

Дипломный проект на тему:

«Классификация изображений по эмоциям»

Слушатели:

Гурова Юлия Алексеевна

Рублева Екатерина Сергеевна

Рудаков Вадим Русланович

Актуальность темы и ее проблематика

Задача автоматического распознавания эмоций по мимике может быть использована:

- в маркетинге (эмоциональный отклик на контент);
- правоохранительных органах (истинность показаний подозреваемого);
- безопасность на дорогах (детекция сонливости и плохого самочувствия водителей).

Задача связана с рядом проблем:

- выбор признаков для анализа;
- определение истинности эмоций;
- контекст исходного изображения

Что такое “эмоция”?

Эмоция – это особый вид психических процессов, которые выражают переживание человеком его отношения к окружающему миру и самому себе.

Универсальные эмоции: гнев, страх, печаль, отвращение, презрение, удивление и радость



Fear

Contempt

Sadness

Happiness

Surprise

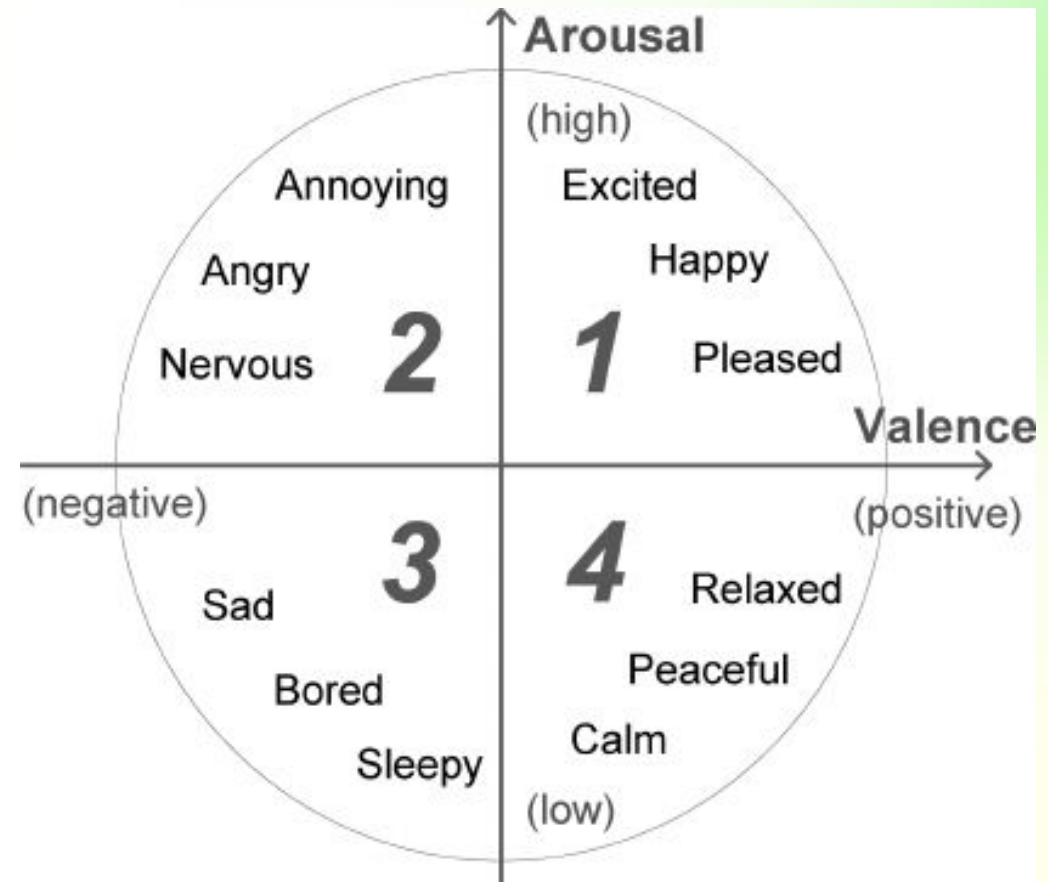
Anger

Disgust

Модель Дж. Рассела с двумерным пространством возбуждения и валентности

Валентность означает положительную или отрицательную аффективность.

Возбуждение измеряет насколько спокойным или волнующим является аффективное состояние.

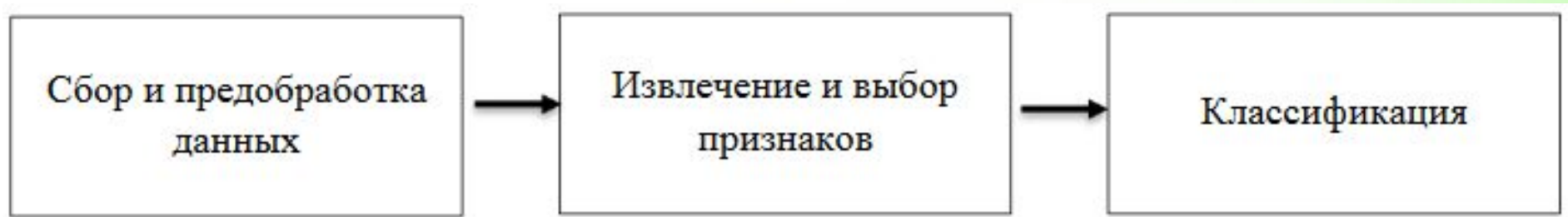


Проблема отсутствия контекста



Существующие подходы к решению задачи распознавания эмоций

Схема работы систем распознавания эмоций:



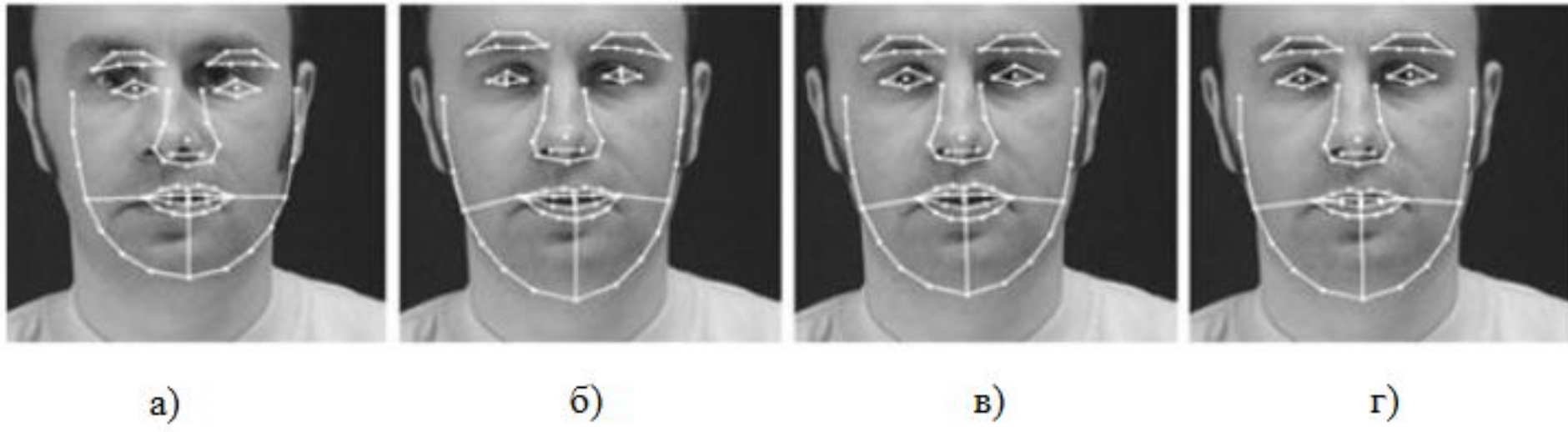
Геометрическими признаками могут являться расстояния, углы и т.д. Они характеризуют особенности расположения, ориентации объектов на изображении. 2D или 3D модели обычно используются для поиска ключевых точек лица.

Текстурные признаки отражают однородность изображения. Текстурными особенностями изображений лица являются в основном изменения образотекстуры, такие как морщины на коже и выпуклости.

Active Shape Models(ASM)

Процесс локализации ASM на изображении:

- а) –начальная позиция,
- б) –спустя 5 итераций,
- в) –спустя 10 итераций,
- г) –итоговая модель



Local Binary Pattern

⁷	⁰	¹
⁶		²
⁵	⁴	³

⁷ 42	⁰ 199	¹ 234
⁶ 177	129	² 199
⁵ 65	⁴ 177	³ 65

⁷ 1	⁰ 0	¹ 0
⁶ 0	X	² 0
⁵ 1	⁴ 0	³ 1

⁷ 1	⁶ 0	⁵ 1	⁴ 0	³ 1	² 0	¹ 0	⁰ 0
-------------------	-------------------	-------------------	-------------------	-------------------	-------------------	-------------------	-------------------

