Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

Пояснительная записка к курсовой работе

по дисциплине «Оптимизационные задачи в машинном обучении»

на тему:

«Разработка системы анализа эмоционального состояния человека»

Выполнил(а):

студент(ка) группы ПМ20-2

Факультета информационных технологий и анализа больших данных

Короткова Вера Владмировна

Научный руководитель:

старший преподаватель

Субботина Ольга Робертовна

2023 г

Оглавление

[ВВЕДЕНИЕ 2](#_Toc153793076)

[Глава 1. Проблема анализа эмоционального состояния человека и подходы к ее решению 3](#_Toc153793077)

[1.1 Задача анализа эмоционального состояния человека 3](#_Toc153793078)

[1.2. Использование искусственных нейронных сетей для решения задач машинного обучения 6](#_Toc153793079)

[1.3. Сверточные нейронные сети 9](#_Toc153793080)

[Глава 2. Реализация программного комплекса для решения задачи анализа эмоционального состояния человека. 13](#_Toc153793081)

[2.1. Подготовка набора данных. 13](#_Toc153793082)

[2.2. Предобработка набора данных 14](#_Toc153793083)

[2.3. Использование предобученной модели для решения поставленной задачи анализа эмоций. 15](#_Toc153793084)

[2.4. Разработка и настройка сверточной нейронной сети для решения задачи компьютерного зрения 18](#_Toc153793085)

[2.4. Результаты 22](#_Toc153793086)

[ЗАКЛЮЧЕНИЕ 25](#_Toc153793087)

[СПИСОК ИСТОЧНИКОВ 26](#_Toc153793088)

[ПРИЛОЖЕНИЕ А. ОПИСАНИЕ ПРОЦЕССОРА 27](#_Toc153793089)

[ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД 28](#_Toc153793090)

ВВЕДЕНИЕ

В современном мире, где информационные технологии стремительно развиваются, анализ эмоционального состояния человека является актуальной задачей. Он необходим различным компаниям, чтобы понимать, как люди принимают решения и ведут себя в зависимости от различных эмоций.

Целью данной курсовой работы является разработка системы анализа эмоционального состояния человека по фотографии. Для достижения поставленной цели будут решены следующие задачи: приведена теоретическая справка по изучаемой области, подготовлен набор данных, построена и оптимизирована собственная модель, использована предобученная модель для сравнения результатов и подведены итоги.

Предметом исследования являются именно такие системы, которые могут классифицировать разные эмоции. А объектом являются фото людей, чьи эмоции необходимо определить.

Распознавание эмоций может проводиться по разным данным, например по голосу (громкости, интонации и т.д.), жестам и фотографиям, то есть по выражению лица и мимике. Для данной работы были выбраны именно изображения, благодаря их универсальности (отсутствие проблем языкового барьера, как в записях голоса) и возможности объективной оценки.

Актуальность данной работы состоит в создании модели, которая сможет соперничать с большой предобученной моделью по метрикам, однако ее архитектура компактнее в разы.

Глава 1. Проблема анализа эмоционального состояния человека и подходы к ее решению

1.1 Задача анализа эмоционального состояния человека

Emotion recognition (или распознавание эмоций) – является одной из самых популярных задач классификации. На данный момент уже разработаны системы, которые распознают эмоции людей по видео, аудио, фотографиям и даже тексту.

В данной работе в качестве исследуемых данных были выбраны именно фотографии по следующим причинам:

1. На фотографиях видно как лицо человека, по которому можно различать эмоции по мимике, так и все тело, по которому классификации могут помочь позы и жесты. Благодаря этому, модель может работать более точно.
2. Больше всего распространены датасеты, содержащие именно фотографии. К тому же существует уже множество различных способов работы с таким типом данных, благодаря чему можно разработать оптимальную структуру системы.
3. Для фотографий существует такой метод как аугментация. Его суть заключается в изменении выборки или даже увеличении ее размера путем модификации данных. Количество примеров для обучения также очень сильно влияет на качество итоговой модели.

Системы машинного обучения, созданные для распознавания эмоций имеют очень широкое применение. Самые популярные направления следующие:

1. Анализ социальных сетей. Определение эмоциональных реакций идей на продукты и услуги может быть полезно для маркетинговых исследований.
2. Психология. Такие системы могут использовать реакции людей на различные стимулы для психологических исследований.
3. Безопасность. Система распознавания эмоций может использоваться в сфере охраны для определения подозрительных и угрожающих личностей.
4. Здравоохранение. По выражению эмоций медики могут выявить развитие различных психических заболеваний.

