НИЯУ МИФИ

Разработка Telegram-бота с использованием сверточной нейросети (CNN) для определения наличия маски на лице по фото

Студент Плотников А. В.

Преподаватель Ильясов А. И.

Москва 2025

Источник данных

Маски играют решающую роль в защите здоровья людей от респираторных заболеваний, а также одну из немногих мер предосторожности, доступных для COVID-19 в отсутствие иммунизации. С помощью этого набора данных можно создать модель, чтобы обнаружить людей в масках, без масок или в неправильно надетых масках. Этот набор данных содержит 853 изображения, принадлежащих к 3 классам, а также ограничивающие их рамки в формате VOC Pascal.

Классы:

- с маской;
- без маски;
- маска носилась неправильно.

Изображения с VOC Pascal

Kaggle: https://www.kaggle.com/datasets/andrewmvd/face-mask-detection/data

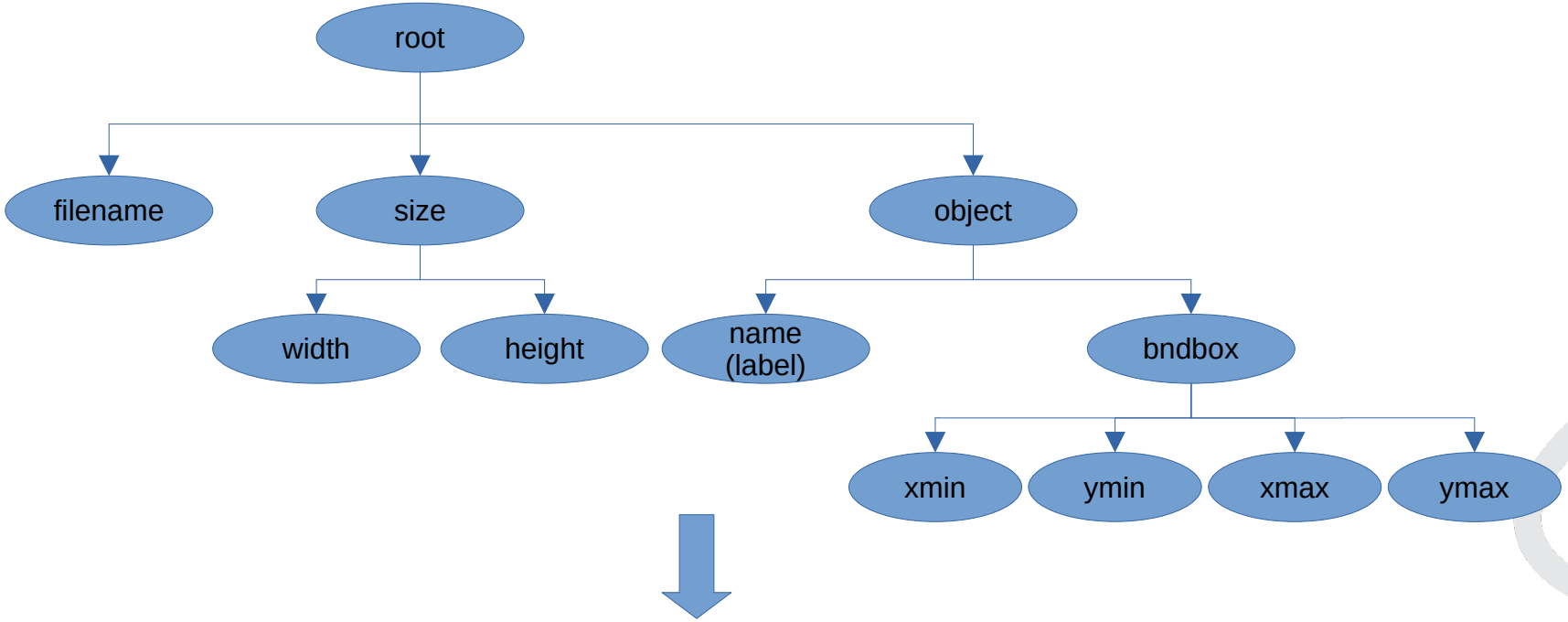
Пример фото из датасета и его описание

```
<annotation>
    <folder>images</folder>
    <filename>maksssksksss0.png</filename>
    <size>
        <width>512</width>
        <height>366</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>without_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>79</xmin>
            <ymin>105</ymin>
            <xmax>109</xmax>
            <ymax>142</ymax>
        </bndbox>
    </object>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>185</xmin>
            <ymin>100</ymin>
            <xmax>226</xmax>
            <ymax>144</ymax>
        </bndbox>
    </object>
    <object>
        <name>without_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>325</xmin>
            <ymin>90</ymin>
            <xmax>360</xmax>
            <ymax>141</ymax>
        </bndbox>
    </object>
</annotation>
```

