**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МО ЭВМ**

# КУРСОВАЯ РАБОТА

**по дисциплине «Производственная практика НИР»**

**Тема: Разработка модели объяснимого искусственного интеллекта для выявления синтетических изображений**

Студент гр. 8304 _____ Облизанов А.Д.

Преподаватель _____ Заславский М.М.

Санкт-Петербург
2023

**ЗАДАНИЕ**
**НА КУРСОВУЮ РАБОТУ**

Студент Облизанов А.Д.

Группа 8304

Тема работы: Разработка модели объяснимого искусственного интеллекта для выявления синтетических изображений

Исходные данные:

Формулирование постановки задачи, проведение экспериментов и обзора литературы

Содержание пояснительной записки:

«Содержание», «Основные термины», «Введение», «Постановка задачи», «Результаты работы в осеннем семестре», «Описание предполагаемого решения», «План работы на весенний семестр», «Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 24.11.2023

Дата сдачи реферата: 20.12.2023

Дата защиты реферата: 22.12.2023

Студент _____ Облизанов А.Д.

Преподаватель _____ Заславский М.М.

## АННОТАЦИЯ

В данной работе рассмотрены существующие подходы к выявлению синтетических изображений (deepfake) лиц людей с помощью технологий искусственного интеллекта с точки зрения объяснимости выходных данных и их применимости для анализа качества синтетической генерации данных. Определены недостатки существующих решений, предложена методология использования методов объяснимого искусственного интеллекта и реализовано ПО, позволяющее визуально выделять синтетические области изображения.



## SUMMARY

This paper examines existing approaches to identifying synthetic images (deepfake) of people's faces using artificial intelligence technologies from the point of view of the explainability of the output data and their applicability for analyzing the quality of synthetic data generation. The shortcomings of existing solutions are identified, a methodology for using explainable artificial intelligence methods is proposed, and software is implemented that allows visually identifying synthetic areas of the image.

**СОДЕРЖАНИЕ**

## ОСНОВНЫЕ ТЕРМИНЫ

- Синтетические данные – данные, сгенерированные компьютерным алгоритмом по заданным параметрам [1].
- Искусственный интеллект (ИИ, англ. Artificial intelligence, AI) – это направление науки и технологий, которое изучает и разрабатывает способы решения задач обучения, решения проблем, распознавания шаблонов [2].
- Deepfake (дипфейк) - методика синтеза изображения, основанная на искусственном интеллекте, чаще всего применяемся для подмены лица человека [3].
- Модель машинного обучения – форма искусственного интеллекта, позволяющая машинному алгоритму обучаться на основе некоторого массива данных.
- Остаточная нейронная сеть — это модель глубокого обучения, в которой весовые слои изучают остаточные функции со ссылкой на входные данные слоя [4].

# ВВЕДЕНИЕ

В последние годы генерация синтетических изображений и видеозаписей людей или других живых существ кратно возросла. Эта концепция также известна как «deepfake», где «deep» отражает использование нейронных сетей Deep Learning, а «fake» - отражает подделку и подмену по отношению к исходному вводу.

Такая подделка изображений используется и во вред, и во благо. Например, она может использоваться для цифровой реанимации исторических личностей [4], для виртуальных примерочных одежды [5]. Такие приложения, как Snapchat, Instagram, Facebook и Reddit, используют подходы DL для изучения конкретных функций изображения и переноса их на другое изображение или видео. Эти достижения deepfake подчеркнули потенциальные последствия цифровых манипуляций с лицом. Несмотря на то, что это интересно и удобно, существует угроза того, что злоумышленники будут использовать deepfake-атаки, чтобы поставить под угрозу безопасность других.

Хотя многие модели генерации deepfake производят неразличимые репродукции, эти подделки все еще могут быть обнаружены либо специализированными криминалистическими методами, либо методами глубокого обучения [6]. Почти каждый генератор deepfake оставляет на изображении следы своих операций свертки. В настоящее время существует большое количество накопленных данных, связанных с анализом deepfake-изображений, особенно таковых, использующих замену или искажение лица человека. Это делает возможным применение методом машинного обучения для распознавания поддельных изображений, по сути, классификации изображений на реальные и deepfake.

Несмотря на высокую точность некоторых моделей, представленных сегодня для решения этой задачи, технологии подделки изображений так же не стоят на месте, что не дает гарантии, что модель будет так же точно распознавать

deepfake изображения, полученные новым методом. В связи с этим все более актуальным становится применение объяснимого искусственного интеллекта для решения задачи классификации изображений. Методы объяснимого искусственного интеллекта главным образом нацелены на то, чтобы повысить доверие пользователей к технологиям AI [7]. Они уже используются в научных исследованиях: они позволяют определять причины успеха той или иной модели AI, устанавливать зависимости в данных, полученные в результате обучения модели на больших объемах данных.

Применение таких методов совместно с наиболее точными моделями определения deepfake изображений позволит объяснить неточности алгоритмов, что укажет на недостатки текущих алгоритмов генерации синтетических данных, а также позволит уберечь пользователей сети Интернет от кибератак с использованием этой технологии, предоставляя аргументированный (объясненный) анализ, показывающий поддельность компрометирующего изображения.