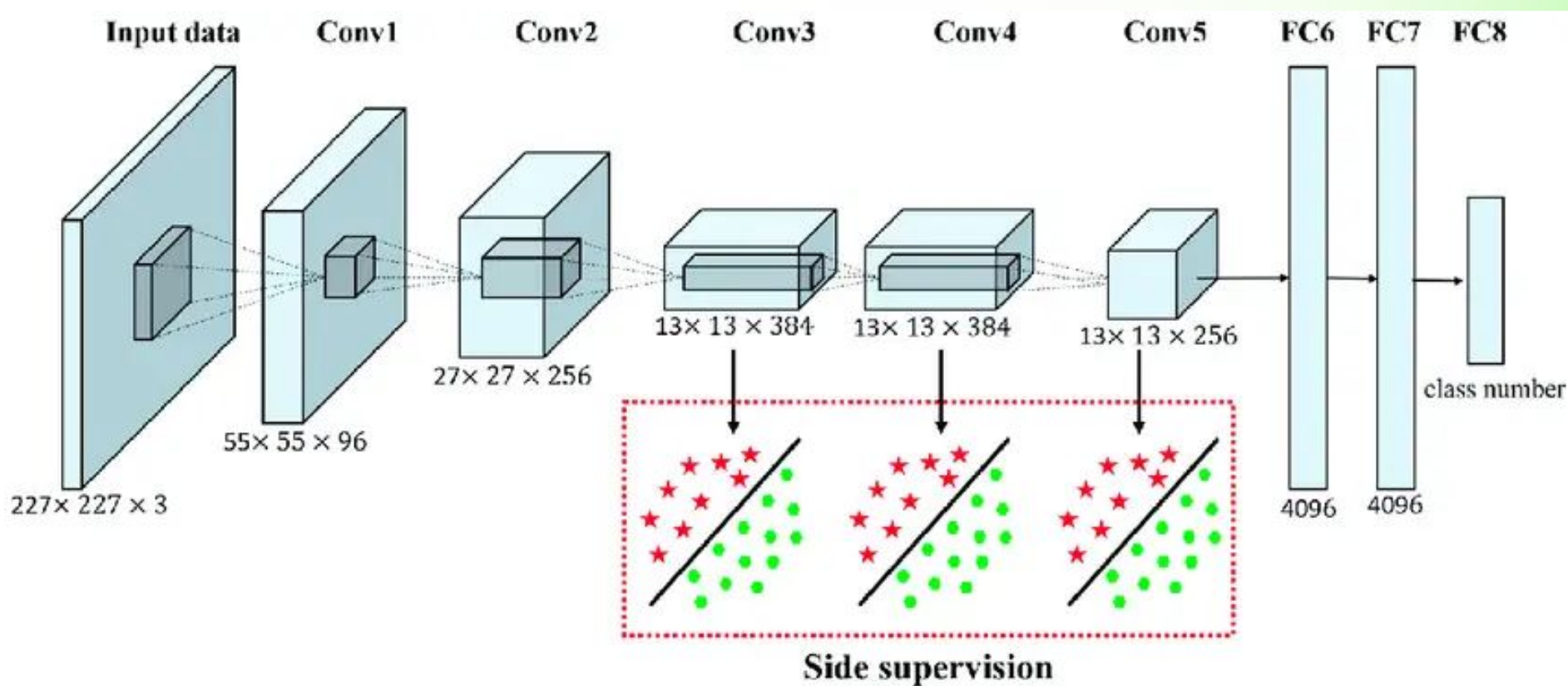
$$128 + 32 + 8 = 168$$

Существующие решения задачи распознавания эмоций (FaceReader)