Изображение maksssksksss0.png

Парсинг данных из xml-формата

```
<annotation>
    <folder>images</folder>
    <filename>maksssksksss0.png</filename>
    <size>
        <width>512</width>
        <height>366</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>without_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>79</xmin>
            <ymin>105</ymin>
            <xmax>109</xmax>
            <ymax>142</ymax>
        </bndbox>
    </object>
    <object>
        <name>with_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>185</xmin>
            <ymin>100</ymin>
            <xmax>226</xmax>
            <ymax>144</ymax>
        </bndbox>
    </object>
    <object>
        <name>without_mask</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>325</xmin>
            <ymin>90</ymin>
            <xmax>360</xmax>
            <ymax>141</ymax>
        </bndbox>
    </object>
</annotation>
```



filename	cropped_filename	width	height	xmin	ymin	xmax	ymax	label
maksssksksss0.png	maksssksksss_0_without_mask.png	512	366	79	105	109	142	without_mask
maksssksksss0.png	maksssksksss_1_with_mask.png	512	366	185	100	226	144	with_mask
maksssksksss0.png	maksssksksss_2_without_mask.png	512	366	325	90	360	141	without_mask

Формирование итоговых изображений

Изображение maksssksksss0.png

а) **б)** **в)**

Итоговые изображения:
а) maksssksksss_0_without_mask.png
б) maksssksksss_1_with_mask.png
в) maksssksksss_3_without_mask.png

Переход к бинарной классификации

with_mask	3232
without_mask	717
mask_weared_incorrect	123

Мало изображений класса mask_weared_incorrect.

При обучении первой версии модели по Confusion Matrix было видно, что класс определяется плохо:

```
Confusion Matrix:
[[  2  13  10]
 [  4 616  27]
 [  0  13 130]]
```

Было решено отбросить данный класс и свести задачу к бинарной классификации.

Приведение изображений к общему формату и добавление аугментации

Разделяем выборку на тренировочную и тестовую части в пропорции 80/20.

Нормализуем изображения [0; 255] → [0; 1] и приводим к одному размеру (224x224).

Размер батча — 32.

Увеличиваем разнообразие выборки с помощью аугментации:

- поворот на случайный угол до 20 градусов;
- сдвиги по ширине и высоте на 20%;
- наклон изображения на 20%;
- изменение масштаба на 20%;
- горизонтальное отражение.

Образованные при аугментации пустые области заполняются ближайшими пикселями.