# ПОСТАНОВКА ЗАДАЧИ

**Актуальность:**

Технологии создания поддельных изображений с помощью технологий Deep Leaning быстро развиваются, что, помимо позитивного влияния на кинопроизводство, восстановление истории и другие сферы, где технологии используются во благо, приводит к рискам кибератак и преступлений с их применением. Поэтому потребность в методах, позволяющий отличить реальное изображение от поддельного, резко возрастает. Существующие методы представляют собой черный ящик, что негативно влияет на доверие пользователей к распознаванию, что делает актуальной **проблему** классификации синтетических изображений с помощью технологий объяснимого искусственного интеллекта.

**Объектом исследования** являются подходы к классификации синтетических изображений.

**Предметом исследования** являются модели объяснимого искусственного интеллекта.

**Целью работы** является применение моделей объяснимого искусственного интеллекта для классификации синтетических изображений и реальных снимков.

Для достижения цели необходимо выполнить следующие **задачи**:

- Анализ существующих наборов данных синтетических изображений;
- Разработка инструмента подготовки данных для моделей ИИ распознавания синтетических изображений;
- Анализ методов объяснимого искусственного интеллекта для объяснения построенной модели;
- Апробация выбранных модели и метода на наборах данных.

# 1. РЕЗУЛЬТАТЫ РАБОТЫ В ОСЕННЕМ СЕМЕСТРЕ

## 1.1. План работы на осенний семестр

В ходе проведенной работы был проведен выбор датасета для получения оригинальных и синтетических изображений. Был разработан инструмент, который покадрово определяет фреймы, в которых содержатся лица людей на видеозаписях и сохраняет получившиеся метаданные для дальнейшего применения фреймов как входных данных в модель машинного обучения. Для достижения цели - применения моделей объяснимого искусственного интеллекта для классификации синтетических изображений и реальных снимков, необходимо предпринять следующие шаги:

- Выбор модели машинного обучения, производящей классификацию синтетических и оригинальных изображений;
- Применение метода GradCAM к модели, настройка параметров метода;
- Проверка возможности определения дипфейка и объяснения причины обнаружения в режиме реального времени;
- Обработка данных, поступающих из метода GradCAM, для дальнейшего анализа полученных объяснений и выявления закономерностей.

## 1.2. Выбор модели машинного обучения

Выбор модели машинного обучения ориентировался на соревновании по датасету DFDC, проводимому на площадке Kaggle в 2019-2020 годах. В результате анализа наиболее успешных решений [8] были определены типы моделей, наиболее подходящие для решения данной задачи

- EfficientNet B3-7 [9]
- ResNeXt 50, 101 [10]

EfficientNet – это класс моделей сверточной нейронной сети, которые получены наиболее эффективным масштабированием и комбинированием параметров глубины, ширины и разрешения изображений. Базовая модель B0 имеет структуру из 8 модулей сверточной нейронной сети (ConvNet). Далее с помощью коэффициента, который отражает вычислительную нагрузку, формируются наборы параметров для моделей B1…B7. Модели EfficientNet демонстрирует отличное сочетание скорости работы и точности при тестировании на датасете ImageNet, что представлено на рис. 1 [11].

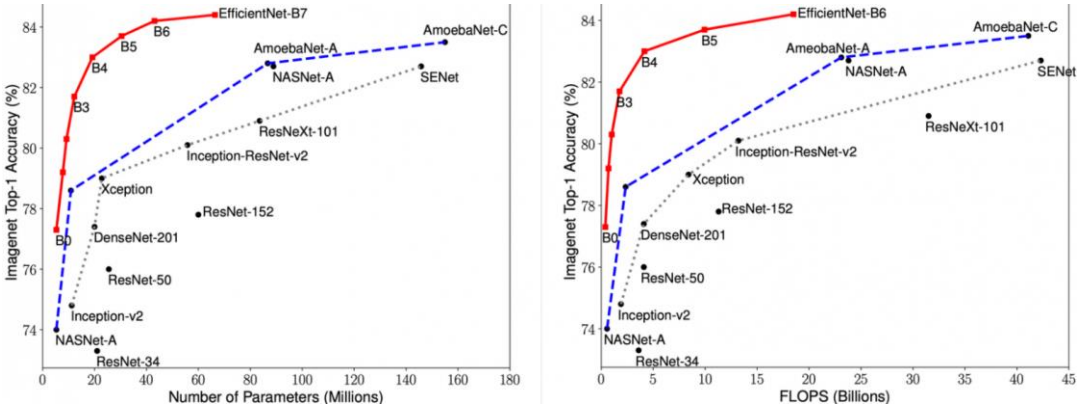


Рисунок 1 – Сравнение точности на ImageNet и сложности моделей

ResNeXt – это развитие модели остаточной нейронной сети ResNet, которая использует identity-функцию для прямой связи между входными весовыми блоками и функцией активации слоя, что дает возможность «перепрыгивать» слои [4]. ResNext повторяет блок ResNet, однако вводит измерение кратности, что позволяет изменять глубину и ширину блоков, что представлено на рис. 2 [12].