Задача ER (emotion recognition) уже давно интересна исследователям. Первым исследования начал Чарльз Дарвин в своей работе “The Expression of the Emotions in Man and Animals” 1872 года [1]. В начале 2000х задача решалась с помощью методов классического машинного обучения, например SVM (опорных векторов) или k ближайших соседей. А в последнее время, как и в моей работе, для распознавания эмоций будет использоваться глубокая нейронная сеть.

Чаще всего для систем распознавания эмоций используются 7 базовых эмоций, которыми являются радость, грусть, страх, доверие, ожидание, удивление, злость, неудовольствие. Фото человека, испытывающего эти эмоции представлены на рис.1:



Рис. 1. Пример фото человека по эмоциям

Существует множество методов классического машинного обучения, на основе которых можно пробовать строить системы по распознаванию эмоций. Однако только глубокая нейронная сеть может распознавать эмоции так, как это мог бы сделать живой человек.

1.2. Использование искусственных нейронных сетей для решения задач машинного обучения

Искусственная нейронная сеть – это реализация математической модели, построенной учеными на основе работы нервных клеток живого организма.

Она представляет собой некоторое количество связанных нейронных слоев, на вход которых передается информация. Данная информация обрабатывается и “движется” по сети до последнего выходного слоя. Во многих источниках, в том числе в учебнике Голдберга [2], говорится о том, что глубокое обучение является разделом машинного обучения, использующим нейронные сети. Оно создано для решения задач из таких областей, как компьютерное зрение, обработка естественного языка, в т.ч. генерация текстов, распознавание образов и т.д.

Для любой задачи глубокого машинного обучения основные элементы всегда одинаковые, это нейронные слои, функции активации, функции потерь и оптимизатор. Меняется только их количество (глубина нейронной сети) и виды. Для каждой области исследований уже изучены и подобраны оптимальные из них. В зависимости от специфики задачи могут добавляться также такие слои как dropout, нормализация и т.д. Схема цикла обучения представлена на рис. 2:

Изображение выглядит как диаграмма

Автоматически созданное описание

Рис. 2. Схема цикла обучения

Рассмотрим несколько основных элементов глубокой нейронной сети подробнее:

Функции активации – это математические функции, которые модифицируют входные данные, помогая нейронной сети выявить нелинейные зависимости между данными. Они могут решать проблему затухания градиента. Также без функций активации не было бы возможности решать сложные нелинейные задачи. На каждом слое можно использовать различные функции активации, но обычно используют либо одинаковые, либо повторяющиеся, но точно не случайные.

Исследователи в области машинного обучения уже достаточно долгое количество времени разрабатывают новые, все более эффективные функции активации для решения определенных задач. На данный момент только в Википедии упоминается более 20 различных вариантов. Чаще всего используются сигмоида, tanh, ReLU и SoftMax.

В представленной в данной работе системе распознавания эмоций в качестве функции активации используется ReLU. Был сделан такой выбор из-за того что при работе с картинками чаще всего используется именно выбранная функция активации или ее аналоги. Частота ее использования обуславливается ее простотой: ReLU(x) = max(0, x). Важно отметить, что в задачах классификации на последнем слое всегда используется SoftMax, для преобразования данных в вероятностное распределение.

Функции ошибки созданы для того, чтобы определять точность модели. Они характеризуют отклонения реальных и предсказанных данных друг от друга. Модель глубокого обучения в процессе обучения всегда стремится уменьшить это значение.

Функций потерь также существует большое разнообразие. На данный момент чаще всего используются MSE, MAE, MAPE, CEL и т.д.

Для решения задач классификации в моделях чаще всего используется именно CrossEntropyLoss в качестве функции активации. Это происходит из-за того что она создана таким образом, что может учитывать вероятность принадлежности к классу.

Во время обучения нейронной сети модель предсказывает значения целевой переменной, которые затем сравниваются с наблюдаемыми значениями при помощи функции потерь. Для обновления весов нейронов в процессе обучения используются оптимизаторы, которые подсчитывают градиенты функции потерь. Один из наиболее распространенных и простых оптимизаторов – стохастический градиентный спуск (SGD).