Создание модели CNN

Total params: 166,650 (650.98 KB)

Trainable params: 166,150 (649.02 KB)

Non-trainable params: 500 (1.95 KB)

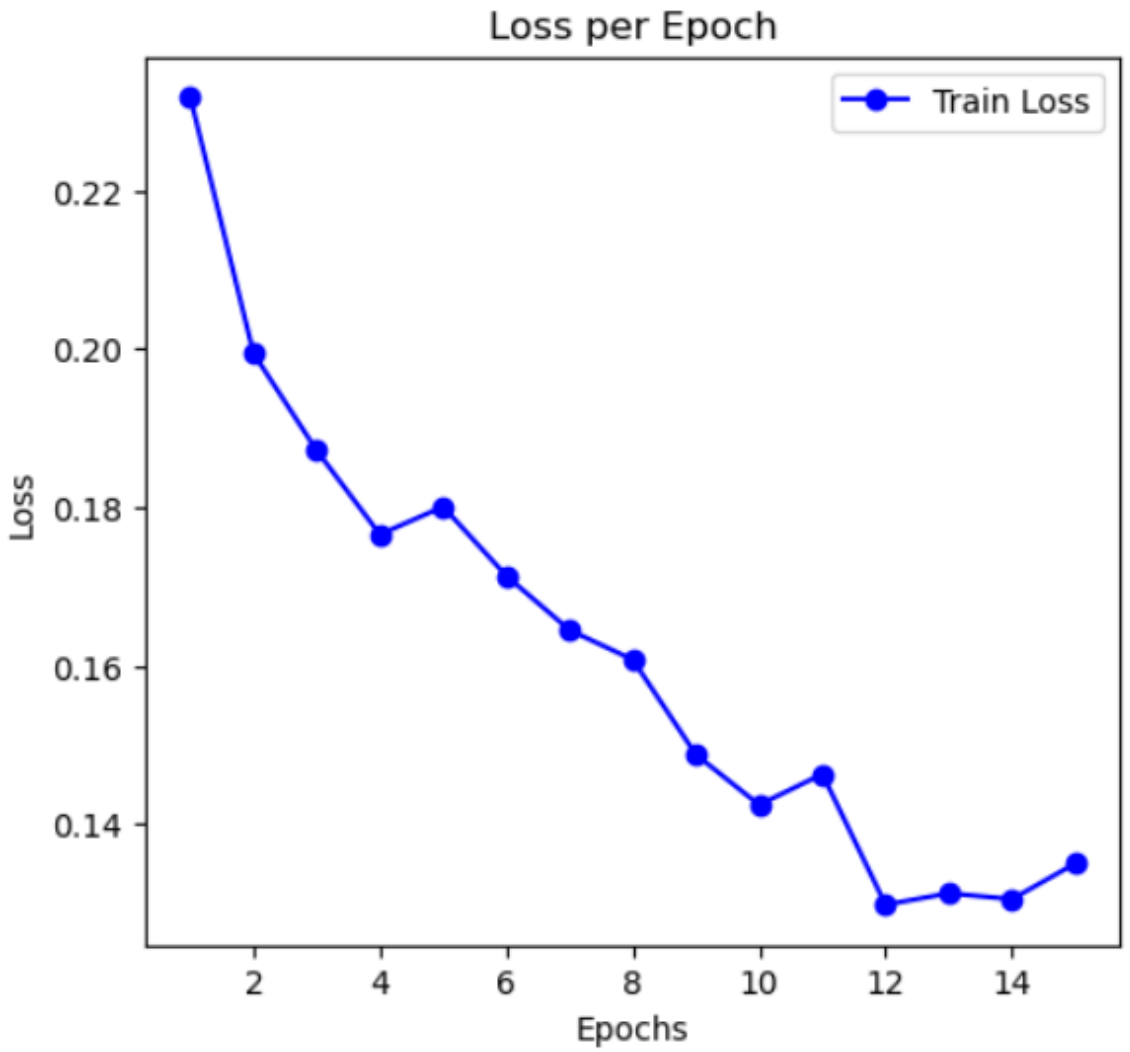
optimizer='adam';
loss='binary_crossentropy';
metrics=['accuracy', metrics.AUC()]