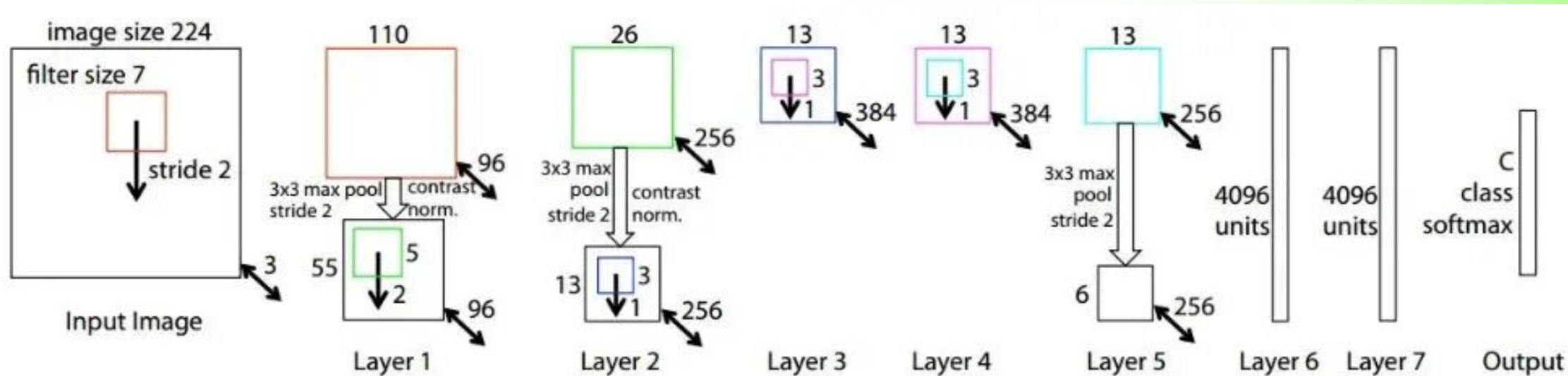
Модели на основе задачи ImageNet

AlexNet | ILSVRC Competition – 2012 (Winner) | Top-5 Error Rate – 15.30%



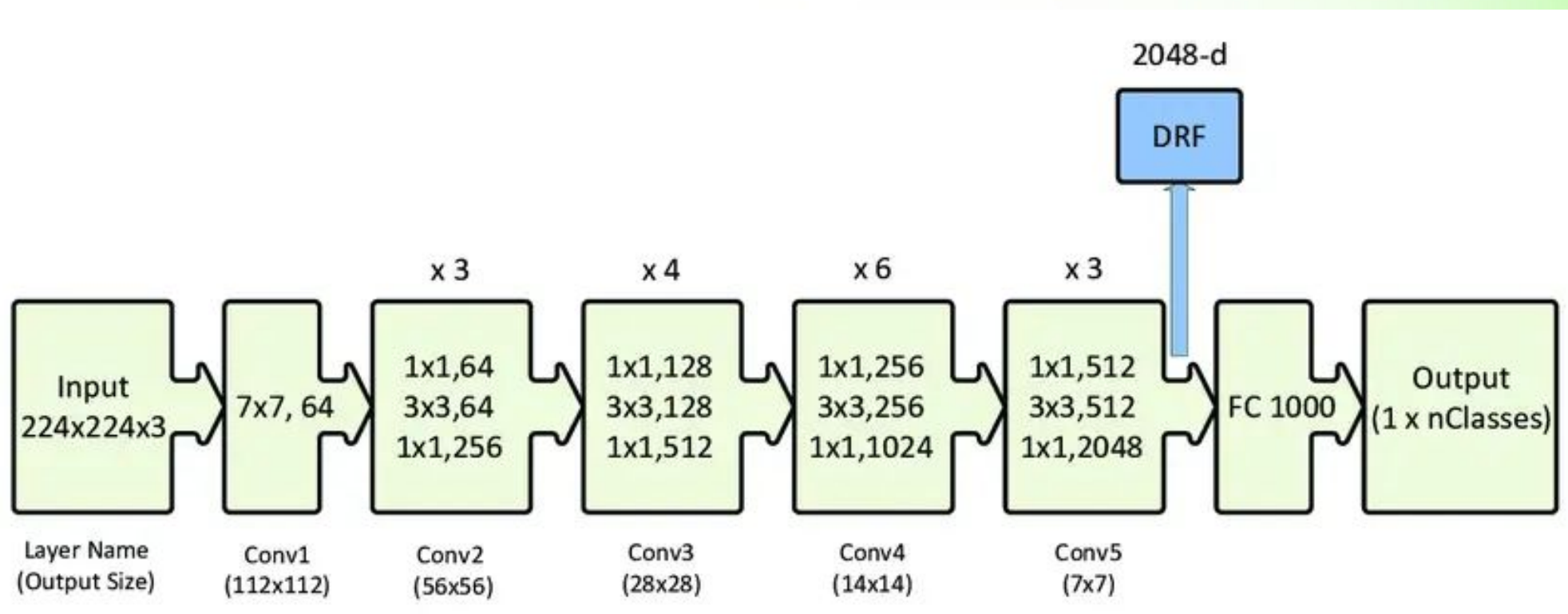
Модели на основе задачи ImageNet

ZFNet | ILSVRC Competition – 2013 (Winner) | Top-5 Error Rate – 11



Модели на основе задачи ImageNet

ResNet | ILSVRC Competition – 2015 (Winner) | Top-5 Error Rate – 3.57%



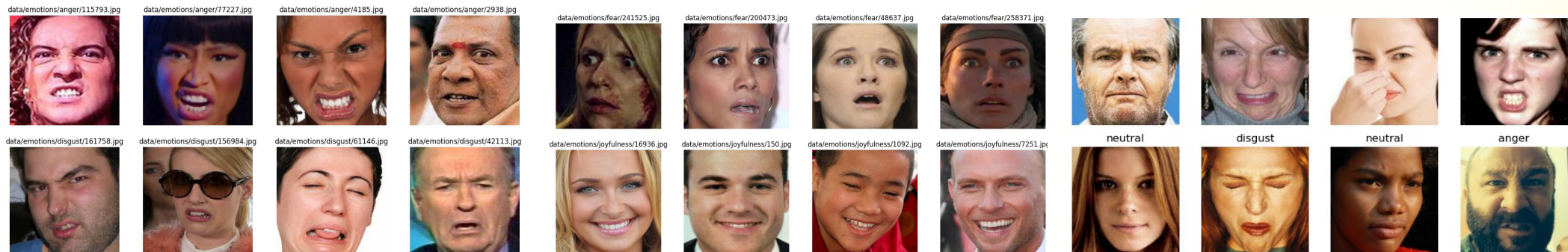
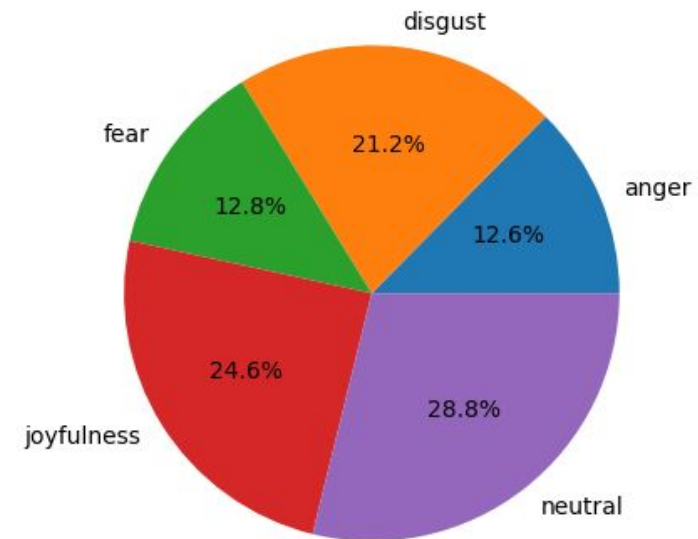
Подготовка исходного датасета

Датасет для дипломного проекта состоит из **1380** изображений по пяти несбалансированным классам эмоций:

- гнев (174 изображения),
- отвращение (292),
- страх (177),
- радость (339),
- нейтрально (398).

Датасет собран авторами работы методом ручной сортировки изображений из имеющейся в нашем распоряжении фотобазы.

Распределение количества изображений по классам



Подготовили: Гурова Юлия Алексеевна, Рублева Екатерина Сергеевна., Рудаков Вадим Русланович

