УДК 004.934

СИСТЕМА ВИЗУАЛЬНОГО РАСПОЗНАВАНИЯ РЕЧИ НА ОСНОВЕ НЕЙРОННОЙ СЕТИ ТРАНСФОРМЕРНОГО ТИПА

Зубрицкая Е.И., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Вашкевич М.И. – д-р. техн. наук, профессор

Аннотация. В работе представлена разработка архитектуры нейронной сети на базе трансформера для задачи визуального распознавания речи. Был собран собственный набор данных, а также выполнена его предобработка, составлена архитектура сверточной сети для извлечения визуальных признаков. При проектировании использовались детектор ключевых точек лица на базе MediaPipe, блоки кодировщика для анализа визуальных признаков и создания их контекстного представления и декодировщика для пошаговой генерации текста на основе выходов кодировщика. В работе показаны примеры полученных данных и архитектур сети.

Ключевые слова. Трансформер, сверточная нейронная сеть, глубокое обучение, компьютерное зрение.

Введение. Визуальное распознавание речи — перспективная, но сложная задача, востребованная в системах помощи людям и в шумных средах [1]. Основными препятствиями являются низкое качество видео и непостоянное положение лица в кадре, из-за чего существующие подходы часто ограничены распознаванием отдельных слов, а не связных предложений.

Целью данной работы является разработка системы распознавания речи на основе трансформерной архитектуры, способной работать с целыми предложениями. Задачи включают выбор архитектуры, формирование набора данных для обучения и тестирования, а также экспериментальную оценку качества и анализ полученных результатов.

Предобработка данных. Исходными данными выступают массив передач BBC с Риз Латиф в качестве ведущей, а также соответствующие текстовые расшифровки, где каждой фразе соответствует временная метка её появления в видео. Задача предобработки состоит в отборе необходимого материала и преобразовании данных в формат, пригодный для обучения сети.

Для устранения избыточной информации выполнена фильтрация видеоматериалов на основе предварительной временной разметки. В результате сформированы компактные видео, содержащие только релевантные фрагменты, и синхронизированные с ними файлы расшифровок с обновленными таймкодами.

Подготовка данных включает сегментацию видео на фрагменты, соответствующие фразам, и извлечение из кадров области рта говорящего. Текстовые фразы преобразовываются в последовательности токенов. Токен – базовая единица текста, представленная числовым кодом, с использованием предобученного BPE-токенизатора от GPT-2. Для оптимизации его исходный словарь урезается так, чтобы содержать только токены, встречающиеся в текстах проекта, что значительно сокращает вычислительные затраты. Итоговые данные, включающие последовательности изображений и их токенизированные фразы, сохраняются в формате PKL. Пример данных представлен на рисунке 1.

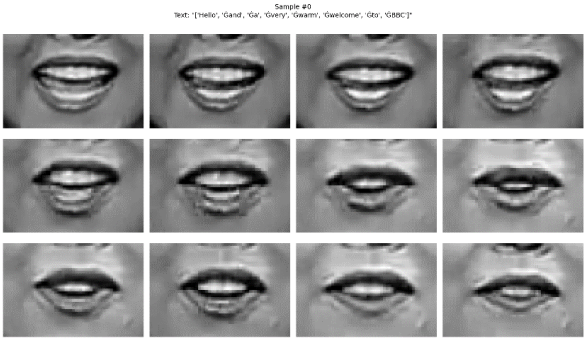

Рисунок 1 – Пример данных

Архитектура кодировщика. Трансформер — это нейросетевая модель, призванная более эффективно решать задачи seq2seq по преобразованию одной последовательности в другую. Архитектура состоит из двух блоков: кодировщика (encoder), представленного на рисунке 2 слева, и декодировщика (decoder), представленного справа. Кодировщик и декодировщик выделены прямоугольниками.