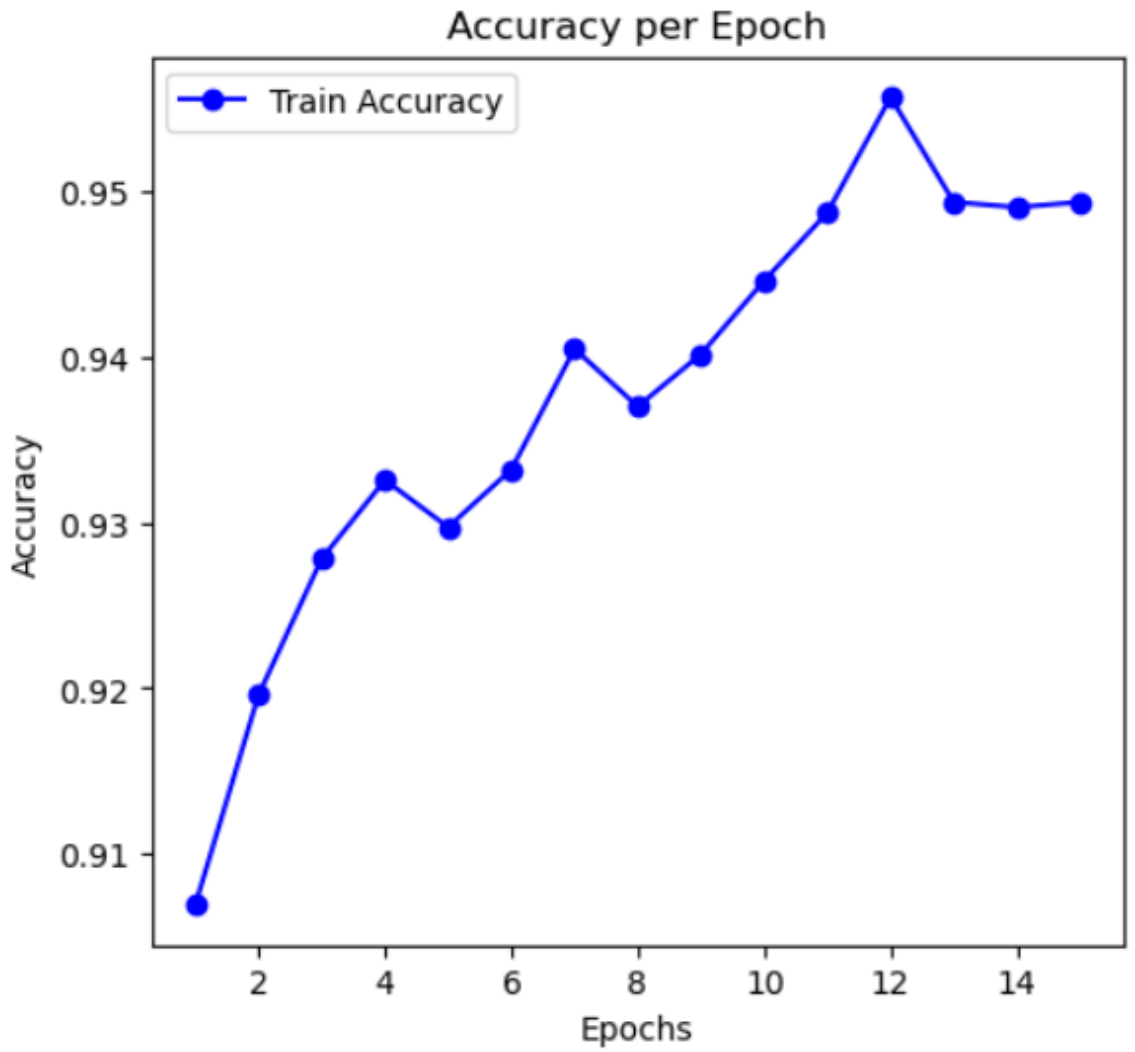
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 25)	700
batch_normalization (BatchNormalization)	(None, 224, 224, 25)	100
re_lu (ReLU)	(None, 224, 224, 25)	0
max_pooling2d (MaxPooling2D)	(None, 112, 112, 25)	0
conv2d_1 (Conv2D)	(None, 112, 112, 50)	11,300
batch_normalization_1 (BatchNormalization)	(None, 112, 112, 50)	200
re_lu_1 (ReLU)	(None, 112, 112, 50)	0
dropout (Dropout)	(None, 112, 112, 50)	0
max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 50)	0
conv2d_2 (Conv2D)	(None, 56, 56, 75)	33,825
batch_normalization_2 (BatchNormalization)	(None, 56, 56, 75)	300
re_lu_2 (ReLU)	(None, 56, 56, 75)	0
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 75)	0
conv2d_3 (Conv2D)	(None, 28, 28, 100)	67,600
batch_normalization_3 (BatchNormalization)	(None, 28, 28, 100)	400
re_lu_3 (ReLU)	(None, 28, 28, 100)	0
max_pooling2d_3 (MaxPooling2D)	(None, 14, 14, 100)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 100)	0
dense (Dense)	(None, 512)	51,712
dropout_1 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1)	513