Однако, в последние годы, было показано, что оптимизатор Adam может быть более эффективным, чем SGD. Adam является адаптивным методом оптимизации, который использует два момента градиента: первый момент (среднее значение градиента) и второй момент (среднеквадратичное отклонение градиента). Это позволяет ему быстрее сходиться к оптимальному решению и уменьшает вероятность застревания в локальных минимумах. Adam также является устойчивым к выбору гиперпараметров и имеет меньшую вероятность переобучения, чем SGD. Эти преимущества делают его предпочтительным выбором для многих задач машинного обучения. В данной работе для увеличения производительности использовался Adam.

1.3. Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks) были созданы для обработки изображений и видео. Главным их отличием от обычных нейронных сетей является наличие операции “свертки”, которая используется вместо минимум одного из линейных слоев. Их главные преимущества – возможность извлечения признаков из изображений, такие как границы, углы, текстуры и другие характеристики. Это делает их полезными для задач компьютерного зрения: классификация изображений, распознавание объектов. Кроме того, сверточные нейронные сети также могут использоваться в обработке звука и текста, но для таких типов данных есть более актуальные модели машинного обучения. Пример работы сверточной нейронной сети приведен на рис. 3:



Рис. 3. Пример работы сверточной сети

Основными элементами CNN являются:

1. Сверточные слои
2. Пулинг (max или average)
3. Нормализация
4. Полносвязные слои

Теперь можно подробнее разобрать, как работает каждый из них и для чего они нужны:

Из учебника Гудфеллоу можно узнать, что свертка – это операция над двумя функциями вещественного аргумента [3]. Формульно ее можно отобразить так:

где x – входы, w – ядро (или веса), а выход можно назвать картой признаков

В PyTorch сверточный слой записывается так:

nn.Conv2d(in\_channels,out\_channels, kernel\_size, stride, padding)

* in\_channels – количество входных каналов. Например, для черно-белого изображение оно равняется 1, а для RGB - 3
* out\_channels – число выходных каналов (или число фильтров)
* kernel\_size – размер используемой свертки
* stride – расстояние, на которое перемещается ядро свертки. По умолчанию это значение равно 1, но если его увеличить то итого изображение будет меньше оригинального
* padding – количество единиц расстояния, на которое увеличится исходное изображение перед сверткой. Оно используется, чтобы не потерять признаки, которые можно извлечь из краев изображения

Теперь можно перейти к пояснению не менее важного элемента любой сверточной сети – субдискретизации (или пулингу).

Пойнтер в своем учебнике пишет о том, что эти слои используются для снижения разрешения сети и уменьшения количества параметров на последних слоях [4]. Наличие пулинга помогает моделям работать быстрее, а также помогает избегать переобучения.

Существует два основных типа пулинга: max pooling и average pooling. Макспулинг выбирает максимальное значение из каждой области определенного размера карты признаков, а усредняющий пулинг вычисляет среднее значение такой области.

В PyTorch для первого типа используется функция nn.MaxPool2d, а для второго типа - nn.AvgPool2d.

В целом, параметры этих слоев такие же, как у сверточных. Только в них не нужна информация о входных и выходных каналах.

Рассмотрим еще один элемент – слой dropout – он не является обязательным для успешного обучения модели, но при этом все равно используется довольно часто.

Разработчики постоянно сталкиваются с проблемой переобучения – то есть модель очень хорошо обучается на тренировочных данных, запоминает их, но не может показать такую же хорошую точность на тестовой выборке. Методов для решения этой проблемы придумано уже большое количество. Одним из них как раз и является слой дропаут. Его суть заключается в том, чтобы обнулять рандомные выходы слоя сети во время цикла обучения.

В PyTorch он используются с помощью следующей функции:

torch.nn.Dropout(p=0.5, inplace=False)

Переменная p как раз определяет – какой процент весов будет обнуляться.

Нормализация в сверточных слоях (batch normalization) используется для ускорения обучения и повышения точности. Он позволяет стабилизировать и оптимизировать модель.

Суть данной математической операции заключается в следующем: изначально данные распределены каким-то образом, а мы приведем их среднее и дисперсию к значениям 0 и 1, чтобы упростить процесс оптимизации обучения.

В конце концов полносвязный слой (fully-connected layer) соединяет все признаки и применяет какие-то линейные преобразования.