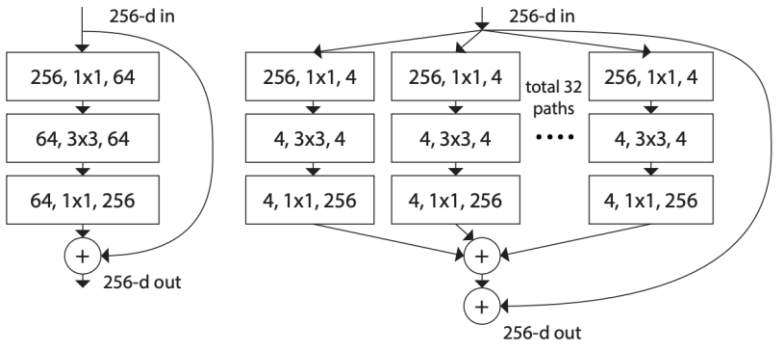


Рисунок 2. Слева – блоки ResNet. Справа – блоки ResNeXt с кратностью 32.

Данные модели будут использоваться вместе с методом объяснимого искусственного интеллекта по отдельности, что позволит создавать ансамбли из моделей такого типа для дальнейших шагов по улучшению точности системы. Объяснение в таком случае будет получено для каждой модели, а затем обобщено согласно весам или правилам ансамблирования моделей.

### 1.3. Применение метода GradCAM к модели, настройка параметров метода

Метод объяснимого искусственного интеллекта Gradient-weighted Class Activation Mapping (Grad-CAM) - использует градиенты любой целевой модели, перетекающие в конечный сверточный слой для создания грубой карты локализации, выделяющей важные области изображения для прогнозирования модели. Grad-CAM относится к локальным методам объяснимого ИИ, то есть предоставляет объяснение для каждого предсказания модели без обобщения. Это позволяет методу не зависит от корреляции входных признаков, неусточивость к которым определена у методов SHAP и Lime [13].

Для практического использования метода создателями разработана библиотека для языка Python pytorch-grad-cam [14].

Для апробации применения метода GradCAM для решения задачи объяснения распознавания дипфейков был разработан Python-блокнот на площадке Kaggle, использующий следующие библиотеки и данные:

- Датасет DFDC
- Skimage – библиотека обработки изображений
- Torch – фрейморк для создания моделей ИИ
- Torchvision – библиотека для работы с моделями распознавания изображений
- Matplotlib – библиотека для построения графиков

- Blazeface – библиотека для распознавания и обрезки лица на фотографии человека
- Deepfakes-inference-demo – датасет весов обученных на DFDC датасете моделей

В блокноте выполняются следующие действия:

- Извлечение n равноотстоящих кадров из видеофайла с дипфейком
- Распознавание и обрезка лица на кадре с помощью Blazeface
- Применение модели ResNeXt-50 к кадрам
- Применение метода GradCAM к блокам модели и визуализация

Метод GradCAM может предоставлять визуальную интерпретацию важности частей изображения по активационным функциям одного из слоев модели. Модель ResNeXt имеет 4 сверточных блока, в конце каждого из которых имеется функция активации. На выходе последнего блока составляются карты признаков размерности 7x7, а на предпоследнем – 14x14. Для более точного определения важности признаков на изображении GradCAM интерпретирует выходы с обоих блоков.

Исходный код блокнота представлен в Приложении А.

### 1.4. Примеры объяснения методом GradCAM

Для демонстрации работы метода объяснимого ИИ были отобраны видеозаписи из датасета DFDC, содержащие синтетические видеозаписи невысокого качества, такие, что искусственность может быть определена человеком без глубокого анализа. В результате запуска блокнота на некоторых таких видео были получены карты важности признаков на изображении с помощью метода GradCAM.

Пример изображения и объяснения предсказания модели представлен на рис. 3.