Обучение модели CNN

```
Epoch 1/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 312s 3s/step - accuracy: 0.8621 - auc_1: 0.8908 - loss: 0.2925
Epoch 2/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 271s 3s/step - accuracy: 0.9162 - auc_1: 0.9523 - loss: 0.2100
Epoch 3/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 266s 3s/step - accuracy: 0.9288 - auc_1: 0.9610 - loss: 0.1762
Epoch 4/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 266s 3s/step - accuracy: 0.9300 - auc_1: 0.9620 - loss: 0.1889
Epoch 5/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 269s 3s/step - accuracy: 0.9226 - auc_1: 0.9606 - loss: 0.1898
Epoch 6/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 267s 3s/step - accuracy: 0.9363 - auc_1: 0.9664 - loss: 0.1654
Epoch 7/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 266s 3s/step - accuracy: 0.9416 - auc_1: 0.9648 - loss: 0.1686
Epoch 8/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 265s 3s/step - accuracy: 0.9318 - auc_1: 0.9682 - loss: 0.1681
Epoch 9/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 266s 3s/step - accuracy: 0.9419 - auc_1: 0.9784 - loss: 0.1445
Epoch 10/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 266s 3s/step - accuracy: 0.9437 - auc_1: 0.9787 - loss: 0.1396
Epoch 11/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 268s 3s/step - accuracy: 0.9443 - auc_1: 0.9755 - loss: 0.1540
Epoch 12/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 267s 3s/step - accuracy: 0.9537 - auc_1: 0.9820 - loss: 0.1311
Epoch 13/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 269s 3s/step - accuracy: 0.9457 - auc_1: 0.9806 - loss: 0.1342
Epoch 14/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 269s 3s/step - accuracy: 0.9439 - auc_1: 0.9799 - loss: 0.1382
Epoch 15/15
99/99 ━━━━━━━━━━━━━━━━━━━━ 268s 3s/step - accuracy: 0.9499 - auc_1: 0.9799 - loss: 0.1327
```