Итого работу сверточной сети можно описать примерно следующим образом:

Входные данные проходят через сверточный слой, затем через слой нормализации, слой активации и пулинговый слой, возможно также добавления слоя дропаут. Затем выход пулингового слоя передается в следующий сверточный слой, и так далее. В конце модели находится полносвязный слой, который преобразует признаки в вероятности для каждого класса.

Таким образом, сверточная нейронная сеть позволяет извлекать признаки из изображений и других типов данных, что делает ее полезной для задач классификации, распознавания объектов и других задач компьютерного зрения.

Глава 2. Реализация программного комплекса для решения задачи анализа эмоционального состояния человека.

2.1. Подготовка набора данных.

Для обучения систем FER – распознавания эмоций по выражениям лиц – существует большое количество разнообразных датасетов. Самыми популярными являются: AffectNet, CK+, FER2013, DISFA, Emotic и другие.

Для написания своей модели была выбрана AffectNet, потому что это открытый доступный набор данных, состоящий из большого количества записей [5].

Датасет состоит более чем из миллиона фотографий людей, которые испытывают одну из 8 самых популярных эмоций. Их нумерация следующая:

* 0 – злость
* 1 – удивление
* 2 – презрение
* 3 – радость
* 4 – нейтральное
* 5 – страх
* 6 – грусть
* 7 - отвращение

Стоит уточнить, что для сокращения времени обучения мной было принято решение взять небольшую часть датасета – около 25000 фото.

В качестве примера данных можно рассмотреть пять первых фотографий из датасета. Они представлены на рис. 4:

Изображение выглядит как Человеческое лицо, Лоб, снимок экрана, улыбка

Автоматически созданное описание

Рис. 4. Пример фотографий из датасета

Для повышения точности модели в задачах компьютерного зрения часто используется уже упомянутой выше прием – аугментация.

Аугментация - это процесс модификации существующих изображений с целью увеличения разнообразия данных обучения и улучшения обобщающей способности модели глубокого обучения. В одной из самых популярных статей на эту тему приводится подробное пояснение, какой выигрыш по метрикам дает использование данной технологии [6].

В своей курсовой работе мной была проведена аугментация без увеличения количества фото, а изменения (например, повороты) оригиналов фотографий. Для увеличения обобщаемости модели этого достаточно.

2.2. Предобработка набора данных

Одной из особенностей выбранного датасета является то, что данные представлены в байтах. Например, самое первое фото записывается так, как показано на рис. 5:

Изображение выглядит как текст, письмо, Шрифт, бумага

Автоматически созданное описание

Рис. 5. Запись фото в байтах

Для преобразования байтов в тензоры была написана небольшая функция convert\_to(). В нее передается запись фото в байтах, а она с помощью специального метода torchvision.transforms.functional.pil\_to\_tensor() библиотеки torch возвращает запись фото в виде тензора. Все фото были пропущены через эту функцию.

Данные в работе представлены следующим образом – pandas датафрейм, по столбцам записаны представление фото в байтах, номер эмоции и представление фото в виде тензора.

Целевой переменной – y - является столбец ‘label’, а экзогенной переменной – x - являются данные из столбца ‘tensor’.

Далее, как и для любой задачи по машинному обучению, данные делятся на тестовую и тренировочную выборку. В данном случае это было сделано с соотношением 4:1, что является самым популярным вариантом разбиения. Делается это с помощью встроенного метода библиотеки scikitlearn - train\_test\_split().

2.3. Использование предобученной модели для решения поставленной задачи анализа эмоций.

Предобученные модели – это модели машинного обучения, написанные профессиональными DL разработчиками, имеющие сложную архитектуру и обученные на огромных наборах данных. Обычно веса всех слоев, кроме последнего, фиксируются. После этого пользователь дообучает выбранную модель на своих данных. Из-за сложности архитектуры они, как правило, дают достаточно высокую точность. Поэтому в данной курсовой работе было сначала будут получены результаты с использованием предобученной модели, чтобы появилось понимание, к какому примерно уровню точности можно стремиться.

Предобученные модели разделяются на несколько видов по разным параметрам, в том числе по типу данных – для текстовых данных используется, например, BERT, для картинок же используются другие модели.

Самыми популярными предобученными моделями для классификации изображений являются VGG-16, ResNet, EfficientNet и т.д. Их точность на самом популярном наборе данных ImageNet представлена на рис. 6:

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

Рис. 6. Метрики предобученных моделей

Согласно данной таблице модель EfficientNet достигает лучших значений точности, именно поэтому для исследуемого набора данных в моей работе была выбрана эта модель.

