ПРАКТИЧЕСКАЯ РАБОТА №5: АВТОКОДИРОВЩИКИ И РЕКУРРЕНТНЫЕ НЕЙРОННЫЕ СЕТИ

Цель

Освоить принципы работы автокодировщиков для сжатия и восстановления данных, а также изучить применение рекуррентных нейронных сетей (RNN, LSTM, GRU) для обработки временных последовательностей и текстов.

Задание

Часть 1: Автокодировщик (1 пара)

1. Подготовка данных:

• Загрузите датасет изображений (например, MNIST или Fashion MNIST).

• Выполните нормализацию данных.

2. Создание автокодировщика:

• Постройте сеть с симметричной архитектурой:

• Входной слой, сжимающий изображение до вектора.

• Скрытые слои, выполняющие кодирование данных.

• Декодирующие слои, восстанавливающие изображение из вектора.

3. Обучение автокодировщика:

• Обучите модель на обучающей выборке.

• Оцените качество восстановления изображений на тестовой выборке.

4. Визуализация:

• Постройте графики исходных и восстановленных изображений.

• Визуализируйте скрытые представления данных.

5. Эксперимент:

• Изучите влияние уменьшения размера скрытого представления (вектора) на качество восстановления.

Часть 2: Рекуррентные нейронные сети (1 пара)

1. Подготовка данных:

• Используйте набор последовательных данных, например, текстовый датасет (предсказание следующего слова) или временной ряд (предсказание следующего значения).

2. Реализация рекуррентной сети:

• Постройте базовую рекуррентную сеть с одним рекуррентным слоем (RNN).

• Обучите её на предоставленных данных.

3. Расширение модели:

• Замените базовую RNN на LSTM и GRU.

• Сравните их поведение и точность на тестовой выборке.

4. Визуализация:

• Графики ошибки и точности для каждой архитектуры (RNN, LSTM, GRU).

• Визуализация предсказаний на временных рядах или текстах.

Что нужно вставить в отчет

1. Введение:

• Описание целей работы.

• Краткое описание автокодировщиков и их применения.

• Описание рекуррентных сетей и их архитектурных вариантов (RNN, LSTM, GRU).

2. Автокодировщик:

• Схема архитектуры сети (размеры слоев, активации).

• Визуализация: исходные и восстановленные изображения.

• Влияние размера скрытого представления на качество восстановления (графики или таблица).

3. Рекуррентные нейронные сети:

• Описание архитектур RNN, LSTM и GRU.

• Графики потерь и точности на обучении и тестировании.

• Таблица с основными метриками для каждого типа сети.

• Визуализация предсказаний на временных рядах или текстах.

4. Сравнительный анализ:

• Влияние архитектурных решений (размер скрытых слоев, использование LSTM/GRU) на качество предсказаний.

5. Заключение:

• Основные выводы о применении автокодировщиков и рекуррентных сетей.

• Рекомендации по выбору архитектуры в зависимости от задачи.

• Описание проблем, с которыми столкнулись студенты, и их решения.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ ПО 5 ПРАКТИЧЕСКОЙ

Часть 1. Автокодировщик

Автокодировщик — это тип нейронной сети, который обучается воспроизводить входные данные на выходе. Он состоит из двух частей:

- Кодер: преобразует входные данные в компактное скрытое представление (вектор).
- Декодер: восстанавливает данные из этого представления.

Цель обучения — минимизировать разницу между входом и выходом. Сеть вынуждена сохранять важную информацию в меньшем количестве признаков.

Этап 1. Подготовка данных

Импортировать необходимые библиотеки:

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
```

Загрузить и нормализовать датасет:

```
(x_train, _), (x_test, _) = mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
x_train = x_train.reshape((len(x_train), 784))  # 28*28
x_test = x_test.reshape((len(x_test), 784))
```

Этап 2. Создание модели автокодировщика

Установить размерность скрытого слоя:

```
encoding_dim = 32  # размер скрытого вектора
```

Построить модель:

```
input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(784, activation='sigmoid')(encoded)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam',
loss='binary_crossentropy')
```

Этап 3. Обучение модели

```
autoencoder.fit(x_train, x_train,
                epochs=20,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

Этап 4. Визуализация результатов

Получить восстановленные изображения:

```
decoded_imgs = autoencoder.predict(x_test)
```

Построить график:

```
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Оригинальные изображения
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.axis('off')

    # Восстановленные изображения
    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(decoded_imgs[i].reshape(28, 28))
    plt.axis('off')
plt.show()
```

Этап 5. Эксперимент с размерностью скрытого слоя

Измените значение encoding_dim на 64, 16, 8 и повторите шаги 2–4. Оцените, как размерность скрытого представления влияет на качество восстановления.

Часть 2. Рекуррентные нейронные сети

Рекуррентные нейронные сети (RNN) предназначены для работы с последовательностями данных. Они используют внутреннюю память, чтобы учитывать контекст предыдущих элементов.