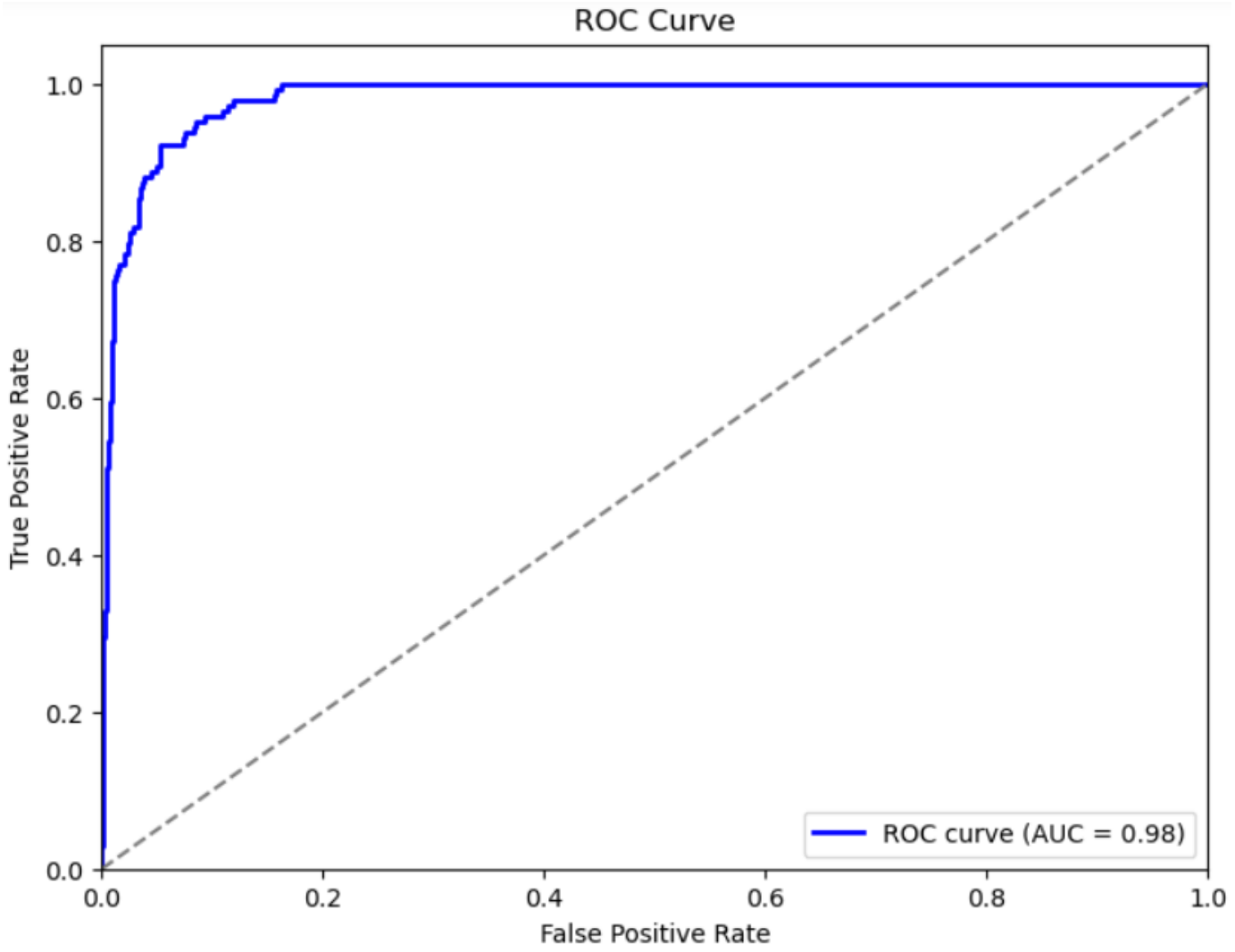
Изменение loss и accuracy в процессе обучения





Проверка модели на тестовых данных

```
Accuracy: 0.9645569620253165
Precision: 0.9652931641010437
Recall: 0.9645569620253165

Confusion Matrix:
[[630  17]
 [ 11 132]]
```



Разработка Telegram-бота
Задание handlers

```python
def main(self) -> None:  1 usage  Артем Плотников *
    """Start the bot."""

    application = Application.builder().token(TOKEN).build()
    application.add_handler(CommandHandler( command: "start", start))
    application.add_handler(CommandHandler( command: "help", help_command))
    application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, echo))
    application.add_handler(MessageHandler(filters.PHOTO & ~filters.COMMAND & ~filters.TEXT, self.predict_mask))
    application.run_polling(allowed_updates=Update.ALL_TYPES)
```

```python
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage  Артем Плотников *
    """Send a message when the command /help is issued."""

    await update.message.reply_text("Бот для определения наличия маски на лице")


async def start(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage  Артем Плотников *
    """Send a message when the command /start is issued."""

    user = update.effective_user
    await update.message.reply_html( text: rf"Здравствуй, {user.mention_html()}! Загрузи фото с лицом в маске или без нее.",
                                    reply_markup=ForceReply(selective=True))


async def echo(update: Update, context: ContextTypes.DEFAULT_TYPE) -> None:  1 usage  Артем Плотников *
    """Echo the user message."""

    await update.message.reply_text("Загрузите фото с лицом в маске или без нее!")
```

Загрузка модели при инициализации

```python
class MaskPredictor:  1 usage  Артем Плотников *
    def __init__(self):  Артем Плотников *
        """Bot initialisation: model and logger loading."""

        logging.basicConfig(format="%(asctime)s - %(name)s - %(levelname)s - %(message)s", level=logging.INFO)
        logging.getLogger("httpx").setLevel(logging.WARNING)

        self.logger = logging.getLogger(__name__)
        self.model = load_model('data/mask_detection_v2.h5')
```

Получение и сохранение изображения

```python
async def predict_mask(self, update: Update, context: CallbackContext):  1 usage  Артем Плотников *
    """Predict if person with mask."""

    photo = update.message.photo[-1]

    file = await context.bot.get_file(photo.file_id)
    img_path = "temp.jpg"
    await file.download_to_drive(img_path)
```

Детектирование лица на изображении

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

img = cv2.imread(img_path)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

if len(faces) == 0:
    await update.message.reply_text("Лицо не обнаружено!")
    os.remove(img_path)
    return
```

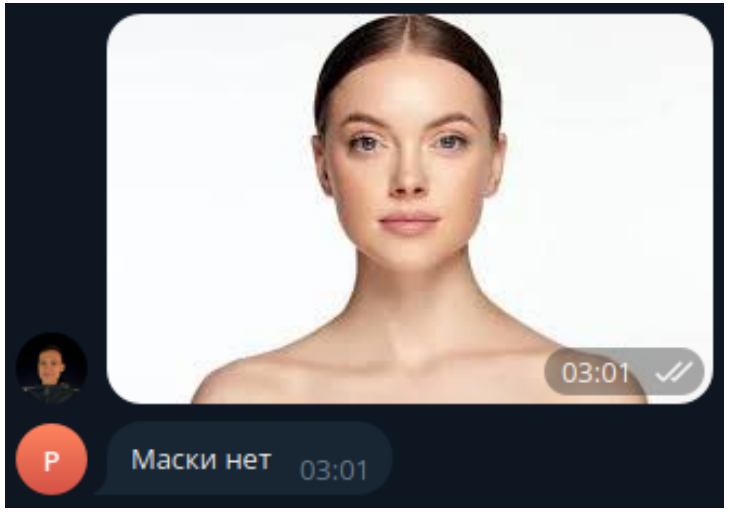
Haar Cascade Classifier — предобученная модель для детектирования лиц

detectMultiScale() применяется для поиска всех лиц на изображении

Преобразование изображения в нужный формат и вызов модели для предсказания

```
    for (x, y, w, h) in faces:
        face_img = img[y:y + h, x:x + w]

        face_img = cv2.cvtColor(face_img, cv2.COLOR_BGR2RGB)
        face_img = image.array_to_img(face_img)
        face_img = face_img.resize((224, 224))
        img_array = image.img_to_array(face_img) / 255
        img_array = np.expand_dims(img_array, axis=0)

        predictions = self.model.predict(img_array)
        predicted_class_index = np.round(predictions).astype(int)

        label_map = {
            0: 'Маска надета',
            1: 'Маски нет'
        }

        await update.message.reply_text(f"{label_map.get(predicted_class_index.item())}")

    os.remove(img_path)
```

Результаты работы Telegram-бота



Спасибо за внимание!