В статье разработчика этой модели [7] говорится о том, что EfficientNet - это новый метод масштабирования сверточных нейронных сетей, который позволяет достигать лучшей эффективности. Она представляет из себя равномерное масштабирование ширины, глубины и разрешения сети с использованием коэффициента. С помощью его балансировки EfficientNet достигает высокой точности на наборе данных ImageNet, используя значительно меньше параметров и вычислительных ресурсов по сравнению с другими моделями.

Ссылка на полный код работы находится в Приложении 2. Далее будут приведены только части кода.

Для скачки модели и ее предобученных весов была придумана библиотека timm. В ней находится большое разнообразие предобученных моделей.

Создается модель следующим образом:

model = timm.create\_model('efficientformerv2\_l', pretrained=True, in\_chans=3, num\_classes=8)

Она имеет сложную архитектуру, на рис. 7 представлена ее часть:

Изображение выглядит как текст, Шрифт, письмо, снимок экрана

Автоматически созданное описание

Рис. 7. Архитектура EfficientNet

Для обучения последнего слоя используется вполне стандартный цикл. Перед этим, как обычно, были выбраны функция потерь – CrossEntropyLoss и оптимизатор – Adam. В цикле на каждой из 25 эпох изменяются веса с помощью оптимизатора, подсчитывается и выводится значение ошибки. Цикл представлен на рис. 8.

Изображение выглядит как текст, программное обеспечение, веб-страница, Веб-сайт

Автоматически созданное описание

Рис. 8. Цикл обучения

Затем уже происходит обучение модели. Длительность каждой эпохи – около 10 минут. Таким образом, обучение длится более 4 часов.

На данной модели были получены следующие результаты:

* accuracy – 68,043 %
* F1 – 68,034 %

2.4. Разработка и настройка сверточной нейронной сети для решения задачи компьютерного зрения

Стоит уточнить, почему были выбраны именно такая функция ошибок и оптимизатор:

* Функция ошибки – CrossEntropyLoss. Использовалась именно CEL, так как для большей части задач классификации разработчики выбирают именно ее.
* Оптимизатор – Adam. Как уже было сказано в теоретической части – опытным путем было доказано его превосходство над другими оптимизаторами для классических задач.

Далее уже будет происходить создание и обучение собственной нейронной сети для исследуемого набора данных.   
В первую очередь необходимо было создать класс Dataset, в данном случае он называется MyDataset. Он создается, для того чтобы в цикле обучения можно было удобно выбирать нужные записи данных. Класс MyDataset имеет два основных метода, которые были также прописаны и мной. Стоит уточнить, что для разных наборов данных написание этих методов немного отличается.

Первый метод – len() – возвращает количество записей (строк) в датасете. Второе метод – getitem(idx) – возвращает значения x и y строки под номером idx.

Далее создаётся конвейер трансформаций картинки с помощью библиотеки torchvision.transforms. В данном случае используется горизонтальный переворот (рандомный), рандомный поворот (максимум на 10 градусов), нормализация картинки и ресайз под один размер (224 на 224, потому что именно такой размер принимает EficientNet, чтобы можно было с ним корректно сравнивать метрики).

Затем мы создаем объект класса Dataset, с параметром transform=transform (то есть передаем туда созданный конвейер). Благодаря этому в дальнейшем при получении элемента из датасета, к i-ой картинке применяется созданный конвейер из изменений над оригинальной картинкой.

Для оптимизации времени обучения, в датасет добавлена возможность перевода данных на необходимый девайс (CPU или GPU). Поэтому при обучении данные и модель находятся на GPU, что ускоряет время обучения, а при валидации (т.к. данные могут не поместиться) они переносятся на CPU (вместе с моделью). Характеристики устройств, на которых проводилось обучение находятся в Приложении 1.

Теперь необходимо прописать цикл обучения, по которому будут обучаться модели, написанные самостоятельно.

Для более точной оценки эффективности моделей машинного обучения часто используется такой прием, как перекрестная кросс-валидация. Этот метод позволяет оценить, насколько хорошо модель будет работать на независимых данных. Принцип его работы состоит в следующем: исходный набор данных разбивается на k подмножеств (в нашем случае - 4) одинакового размера. Для каждого из них модель обучается на оставшихся 3 подмножествах и проверяется на том, который был исключен из обучения. Процесс повторяется k раз. Оценка производительности модели равна среднему среди полученных результатов k прогонов.