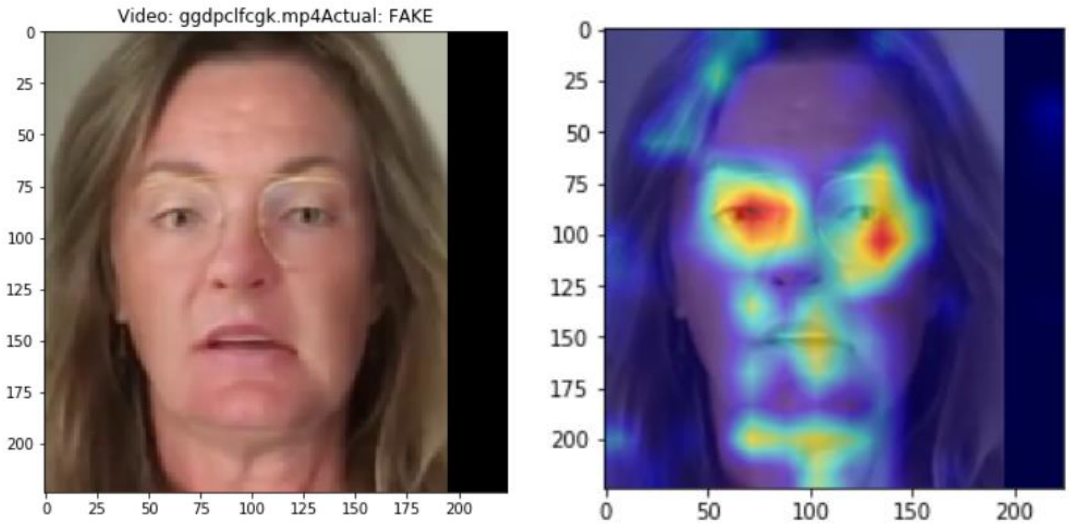


Рисунок 3 – Пример работы GradCAM на синтетическом изображении

На оригинальном изображении можно заметить искажение –дужка очков в некоторых областях отсутствует, пропадает. Кроме того: тон кожи в левой части около глаза сильно отличается от тона в остальных частях лица. На карте важности признаков, полученной методом GradCAM, можно заметить, что описанные выше искажения выделены. В данном случае изображено объяснение GradCAM на картах признаков предпоследнего блока ResNeXt модели, так как карты признаков последнего блока предоставляют слишком обобщенное представление.

Еще один пример работы представлен на рис. 4-5.


Рисунок 4 – Применение GradCAM на последнем блоке ResNeXt модели

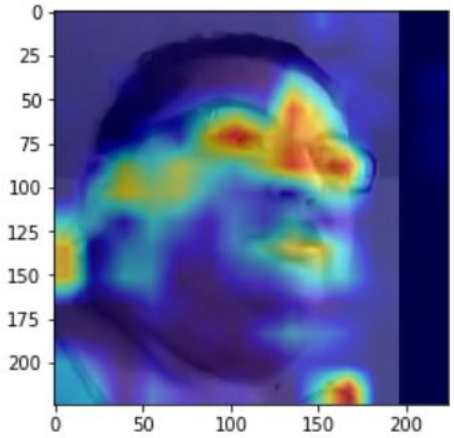

Рисунок 5 – Применение GradCAM на предпоследнем блоке ResNeXt модели

На исходном изображении можно заметить неестественную форму лба и искаженную форму очков – искажения в верхней части лица. На интерпретации GradCAM последнего блока можно заметить, что область лица, выдающая дипфейк, выделена корректно. Однако более мелкие признаки (лоб, очки) можно наблюдать только на объяснении по предпоследнему блоку модели. В то же время, на рис. 5 отмечен объект вне лица, расположенный на стене, который тоже был «отмечен» моделью на предпоследнем блоке свертки. При этом судя по интерпретации последнего блока, этот объект не повлиял на объяснение модели.

В результате экспериментов было определено, что для наиболее эффективного использования метода GradCAM с моделями ResNeXt необходимо анализировать интерпретацию объяснения с последних двух слоев модели (а для более детальных изображений – на большем количестве слоев):

- Интерпретация последнего блока 7x7 предоставляет области наибольшего интереса на изображении (области лица)
- Интерпретация предпоследнего блока 14x14 представляет признаки (части лица), наибольшим образом влияющие на предсказание модели

# 2. ОПИСАНИЕ ПРЕДПОЛАГАЕМОГО РЕШЕНИЯ

Необходимо разработать ПО, позволяющее использовать настраиваемые ансамбли моделей типа ResNeXt и EfficientNet, применяющее метод объяснимого ИИ GradCAM для получения визуальной интерпретации важности признаков на изображении каждой модели отдельно и ансамбля моделей для итоговой интерпретации. Данное ПО должно работать с видеозаписями, производя предварительную обработу – выбор кадров, преобразование изображений, обрезка лица человека на кадре. Помимо этого, необходимо предусмотреть возможность использования технологий Facial mapping для разметки областей лица и сопоставлении компрометирующих признаков на изображении с разметкой.

В качестве ЯП был выбран язык Python, формат блокнота IPYNB. Площадка для размещения и выполнения кода – Kaggle.

Алгоритм работы инструмента состоит из следующих шагов:

1. На вход инструмента подается видеозапись в формате mp4, над которой производятся следующие преобразования:
   1.1. Выбор равноотстоящих n кадров изображения
   1.2. Определение и обрезка лица на кадре
   1.3. Составление батча изображений для подачи в модели
2. Используются обученные модели ИИ для получения предсказаний
   2.1. Выполняются модели ResNeXt различной конфигурации
   2.2. Выполняются модели EfficientNet различной конфигурации
3. Используется метод GradCAM, адаптированный под каждую из моделей, в результате чего фиксируются карты признаков для каждой их моделей ансамбля при работе метода на различных слоях
   3.1. Для каждой модели метод выполняется на предпоследнем и последнем блоке модели

3.2. Полученные карты признаков предпоследнего слоя фильтруются картам признаков последнего слоя, чтобы оставить только признаки, повлиявшие на итоговое предсказание модели

4. Происходит ансамблирование предсказаний моделей и объяснений (карт признаков) метода GradCAM для каждого изображения
5. Определяются максимальные значения предсказания о вероятности дипфейка. Эти значения, а также интерпретации GradCAM сохраняются.
6. Выполняются методы дополнительного анализа в случае, если с высокой вероятностью был обнаружен дифейк
   6.1. Выполняется модель определения карты лица (facial mapping)
   6.2. Рассчитывается удаленность наиболее важных признаков в интерпретации GradCAM от меток объектов лица
   6.3. Определяются и сохраняются части лица, компрометирующие дипфейк

Обобщенная схема решения представлена на рис. 6.