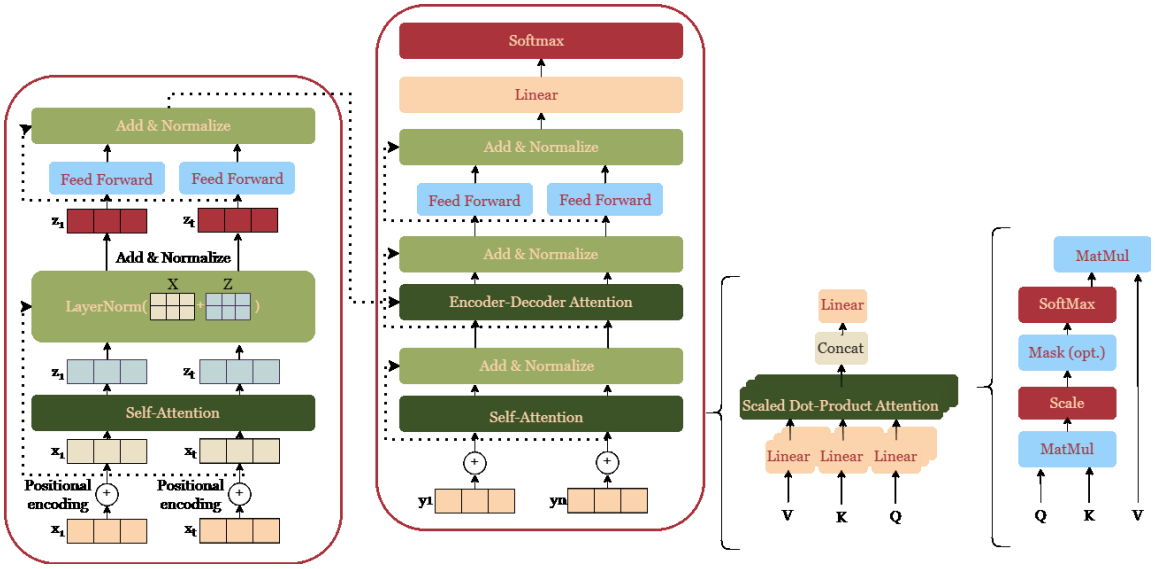

Рисунок 2 – Архитектура сети

Кодировщик модели Transformer принимает на вход T эмбеддингов элементов входной последовательности и выдаёт T уточнённых эмбеддингов для каждого элемента с учётом его контекста с которыми впоследствии будет работать декодировщик модели, состоящий из отдельных идентичных блоков. Кодировщик также состоит из N последовательных применений идентичных блоков, каждый - со своими параметрами [2].

Для получения входных эмбедингов каждый кадр пропускается через свёрточную нейронную сеть с трехмерной сверткой, которая извлекает из изображений компактные числовые признаки. На выходе свёрточной сети получается T векторов, каждый размерностью D. После чего первый блок кодировщика прибавляет к ним эмбеддинги позиции кадра в последовательности, затем выдаёт такое же количество D-мерных выходных эмбеддингов, которые учитывают контекст всей входной последовательности.

Позиционное кодирование заключается в том, что каждой позиции сопоставляется свой эмбеддинг. Вектор позиционного кодирования e^p для токена на позиции $p = 1,2,\dots T$ заполняется синусами от этой позиции на чётных элементах вектора и косинусами от позиции - на нечётных, причем для разных элементов используются синусы и косинусы разных периодов от 2π до $10000 \cdot 2\pi$:

$$e^p_{2i} = sin(\frac{pos}{10000^{2i/D}}) \quad (1)$$

$$e^p_{2i+1} = cos(\frac{pos}{10000^{2i/D}}) \quad (2)$$

Можно объединить эти вектора в матрицу E размером T×D [3].

Каждый блок кодировщика имеет свои параметры, но функционально устроен единообразно:

1. каждый входной эмбеддинг преобразуется через блок самовнимания (self-attention) в выходной эмбеддинг;
2. входной и выходной эмбеддинги суммируются, как показано пунктиром на схеме;
3. суммарный эмбеддинг пропускается через послойную нормализацию (LayerNorm);
4. результирующий эмбеддинг преобразуется двухслойным персептроном (Feed Forward) в выходной;
5. затем опять входной и выходной эмбеддинги (в контексте предыдущего шага) суммируются, как показано пунктиром на схеме (рисунок 2);
6. суммарный эмбеддинг снова пропускается через послойную нормализацию (LayerNorm).

Для каждого из N блоков декодировщика этапы 1-6 применяются к элементу входной последовательности независимо с одинаковыми весами. При этом веса внутри каждого блока свои.