Решение задачи через библиотеку PyTorch

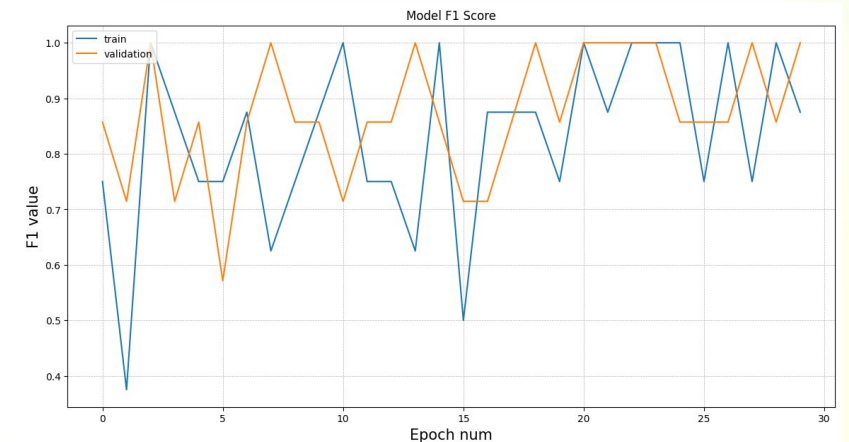
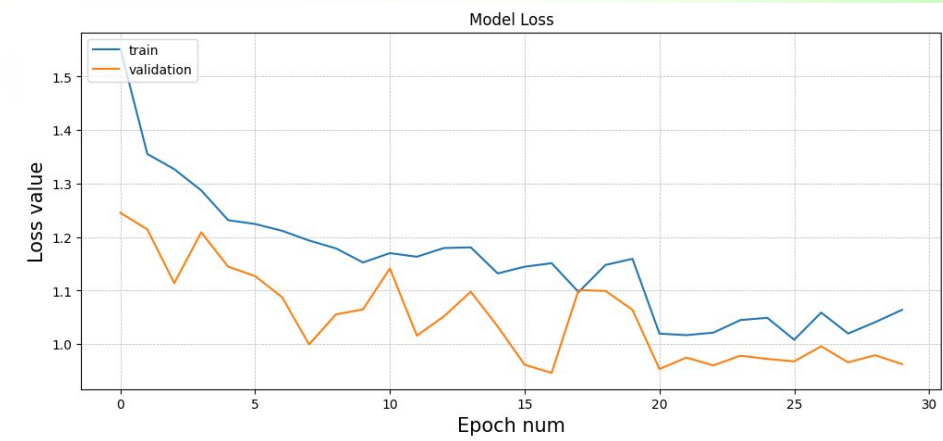
Resnet18

```
ResNet
├── Conv2d: 1-1
├── BatchNorm2d: 1-2
├── ReLU: 1-3
├── MaxPool2d: 1-4
├── Sequential: 1-5
│   ├── BasicBlock: 2-1
│   │   ├── Conv2d: 3-1
│   │   ├── BatchNorm2d: 3-2
│   │   └── ReLU: 3-3
│   ├── Conv2d: 3-4
│   ├── BatchNorm2d: 3-5
│   └── ReLU: 3-6
│   ├── BasicBlock: 2-2
│   │   ├── Conv2d: 3-7
│   │   ├── BatchNorm2d: 3-8
│   │   └── ReLU: 3-9
│   └── Conv2d: 3-10
├── Sequential: 1-8
│   ├── BasicBlock: 2-7
│   │   ├── Conv2d: 3-39
│   │   ├── BatchNorm2d: 3-40
│   │   ├── ReLU: 3-41
│   │   ├── Conv2d: 3-42
│   │   └── BatchNorm2d: 3-43
│   ├── Sequential: 3-44
│   │   ├── ReLU: 3-45
│   │   └── BasicBlock: 2-8
│   │       ├── Conv2d: 3-46
│   │       ├── BatchNorm2d: 3-47
│   │       ├── ReLU: 3-48
│   │       ├── Conv2d: 3-49
│   │       └── BatchNorm2d: 3-50
│   └── ReLU: 3-51
├── AdaptiveAvgPool2d: 1-9
└── Linear: 1-10
```

[8, 3, 224, 224]	[8, 5]	--
[8, 3, 224, 224]	[8, 64, 112, 112]	(9,408)
[8, 64, 112, 112]	[8, 64, 112, 112]	(128)
[8, 64, 112, 112]	[8, 64, 112, 112]	--
[8, 64, 112, 112]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	(36,864)
[8, 64, 56, 56]	[8, 64, 56, 56]	(128)
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	(36,864)
[8, 64, 56, 56]	[8, 64, 56, 56]	(128)
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 64, 56, 56]	[8, 64, 56, 56]	(36,864)
[8, 64, 56, 56]	[8, 64, 56, 56]	(128)
[8, 64, 56, 56]	[8, 64, 56, 56]	--
[8, 256, 14, 14]	[8, 512, 7, 7]	--
[8, 256, 14, 14]	[8, 512, 7, 7]	--
[8, 256, 14, 14]	[8, 512, 7, 7]	(1,179,648)
[8, 512, 7, 7]	[8, 512, 7, 7]	(1,024)
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 7, 7]	(2,359,296)
[8, 512, 7, 7]	[8, 512, 7, 7]	(1,024)
[8, 256, 14, 14]	[8, 512, 7, 7]	(132,096)
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 7, 7]	(2,359,296)
[8, 512, 7, 7]	[8, 512, 7, 7]	(1,024)
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 7, 7]	(2,359,296)
[8, 512, 7, 7]	[8, 512, 7, 7]	(1,024)
[8, 512, 7, 7]	[8, 512, 7, 7]	--
[8, 512, 7, 7]	[8, 512, 1, 1]	--
[8, 512]	[8, 5]	2,565

Total params: 11,179,077
Trainable params: 2,565
Non-trainable params: 11,176,512
Total mult-adds (G): 14.51

Input size (MB): 4.82
Forward/backward pass size (MB): 317.92
Params size (MB): 44.72
Estimated Total Size (MB): 367.45