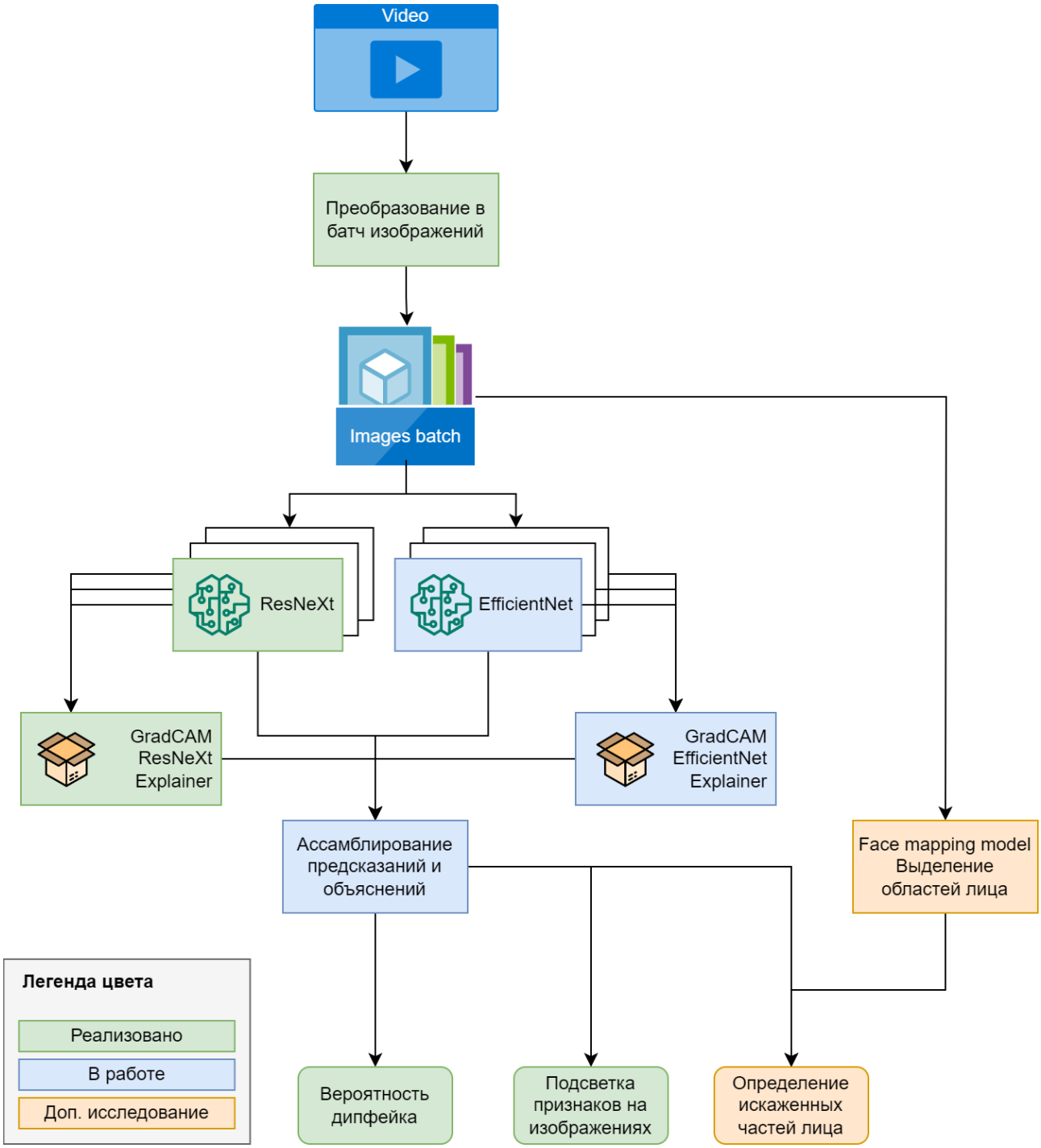


Рисунок 6 – Обобщенная схема решения

# ПЛАН РАБОТЫ НА ВЕСЕННИЙ СЕМЕСТР

По результатам работы осеннего семестра был разработан прототип решения, который необходимо доработать согласно описанию предполагаемого решения. Также необходимо провести ряд экспериментом для апробации решения и проанализировать полученные результаты. Показать практическую применимость разработанного инструмента и предложить потенциальные пути использования и развития архитектуры. Для этого необходимо:

- Разработать ансамбль моделей ResNeXt и EfficientNet
- Применить метод GradCAM для каждой модели ансамбля и обобщить результаты
- Разработать алгоритм выделения признаков на основе применения GradCAM на предпоследнем и последнем блоке моделей
- Составить датасет корректных интерпретаций дипфейков с помощью разработанного инструмента
- Проанализировать полученные результаты и предложить дальнейшие шаги

Одним из дальнейших шагов является определение частей лица на изображении и сопоставление интерпретации дипфейков с частями лица. Это может позволить выявить слабые стороны алгоритмов дипфейков с биологической точки зрения, например, определить, какие части лица наиболее сложно подделать современным моделям создания синтетических изображений.