Блок самовнимания призван обогатить эмбеддинг каждого кадра информацией о других кадрах последовательности. Если обрабатывается последовательность длины T из D-мерных эмбедингов, тогда входную последовательность можно представить в виде матрицы X размером $T \times D$. При использовании механизма самовнимания по эмбедингу каждого кадра генерируются:

1. D- мерный запрос (query) $q_{1\times d} = x_{1\times D} \cdot W^Q_{D\times d}$
2. D-мерный ключ (key) $k_{1\times d} = x_{1\times D} \cdot W^Q_{D\times d}$

3. $\bar{d}$-мерное значение (value) $v_{1\times\bar{d}} = x_{1\times D} \cdot W^{V}_{D\times\bar{d}}$

Объединяя для каждого токена последовательности вектора их запросов, ключей и значений, получим: матрицу запросов $Q_{T\times d}$, матрицу ключей $K_{T\times d}$, матрицу значений $V_{T\times\bar{d}}$. Тогда выходной эмбеддинг вычисляется агрегацией значений для всех токенов последовательности:

$$y_{1\times\bar{d}} = \operatorname{softmax}\left(\frac{1}{\sqrt{d}} q_{1\times d}(K^T)_{d\times T}\right)_{1\times T} V_{T\times\bar{d}} \tag{3}$$

Агрегация производится суммированием значений с весами, равными соответствиям соответствующих ключей запросу. Соответствия вычисляются по их отмасштабированному скалярному произведению. Для повышения производительности вычисления производятся для всех токенов одновременно, используя матричную запись. Результат самовнимания для всех токенов записывается как:

$$Y_{T\times\bar{d}} = \operatorname{softmax}\left(\frac{1}{\sqrt{d}} Q_{T\times d}(K^T)_{d\times T}\right)_{T\times T} V_{T\times\bar{d}} \tag{4}$$

В трансформере используется несколько голов самовнимания, каждая - со своими весами., что позволяет собирать разную контекстную информацию. После применения голов самовнимания их результаты конкатенируются и пропускаются через линейное преобразование, чтобы вернуть размерность эмбеддинга к исходному вектору [3]. Визуализация процесса показана на рисунке 3.

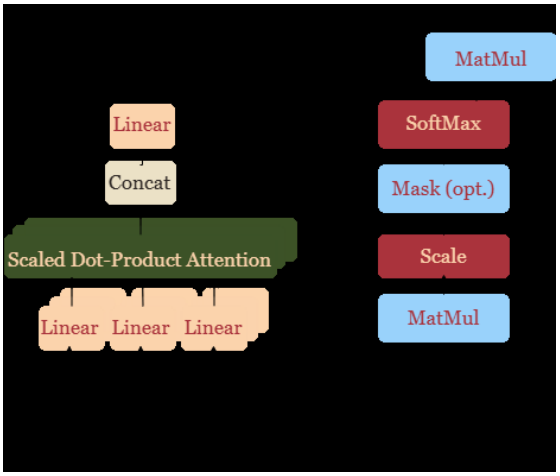

Рисунок 3 – Механизм самовнимания

Слой LayerNorm принимает на вход активации выходы предыдущего слоя сети и линейно перемасштабирует их:

$$z_M \to \gamma_M \frac{z_M - \mu}{\sqrt{\sigma^2 + \delta}} + \beta_M, \tag{5}$$

где $\delta > 0$ - малая константа, исключающая деление на ноль, а параметры μ и σ вычисляются как выборочное среднее и СКО по активациям всех нейронов слоя в рамках одного объекта:

$$\mu = \frac{1}{M}\sum_{m=1}^{M} z_m \tag{6}$$

$$\sigma = \sqrt{\frac{1}{M}\sum_{m=1}^{M}(z_m - \mu)^2} \tag{7}$$

Feed Forward преобразование представляет собой двухслойный персептрон с активацией ReLU:

$$y = ReLU(xW_1 + b_1)W_2 + b_2 \quad (8)$$

где x - входной эмбеддинг кадра, y - его нелинейно преобразованная версия той же размерности, W_1, W_2 - обучаемые матрицы, b_1, b_2 - обучаемые вектора смещений. Этот блок позволяет модели настраивать сложные зависимости и строить более богатые признаковые представления.

Архитектура декодировщика. Декодировщик модели трансформера принимает на вход M эмбеддингов элементов уже сгенерированной последовательности и выдаёт вероятностное распределение следующего предсказываемого элемента последовательности.