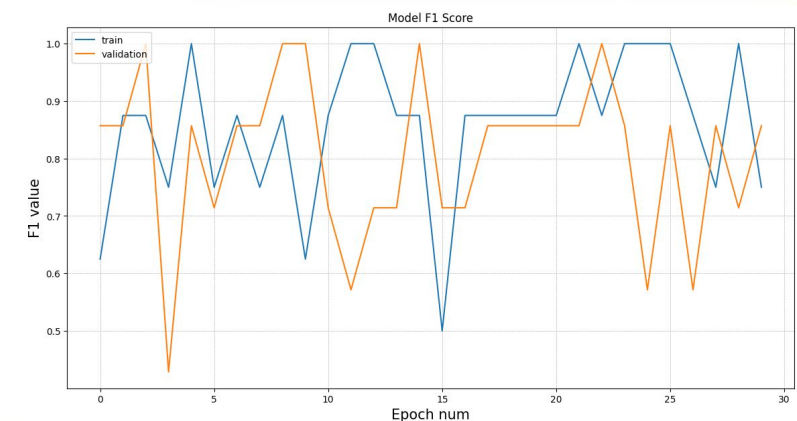
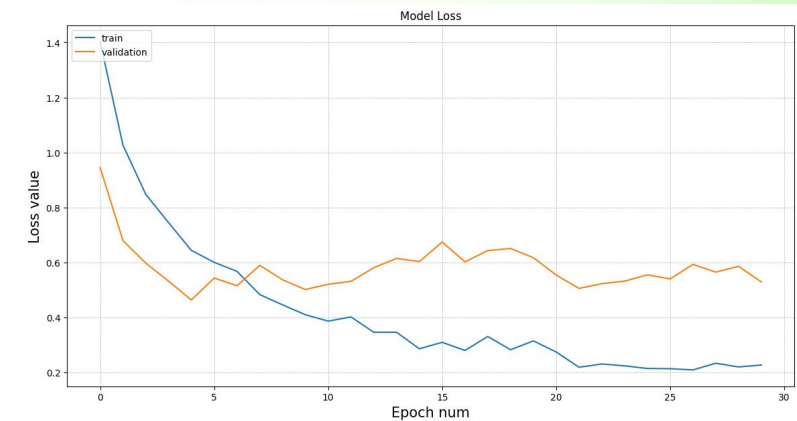
Решение задачи через библиотеку PyTorch

EfficientNet_b0

```
My_Net_efficientnet_b0
EfficientNet: 1-1
  Conv2d: 2-1
  BatchNormAct2d: 2-2
    Identity: 3-1
    SiLU: 3-2
  Sequential: 2-3
    Sequential: 3-3
    Sequential: 3-4
    Sequential: 3-5
    Sequential: 3-6
    Sequential: 3-7
    Sequential: 3-8
    Sequential: 3-9
  Conv2d: 2-4
  BatchNormAct2d: 2-5
    Identity: 3-10
    SiLU: 3-11
  SelectAdaptivePool2d: 2-6
    AdaptiveAvgPool2d: 3-12
    Flatten: 3-13
  Sequential: 2-7
    Linear: 3-14
    ReLU: 3-15
    Linear: 3-16
```

[8, 3, 256, 256]	[8, 5]	--
[8, 3, 256, 256]	[8, 5]	--
[8, 3, 256, 256]	[8, 32, 128, 128]	864
[8, 32, 128, 128]	[8, 32, 128, 128]	64
[8, 32, 128, 128]	[8, 32, 128, 128]	--
[8, 32, 128, 128]	[8, 32, 128, 128]	--
[8, 32, 128, 128]	[8, 320, 8, 8]	--
[8, 32, 128, 128]	[8, 16, 128, 128]	1,448
[8, 16, 128, 128]	[8, 24, 64, 64]	16,714
[8, 24, 64, 64]	[8, 40, 32, 32]	46,640
[8, 40, 32, 32]	[8, 80, 16, 16]	242,930
[8, 80, 16, 16]	[8, 112, 16, 16]	543,148
[8, 112, 16, 16]	[8, 192, 8, 8]	2,026,348
[8, 192, 8, 8]	[8, 320, 8, 8]	717,232
[8, 320, 8, 8]	[8, 1280, 8, 8]	409,600
[8, 1280, 8, 8]	[8, 1280, 8, 8]	2,560
[8, 1280, 8, 8]	[8, 1280, 8, 8]	--
[8, 1280, 8, 8]	[8, 1280, 8, 8]	--
[8, 1280, 8, 8]	[8, 1280]	--
[8, 1280, 1, 1]	[8, 1280, 1, 1]	--
[8, 1280, 1, 1]	[8, 1280]	--
[8, 1280]	[8, 5]	--
[8, 1280]	[8, 256]	327,936
[8, 256]	[8, 256]	--
[8, 256]	[8, 5]	1,285

```
=====
Total params: 4,336,769
Trainable params: 4,336,769
Non-trainable params: 0
Total mult-adds (G): 4.02
=====
Input size (MB): 6.29
Forward/backward pass size (MB): 563.83
Params size (MB): 17.18
```



Решение задачи через библиотеку TensorFlow

InceptionV3

Model: "InceptionV3_based"

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 299, 299, 3)]	0
sequential (Sequential)	(None, 299, 299, 3)	0
sequential_1 (Sequential)	(None, 299, 299, 3)	0
inception_v3 (Functional)	(None, 8, 8, 2048)	21802784
flatten (Flatten)	(None, 131072)	0
dropout (Dropout)	(None, 131072)	0
batch_normalization_94 (Batch Normalization)	(None, 131072)	524288
dense (Dense)	(None, 5)	655365

=====
Total params: 22,982,437
Trainable params: 17,559,493
Non-trainable params: 5,422,944



обучение классификатора



разморозка слоёв

Решение задачи через библиотеку TensorFlow

MobileNetV2

Model: "MobileNetV2_RMSprop_mse"

Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 299, 299, 3)]	0
sequential (Sequential)	(None, 299, 299, 3)	0
sequential_1 (Sequential)	(None, 299, 299, 3)	0
mobilenetv2_1.00_224 (Functional)	(None, 10, 10, 1280)	2257984
flatten_1 (Flatten)	(None, 128000)	0
dropout_1 (Dropout)	(None, 128000)	0
batch_normalization_95 (Batch Normalization)	(None, 128000)	512000
dense_1 (Dense)	(None, 5)	640005

