Тестовое задание Vision Labs

Ветошкин Лука

21 июня 2024 г.

1 Постановка задачи

Задание заключается в классификации изображений, а именно в определении, открыт или закрыт глаз, который находится на изображении. Проблемой является то, что представленный набор данных не имеет разметки. Требованием является получить EER(Equal error rate) меньше 0.05. В этом отчёте будет проведён анализ данных, рассмотрены и применены различные методы для решения этой задачи, а также представлены полученные результаты.

2 Анализ данных

Датасет состоит из 4000 неразмеченных одноканальных изображений размером 24x24 пикселей. На рисунке 1 показаны примеры изображений из датасета.

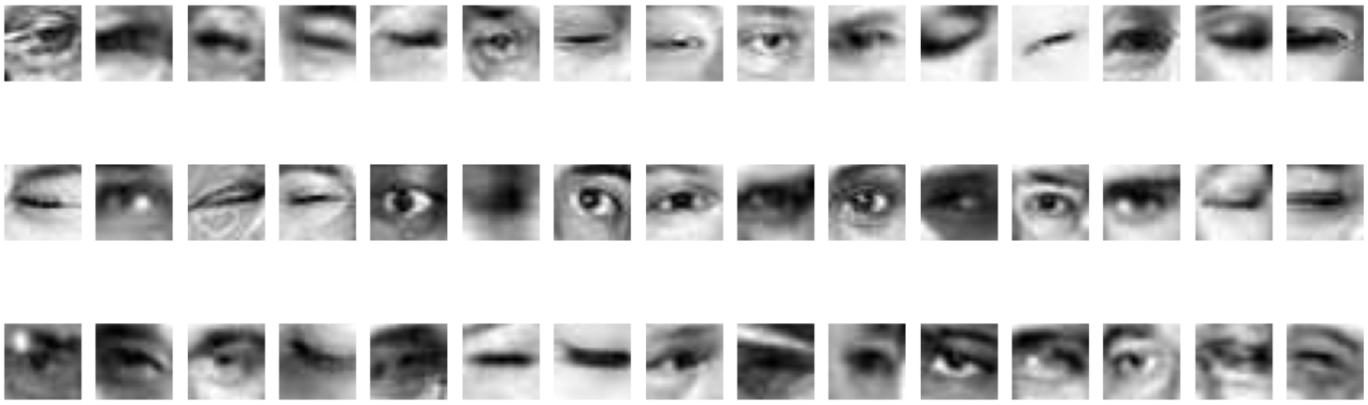

Рис. 1: Пример изображений из датасета

Посмотрим на распределение изображений с помощью алгоритма t-SNE в комбинации с PCA. Результат показан на рисунке 2. Как видно из визуализации сложно найти гиперплоскость, которая явно разделит данную выборку на две подвыборки, поэтому алгоритмы кластаризации будут работать не слишком точно.

Рис. 2: Распределение изображений с помощью t-SNE

3 Обзор методов

В рамках данного исследования были применены два подхода. Первый подход представляет собой semi-supervised метод, который включает в себя обучение векторного представления для изображений на всей имеющейся выборке, после чего на небольшой подвыборке обучается нейронная сеть для выполнения задачи классификации. Во втором подходе используется предварительно обученная нейронная сеть для полуавтоматической разметки исходного набора данных, который впоследствии используется для обучения модели классификации.

3.1 VAE

Обучение вариационного автокодировщика (VAE) относится к обучению без учителю. Архитектура VAE (рис. 3) состоит из кодировщика и декодирвщика. В нашем случае $\boldsymbol{x}$ это входные изображения. Кодировщик должен предсказывать параметры μ и σ Гауссового распределения. Латентное пространство формируется с помощью KL-дивергенции между распределениями $p(\mathbf{z})$ и $p_\theta(\mathbf{x}|\mathbf{z})$. Для обеспечения дифференцируемости $\mathbf{z}$ считается следующим образом:

$$\mathbf{z} = \mu + \sigma \cdot \epsilon, \epsilon \sim \mathcal{N}(0, I), \mathbf{z} \sim \mathcal{N}(\mu, \epsilon \cdot I) \tag{1}$$

Функция потерь считается по формуле 2:

$$\begin{aligned} L &= L_{rec} + \beta \cdot L_{KL} \\ L_{rec} &= \|\mathbf{x}_{gt} - \mathbf{x}_{pred}\|^2 \end{aligned} \tag{2}$$

$$L_{KL} = D_{KL}(\mathcal{N}(\mu, \sigma \cdot I)||\mathcal{N}(0, I)) = -\frac{1}{2}\sum_{i=1}^{N}(1 + \log \sigma_i^2 - \mu_i^2 - \sigma_i^2)$$

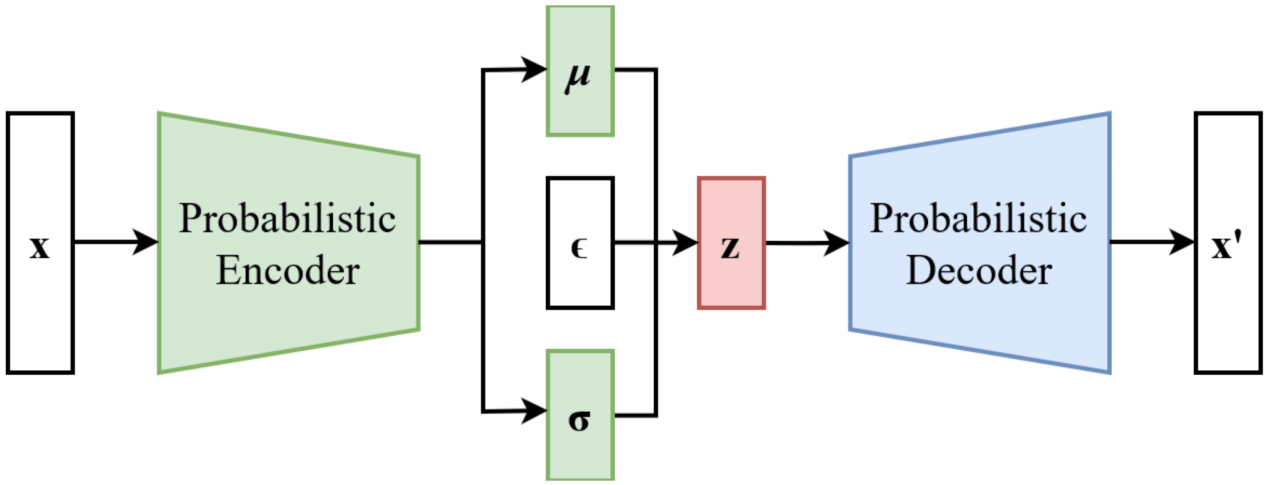

Рис. 3: Архитектура VAE

Обучаться вариационный автокодировщик будет на 3900 изображениях, на остальных 100 изображениях будет происходить дообучение для задачи классификации. Данные 100 изображений состоят из 50 изображений с открытым и 50 с закрытым глазом. Подробнее в разделе эксперименты.

3.2 Image-Text Matching

Для полуавтоматической разметки данных будем использовать Image-Text Matching с помощью модели BLIP-2. BLIP-2 (рисунок 4) является современной, экономичной архитектурой, в основе которой лежит Querying Transformer(Q-Former), который позволяет работать с предобученными языковой моделью (LLM) и Vision Transformer (ViT). С помощью данной модели будем подсчитывать вероятности связей между описаниями для закрытого и открытого глаза и самого изображения.

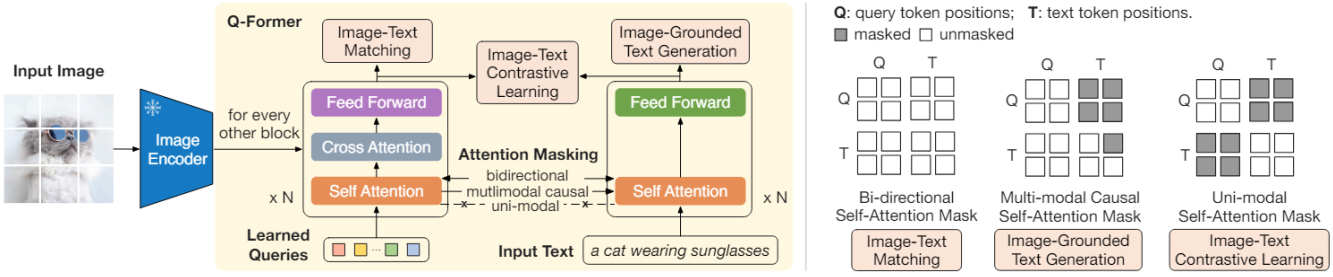

Рис. 4: Архитектура BLIP-2

4 Результаты

4.1 VAE

На рисунке 5 графики обучения VAE со следующими параметрами обучения: $batch\_size = 64$, $\beta = 0.001$, $latent\_dim = 64$, $learning\_rate = 0.01$, оптимизатор Adam:

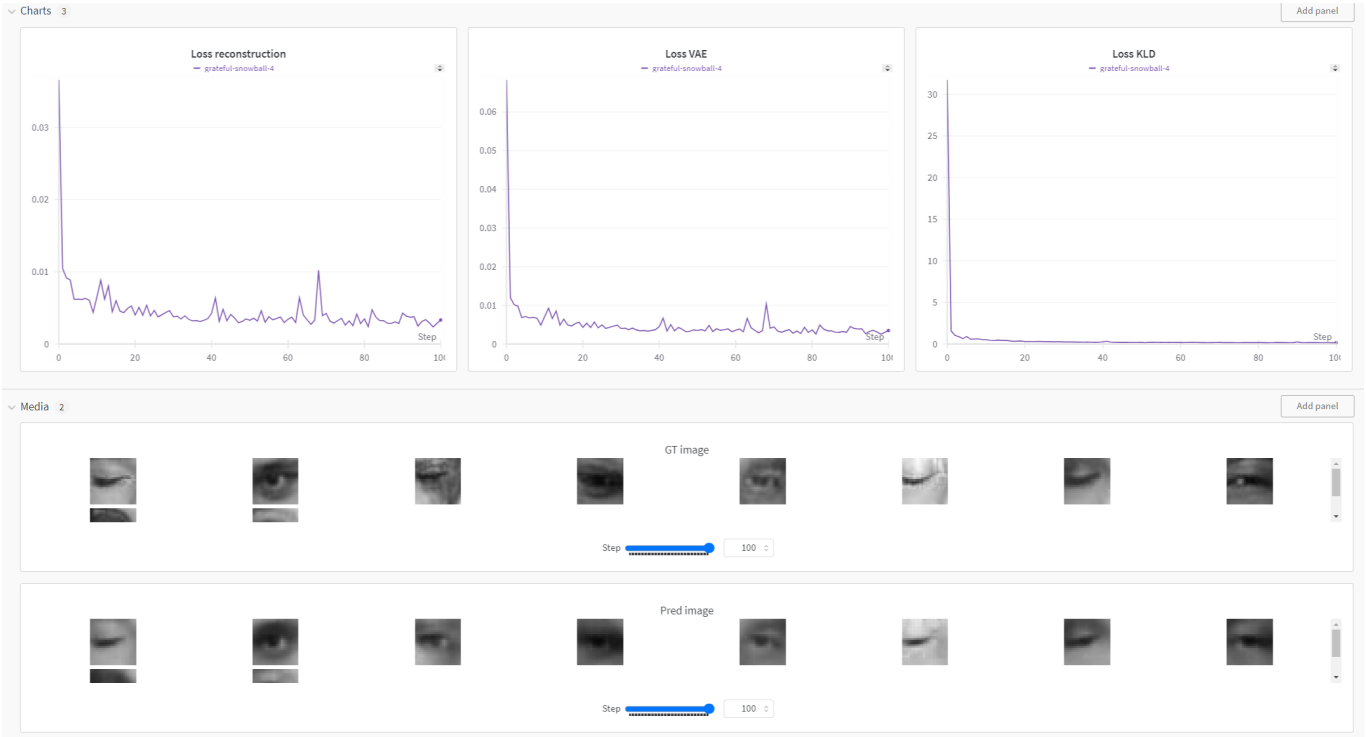

Рис. 5: График обучения VAE

После обучения VAE, используем дополнительно линейный слой для классификации (рис. 6). Обучение происходило для 70 изображений и тестировалось на 30 изображениях. Результаты представлены на рисунке (рис. 7). Как видно результат оказался относительно неплохой (Accuracy = 0.87), так как выборка обучения была небольшой, однако EER оказался равен 0.097, что не подходит по ТЗ.

```python
class EyeClassifier(nn.Module):
    def __init__(self, latent_dim, pretrained_vae=None):
        self.encoder = Encoder(latent_dim)
        self.freeze_backbone = pretrained_vae is not None
        if pretrained_vae is not None:
            try:
                state_dict = torch.load(pretrained_vae)
                state_dict = OrderedDict(((k[len("encoder."):], v)
                                          for k, v in state_dict.items()
                                          if "encoder." in k))
                self.encoder.load_state_dict(state_dict, strict=True)
                for param in self.encoder.parameters():
                    param.requires_grad = False
            except FileNotFoundError as e:
                print(e)
                print('Укажите путь к весам VAE!')
        self.encoder.eval()
        self.class_fc = nn.Linear(latent_dim, 1)


    def forward(self, x):
        with torch.no_grad():
            mu, log_var = self.encoder(x)

        logits = self.class_fc(mu)
        x = torch.sigmoid(logits)
        x = x.squeeze(-1)
        return x
```

Рис. 6: Код классификатора VAE

Рис. 7: Результат обучения VAE

4.2 Image-Text Matching

Для текствого описания открытого глаза будем использовать запрос «open eye or pupil» , для закрытого «closed eye». Проверим качество работы алгоритма для датасета размеченного ранеее для дообучения VAE. Тогда картинки на которых изображен открытый глаз данный алгоритм распознает с точностью 0.94, с закрытым глазом с точностью 0.88. Тогда будем использовать данный алгоритм для разметки всех изображений и затем вручную отбросим ошибочные результаты. После фильтрации изображений содержащих закрытый глаз осталось 1471 штук и изображений с открытым глазом 1719. Для обучения классификации будем использовать предобученную модель EfficientNet-b4, поверх которой используются два линейных слоя, функция активации ReLU, BatchNorm (рис.8).

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()

        self.model_ft = models.efficientnet_b4(weights="EfficientNet_B4_Weights.DEFAULT")
        self.linear1 = nn.Linear(1000,256)
        self.relu = nn.ReLU()
        self.batch_norm = nn.BatchNorm1d(256)
        self.out = nn.Linear(256,2)
    def forward(self, x):
        x = self.model_ft(x)
        x = self.linear1(x)
        x = self.batch_norm(x)
        x = self.relu(x)
        x = self.out(x)
        return x
```

Рис. 8: Код модели обученной на размеченном датасете с помощью BLIP-2

На рисунке 9 показаны результаты обучения данной нейронной сети с параметрами обучения: $batch\_size = 32$, $learning\_rate = 0.0001$, оптимизатор Adam. В данном случае удалось получить Accuracy = 0.9728 и EER = 0.0230, что удовлетворяет условиям ТЗ.

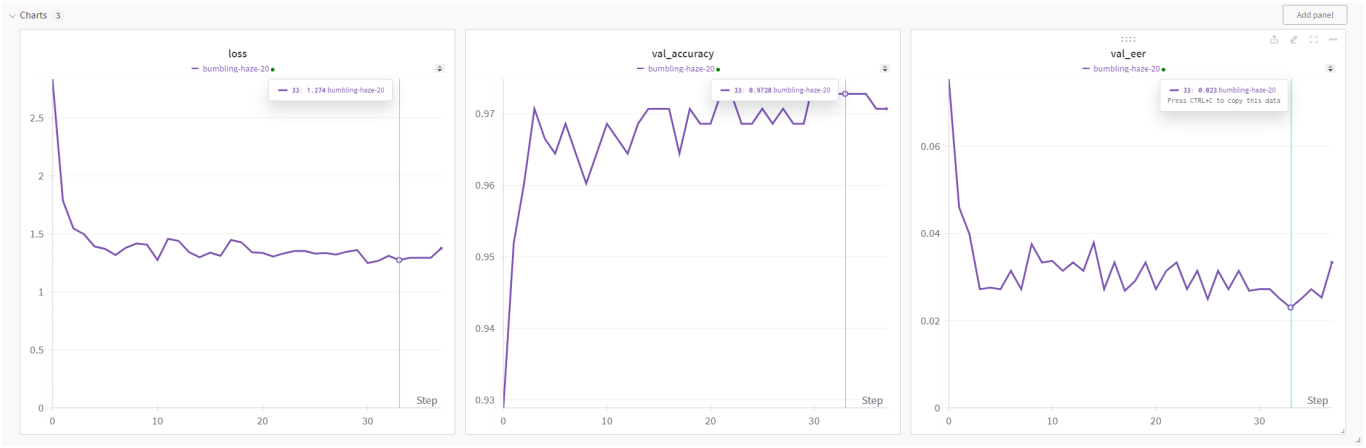

Рис. 9: Результат обучения модели на размеченном датасете с помощью BLIP-2

Также посмотрим на изображения, на которых ошибается данная нейронная сеть. На рисунке 10 показаны изображения, которые алгоритм распознал как закрытые глаза, хотя помечены они как открытые глаза. На рисунке 11 показана противоположная ситуация.

Рис. 10: Ошибка нейронной сети на изображениях с открытыми глазами

Рис. 11: Ошибка нейронной сети на изображениях с закрытыми глазами

Как видно данные изображения имеют плохое качество, что даже человеку будет проблематично ответить где изображен открытый глаз, а где закрытый, ведь может быть ошибка на самом деле в разметке, а не в работе алгоритма.

5 Вывод

В рамках данного исследования были проверены различные подходы к решению задачи бинарной классификации без разметки данных. Как не странно было получено лучшее качество для нейронной сети обученной на размеченном датасете. Однако нельзя сказать, что первый способ оказался нерабочим или плохим, данный алгоритм был обучен всего лишь на 2% данных и показал точность в 87%, данный алгоритм можно также улучшать, а именно поискать лучшее значение для размера латентного вектора, усложнить архитектуру кодировщика/декодеровщика и т.д., тогда качество может вырасти на 5-10%. Также стоит отметить хорошее качество алгоритма BLIP-2, потому что для изображений низкого разрешения данная модель имееет точность примерно 90%, что позволило быстро разметить датасет.