Основные разновидности:

RNN — базовая архитектура, плохо справляется с долгосрочными зависимостями.

LSTM — более устойчивая к затухающим градиентам, хорошо работает на длинных последовательностях.

GRU — упрощённая версия LSTM, работает быстрее при схожем качестве.

Этап 1. Подготовка текста

Загрузить текст и привести к нижнему регистру:

```
text = open("input.txt").read().lower()
```

Сформировать выборки:

```
seq_length = 40
step = 3
sentences = []
next_chars = []

for i in range(0, len(text) - seq_length, step):
    sentences.append(text[i: i + seq_length])
```

```
    next_chars.append(text[i + seq_length])
```

Закодировать данные:

```
chars = sorted(list(set(text)))
char_indices = dict((c, i) for i, c in enumerate(chars))
indices_char = dict((i, c) for i, c in enumerate(chars))

X = np.zeros((len(sentences), seq_length, len(chars)),
dtype=bool)
y = np.zeros((len(sentences), len(chars)), dtype=bool)

for i, sentence in enumerate(sentences):
    for t, char in enumerate(sentence):
        X[i, t, char_indices[char]] = 1
    y[i, char_indices[next_chars[i]]] = 1
```

Этап 2. Базовая RNN

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, Dense

model_rnn = Sequential()
model_rnn.add(SimpleRNN(128, input_shape=(seq_length,
len(chars))))
model_rnn.add(Dense(len(chars), activation='softmax'))

model_rnn.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
history_rnn = model_rnn.fit(X, y, batch_size=128, epochs=10,
validation_split=0.2)
```

Этап 3. Модификации: LSTM и GRU

Модель LSTM:

```
from tensorflow.keras.layers import LSTM

model_lstm = Sequential()
model_lstm.add(LSTM(128, input_shape=(seq_length, len(chars))))
model_lstm.add(Dense(len(chars), activation='softmax'))
model_lstm.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
history_lstm = model_lstm.fit(X, y, batch_size=128, epochs=10,
validation_split=0.2)
```

Модель GRU:

```
from tensorflow.keras.layers import GRU

model_gru = Sequential()
model_gru.add(GRU(128, input_shape=(seq_length, len(chars))))
model_gru.add(Dense(len(chars), activation='softmax'))
model_gru.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
history_gru = model_gru.fit(X, y, batch_size=128, epochs=10,
validation_split=0.2)
```

Этап 4. Визуализация результатов

Построить графики потерь:

```
import matplotlib.pyplot as plt

plt.plot(history_rnn.history['loss'], label='RNN')
plt.plot(history_lstm.history['loss'], label='LSTM')
plt.plot(history_gru.history['loss'], label='GRU')
plt.title("График потерь на обучении")
plt.xlabel("Эпоха")
plt.ylabel("Потери")
plt.legend()
plt.show()
```

Аналогично можно построить графики точности ('accuracy').

ПРАКТИЧЕСКАЯ РАБОТА №6: ОПИСАНИЕ РЕЗУЛЬТАТОВ ДЛЯ ВКР И ШАГИ РЕАЛИЗАЦИИ ПРОЕКТОВ НА ОСНОВЕ НЕЙРОННЫХ СЕТЕЙ

Цель

Научиться систематизировать результаты работы нейронных сетей, правильно оформлять графики, таблицы и выводы для ВКР. Освоить базовые шаги при реализации проектов на основе нейронных сетей и подготовке научных отчетов.

Задание

Часть 1: Описание результатов обучения нейронной сети (1 пара)

1. Выберите один из завершенных проектов из предыдущих практических занятий (например, полносвязная сеть, сверточная сеть или автокодировщик).

2. Подготовьте следующие графики:

• Потери (loss) на обучающей и тестовой выборках.

• Точность классификации или метрики качества (точность, F1-мера) на тестовой выборке.

• Визуализация ошибок классификации или восстановления данных.

3. Оформите результаты в виде краткого отчета с основными выводами о качестве модели, возможных проблемах и их решениях.

Часть 2: Шаги реализации проекта на основе нейронных сетей (1 пара)

1. Опишите процесс реализации проекта в формате шагов:

• Постановка задачи: краткое описание задачи, набор данных, цель.

• Предварительная обработка данных: нормализация, преобразования.

• Архитектура модели: выбор слоев, функций активации, оптимизаторов.

• Обучение модели: настройка гиперпараметров, визуализация процесса обучения.

• Оценка модели: метрики, анализ ошибок.

2. Укажите, как данные шаги могут быть применены в рамках выпускной работы.

3. Подготовьте сводную таблицу с параметрами, метриками и рекомендациями по улучшению модели.

Часть 3: Итоговая защита и обсуждение (1 пара)

1. Представьте подготовленный отчет на защите.

2. Обсудите:

• Какую роль играет предварительная обработка данных?

• Какие улучшения можно внести в архитектуру модели?

• Какие проблемы возникли в процессе выполнения проекта, и как они были решены?

Что нужно вставить в отчет:

1. Введение:

• Цели работы и задачи исследования.

• Описание используемого датасета и задачи (например, классификация, восстановление изображений).

2. Описание проекта:

• Постановка задачи, выбранные данные и их предварительная обработка.

• Архитектура нейронной сети (слои, активации, параметры).

• Описание экспериментов: количество эпох, learning rate и другие гиперпараметры.

3. Результаты:

• Графики потерь и точности.

• Таблица с основными метриками качества (точность, F1-мера, recall).

• Визуализация ошибок классификации или восстановления изображений.

4. Сводная таблица параметров:

• Гиперпараметры сети, количество слоев, функции активации и методы оптимизации.

• Влияние параметров на итоговые результаты.

5. Заключение:

• Основные выводы о работе модели.

• Рекомендации по улучшению результатов.

• Влияние параметров сети и методов обработки данных на итоговую метрику.

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ ПО 6 ПРАКТИЧЕСКОЙ

Часть 1. Описание результатов обучения нейронной сети

Результаты работы нейронной сети следует оформлять так, чтобы они наглядно отражали процесс обучения, качество модели и возникшие проблемы. Наиболее важными элементами являются:

- Графики потерь — отображают динамику ошибки на обучении и валидации.
- Метрики качества — такие как точность, F1-мера, полнота.
- Визуализация ошибок — примеры неправильных предсказаний, чтобы понять слабые стороны модели.

Этап 1. Выбор завершённого проекта

Выберите один из ранее выполненных проектов:

- Полносвязная нейронная сеть (например, для классификации изображений).
- Сверточная сеть (например, CNN на MNIST/FashionMNIST).
- Автокодировщик.

Этап 2. Построение графиков и анализ

Построение графиков потерь и точности:

```
plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
```

```
plt.ylabel('Loss')
plt.title('График потерь')
plt.legend()
plt.grid()
plt.show()

plt.plot(history.history['accuracy'], label='train_acc')
plt.plot(history.history['val_accuracy'], label='val_acc')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('График точности')
plt.legend()
plt.grid()
plt.show()
```

Вычисление дополнительных метрик:

```
from sklearn.metrics import classification_report

y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)

print(classification_report(y_test, y_pred_classes))
```

Визуализация ошибок классификации:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d')
plt.title('Матрица ошибок')
plt.xlabel('Предсказанный класс')
plt.ylabel('Истинный класс')
plt.show()
```

Для автокодировщиков можно визуализировать восстановленные изображения и их отличия от оригиналов.

Этап 3. Оформление краткого отчета

Укажите краткие выводы:

- Общее качество модели.
- Наличие переобучения или недообучения.
- Сильные и слабые стороны модели.

Приведите примеры некорректных предсказаний и объясните возможные причины.

Часть 2. Шаги реализации проекта на основе нейронных сетей

Проект на основе нейронной сети включает несколько обязательных этапов, каждый из которых должен быть описан в выпускной работе. Последовательность реализации проекта стандартна и должна быть представлена структурированно.

Этапы реализации проекта

1. Постановка задачи

- Определение цели (например: классификация изображений, прогнозирование временных рядов).
- Выбор набора данных (например, MNIST, CIFAR-10, текстовый корпус).
- Формулировка ожидаемого результата.

2. Предварительная обработка данных

- Очистка данных, удаление пропусков.
- Масштабирование и нормализация признаков.
- Кодирование категориальных переменных, если необходимо.

3. Архитектура модели

- Выбор типа модели (полносвязная, сверточная, RNN, автокодировщик).
- Выбор функций активации (ReLU, sigmoid и др.).
- Оптимизатор (Adam, SGD).
- Структура сети: число слоёв, размерности, наличие dropout или batch normalization.

4. Обучение модели

- Задание числа эпох, размера батча, learning rate.
- Контроль переобучения (валидация, регуляризация).
- Визуализация динамики обучения.

5. Оценка модели

- Расчёт метрик: точность, F1-мера, recall, precision.
- Анализ ошибок.
- Визуализация результатов: графики, примеры неверных предсказаний.

Применение в ВКР

Данные шаги полностью соответствуют требованиям к описанию экспериментальной части выпускной квалификационной работы. Они позволяют показать:

- Четкую логику построения модели.
- Осознанный выбор архитектурных и технических решений.
- Умение анализировать полученные результаты.

Часть 3. Итоговая защита и обсуждение

1. Презентация отчета

- Представьте оформленные графики, таблицы и выводы.
- Укажите сильные и слабые стороны модели.
- Подчеркните, какие шаги улучшили результат.

2. Ответы на обсуждение

- Предварительная обработка: влияет на стабильность обучения, скорость сходимости и итоговую метрику.
- Улучшения архитектуры: добавление слоёв, использование

современных блоков (например, residual connections), увеличение обучающей выборки.

- Проблемы и решения:
- Перенасыщение модели — решается через dropout или регуляризацию.
- Нестабильная сходимость — корректируется learning rate.
- Недостаток данных — решается аугментацией или генерацией.