=====
Total params: 3,409,989
Trainable params: 3,119,877
Non-trainable params: 290,112



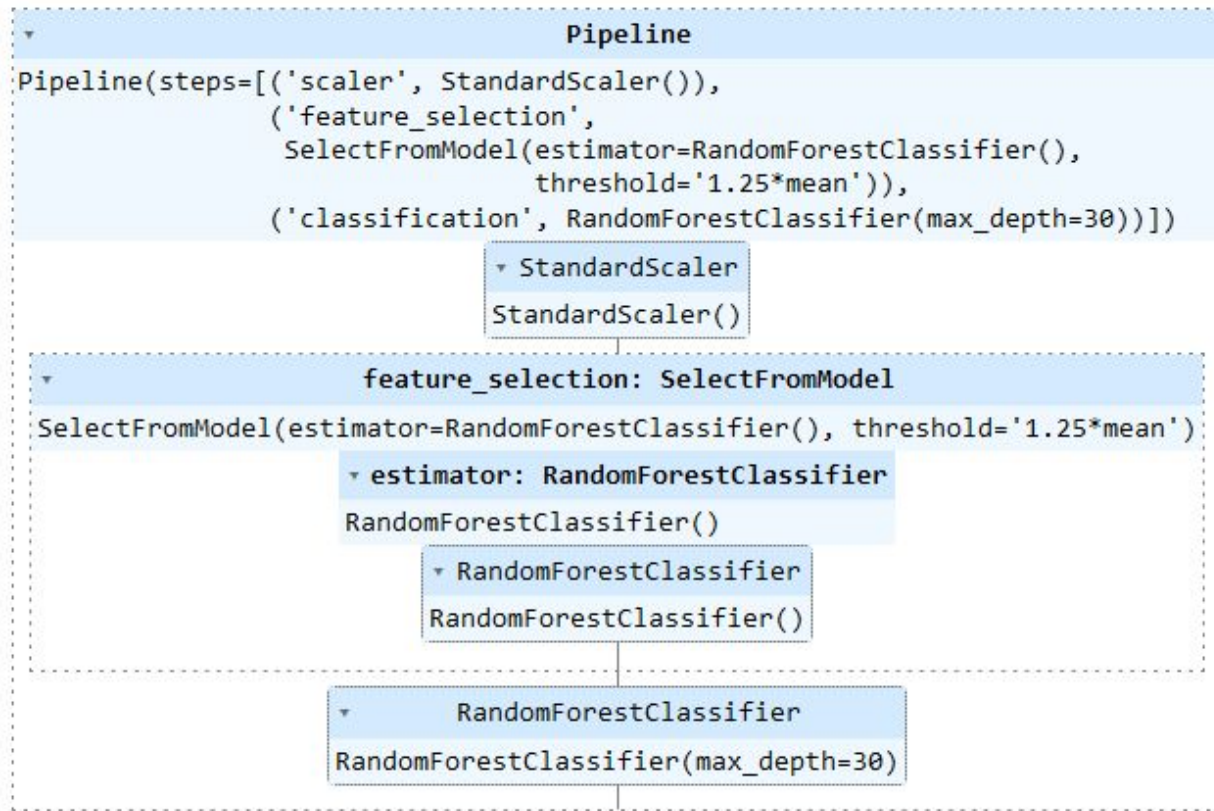
обучение классификатора



разморозка слоёв

Решение задачи через библиотеку Scikit-learn

RandomForestClassifier



Accuracy: 0.5131195335276968
Precision (macro): 0.5175410303981731
Recall (macro): 0.43463177171042344
F1-score (macro): 0.43779029300233124

Precision (micro): 0.5131195335276968
Recall (micro): 0.5131195335276968
F1-score (micro): 0.5131195335276968

Precision (None): [0.5 0.58181818 0.46153846 0.60204082 0.44230769]
Recall (None): [0.10810811 0.43243243 0.27272727 0.66292135 0.6969697]
F1-score (None): [0.17777778 0.49612403 0.34285714 0.63101604 0.54117647]

Решение задачи через библиотеку Scikit-learn

Модель Маркова

1. Разбивка данных на тренировочную и тестовую выборку;
2. Нормализация данных;
3. Подбор числа компонент для PCA;
4. Сокращение размерности;
5. Подбор компонент под модель Маркова
6. Создание модели Маркова

```
GaussianHMM
GaussianHMM(n_components=10)
```

```
Accuracy: 0.22448979591836735
Precision (macro): 0.14470588235294118
Recall (macro): 0.20266666666666667
F1-score (macro): 0.07827172827172826
```

```
Precision (micro): 0.22448979591836735
Recall (micro): 0.22448979591836735
F1-score (micro): 0.22448979591836735
```

```
Precision (None): [0.          0.5         0.          0.22352941 0.          ]
Recall (None): [0.          0.01333333 0.          1.          0.          ]
F1-score (None): [0.          0.02597403 0.          0.36538462 0.          ]
```


Лучшие показатели предсказания модели

Модель	F1 - Score
Resnet18	0.90
EfficientNet_b0	0.86
InceptionV3	0,90
MobileNetV2	0,93
RandomForestClassifier	0.51
Модель Маркова	0.22

Выводы

Нейросети:

1. Использование “переноса обучения” позволило достичь высоких результатов даже на небольшом датасете в объеме 1380 изображений с самыми простыми моделями классификатора. На всех моделях удалось достичь результата от 85% до 93%.
2. Все модели столкнулись с проблемой определения классов “гнев” и “страх”, что связано с несбалансированностью классов. Для повышения качества работы моделей, в первую очередь, необходимо поработать над улучшением самого датасета: сбалансировать классы и увеличить общее количество изображений.