Помимо разработки решения необходимо:

- Написать научное исследование на тему ВКР
- Выступить на локальной конференции
- Выполнить остальные шаги согласно плану подготовки ВКР

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Nvidia // What Is Synthetic Data? URL: https://blogs.nvidia.com/blog/2021/06/08/what-is-synthetic-data/ (дата обращения: 02.11.2022).
2. TechTarget // What is artificial intelligence (AI)? URL: https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence (дата обращения: 02.11.2022)
3. INSIDER // What is a deepfake? Everything you need to know about the AI-powered fake media. URL: https://www.businessinsider.com/guides/tech/what-is-deepfake (дата обращения: 23.11.2023)
4. Wu Z., Shen C., Van Den Hengel A. Wider or deeper: Revisiting the resnet model for visual recognition //Pattern Recognition. – 2019. – Т. 90. – С. 119-133.
5. Mirsky Y., Lee W. The creation and detection of deepfakes: A survey //ACM Computing Surveys (CSUR). – 2021. – Т. 54. – №. 1. – С. 1-41.
6. Guarnera L. et al. Preliminary forensics analysis of deepfake images //2020 AEIT international annual conference (AEIT). – IEEE, 2020. – С. 1-6.
7. Tolosana R. et al. Deepfakes and beyond: A survey of face manipulation and fake detection //Information Fusion. – 2020. – Т. 64. – С. 131-148.
8. Kaggle // Deepfake Detection Challenge. URL: https://www.kaggle.com/competitions/deepfake-detection-challenge/leaderboard (дата обращения: 24.11.2023)
9. GitHub // selimsef/dfdc_deepfake_challenge repository. URL: https://github.com/selimsef/dfdc_deepfake_challenge (дата обращения: 25.11.2023)

10. Kaggle // Deepfake Detection Challenge, 5th (from 20th) place approach. URL: https://www.kaggle.com/competitions/deepfake-detection-challenge/discussion/140364 (дата обращения: 25.12.2023)
11. Tan M., Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks //International conference on machine learning. – PMLR, 2019. – С. 6105-6114.
12. Xie S. et al. Aggregated residual transformations for deep neural networks //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 1492-1500.
13. Oblizanov A. et al. Evaluation Metrics Research for Explainable Artificial Intelligence Global Methods Using Synthetic Data //Applied System Innovation. – 2023. – Т. 6. – №. 1. – С. 26.
14. GitHub // jacobgil/pytorch-grad-cam repository. URL: https://github.com/jacobgil/pytorch-grad-cam (дата обращения: 25.12.2023)