Таким образом для кросс-валидации был создан цикл. В нем будет происходить следующее:

Создается модель вместе функцией ошибки и оптимизатором; тренировочный и валидационный датасеты; а также переменная для замеров времени.

Затем в цикле по 25 эпохам будут изменяться веса с помощью оптимизатора, подсчитываться и выводиться (каждую 5 эпоху) значения функции ошибки и метрик accuracy и F1-score.

При обучении модель переводится в положение обучения model.train() а при валидации и при тестировании - она переводится в положение тестирования model.eval() (при этом Dropout не работает).

После этого переходим непосредственно к созданию архитектуры модели.

Первая модель – MyCNNModel – является достаточно простой и состоит из 5 слоев, а именно двух сверток, макспулинга и двух линейных слоев, а также двух нормализаций, дропаута, функций активации ReLU и tanh на последнем слое.

Можно сказать, что ее архитектура создавалась почти рандомно. Примерное количество слоев перебиралось до достижения максимально возможного значения точности.

Значения на данной модели получились следующие:

* Accuracy – 43 %
* F1-score – 39,86 %
* Time – 20,28 минут в среднем

По сравнению с результатами, полученными на предобученной модели – значения этих метрик очень низкие. Поэтому далее будет создана намного более глубокая и сложная модель.

Вторая – оптимизированная модель – MyCNNModel1 – была написано после вдохновения архитектурой модели VGG16. Она состоит из 6 слоев nn.Sequential().

nn.Sequential() - это класс в библиотеке PyTorch, созданный для последовательного вызова нескольких слоев нейронной сети. В него можно передать несколько слоев в качестве аргументов.

В каждый из таких конвейеров передаются свертка, нормализация, функция активации и макспулинг. Внутри одного sequential они повторяются, по 2 или 3 раза. Однако на последнем слое добавляется два простых линейных слоя и дропаут. Таким образом данная модель состоит из 20 слоев.

Результаты по ней следующие:

* Accuracy – 62,28 %
* F1-score – 62,03 %
* Time – 39,56 минут в среднем

Очевидно, что значения метрик сильно выросли – примерно в 1.5 раза. Но между значениями этой модели и предобученной еще остается разрыв. Проанализировав данные результаты, мной был сделан вывод, что построенная модель слишком глубокая для того количества данных, на которых она обучается. Скорее всего, при большем количестве фотографий она бы дала результат лучше. Поэтому было принято решений упростить эту модель.

Вторая модель была взята за основу третьей модели – MyCNNModel2. В целом, ее архитектура примерно такая же. Отличия следующие: количество слоев sequential уменьшено на 1, из всех слоев sequential удалены повторы – в каждом не больше одной свертки и т.д. Так общее количество слоев составляет 10.

Результаты после обучения данной модели следующие:

* Accuracy – 66,15 %
* F1-score – 66,12 %
* Time – 30,12 минут в среднем

Как видно из результатов, предположения о слишком большой глубине предыдущей модели подтвердились. Более простая модель показывает лучшую эффективность.

2.4. Результаты

Для определения точности в исследуемой задаче были использованы метрики accuracy и f1-score. То, как они рассчитываются, представлено на формулах 3.1 и 3.2.

где TP – правильно предсказанный положительный результат, TN – правильно предсказанный негативный результат, FP – неправильно предсказанный положительный результат, FN – неправильно предсказанный негативный результат.

Время замерялось с помощью функций библиотеки time. В качестве визуализаций обучения моделей представлены графики функций потерь для каждой из моделей на рис. 9 - 12:

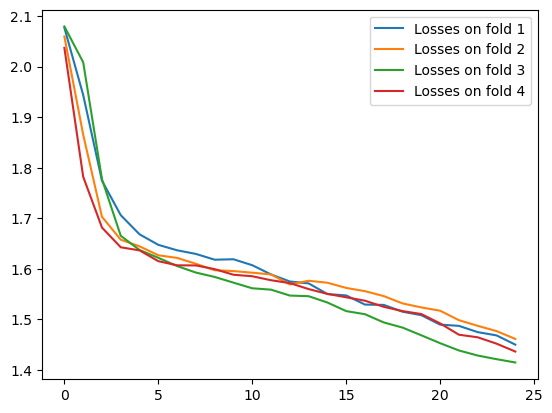


Рис. 9. График функции потерь MyCNNModel

Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание

Рис. 10. График функции потерь MyCNNModel1

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рис. 11. График функции потерь MyCNNModel2

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Рис. 12. График функции потерь EfficientNet

По рисункам можно сделать вывод о том, что каждая из моделей еще может обучаться дальше, но она уже достигла предела своей точности.

