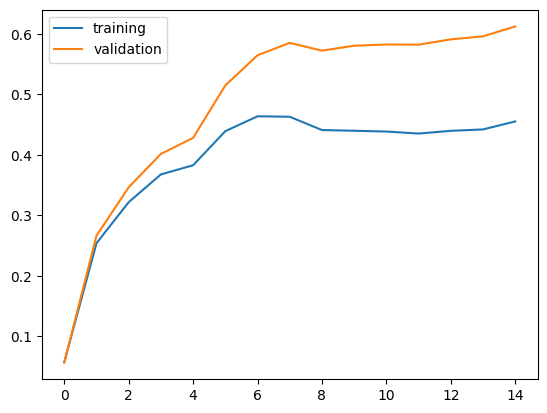


Рисунок 4 – Точность на исходной конфигурации.

Изображение выглядит как снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Спектрограмма оригинального аудио

Изображение выглядит как снимок экрана, линия, текст

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 – Спектрограмма аудио с шумом

Изображение выглядит как снимок экрана, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 – Спектрограмма очищенного аудио

loss: 0.01911

R2 score on train: 0.455

R2 score on val: 0.612

Таблица 1 - Результаты исходной модели очистки шума.

|  |  |
| --- | --- |
| **Описание** | **R2** |
| Исходный - Шум | 0.455 |
| Восстановленный - Шум | 0.612 |

Попробуем уменьшить количество нейронов вдвое во всех слоях. Результаты в таблице ниже.

Таблица 2 - Результаты -кол-во нейронов уменьшено вдвое.

|  |  |
| --- | --- |
| **Описание** | **R2** |
| Исходный - Шум | 0.435 |
| Восстановленный - Шум | 0.579 |

Таблица 3- Результаты - кол-во нейронов увеличено вдвое.

|  |  |
| --- | --- |
| **Описание** | **R2** |
| Исходный - Шум | 0.478 |
| Восстановленный - Шум | 0.652 |

# Итоговая таблица

| **Конфигурация сети** | **Гиперпараметры** | **Точность** | **Комментарии** |
| --- | --- | --- | --- |
| Linear(in\_features=3072, out\_features=512, bias=True) ELU(alpha=1.0) Linear(in\_features=512, out\_features=256, bias=True) ELU(alpha=1.0) Linear(in\_features=256, out\_features=64, bias=True) Tanh() | lr = 0.001,  batch\_size = 128,  epochs = 200 | test = 87.77%,  train = 88.17% | Базовый вариант |
| Linear(in\_features=3072, out\_features=512, bias=True) ELU(alpha=1.0) Linear(in\_features=512, out\_features=256, bias=True) ELU(alpha=1.0) Linear(in\_features=256, out\_features=64, bias=True) Tanh() | lr = 0.001, batch\_size = 256, epochs = 400 | test = 85.01%,  train = 85.30% | Изменили размер батча в 2 раза, увеличили число эпох в 2 раза |
| Linear(in\_features=3072, out\_features=512, bias=True) ELU(alpha=1.0) Linear(in\_features=512, out\_features=256, bias=True) ELU(alpha=1.0) Linear(in\_features=256, out\_features=64, bias=True) Tanh() | lr = 0.0005, batch\_size = 128, epochs = 400 | test = 87.72%,  train = 87.88% | Уменьшили скорость обучения, увеличили число эпох |
| Linear(in\_features=3072, out\_features=256, bias=True) ELU(alpha=1.0) Linear(in\_features=256, out\_features=128, bias=True) ELU(alpha=1.0) Linear(in\_features=128, out\_features=32, bias=True) Tanh() |  | test = 86,43%,  train = 86,79% | Уменьшили параметр hidden\_size в 2 раза |
| Linear(in\_features=3072, out\_features=1024, bias=True)  ELU(alpha=1.0)  Linear(in\_features=1024, out\_features=512, bias=True)  ELU(alpha=1.0)  Linear(in\_features=512, out\_features=256, bias=True)  Tanh() |  | test = 91.45%,  train = 92.32% | Увеличили параметр hidden\_size в 2 раза |
| **Часть 2** | | | |
| Conv1d(2, 256, kernel\_size=3, stride=2, padding=1) Mish()  Conv1d(256, 512, kernel\_size=3, stride=2, padding=1)  Mish()  Conv1d(512, 1024, kernel\_size=3, stride=2, padding=1)  Mish() | lr = 0.0001,  epoch = 15 | r2\_score = 89,4% | Исходная модель |
| Conv1d(2, 128, kernel\_size=3, stride=2, padding=1) Mish()  Conv1d(128, 256, kernel\_size=3, stride=2, padding=1)  Mish()  Conv1d(256, 512, kernel\_size=3, stride=2, padding=1)  Mish() | lr = 0.0001,  epoch = 15 | r2\_score = 88,2% | Уменьшили число нейронов в 2 раза |
| Conv1d(2, 512, kernel\_size=3, stride=2, padding=1) Mish()  Conv1d(512, 1024, kernel\_size=3, stride=2, padding=1)  Mish()  Conv1d(1024, 2048, kernel\_size=3, stride=2, padding=1)  Mish() | lr = 0.0001,  epoch = 15 | r2\_score = 90,1% | Увеличили число нейронов в 2 раза |

## Вывод

В ходе лабораторной работы мы исследовали применение автоэнкодеров для обработки изображений и аудиозаписей. В процессе мы изменяли количество нейронов в конечном слое энкодера и декодера соответственно для сравнения, полученных результатов при помощи r2 score.

Наилучший вариант получился при большом количестве нейронов в конечном слое энкодера.

Также, использование метода PCA при отображении на двумерную плоскость, помогло проиллюстрировать структуру данных и выделить наглядно определенные кластеры, что может быть полезно для дальнейшего исследования и анализа данных.

Применение сверточного автоэнкодера для удаления шума из аудиозаписей продемонстрировало его способность эффективно фильтровать ненужные звуки. Сравнение результатов r2-score показывает, что очищенные данные довольно близки к исходным аудиозаписям.