# ПРИЛОЖЕНИЕ А

# ИСХОДНЫЙ КОД ПРОГРАММЫ

```
!pip install grad-cam
# Loading the required libraries
import os, sys, time
import cv2
import random
import numpy as np
import pandas as pd
import skimage.transform

import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import models, transforms
from torch.autograd import Variable
from torch import topk

%matplotlib inline
import matplotlib.pyplot as plt
from PIL import Image
from matplotlib.pyplot import imshow
from tqdm.notebook import tqdm

# Read the test videos
test_dir = "/kaggle/input/deepfake-detection-challenge/test_videos/"

test_videos = sorted([x for x in os.listdir(test_dir) if x[-4:] == ".mp4"])
len(test_videos)

#Load the test labels -- this was created by mapping the metadata file with the
test file names
test_labels               =               pd.read_csv('/kaggle/input/sample-face-
crop/test_video_labels.csv')
test_labels.head()
test_labels['label'].value_counts()
# Confirm the env variables
print("PyTorch version:", torch.__version__)
print("CUDA version:", torch.version.cuda)
print("cuDNN version:", torch.backends.cudnn.version())
# Check if GPU is available
gpu = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
gpu
# Attach the required libraries to the system path
# This have a few helper functions & path to the pre-trained model
```

```
import sys
sys.path.insert(0, "/kaggle/input/blazeface-pytorch")
sys.path.insert(0, "/kaggle/input/deepfakes-inference-demo")
# Initalize blazeface

from blazeface import BlazeFace
facedet = BlazeFace().to(gpu)
facedet.load_weights("/kaggle/input/blazeface-pytorch/blazeface.pth")
facedet.load_anchors("/kaggle/input/blazeface-pytorch/anchors.npy")
_ = facedet.train(False)
from helpers.read_video_1 import VideoReader
from helpers.face_extract_1 import FaceExtractor

frames_per_video = 16

video_reader = VideoReader()
video_read_fn = lambda x: video_reader.read_frames(x,
num_frames=frames_per_video)
face_extractor = FaceExtractor(video_read_fn, facedet)
input_size = 224 # Define the input size of the image

# Define the normalizing functions with ImageNet parameters
from torchvision.transforms import Normalize

mean = [0.485, 0.456, 0.406]
std = [0.229, 0.224, 0.225]
normalize_transform = Normalize(mean, std)

# Define some helper functions for re-sizing image & making them into perfect
squares
def isotropically_resize_image(img, size, resample=cv2.INTER_AREA):
    h, w = img.shape[:2]
    if w > h:
        h = h * size // w
        w = size
    else:
        w = w * size // h
        h = size

    resized = cv2.resize(img, (w, h), interpolation=resample)
    return resized


def make_square_image(img):
    h, w = img.shape[:2]
    size = max(h, w)
    t = 0
```

```
    b = size - h
    l = 0
    r = size - w
    return cv2.copyMakeBorder(img, t, b, l, r, cv2.BORDER_CONSTANT, value=0)
# Define the ResNext Model with the given blocks & load the checkpoint

import torch.nn as nn
import torchvision.models as models

class MyResNeXt(models.resnet.ResNet):
    def __init__(self, training=True):
        super(MyResNeXt, self).__init__(block=models.resnet.Bottleneck,
                                        layers=[3, 4, 6, 3],
                                        groups=32,
                                        width_per_group=4)

        self.fc = nn.Linear(2048, 1)
#Load the checkpoint & update the model for prediction
checkpoint  =  torch.load("/kaggle/input/deepfakes-inference-demo/resnext.pth",
map_location=gpu)

model = MyResNeXt().to(gpu)
model.load_state_dict(checkpoint)
_ = model.eval()

del checkpoint
#model.layer4 <<- You can use this to inspect the 4th layer or any other layers
y = 'REAL'
while y == 'REAL':
    sample_video= random.choice(test_videos) # Select a random test video
    video_path = os.path.join(test_dir, sample_video)
    y          =          test_labels[test_labels['processedVideo']          ==
sample_video]['label'].values[0]
print("Selected Video: ", sample_video)
print("True Value: ",y)
batch_size = 16 # Extract faces from 16 frames in the video
faces = face_extractor.process_video(video_path)
print("No. of frames extracted: ", len(faces))
print("Keys in the extracted info: ", faces[0].keys())
try:
    print("Shape  of  extracted  face_crop:  ",  faces[0]['faces'][0].shape)  #
multiple faces can be captured. In this set only a single face is detected
    print("Scores of the face crop: ", faces[0]['scores'][0])
except:
    print("====================================")
    print("No faces detected! Please run again.")
```

```
# Only look at one face per frame. This removes multiple faces from each frame,
keeping only the best face
face_extractor.keep_only_best_face(faces)
def preprocessTensor(sample_face):
    """
    param imageArray: face crop passed to the function
    return imageTensor(x), resized_face: The processed tensor and resized_face

    The following activities are performed:
    1# resizing the image, via zero padding
    2# make the crop into a square
    3# Convert to a tensor and load to the gpu
    4# Permutate the axis so that the channel is at zero index
    5# Normalize & add a new dimension in zero index
    """
    resized_face = isotropically_resize_image(sample_face, input_size)
    resized_face = make_square_image(resized_face)

    x = torch.tensor(resized_face, device=gpu).float()
    x = x.permute(( 2, 0, 1))
    x = normalize_transform(x / 255.)
    x = x.unsqueeze(0)

    return x, resized_face
# Resize to the model's required input size.
# We keep the aspect ratio intact and add zero --???? This could be a problem!
# padding if necessary.

sample_face = faces[0]['faces'][0]
x, resized_face = preprocessTensor(sample_face)
print(x.shape)

# Make a prediction.
with torch.no_grad():
    y_pred = model(x)
    y_pred = torch.sigmoid(y_pred.squeeze())

print("Prediction: ", y_pred)
# Resize to the model's required input size.
# We keep the aspect ratio intact and add zero --???? This could be a problem!
# padding if necessary.
sample_face = faces[0]['faces'][0]
resized_face = isotropically_resize_image(sample_face, input_size)
resized_face = make_square_image(resized_face)
resized_face.shape
x = torch.tensor(resized_face, device=gpu).float()
print(x.shape)
```

```
# Preprocess the images.
x = x.permute(( 2, 0, 1))
x = normalize_transform(x / 255.)
x = x.unsqueeze(0)
print(x.shape)
# Make a prediction.
with torch.no_grad():
    y_pred = model(x)
    y_pred = torch.sigmoid(y_pred.squeeze())

print("Prediction: ", y_pred)
class SaveFeatures():
    features=None
    def __init__(self, m): self.hook = m.register_forward_hook(self.hook_fn)  # attach the hook to the specified layer
    def hook_fn(self, module, input, output): self.features = ((output.cpu()).data).numpy() # copy the activation features as an instance variable
    def remove(self): self.hook.remove()
final_layer = model._modules.get('layer4') # Grab the final layer of the model
activated_features = SaveFeatures(final_layer) # attach the call back hook to the final layer of the model
prediction_var = Variable(x.cuda(), requires_grad=True) # Squeeze the  variable to add an additional dimension & then
prediction_var.shape                                    # wrap it in a Variable which stores the grad_training weights
y_pred = model(prediction_var)
y_pred = torch.sigmoid(y_pred.squeeze())

print("Prediction: ", y_pred)
pred_probabilities = F.softmax(y_pred).data.squeeze() # Pass the predictions through a softmax layer to convert into probabilities for each class
print("Predicted Class: ", pred_probabilities)
def getCAM(feature_conv, weight_fc, class_idx):
    _, nc, h, w = feature_conv.shape
    cam = weight_fc.dot(feature_conv.reshape((nc, h*w)))
    cam = cam.reshape(h, w)
    cam = cam - np.min(cam)
    cam_img = cam / np.max(cam)
    return [cam_img]

#overlay = getCAM(activated_features.features, weight_softmax, 0. )
weight_softmax_params = list(model._modules.get('fc').parameters()) # This gives a list of weights for the fully connected layers
print(len(weight_softmax_params))
print(weight_softmax_params[0].shape, weight_softmax_params[1].shape) # weghts for the last two layers
```

```
weight_softmax = weight_softmax_params[0].cpu().data.numpy() # what does this do
??
print(weight_softmax.shape)
#activated_features.remove()
cam_img = getCAM(activated_features.features, weight_softmax, pred_probabilities
)
print(cam_img[0].shape)
imshow(cam_img[0], alpha=0.5, cmap='jet')
from pytorch_grad_cam import GradCAM
from pytorch_grad_cam.utils.model_targets import BinaryClassifierOutputTarget
from pytorch_grad_cam.utils.image import show_cam_on_image
targets = [BinaryClassifierOutputTarget(1)]
target_layers = [model.layer3[-1]]
cam = GradCAM(model=model, target_layers=target_layers, use_cuda=False)
grayscale_cam = cam(input_tensor=prediction_var, targets=targets)
grayscale_cam = grayscale_cam[0, :]
resized_face_norm = resized_face / 255
visualization     =     show_cam_on_image(resized_face_norm,     grayscale_cam,
use_rgb=True)
imshow(visualization)
fig, ax = plt.subplots(1,2, figsize=(10,10))

ax[0].imshow(resized_face)
ax[0].set_title("Video: " + sample_video + "Actual: " + y )
ax[1].imshow(resized_face)
ax[1].imshow(skimage.transform.resize(cam_img[0],
(resized_face.shape[0],resized_face.shape[1] )), alpha=0.25, cmap='jet')
y_pred = str(y_pred.cpu().data.numpy())
ax[1].set_title(y_pred)
fig.tight_layout()

test_labels['y_pred'] = ypred

test_labels['label_10'] = test_labels['label'].apply(lambda x: 1. if x == 'FAKE'
else 0.) # Converting the labels to an ordinal value
test_labels['diff'] = abs(test_labels['label_10'] - test_labels['y_pred']) #
Taking the naive absolute difference of the prediction to the actual ordinal
value
test_labels.sort_values(by=['diff'], ascending=False, inplace=True) # Sorting
the dataframe on difference gives us the samples with the most digression
test_labels.reset_index(drop=True, inplace=True)
test_labels.head() # LEts check the samples with the highest loss
test_labels['diff'].describe()
test_labels[test_labels['label_10'] == 0].head()
# Store the predictions. Since I don't want to keep running the above loop
test_labels.to_csv('test_labels.csv')
idx = 16
```

```
sample_video = test_labels.iloc[idx]['processedVideo']
video_path = os.path.join(test_dir, sample_video)
y = test_labels.iloc[idx]['label']
print(sample_video)

batch_size = 1 # Extract faces from 16 frames in the video
faces = face_extractor.process_video(video_path)
try:
    sample_face = faces[0]['faces'][0]
    x, resized_face = preprocessTensor(sample_face)

    final_layer = model._modules.get('layer4')
    activated_features = SaveFeatures(final_layer)

    prediction_var = Variable(x.cuda(), requires_grad=True)
    y_pred = model(prediction_var)
    y_pred = torch.sigmoid(y_pred.squeeze())

    pred_probabilities = F.softmax(y_pred).data.squeeze()
    weight_softmax_params = list(model._modules.get('fc').parameters())

    weight_softmax = weight_softmax_params[0].cpu().data.numpy()

    cam_img      =      getCAM(activated_features.features,      weight_softmax,
pred_probabilities )

    fig, ax = plt.subplots(1,2, figsize=(10,10))

    ax[0].imshow(resized_face)
    ax[0].set_title("Video: " + sample_video + " Actual: " + y )
    ax[1].imshow(resized_face)
    ax[1].imshow(skimage.transform.resize(cam_img[0],
(resized_face.shape[0],resized_face.shape[1] )), alpha=0.25, cmap='jet')
    y_pred  =  "PredProb:"  +  str(y_pred.cpu().data.numpy())  +  "  DIFF:  "  +
str(test_labels.iloc[idx]['diff'])
    ax[1].set_title(y_pred)
    fig.tight_layout()
except:
    print("Error processing file: ", sample_video)
```