Для большей наглядности соберем все результаты в одну сводную таблицу:

Таблица 1. Метрики моделей

|  |  |  |  |
| --- | --- | --- | --- |
| **Модель** | **Accuracy** | **F1-score** | **Time** |
| **MyCNNModel** | 0,43 | 0,4 | 20 min |
| **MyCNNModel1** | 0,62 | 0,62 | 40 min |
| **MyCNNModel2** | 0,66 | 0,66 | 30 min |
| **EfficientNet** | 0,68 | 0,68 | 224 min |

В качестве финальной визуализации работы лучшей модели – MyCNNModel2 – представлены 16 картинок из тестового датасета на рис. 13. Над каждой из них подписаны реальный класс и предсказанный.

Изображение выглядит как Человеческое лицо, коллаж, улыбка, человек

Автоматически созданное описание

Рис. 13. Пример предсказания эмоций

ЗАКЛЮЧЕНИЕ

Компьютерное зрение это большое разнообразие технологий, которые решают важные для современного мира задачи. Одной из этих задач является распознавание человеческих эмоций по выражению лица. В данной курсовой работе была разработана система, которая как раз решает задачу FER. Такие системы можно внедрять в различные предприятия для отслеживания эмоционального состояния работников, реакции клиентов и т.д.

В результате выполнения работы мной было создано три модели глубокого обучения, лучшая из которых может соревноваться по метрикам с предобученной на большом наборе данных глубокой модели EfficientNet.

Однако стоит отметить, что данная модель может достичь и более высоких результатов при увеличении количества фотографий для обучения, а также при использовании новых передовых технологий в области глубокого обучения – трансформеров и автоэнкодеров.

СПИСОК ИСТОЧНИКОВ

1. Дарвин Ч. О выражении ощущений у человека и животных/ пер. под ред. Академика А. О. Ковалевского: Типография А. Пороховщикова, 1896, Москва.
2. Гольдберг Й. Нейросетевые методы в обработке естественного языка / пер. с анг. А. А. Слинкина. – Москва: ДМК Пресс, 2019. – 282 с.

Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / Я. Гудфеллоуб И. Бенджио, А. Курвилль. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.

Пойнтнер Ян. Программируем с PyTorch: Создание приложений глубокого обучения / Ян Пойнтнер. – СПб.: Питер, 2020. – 256 с.

Ali Mollahosseini. AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild / Ali Mollahossein, Mohammad H. Mahoor. – arxiv.org, 2017 – 18 p. – URL: <https://arxiv.org/pdf/1708.03985.pdf> (дата обращения 08.12.2023). – Текст: электронный.

Mingle Xu. A Comprehensive Survey of Image Augmentation Techniques for Deep Learning / Mingle Xu, Sook Yoon, Alvaro Fuentes, Dong Sun Park. – arxiv.org, 2022 – 43 p. – URL: <https://arxiv.org/pdf/2205.01491.pdf> (дата обращения 08.12.2023). – Текст: электронный.

Mingxing Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks / Mingxing Tan, Quoc V. Le. – arxiv.org, 2020 – 11 p. – URL: <https://arxiv.org/pdf/1905.11946.pdf> (дата обращения 08.12.2023). – Текст: электронный.

ПРИЛОЖЕНИЕ А. ОПИСАНИЕ ПРОЦЕССОРА

Процессор - AMD Ryzen 7 4800HS with Radeon

Ядра - 16

Оперативная память - 16 Гб DDR4

Видеокарта - NVIDIA GeForce GTX 1660 Ti with Max-Q Design

Видеопамять - 6 Гб GDDR6

Тактовая частота - 2.90 GHz

Частота системной шины - 100 MHz

Объём кэша второго уровня - 512 Кб

ПРИЛОЖЕНИЕ Б. ИСХОДНЫЙ КОД

Открытый репозиторий GitHub с исходным кодом: