Содержание

1 Введение

Производственная практика пройдена в Институте проблем управления им. В. А. Трапезникова Российской академии наук (ИПУ РАН).

Целью прохождения производственной практики является закрепление и развитие профессиональных компетенций научно-исследовательской и проектной деятельности.

Для достижения поставленной цели потребовалось решить следующие задачи практики (в соответствии с программой практики):

- Закрепление и расширение теоретических и практических знаний, полученных студентом в процессе обучения.
- Получение навыков самостоятельной работы, а также работы в составе научно-исследовательских коллективов.
- Работа над проектом по созданию детектора наличия медицинской маски на человеке.
- Обработка полученных материалов и оформление отчета о прохождении практики.

В ходе прохождения производственной практики для выполнения задания языком программирования был выбран python3, для обработки изображений применялась библиотека fastai, графический интерфейс был создан при помощи фреймворка tkinter.

2 Содержательная часть

2.1 Описание профессиональных задач студента

Содержание практики (вопросы, подлежащие изучению):

1. Прохождение инструктажа по технике безопасности на предприятии.
2. Исследование текущего состояния систем проверки наличия медицинской маски на человеке.
3. Подготовка дата-сета для обучения нейронной сети для проверки наличия медицинской маски на человеке.
4. Разработка архитектуры нейронной сети для проверки наличия медицинской маски на человеке.
5. Разработка детектора наличия медицинской маски на человеке.
6. Обучение нейронной сети для проверки наличия медицинской маски на человеке.
7. Сбор, обобщение и анализ полученных в ходе производственной практики материалов и подготовка отчета по практике.

В последнее время проблема определения наличия медицинской маски на лице человека стала особо актуальной. Особенно, после введения всевозможных мер, требующих от предприятий и организаций повышенной внимательности и контроля за соблюдением санэпидемиологических требований.

Так как тема новая, аналогов данного детектора не много. Удалось найти лишь одно готовое решение данной проблемы, это CVizi Fisher: Masks. Компания занимается видеоаналитикой для наблюдения и контроля появления людей в определенных зонах с определенными условиями. Несомненным достоинством данного решения является простота использования, но есть и недостатки. Первый недостаток в том, что эта услуга платная и, помимо единоразовой платы за оборудование, придется платить и за абонентскую плату, 15 рублей в день за каждую камеру. Второй недостаток в том, что все это решение в «коробке», и у пользователя нет возможности как-то добавить или изменить функционал, Кроме этого, нигде не упоминается о качестве модели, используемой в приложении.

Для обучения модели необходим размеченный дата-сет с изображениями.

2.1.1 Принцип работы сверточных нейронных сетей

Для обработки видео и изображений лучше всего подходят сверточные нейронные сети, так как они лучше справляются с однородными данными изображения.

Основой сверточных нейронных сетей являются сверточные слои. Им на вход подается массив матриц с данными. В случае цветного изображения на вход первого сверточного слоя попадают 3 матрицы, соответствующие 3 цветовым каналам RGB, размеры этих матриц соответствуют размерам исходного изображения. Слой свертки включает для каждого канала свой фильтр, ядро свертки которого обрабатывает предыдущий слой по фрагментам, суммируя результаты поэлементного произведения каждого фрагмента. Ядро свертки представляет из себя небольшую матрицу обычно размером 3×3, числовые коэффициенты внутри матрицы подбираются в процессе обучения. Пример работы продемонстрирован на рис. 1.

Далее, скалярный результат каждой свертки попадает в функцию активации. Функция активации может быть разной, главное, чтобы она обладала свойством нелинейности. Пример такой функции – ReLU, вычисляется на как $f(x) = max(0,x)$.

Слой пулинга представляет собой нелинейное уплотнение карты признаков, при этом группа пикселей (обычно размера 2×2) уплотняется до одного пикселя, проходя нелинейное преобразование. Наиболее употребительна при этом функция максимума. Преобразования затрагивают непересекающиеся прямоугольники или квадраты, каждый из которых ужимается в один пиксель, при этом выбирается пиксель. Операция пулинга позволяет существенно уменьшить пространственный объём изображения. Пример работы продемонстрирован на рис. 2.

После нескольких прохождений свёртки изображения и уплотнения с помощью пулинга система перестраивается от конкретной сетки пикселей с высоким разрешением к более абстрактным картам признаков. После сверточных слоев остается большое число каналов, хранящих небольшое число данных, которые воспринимаются как различные очертания, выявленные из исходного изображения. Эти данные объединяются и передаются в обычную полносвязную сеть.

Наиболее популярны способом обучения является метод обучения с учителем – метод обратного распространения ошибки.

2.1.2 Архитектуры сверточных нейронных сетей

Наиболее популярным архитектурами нейронных сетей для обработки изображений на данный момент являются:

- AlexNet
- VGGNet
- GoogleNet
- ResNet

AlexNet – прорывная для своего времени архитектура сверточной нейронной сети. AlexNet была первой свёрточной нейросетью, выигравшей соревнование по классификации ImageNet в 2012 году с результатом 16.4%. Её архитектура состоит из пяти сверточных слоёв, между которыми располагаются pooling-слои и слои нормализации, а завершают нейросеть три полносвязных слоя.

В 2014 году VGGNet достигла более чем в два раза лучшего результата по сравнению с AlexNet. Основная идея VGG-архитектур — использование большего числа слоёв с фильтрами меньшего размера.

GoogleNet — ещё более глубокая архитектура с 22 слоями. Целью Google было разработать нейросеть с наибольшей вычислительной эффективностью. Для этого они придумали так называемый модуль Inception — вся архитектура состоит из множества таких модулей, следующих друг за другом. Идея основного модуля Inception заключается в том, что он сам по себе является небольшой локальной сетью. Вся его работа состоит в параллельном применении нескольких фильтров на исходное изображение. Данные фильтров объединяются, и создаётся выходной сигнал, который переходит на следующий слой. Пример архитектуры GoogleNet рис. 3.

В 2015 году ResNet произвела настоящую революцию глубины нейросетей. Она состояла из 152 слоёв и снизила процент ошибок до 3,57% в соревновании классификации ImageNet. Это сделало её почти в два раза эффективнее GoogleNet.

Обычная нейростеть при сильном увеличении количества слоев показывает результат хуже как при обучении, так и при тестировании. Создатели ResNet предположили, что загвоздка кроется в оптимизации — более глубокие модели гораздо хуже поддаются настройке. Тогда они решили не складывать слои друг на друга для изучения отображения нужной функции напрямую, а использовать остаточные блоки, которые пытаются «подогнать» это отображение. Нейросеть «перепрыгивает»

через некоторые слои. Они больше не содержат признаков и используются для нахождения остаточной функции $H(x) = F(x) + x$ вместо того, чтобы искать $H(x)$ напрямую. Нейросеть состоит из большого стека одинаковых остаточных блоков, каждый из которых имеет два свёрточных слоя 3×3. Периодически число фильтров удваивается, а их размерность уменьшается с шагом 2 (/ 2 в каждом измерении). В самом начале архитектуры присутствует дополнительный свёрточный слой. Также у ResNet нет полносвязных слоёв в конце — используется только один слой с выходными классами. С увеличением числа слоёв для уменьшения размерности изображения применяются точно такие же дополнительные слои, как и в GoogleNet. В результате экспериментов с ResNet выяснилось, что очень глубокие сети действительно можно обучить без ухудшения точности. Нейросеть достигла наименьшей ошибки в задачах классификации, которая превзошла даже человеческий результат. Пример архитектуры ResNet рис. 4.

Результаты популярных архитектур на соревновании ImageNet представлены на рис. 5.

2.1.3 Обучение нейросети

Обучить нейросеть можно двумя способами.

Для первого способа обучения нейронной сети необходимо задать все коэффициенты случайными значениями, а потом подбирать при помощи обратного распространения ошибки. Данный способ требует много времени и вычислительных мощностей. Кроме этого, необходим очень большой размеченный датасет для обучения.

Второй способ называется fine-tuning (тонкая настройка). Для него нужны куда меньшие вычислительные мощности и меньший размер датасета, при этом можно достичь лучшего качества. Суть способа заключается в том, чтобы взять уже готовую, предобученную модель и дообучить ее на наших данных. У модели будут уже подобранные коэффициенты для извлечения нужных очертаний объектов. Остается лишь дообучить модель на новом промаркированном датасете.

2.2 Описание выполнения пунктов

2.2.1 Обучение нейросети

Для того, чтобы приступить к обучению нейросети, необходим датасет с размеченными изображениями.

В интернете был найден репозиторий, в котором решается похожая задача по распознаванию маски на лице человека. В репозитории есть csv файл с промаркированными ссылками на изображения.

Для обучения модели была использована библиотека fast.ai. Библиотека fast.ai упрощает обучение качественных нейронных сетей с использованием современных передовых практик. Она основана на исследованиях в области передового опыта глубинного обучения, включая поддержку «из коробки» для моделей обработки изображений, текста и коллаборативной фильтрации.

В качестве архитектуры нейронной сети выбрана архитектура ResNet, так как она показала лучшие результаты в соревновании классификации ImageNet. В библиотеке fast.ai для архитектуры ResNet есть обученные модели с разным числом слоев: 18, 34, 50, 101 и 152. Детальная архитектура представлена на рис. 6.

Были протестированы все варианты нейронной сети. Нейросеть из 18 слоев занимает меньше всего места и работает быстрее остальных, но качество уступает моделям с большим числом слоев. Качество остальных 4 моделей примерно одинаковое, при этом скорость обработки изображения падает с увеличением числа слоев. Была использована модель с 34 слоями, так как она оптимальна с точки зрения качества и времени обработки изображения. Матрицы ошибок для протестированных моделей приведены на рис. 7 – 11.

2.2.2 Создание демонстрационного варианта

Для демонстрации работы модели было создана программа на Python, которая обрабатывает видео с веб камеры и выводит его и вероятность нахождения маски на изображении на экран. Демонстрационный вариант работает следующим образом: Кадр с веб камеры берется при помощи библиотеки openCV, и хранится в памяти в качестве трехмерного тензора. Далее этот тензор оправляется в обученную модель, которая на выход дает вероятность нахождения маски в кадре. Для исключения выбросов итоговый результат берется как среднее последних 3 измерений. В завершении итоговый результат и кадр обновляются в графическом окне.

Графическая часть реализована при помощи библиотеки tkinter. Итоговый вид окна представлен на рис. 12.

2.2.3 Тестирование

На рис. 12 – 14 представлены скриншоты демонстрационного окна. На рис. 12 и рис. 13 все просто, где нет маски вероятность близка к нулю, где маска хорошо надета вероятность 100%. На рис. 14 маска надета не полностью и нейросеть выдает уже меньший результат.

2.2.4 Дальнейшее развитие

Данная тема имеет много возможностей для развития.

Первое, что необходимо сделать, это добавить модуль для решения задач нахождения объектов (object detection). Лицо человека в маске или без необходимо сначала выделить и после этого определять наличие маски на выделенном участке изображения. Данное дополнение может сильно улучшить качество работы детектора масок.

Как дополнение можно сделать детектор и других средств индивидуальной защиты, например, перчаток. Также можно определять качество защитной маски.

3 Заключение

В ходе практики, был изучен принцип работы сверточной нейронной сети, рассмотрены самые популярные архитектуры сверточных нейронных сетей и подходы к их обучению.

Были приобретены навыки по поиску необходимого датасета, по работе со сверточными нейронными сетями, их дообучению и применению в реальном приложении. Помимо этого, были получены навыки по работе с графическим интерфейсом в python.

По окончанию практики была достигнута главная цель - применение теоретических знаний, полученных в процессе обучения, при решении реальных задач.

А также приобретены навыки и опыт практической работы. Данная практика является хорошим практическим опытом для дальнейшей самостоятельной деятельности.

4 Приложения

С кодом можно ознакомиться по ссылке: https://github.com/andrsolo21/Mask_detection.

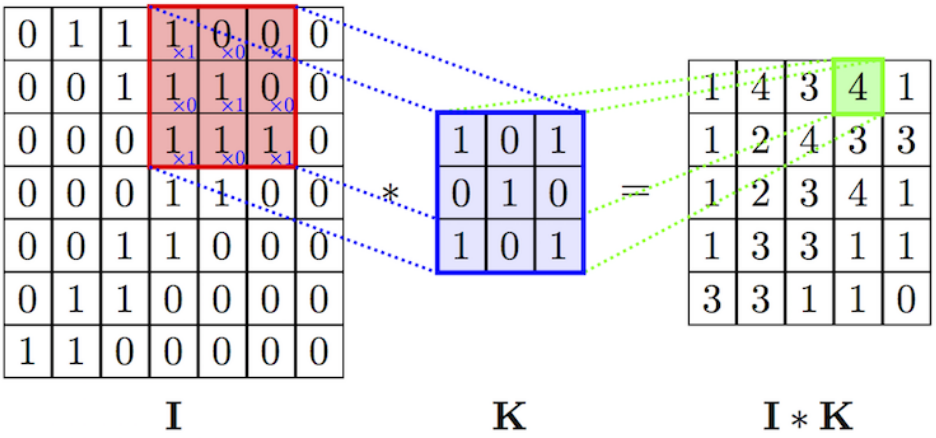

Рис. 1: Пример работы свертки

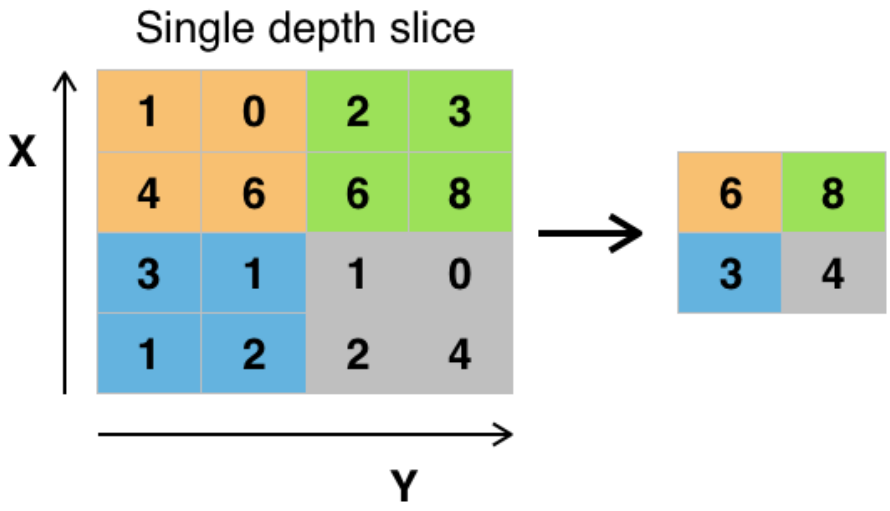

Рис. 2: Пример работы пулинг слоя

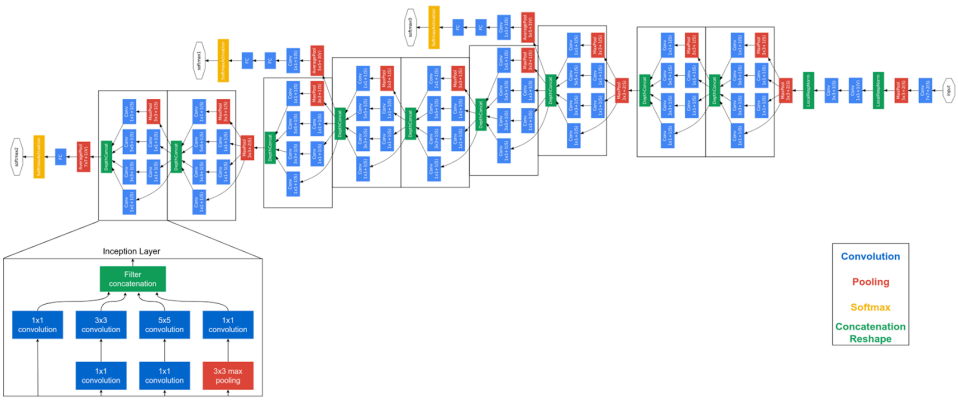

Рис. 3: Архитектура GoogleNet

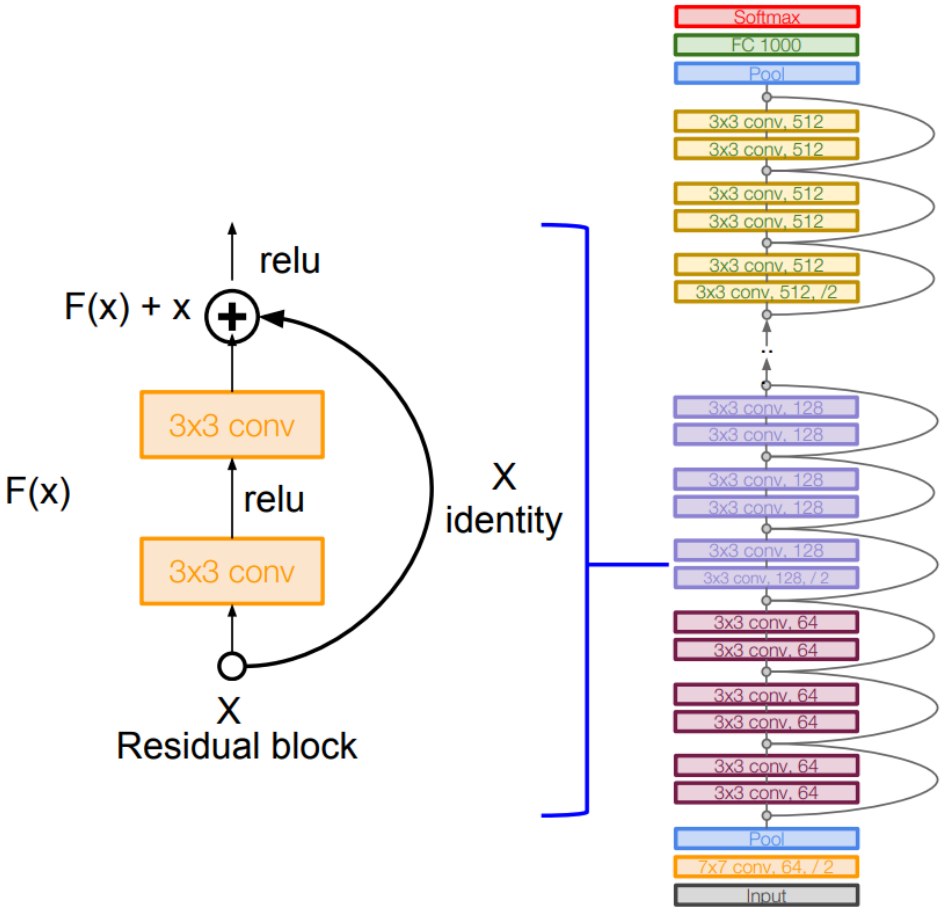

Рис. 4: Пример архитектуры ResNet

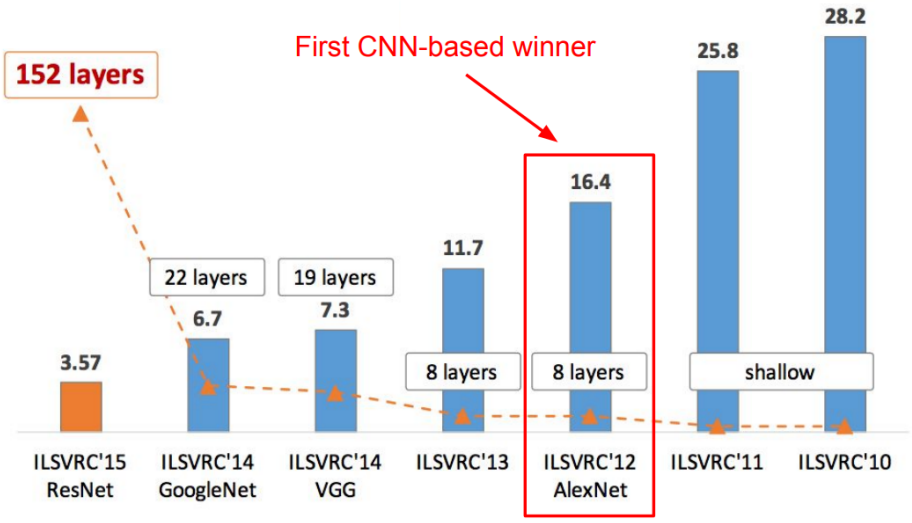

Рис. 5: Результаты популярных архитектур на соревновании ImageNet

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рис. 6: Архитектуры сетей ResNet

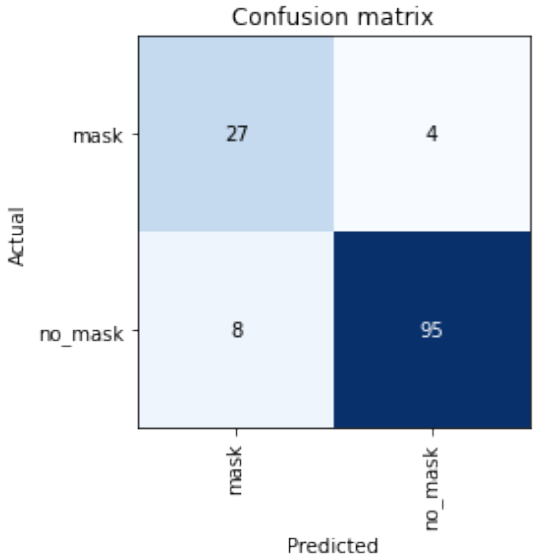

Рис. 7: Матрица ошибок для resnet18

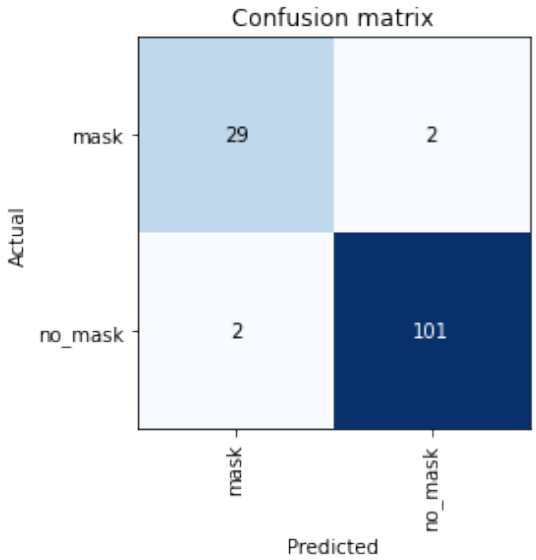

Рис. 8: Матрица ошибок для resnet34

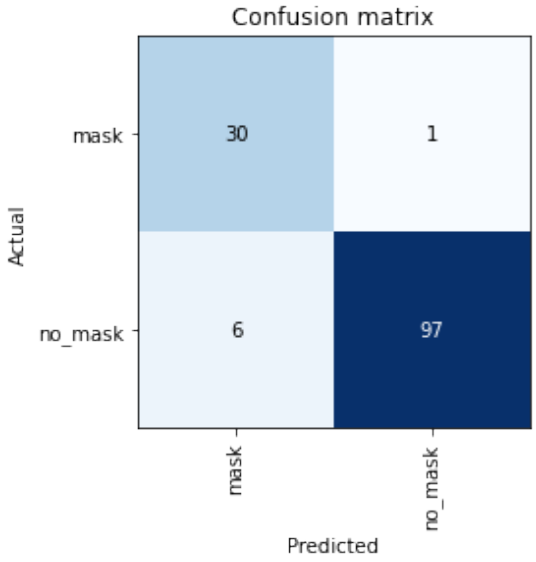

Рис. 9: Матрица ошибок для resnet50

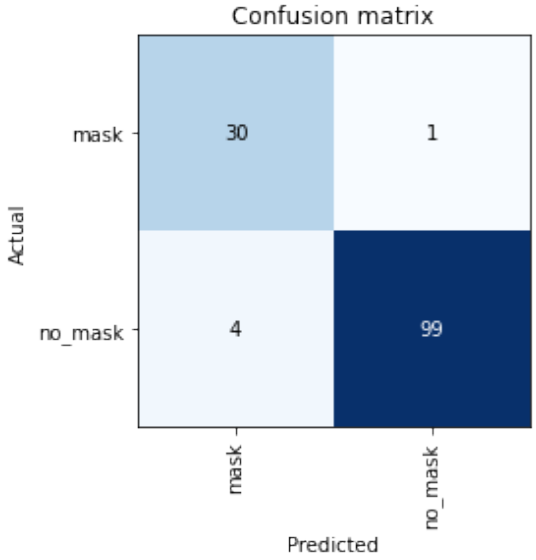

Рис. 10: Матрица ошибок для resnet101

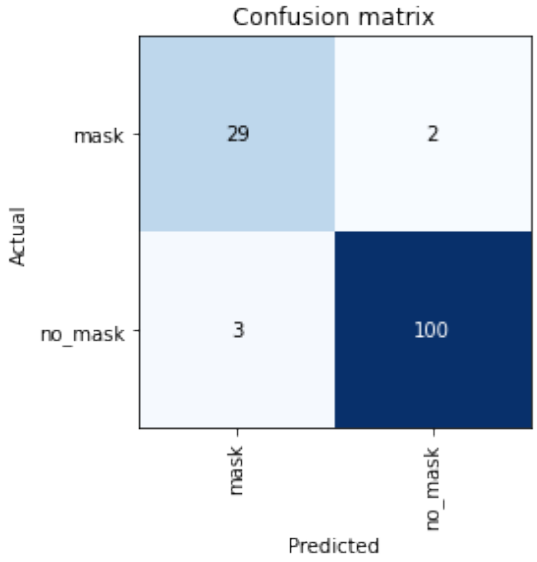

Рис. 11: Матрица ошибок для resnet152

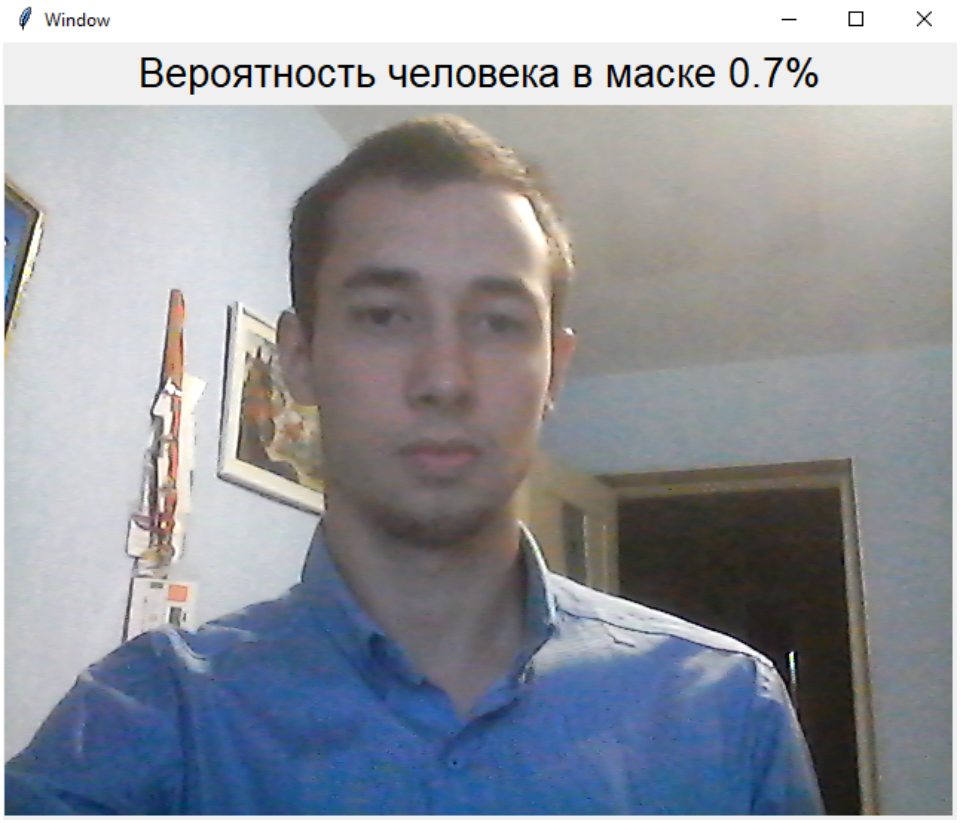

Рис. 12: Тестирование без маски

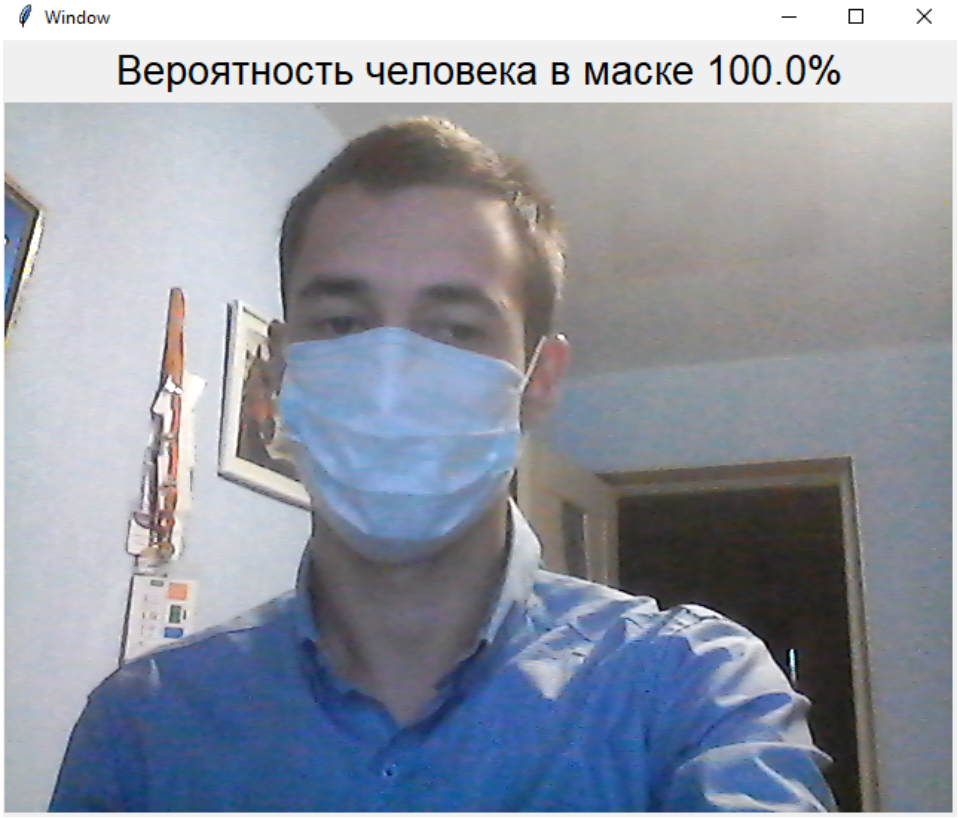

Рис. 13: Тестирование в маске

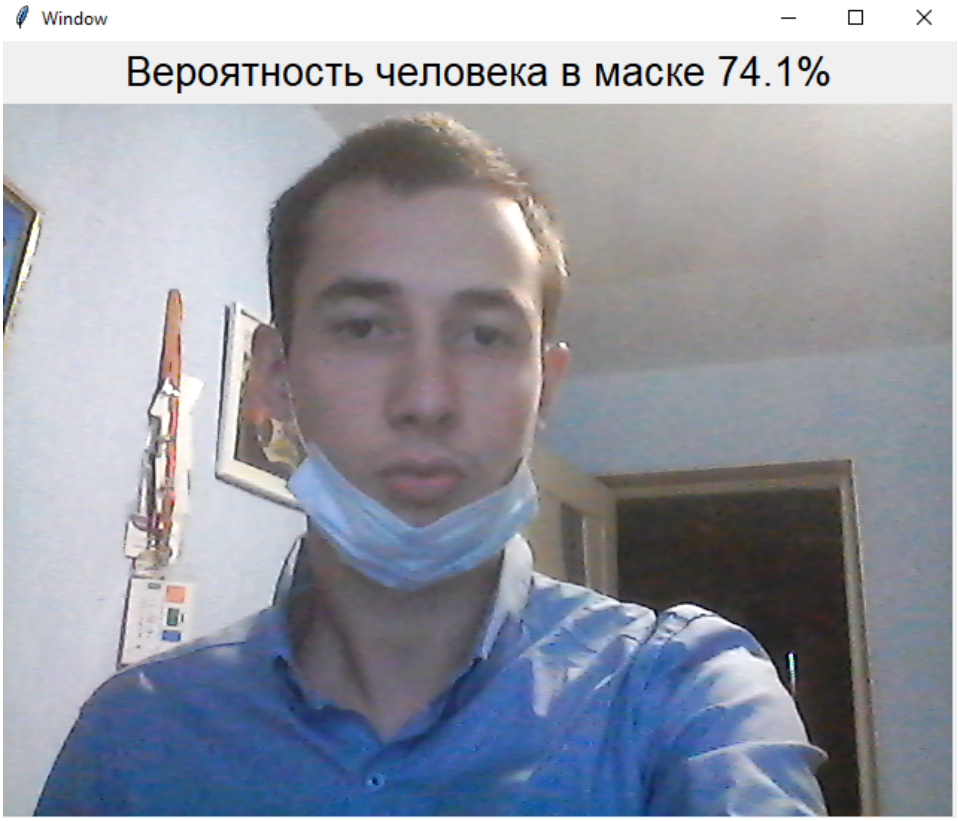

Рис. 14: Тестирование с недоконца надетой маской