Декодировщик состоит из N последовательных применений блоков декодировщика, каждый раз - со своими параметрами. Задача каждого блока - уточнить эмбеддинги сгенерированных токенов с учётом их контекста из других сгенерированных токенов. Последний блок выдаёт окончательные эмбеддинги, к каждому из которых независимо применяется линейный слой и SoftMax-преобразование. Итоговый вектор выходов интерпретируется как вероятность токенов на каждой позиции выходной последовательности. Интересен только токен на позиции t, который выбирается максимально вероятным. После этого декодировщик перезапускается для предсказания токенов в позициях t+1, t+2,... пока не будет сгенерирован токен [EOS], означающий конец генерации.

Для декодировщика входными токенами является уже не входная, а выходная последовательность, сдвинутая на один шаг вперёд специальным токеном [BOS]. Это сделано для того, чтобы каждый раз для всех ранее сгенерированных токенов предсказывался следующий, а в самом начале - первый токен.

Блок самовнимания применяется точно так же, как и в случае кодировщика, с отличием, что степени внимания маскируются таким образом, чтобы из позиции t нельзя было смотреть в будущие позиции t+1,t+2,..., которые ещё не сгенерированы. Данный механизм называется маскируемое внимание.

Также в декодировщике используется блок перекрестного внимания, уточняющий эмбеддинги декодировщика, опираясь на информацию о входной последовательности. Он устроен точно так же, как и блок самовнимания в кодировщике, но внимание в нём направлено от токенов выходной последовательности к итоговым токенам входной последовательности, полученных с выходов последнего блока кодировщика. Это реализуется тем, что:

1. запросы генерируются по уже сгенерированным токенам выходной последовательности;
2. ключи и значения генерируются по эмбеддингам входной последовательности.

Таким образом наблюдаются 3 вида внимания в трансформере: самовнимание в кодировщике, маскируемое внимание в декодировщике и перекрестное внимание в декодировщике. Эти типы вниманий представлены на рисунке 4.

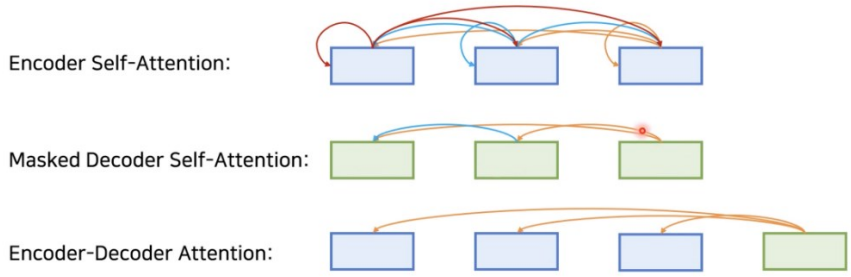

Рисунок 4 – Виды внимания в трансформере

Вывод. В результате проделанной работы была сформирована архитектура модели трансформера для визуального распознавания речи с использованием сверточной сети с трехмерной сверткой для извлечения визуальных признаков..

Список использованных источников:

1. Макар, Д. А. Нейронная сеть на основе сверточных, рекуррентных слоев и механизма внимания для визуального распознавания речи / Д. А. Макар, М. И. Вашкевич // Доклады БГУИР. – 2026. – Т. 24, № 1. – С. 75–82.

2.Кодировщик трансформера [Электронный ресурс]. – Режим доступа : https://deepmachinelearning.ru/docs/Neural-networks/Transformer/Encoder-self-attention.

3.The Transformers [Электронный ресурс]. – Режим доступа : https://www.vizuaranewsletter.com/p/the-transformers.

4. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. – 2017. – Т. 30.

.

UDC 004.934

A VISUAL SPEECH RECOGNITION SYSTEM BASED ON A TRANSFORMER-TYPE NEURAL NETWORK

Zubritskaya K.I. student

Belarusian State University of Informatics and Radioelectronics[1]
Minsk, Republic of Belarus

Vashkevich M.I. – Doctor of Science

Annotation. The work presents the development of a transformer-based neural network architecture for the task of visual speech recognition. A proprietary data set was collected, as well as its preprocessing, and the architecture of a convolutional network for extracting visual features was compiled. During the design, a MediaPipe-based face key point detector, encoder blocks for analyzing visual features and creating their contextual representation, and a decoder for step-by-step text generation based on encoder outputs were used. The work shows examples of the received data and network architectures.

Keywords. Transformer, convolutional neural network, deep learning, computer vision.