Классические методы:

1. Обе модели показывают низкие показатели качества.
2. Модель “случайного леса” требует большой набор признаков, по которым можно производить классификацию.
3. Модель Маркова подходит для предсказания будущих событий, а не статичных фотографий.

Список использованных источников

Общее

1. Convolutional neural network // https://duchesnay.github.io/pystatsml/deep_learning/dl_cnn_cifar10_pytorch.html
2. Lakhani N.D. Statistical Evaluation Metrics // <https://iust-projects.ir/post/minidm01/>
3. AI, практический курс. Базовая модель распознавания эмоций на изображениях // <https://habr.com/ru/company/intel/blog/420635/>
4. CS231n: Свёрточные нейронные сети для распознавания образов // <https://habr.com/ru/post/456186/>
5. Правильная настройка случайного начального значения в экспериментах ML // <https://odsc.medium.com/properly-setting-the-random-seed-in-ml-experiments-not-as-simple-as-you-might-imagine-219969c84752>
6. Как рассчитать оценку f1 // <https://stackoverflow.com/questions/67959327/how-to-calculate-the-f1-score>

Модели

1. SOURCE CODE FOR TORCHVISION.MODELS.EFFICIENTNET // https://pytorch.org/vision/master/_modules/torchvision/models/efficientnet.html
2. RESNET50 документация модели // <https://pytorch.org/vision/stable/models/generated/torchvision.models.resnet50.html#torchvision.models.resnet50>
3. InceptionV3 // <https://keras.io/api/applications/inceptionv3/>
4. MobileNet // <https://keras.io/api/applications/mobilenet/>
5. Внедрение EfficientNet в PyTorch, часть 2: решение // <https://questu.ru/articles/155907/>

Pytorch

1. TorchEval документация // <https://github.com/pytorch/torcheval>
2. Передача обучения с использованием EfficientNet PyTorch // <https://debuggercafe.com/transfer-learning-using-efficientnet-pytorch/>
3. Документация STEPLR // https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html#torch.optim.lr_scheduler.StepLR
4. ОБУЧЕНИЕ КЛАССИФИКАТОРА // https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html#training-an-image-classifier
5. ПЕРЕДАЧА ОБУЧЕНИЯ ДЛЯ КОМПЬЮТЕРНОГО ЗРЕНИЯ УЧЕБНОЕ ПОСОБИЕ // https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
6. multiclass_f1_score документация // https://pytorch.org/torcheval/stable/generated/torcheval.metrics.functional.multiclass_f1_score.html#torcheval.metrics.functional.multiclass_f1_score
7. Классификация изображений с помощью PyTorch // <https://medium.com/@nutanbhogendrasharma/image-classification-with-pytorch-351a0a6cc09d>
8. Facial Expression Recognition using Pytorch // <https://www.kaggle.com/code/veb101/facial-expression-recognition-using-pytorch>
9. Как обучать свои собственные модели с помощью timm // <https://timm.fast.ai/training>
10. Классификация изображений с использованием логистической регрессии в PyTorch // <https://digitrain.ru/articles/369516/>
11. PyTorch — многоклассовая классификация // <https://towardsdatascience.com/pytorch-tabular-multiclass-classification-9f8211a123ab>
12. Измерение оценки F1 для многоклассовой классификации в PyTorch // <https://stackoverflow.com/questions/62265351/measuring-f1-score-for-multiclass-classification-natively-in-pytorch>
13. Матрица и точность тестирования для учебного пособия по передаче обучения PyTorch // <https://stackoverflow.com/questions/53290306/confusion-matrix-and-test-accuracy-for-pytorch-transfer-learning-tutorial>
14. Как использовать GradScaler в PyTorch // https://wandb.ai/wandb_fc/tips/reports/How-To-Use-GradScaler-in-PyTorch-VmldzoyMTY5MDA5

Список использованных источников

Ignite

1. <https://pytorch-ignite.ai/tutorials/beginner/01-getting-started/> - краткое руководство по запуску PyTorch-Ignite.
2. <https://uproger.com/obj-instrukciya-po-rabote-s-bibliotekoj-pytorch-igniteobj/> - Инструкция по Работе с Библиотекой PyTorch-Ignite.
3. <https://habr.com/ru/company/ods/blog/424781/> - Обучение и тестирование нейронных сетей на PyTorch с помощью Ignite.
4. <https://pytorch.org/ignite/metrics.html> - метрики PyTorch-Ignite.

Tensorflow

Курсы

1. Holbrook R. Intro to Deep Learning // <https://www.kaggle.com/learn/intro-to-deep-learning>
2. Moroney L. Device-based Models with TensorFlow Lite // <https://www.coursera.org/learn/device-based-models-tensorflow>
3. Tensorflow Guide. TensorFlow Basics // <https://www.tensorflow.org/guide>

Статьи

1. Data augmentation // https://www.tensorflow.org/tutorials/images/data_augmentation
2. Image classification // <https://www.tensorflow.org/tutorials/images/classification>
3. Load and preprocess images // https://www.tensorflow.org/tutorials/load_data/images
4. Preprocessing layers // https://keras.io/api/layers/preprocessing_layers/
5. Transfer learning and fine-tuning // https://www.tensorflow.org/tutorials/images/transfer_learning
6. Transfer learning and fine-tuning // https://keras.io/guides/transfer_learning/
7. Working with preprocessing layers // https://www.tensorflow.org/guide/keras/preprocessing_layers

Спасибо за внимание!