Распознавание Искусственных Лиц

Сюй Цзыи
xuziyiwork@gmail.com - by LaTeX

10 мая 2021 г.

Содержание

Аннотация

Распознавание изображений уже используется во многих отраслях, и распознавание лиц уже является очень зрелым. Наша задача состоит в том, чтобы определить подмножество человеческого лица. Дело в том, что есть много искусственных лиц, например: портреты человека, скульптуры, комиксы, и даже изображения лиц, полученные через искусственные сети (GAN - Generative Adversarial Networks). Надеемся, что мы сможем применить его к этой теме через сеть глубинных дефектов (DRN - Deep Residual Networks).

Задача состоит в том, чтобы отличить истинное лицо от Искусственных Лиц. Через сеть в конце концов достигли более 90% точности прогнозов.

1 Предисловие

Статья включает в себя постановку задачи, предложенное решение, математическую модель, вывод, списка источник, приложений.

Сначала, рассмотрим вопрос о том в каком направлении следует двигаться, с учетом того что лежит в основе проблемы ситуационного анализа. Потом, столкнувшись с проблемой, обсудим как обрабатывать данные, как конструировать нервную сеть, как регулировать параметры и как оценивать возможности модели. При проектировании модели и реализации кода мы столкнемся со многими проблемами, которые потом потребуют усилий, чтобы найти решение. Одним из важных шагов в этом направлении является математическое объяснение причин проблемы и процесс ее устранения. В конце статьи помечаем цитируемые статьи и произносим код как вложение.

2 Понимание проблем

Повторите вопрос: надеемся, что можно отделить набор данных, смешивающих настоящее лицо с искусственным лицом. Но как отделить правильное лицо из набора данных? Очевидно, что это проблема с классификацией, которая контролируется.

В настоящее время общепринятые алгоритмы классификации включают: классификация по k ближайшим соседям, алгоритм классификации Бейеса, SVM, статистические методы, случайные леса, нейронная сеть, свёрточная нейронная сеть. Свёрточная нервная сеть вместо полносвязной подключенной нервной сети стала популярным алгоритмом. Однако экспоненциальный взрыв и исчезновение требуют рассмотрения вопроса о том, как можно усовершенствовать нейронную сеть. Создать новую нейронную сеть, которая была бы более совершенной и менее проблематичной. Таким образом, ResNet и SeNet стали ответом на этот вопрос.

Мы проведем идентификацию поддельного лица через ResNet и SeNet. Подумаем над тем, как добиться большей точности и что ещё нужно делать после этого.

3 О подходе к решению

После отбора подходов к решению задачи необходимо решить следующие вопросы:

Во - первых, как будет осуществляться сбор данных. Во - вторых, как можно улучшить качество данных и как они обрабатываются. В - третьих, как выбрать мобильное обучение. В - четвертых, как проектировать сетевую структуру. В - пятых, как выбрать параметр. В - шестой, какие интерфейсы нужны.

Затем эти вопросы будут решены следующим образом.

3.1 Набор данных

Сначала на платформе, располагающей значительными ресурсами для сбора данных, мы можем найти необходимые наборы данных.

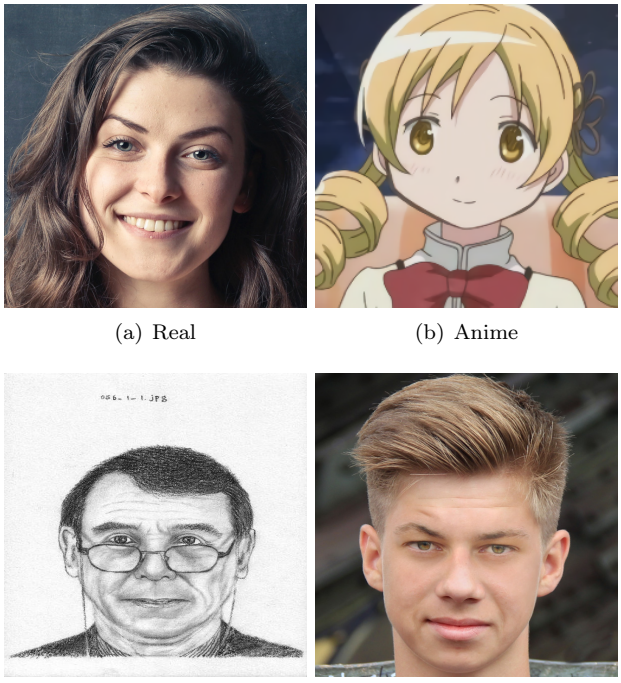
3.1.1 Платформа набора данных

Kaggle, ImageNet, UCI, Github, AWS, Datasets Search.

3.1.2 Набор данных

А. Действительные данные: 1269 + 3400

Б. Ложные данные: Мультяшные лица (self2anime 3400); Эскизные лица (CUFSF, CUFS 1800); Лица, созданные генеративно-состязательными сетями (Real and Fake face Detection 960)



(a) Real (b) Anime

(c) Sketch (d) GAN

Рис. 1: Datasets

3.2 Обработка данных

После сбора данных картина должна быть обработана в удобном для нас формате.

3.2.1 Анализ данных

По пропорции 8: 1: 1 делится на тренировочный набор, контрольное множество, наборы тестов.

Тренировочный набор подходит под параметры, контрольное множество проверяет степень оптимизации тестовой модели, набор тестов получает точность прогноза модели.

Нужно установить все картинки в едином размере для ввода. Потом будет случайный выбор фрагмента и отражение по горизонтали. Наконец, полученное изображение было преобразовано в тензорную форму и стандартизировано.

Сбросить размер изображения, вырезать центральный район, переключиться в тензорную форму, стандартизать.

3.2.2 Характеристический объект

А. Выбор характеристик: можно сделать целевой выбор путем изучения типов данных, которые будут извлекаться из характеристик высокого коэффициента корреляции и прогнозирования категорий.

Б. Обработка характеристик: уже реализовались при предварительной обработке.

В. Извлечения характеристик: в отличие от обычных машинных обучений, в свёртывающей нервной сети вид может осуществлять процесс свёртывания и бассейнообразования.

Дальнейшие обсуждения будут проведены в следующем разделе

3.3 Мобильное обучение

Учитывать размер набора данных, скорость вычисления Gpu, степень соответствия модели. здесь принята миграция обучения. Загрузить параметры и макет, которые уже подготовлены в глубинной сети дефектов, и продолжить работу по имитации лица.

А. Изучение на основе примеров переноса (instance - based transfer learning): часть данных (data) в исходном поле (source domain) может быть использована для обучения Таргет domain с помощью методов повторного поиска.

Б. Обучение на основе профилей (feature - representation transfer learning): обучение с помощью source domain (good) характеристики, кодирование знаний через их собственные формы и передача знаний от suorce domain к target domain, повышение эффективности задачи target domain.

В. Изучение на основе параметров переноса (parameter - transfer learning): target domain и source dominian задания делятся одинаковыми параметрами моделирования (model parameters) или подчиняются одному и тому же априорному распределению (prior distribution).

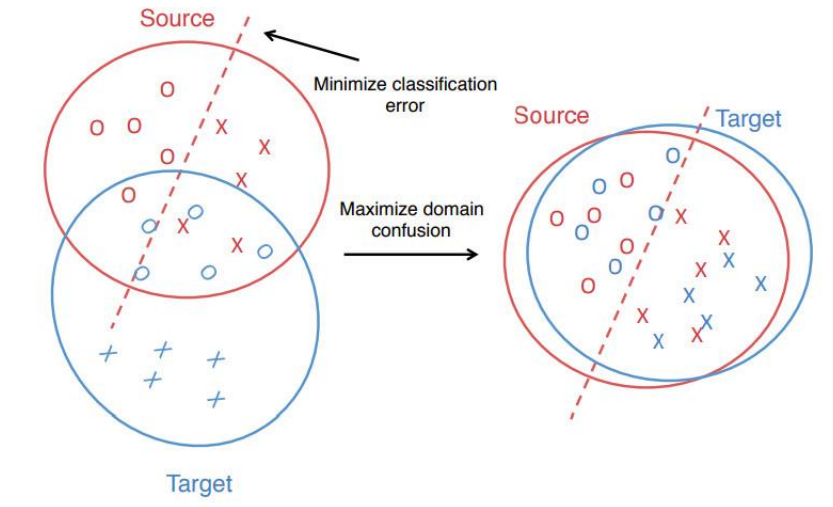


Рис. 2: Мобильное обучение

3.4 Глубокая сеть дефектов

При рассмотрении вопроса мы определили, что должны использовать глубокую сеть дефектов. В итоге были отобраны две модели ResNet и SENet. И кратко опишу эти две модели. А потом о строгой математической интерпретации.

3.4.1 ResNet

Хэ каймин и другие официально представили ResNet в документе в 2015 году. непосредственно добавляем ввод x к выходу, вывод y может быть четко разделен на H (x, wh) и x линейное наложение, что позволяет градиентам больше одного постоянного канала отображения, что считается очень важным для обучения в глубинных сетях. Большинство исследований по остаточным сетям сосредоточено в двух направлениях: во - первых, структурные исследования, и, во - вторых, изучение сетевых принципов.

3.4.2 SENet

А процесс SENet включает модуль Squeeze and Excitation (SE). Модуль SE сначала осуществлял операцию Squeeze для получения рисунков, получая глобальные характеристики уровня channel, затем осуществлял операции Excitation для изучения глобальных характеристик, изучал взаимоотношения между различными каналами и получал вес различных каналов, а затем умножал их на оригинальные характеристики, чтобы получить окончательный признак. В сущности, модуль SE выполняет функции attention или gating в измерении канала, что позволяет моделям уделять больше внимания характеристикам канала с наибольшим объемом информации, а не тем, которые не имеют значения для канала. Еще один момент заключается в том, что модуль SE является общим, что означает, что он может быть встроен в существующую сетевую архитектуру.

Дальнейшие обсуждения будут проведены в следующем разделе

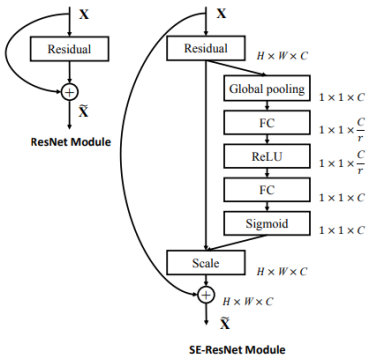


Рис. 3: Глубинная сеть дефектов ResNet и SENet

3.5 Параметры

Даем формат, в котором устанавливаются сети и параметры

3.5.1 Власти

А. Свёртывающее ядро и бассейнообразование являются ядром CNN, их параметры уже определены сетью.

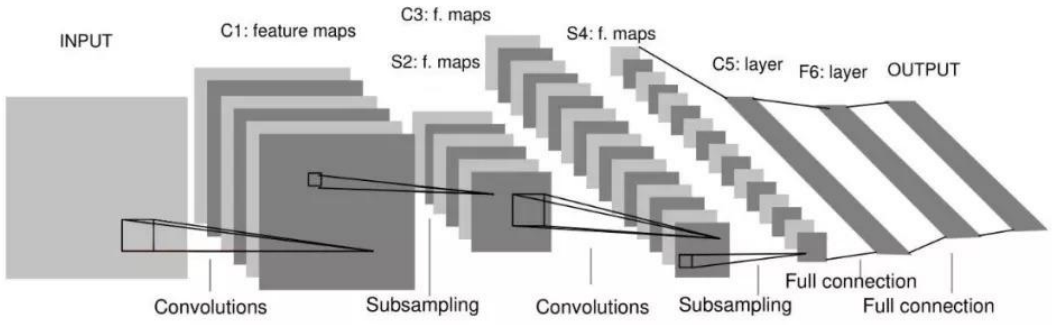


Рис. 4: Классический CNN

Б. Сплошной соединительный слой имеют значение веса w и смещения b.

В. Слоя BatchNorm имеет scale и параметр shift, которые также могут рассматриваться как вес и смещение.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Рис. 5: Слоя BatchNorm

3.5.2 Обратное распространение

А. Функция потери выбирается как функция перекрестной энтропии, и добавить регулярные элементы.

Б. О оптимизаторе заменить SGD Adam, и использовать автоматическое обновление скорости обучения.

В. Регулировать каждый раз обновление данных batch size.

3.6 Интерфейс

Неизвестный рисунок прогноза может вывести тип суждения по модели прогноза. Вычисление точности трех наборов данных top1 и top5 будет выполнен.

Создание модели для указанной категории находиться в стадии изготовления.

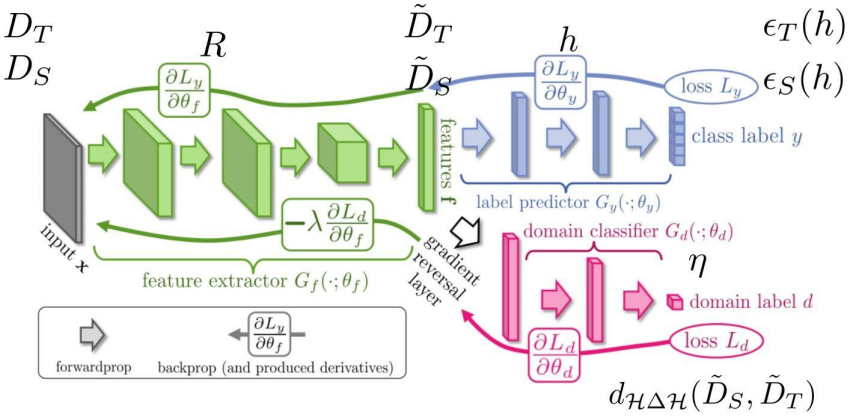


Рис. 6: GAN

4 Математическая модель

4.1 Тензор

Сначала вкратце представляем строго определённый тензор в математике. Затем показывает огромное преимущество тензора в глубокой учебе.

4.1.1 Определения

Тензором типа на векторном пространстве V (размерности n) называется объект, задаваемый в произвольнОм базисе $\{e_i\}$ набором чисел $T^{i_1 i_2 \dots i_s}_{j_1 j_2 \cdots j_r}$ (каждый из индексов может принимать значения от 1 до n), которые при переходе к другому базису $\{e'_i\}$ изменяются по следующему закону (применяется правило Эйнштейна):

$$T^{i'_1 i'_2 \dots}_{j'_1 j'_2 \dots j'_r} = T^{i_1 i_2 \dots i_s}_{j_1 j_2 \dots j_r} c^{i'_1}_{i_1} c^{i'_2}_{i_2} \dots c^{i'_s}_{i_s} c^{j_1}_{j'_1} c^{j_2}_{j'_2} \dots c^{j_r}_{j'_r} \tag{1}$$

Правило тензора низкой размерности:

$$\begin{array}{lll} Rule & f(x) & Scalar \\ Constant & c & 0 \\ MultiplicationRule & cf & c\frac{df}{dx} \\ PowerRule & x^n & nx^{n-1} \\ SumRule & f+g & \frac{df}{dx} + \frac{dg}{dx} \\ DifferenceRule & f-g & \frac{df}{dx} - \frac{dg}{dx} \\ ProductRule & fg & f\frac{dg}{dx} + \frac{df}{dx}g \\ ChainRule & f(g(x)) & \frac{df(u)}{du}\frac{du}{dx} \end{array} \tag{2}$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \dots \\ \nabla f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x} f_2(\mathbf{x}) \\ \dots \\ \frac{\partial}{\partial x} f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_1(\mathrm{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & \dots & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \dots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix} \tag{3}$$

При вычислении тензора можем имитировать прямое и обратное распространение в нервной сети.

4.1.2 Применение

Существует функция ввода и вывода, через которую можно получить функцию потери:

$$X = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_N]^T \tag{4}$$

$$\mathbf{y} = [target\,(\mathbf{x}_1)\,, target\,(\mathbf{x}_2)\,, \ldots, target\,(\mathbf{x}_N)]^T \tag{5}$$

$$C(\mathbf{w}, b, X, \mathbf{y}) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - activation\,(\mathbf{x}_i)\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \mathbf{x}_i + b\right)^2 \tag{6}$$

При обратном распространении и снижении градиента параметр можно обновить формулой тензора:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta\frac{\partial C}{\partial \mathrm{w}} \tag{7}$$

$$b_{t+1} = b_t - \eta\frac{\partial C}{\partial b} \tag{8}$$

Внимание:

А. При построении программы используется функция перекрестной энтропии, а не пример.

Б. Скорость обучения η будет автоматически обновляться, а не константа.

4.2 Извлечение характеристик

Тензор также может быть использован для извлечения характеристик. типичные методы обучения машин: PCA, LDA, SVD. Пример анализа главного компонента(Principal components analysis):

$$W^* = w^T \text{ W} = I_e argmin \left\| X - W^* W^T * X \right\|_F \tag{9}$$

$$argmax \left(w_1^T X X^T w_1 + \ldots + w_d^T X X^T w_{d'} \right) = \lambda_1 + \ldots + \lambda_{\vec{d}} \tag{10}$$

Использовать различные свёрточные ядра для получения различных характеристик.

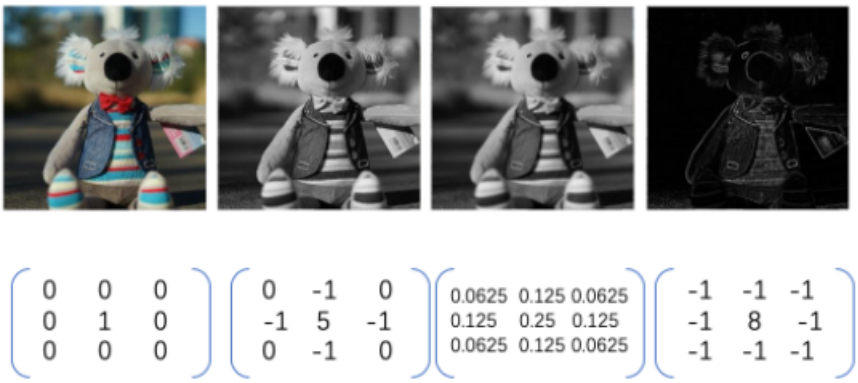


Рис. 7: Свёртывающее ядро

Обучение путём миграции может непосредственно получить хорошо обученную сеть моделей.

4.3 Глубинная сеть дефектов

Ниже дается математическое объяснение двух сетей.

4.3.1 ResNet

ResBlock может быть обозначен как:

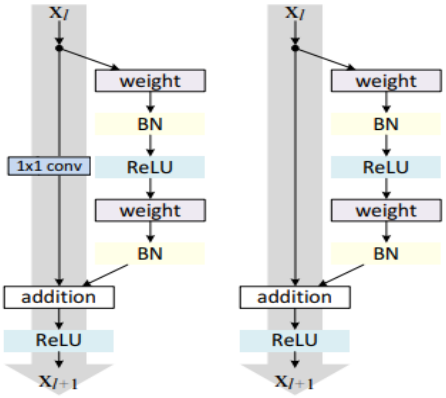
$$x_{l+1} = x_l + \mathcal{F}(x_l, W_l) \tag{11}$$

ResBlock на свёртке может быть обозначен как:

$$x_{l+1} = h(x_l) + \mathcal{F}(x_l, W_l) \tag{12}$$

ResBlock на многоэтажном может быть обозначен как:

$$x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, W_i) \tag{13}$$



(a) ResBlock1 (b) ResBlock2

Рис. 8: ResBlock

Процесс снижения градиента при обратном распространении произошли большие изменения.

$$\frac{\partial \varepsilon}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \varepsilon}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1}\mathcal{F}(x_i, W_i)\right) = \frac{\partial \varepsilon}{\partial x_L} + \frac{\partial \varepsilon}{\partial x_L}\frac{\partial}{\partial x_l}\sum_{i=l}^{L-1}\mathcal{F}(x_i, W_i) \tag{14}$$

4.3.2 SENet

Модуль Squeeze и Excitation (SE):
А. Операция Squeeze:

$$z_c = F_{sq}(u_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i,j), z \in R^C \tag{15}$$

Б. Операция Excitation:

$$s = F_{ex}(z, W) = \sigma(g(z, W)) = \sigma\left(W_2 ReLU\left(W_1 z\right)\right) \tag{16}$$

В. Наконец, чтобы узнать значение активации каждого канала (sigmoid значение от 0 до 1) умножить на оригинальные свойства на U:

$$\tilde{x}c = Fscale\left(u_c, s_c\right) = s_c \cdot u_c \tag{17}$$

5 Анализ

Будет проведено несколько контрольных испытаний с одной переменной. Цель: Во - первых, проверить наши мысли, Некоторые настройки действительно необходимы? например, некоторые наборы данных Anime SKetch, Мобильное обучение поможет ли нам сократить время и повысить точность; во - вторых, найти оптимальный параметр гиперпараметр, правилах аугментации, число слоев сети; В - третьих, найдите непредвиденные вопросы например, требуется ли больше данных, методов.

5.1 отдельная модель

Но сначала мы тренировали несколько моделей по разным наборам данных. после почти полного аугментации данных, цитируется хорошо разработанная модель Мобильного обучения, с часто используемым гиперпараметром, в самой обычной сети Res-Se-net, подготовить три модели Anime, Sketch, Ganfake. Затем исходя из точных показателей, можно судить о соотношении истинного и ложного лица.

Опыт показывает, что на аниме и рисунках точность тренировочных моделей приближается к 100%, в то время как фальсифицированные лица только на 50%.

5.1.1 Anime Sketch

Anime: Test Loss: 0.020 | Test Acc @1: 99.10%

Sketch: Test Loss: 0.000 | Test Acc @1: 100.00%

Можно доказать, что между карикатурами и картинами и реальным лицом особенно большие различия. особенно хорошие результаты можно получить за счет почти не требующейся подготовки. Тогда наша задача изменилась: как обучить более точную сеть, чтобы лучше предсказать GANfake и настоящего человека.

5.1.2 Ganfake

Ganfake: Test Loss: 0.634 | Test Acc @1: 56.18%

Необходимо повысить точность этой модели. В дополнение к перечисленным ниже параметрам и сетям, на самом деле, слишком мало данных также является причиной, мы удвоили объем данных для эксперимента.

Ganfake: Test Loss: 0.498 | Test Acc @1: 69.87%

Удвоили объем данных для эксперимента ещё раз.

Ganfake: Test Loss: 0.331 | Test Acc @1: 76.75%

Количество данных особенно сильно влияет на эксперимент, затем посмотрим, есть ли другие способы повышения точности.

5.2 правила аугментации

В тех случаях, когда другие условия были идентичны, было введено несколько групп контрольных экспериментов. Поскольку собранные данные уже обработаны, не так хорошо как ожидалось, но есть некоторые улучшения.

5.2.1 повороты

transforms.CenterCrop(224): Test Loss: 0.175 | Test Acc @1: 91.85%
transforms.RandomCrop(224): Test Loss: 0.159 | Test Acc @1: 92.10%
transforms.FiveCrop(224): Test Loss: 0.165 | Test Acc @1: 91.97%

5.2.2 масштабирование

None: Test Loss: 0.187 | Test Acc @1: 91.37%
transforms.RandomRotation(5): Test Loss: 0.148 | Test Acc @1: 93.08%

5.2.3 изменение цветовых характеристик

None: Test Loss: 0.211 | Test Acc @1: 86.87%
transforms.ColorJitter(brightness=0.5, contrast=0.5, saturation=0.5, hue=0.5):
Test Loss: 0.197 | Test Acc @1: 90.06%

5.3 Мобильное обучение

Провели ряд экспериментов, первый из которых привел к инициализации параметров, а второй - к загрузке модели предварительной подготовки открытого источника.

```
Epoch: 01 | Epoch Time: 9m 16s
        Train Loss: 0.672 | Train Acc @1:  72.12% | Train Acc @5:  87.71%
        Valid Loss: 1.306 | Valid Acc @1:  61.88% | Valid Acc @5:  74.87%
Epoch: 02 | Epoch Time: 8m 46s
        Train Loss: 0.310 | Train Acc @1:  84.04% | Train Acc @5:  97.61%
        Valid Loss: 1.166 | Valid Acc @1:  58.76% | Valid Acc @5:  82.68%
Epoch: 03 | Epoch Time: 8m 33s
        Train Loss: 0.278 | Train Acc @1:  84.77% | Train Acc @5:  98.56%
        Valid Loss: 0.294 | Valid Acc @1:  83.01% | Valid Acc @5:  98.75%
Epoch: 04 | Epoch Time: 8m 22s
        Train Loss: 0.230 | Train Acc @1:  86.28% | Train Acc @5:  99.04%
        Valid Loss: 0.245 | Valid Acc @1:  86.33% | Valid Acc @5:  98.69%
Epoch: 05 | Epoch Time: 8m 25s
        Train Loss: 0.226 | Train Acc @1:  86.04% | Train Acc @5:  99.33%
        Valid Loss: 0.243 | Valid Acc @1:  82.64% | Valid Acc @5:  99.38%
Epoch: 06 | Epoch Time: 8m 1s
        Train Loss: 0.210 | Train Acc @1:  87.13% | Train Acc @5:  99.56%
        Valid Loss: 0.201 | Valid Acc @1:  87.80% | Valid Acc @5: 100.00%
Epoch: 07 | Epoch Time: 7m 57s
        Train Loss: 0.198 | Train Acc @1:  87.73% | Train Acc @5:  99.85%
        Valid Loss: 0.195 | Valid Acc @1:  86.70% | Valid Acc @5: 100.00%
Epoch: 08 | Epoch Time: 7m 58s
        Train Loss: 0.192 | Train Acc @1:  88.43% | Train Acc @5:  99.92%
        Valid Loss: 0.208 | Valid Acc @1:  85.72% | Valid Acc @5:  99.84%
Epoch: 09 | Epoch Time: 7m 52s
        Train Loss: 0.186 | Train Acc @1:  88.32% | Train Acc @5:  99.95%
        Valid Loss: 0.192 | Valid Acc @1:  87.64% | Valid Acc @5: 100.00%
Epoch: 10 | Epoch Time: 7m 44s
        Train Loss: 0.181 | Train Acc @1:  89.29% | Train Acc @5:  99.97%
        Valid Loss: 0.193 | Valid Acc @1:  87.48% | Valid Acc @5: 100.00%
Test Loss: 0.185 | Test Acc @1:  89.19% | Test Acc @5: 100.00%
```

(a) False

```
Epoch: 01 | Epoch Time: 8m 6s
        Train Loss: 0.409 | Train Acc @1:  80.38% | Train Acc @5:  94.39%
        Valid Loss: 0.208 | Valid Acc @1:  87.64% | Valid Acc @5: 100.00%
Epoch: 02 | Epoch Time: 8m 2s
        Train Loss: 0.239 | Train Acc @1:  86.01% | Train Acc @5:  99.28%
        Valid Loss: 0.472 | Valid Acc @1:  85.10% | Valid Acc @5:  99.69%
Epoch: 03 | Epoch Time: 7m 53s
        Train Loss: 0.224 | Train Acc @1:  85.93% | Train Acc @5:  99.58%
        Valid Loss: 0.202 | Valid Acc @1:  87.33% | Valid Acc @5:  99.69%
Epoch: 04 | Epoch Time: 7m 51s
        Train Loss: 0.211 | Train Acc @1:  86.88% | Train Acc @5:  99.85%
        Valid Loss: 0.204 | Valid Acc @1:  86.86% | Valid Acc @5:  99.84%
Epoch: 05 | Epoch Time: 8m 36s
        Train Loss: 0.201 | Train Acc @1:  86.74% | Train Acc @5:  99.92%
        Valid Loss: 0.185 | Valid Acc @1:  88.68% | Valid Acc @5: 100.00%
Epoch: 06 | Epoch Time: 8m 16s
        Train Loss: 0.194 | Train Acc @1:  87.25% | Train Acc @5:  99.95%
        Valid Loss: 0.179 | Valid Acc @1:  89.10% | Valid Acc @5: 100.00%
Epoch: 07 | Epoch Time: 8m 25s
        Train Loss: 0.187 | Train Acc @1:  88.42% | Train Acc @5:  99.98%
        Valid Loss: 0.175 | Valid Acc @1:  90.10% | Valid Acc @5: 100.00%
Epoch: 08 | Epoch Time: 9m 17s
        Train Loss: 0.180 | Train Acc @1:  89.30% | Train Acc @5: 100.00%
        Valid Loss: 0.172 | Valid Acc @1:  90.41% | Valid Acc @5: 100.00%
Epoch: 09 | Epoch Time: 8m 6s
        Train Loss: 0.175 | Train Acc @1:  90.02% | Train Acc @5: 100.00%
        Valid Loss: 0.164 | Valid Acc @1:  91.09% | Valid Acc @5: 100.00%
Epoch: 10 | Epoch Time: 9m 21s
        Train Loss: 0.171 | Train Acc @1:  90.38% | Train Acc @5:  99.98%
        Valid Loss: 0.161 | Valid Acc @1:  91.19% | Valid Acc @5: 100.00%
Test Loss: 0.181 | Test Acc @1:  89.32% | Test Acc @5: 100.00%
```

(b) True

Рис. 9: Мобильное обучение

Хотя при предварительной тренировке параметры могут быстрее сходиться, но в конце, точность имеет лишь небольшое преимущество. Но думаю, что это имеет значение, может сократить время тренировки и предотвратить проблемы градиента.

5.4 Параметры

5.4.1 Learning rate

Установите очень малый коэффициент обучения, сначала подготовьте небольшой волны данных для тестирования: по мере увеличения коэффициента учебы, когда функция потери падает больше. проверили три раза, и результаты были остановлены между 0.001 и 0.01. Затем используя этот коэффициент обучения, мы получаем действительно удовлетворительные темпы обучения по этой модели.

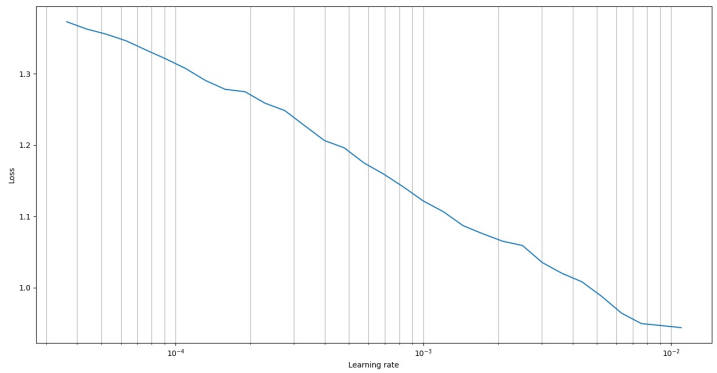


Рис. 10: Learning rate

Таким образом, эксперимент будет измеряться в качестве исходного значения, и на каждом этаже будет устанавливаться градиент, который в конечном счете вводится с помощью написанной функции автоматического изменения.

```
params = [
      {'params': model.conv1.parameters(), 'lr': FOUND_LR / 10},
      {'params': model.bn1.parameters(),    'lr': FOUND_LR / 10},
      {'params': model.layer1.parameters(), 'lr': FOUND_LR / 8},
      {'params': model.layer2.parameters(), 'lr': FOUND_LR / 6},
      {'params': model.layer3.parameters(), 'lr': FOUND_LR / 4},
      {'params': model.layer4.parameters(), 'lr': FOUND_LR / 2},
      {'params': model.fc.parameters()}
      ]
optimizer = optim.Adam(params, lr = FOUND_LR)

TOTAL_STEPS = EPOCHS * STEPS_PER_EPOCH
MAX_LRS = [p['lr'] for p in optimizer.param_groups]
scheduler = lr_scheduler.OneCycleLR(optimizer, max_lr = MAX_LRS, total_steps = TOTAL_STEPS)
```

Рис. 11: Learning rate

5.4.2 Optimizer

Эксперимент показал, что влияние на точность невелико, но Adam сходимость быстрее.

SGD: Test Loss: 0.315 | Test Acc @1: 83.64%

RMSprop: Test Loss: 0.313 | Test Acc @1: 83.38%

Adam: Test Loss: 0.266 | Test Acc @1: 83.50%

5.4.3 Loss function

Эксперимент показал, что функция потери перекрестной энтропии показает себя лучше.

MSELoss: Test Loss: 0.261 | Test Acc @1: 85.17%

CrossEntropyLoss: Test Loss: 0.249 | Test Acc @1: 85.85%

5.5 Визуализация

Анализируя полученные неточные фотографии, просто наблюдать почему неправильно оценил их.

Есть несколько выводов:

5.5.1 Переоборудование

Результаты обучения лучше результатов испытаний.

Epoch: 10 | Epoch Time: 1m 59s

Train Loss: 0.491 | Train Acc @1: 74.60% | Train Acc @5: 100.00%

Valid Loss: 0.445 | Valid Acc @1: 78.88% | Valid Acc @5: 100.00%

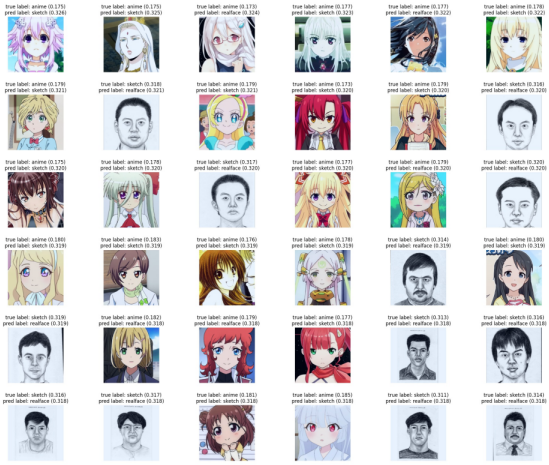
Test Loss: 0.701 | Test Acc @1: 63.51% | Test Acc @5: 100.00%

Обработанные снимки прогнозируют лучшие результаты. Когда отмена части предварительной обработки, точность тестирования будет повышена.

5.5.2 Пример ошибки

Следующая картина неверная классификация с вероятностью категории прогнозов и реальной вероятностью.

(a) Ganfake

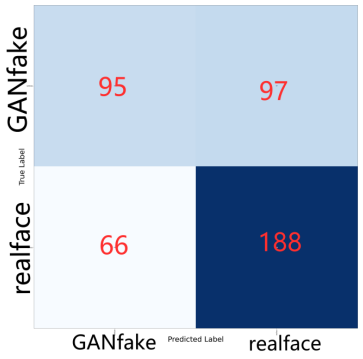


(b) Anime/sketch

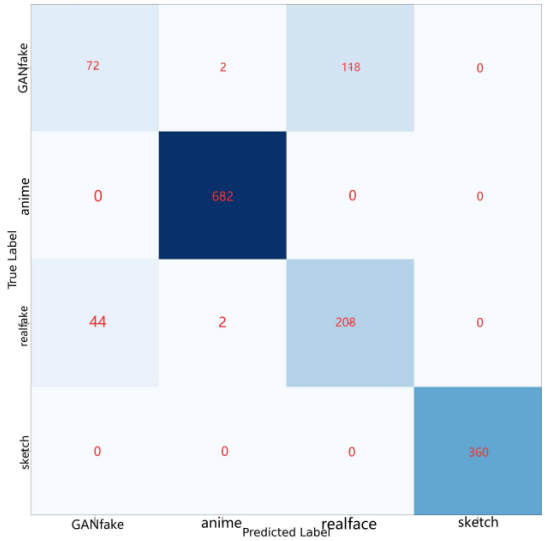
Рис. 12: ошибка-other

5.5.3 Распределение отображений

Ниже приводится статистическая классификация, абсцисса - реальное значение, а ордината - прогнозируемое значение.



(a) real/fake



(b) All

Рис. 13: Распределение отображений

5.5.4 Классификация снижения размерности

По методу pca классифицировать ситуацию на двухмерной плоскости.

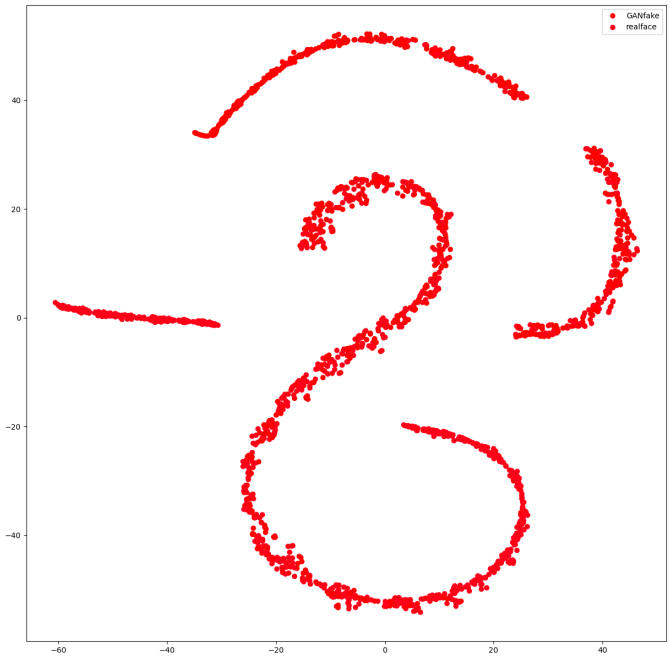
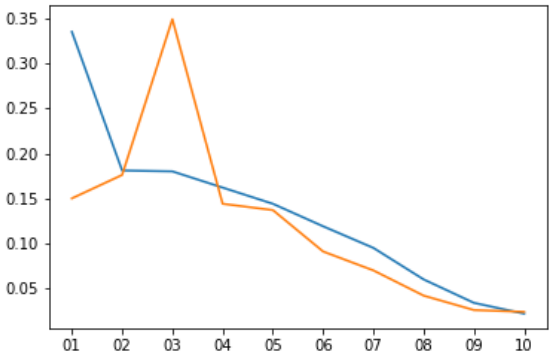
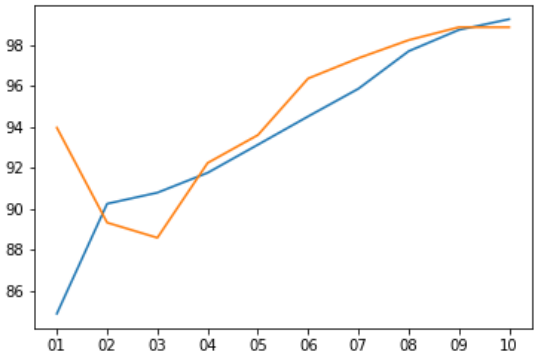


Рис. 14: PCA

5.5.5 Кривая точности



(a) Loss



(b) Acc

Рис. 15: Распределение отображений

6 Вывод

Работа по обработке данных и определению характеристик имеет очень важное значение. Существует простой ввод исходных данных в сеть, тогда функция будет сходиться через долгое время. Это требует особенно больших затрат и очень больших наборов данных. Но после того, как прошли через ряд обработки, в конце получим тензор будет лучше приспособлен к модели.

Мобильное обучение в больших данных, в более глубокой сети и в компьютерном визуальном применении вести себя очень хорошо. В будущем можно больше применить Мобильное обучение, чтобы заменить работу по инициализации параметров.

Глубинная сеть дефектов можно решить вопрос, что хотя есть очень глубокая сеть, но не очень высокая точность. На этой основе преимущество заключается в том, что другие операции могут иметь больше возможностей.

7 Список источников

Pytorch:https://pytorch.org/docs/stable/index.html
Tensorflow:https://www.tensorflow.org/api_docs/python/tf
Kaggle:https://www.kaggle.com/
Imagenet:http://www.image-net.org/
Tianchi:https://tianchi.aliyun.com/competition/gameList/activeList

8 Приложений

8.1 Точность тренировки

Pytorch: (resnet50 93%, senet50 94%)
Tensorflow: (resnet50 46%, senet50 88%)

8.2 Набор данных

https://drive.google.com/file/d/1-5djvNp4hff3rlQWaiq8CC9278dTUQPQ/view?usp=sharing

8.3 Код

Pytorch: https://drive.google.com/file/d/1PFrB9AIT-KrBNOKoBj4SuoO1yFRT149U/view?usp=sharing
Tensorflow: https://drive.google.com/file/d/1g4huMqAfzoZ8_So4zRWrfRSP5lJI8e8_/view?usp=sharing