ДОДАТОК А
ЛІСТИНГ КОДУ

//клас обробки вхідного зображення і отримання маски

```
public class MF {       //MatrixForeground

    private String TAG = "--------------------";

    private static MF instance = new MF();

    MF() {

    }

    public static MF getInstance() {

        return instance;

    }

    public final Bitmap/*Pair<byte[][], Bitmap>*/
getMatrix(TensorFlowInferenceInterface tfg, int[] source) {

        Log.d(TAG, "Pixel array size " + source.length + "");

        float[] bm = new float[source.length * 3];

        for (int i = 0, j = 0; i < source.length; i++, j += 3) {

            bm[j + 0] = Color.red(source[i]) / 255.0F;

            bm[j + 1] = Color.green(source[i]) / 255.0F;

            bm[j + 2] = Color.blue(source[i]) / 255.0F;

        }

        Log.d(TAG, "Array conversion");

        if (tfg != null) tfg.feed("inputs/images", bm, 1, 1280, 960, 3);

        Log.d(TAG, "------------FillNodeFloat");


        if (tfg != null)

            tfg.run(new String[]{/*"person", */"person_2"/*, "skin","hair",
"skin_and_lips"*/}); // person_2 ; skin ; person", "hair", "skin_and_lips

        Log.d(TAG, "Runstats " + (tfg != null ? tfg.getStatString() :
null));

        float[] maskArr = new float[960 * 1280];

        if (tfg != null) tfg.fetch("person_2", maskArr);
```

```
        Log.d(TAG, "Read person_2 Ok");
int[] resultMask = new int[960 * 960];
for (int i = 307200; i < maskArr.length; i++){
}
        byte[] bmHair = new byte[maskArr.length];


        for (int i = 0; i < maskArr.length; i++) {
            if (maskArr[i] > 128)
                result[i+38400] = Color.argb((int) maskArr[i] > 150 ? (int)
maskArr[i] : 0, 255, 0, 0);
            bmHair[i] = (byte) (maskArr[i]);
        }
        float[] maskArr = new float[960 * 1200];
        if (tfg != null) tfg.fetch("person_2", maskArr);
        Log.d(TAG, "Read person_2 Ok");
int [] resultMask = new int[960 * 960];
// 320 строк пустых сверху; 40 строк срезает модель сверху и снизу; 280x960
= 268800
for (int i = 172800, j = 0; j < resultMask.length; i++, j++){
resultMask[j] = Color.argb( /*(maskArr[i] > 150 ? (int) maskArr[i] : 0)*/
(int) maskArr[i], 0,0,0);
}
        Bitmap resBmp = Bitmap.createBitmap(resultMask, 960, 960,
Bitmap.Config.ARGB_8888);
        return resBmp;
    }
}
```

//клас підготовки вихідного зображення і поєднання вхідного зображення і маски //переднього плану

```
    public void getFG(final Bitmap bitmap, final
TensorFlowInferenceInterface tf) {
        int sizeResult = 960;  //bitmapSource.getHeight(); 2448x2448
```

```
        int bitmapSourceHeight = bitmap.getHeight();

        float scaleHeight = ((float) sizeResult) / bitmapSourceHeight;

        float scaleWidth = scaleHeight;


        Matrix matrix = new Matrix();

        matrix.postScale(scaleWidth, scaleHeight);

// преобразование в 960x960

        Bitmap temporaryBM = Bitmap.createBitmap(bitmap, 0, 0,
bitmapSourceHeight, bitmapSourceHeight, matrix, false);

        // преобразование в 960x1280 - добавляем 320 точек сверху (баласт)

        Bitmap sourceForFG = Bitmap.createBitmap(960, 1280,
Bitmap.Config.ARGB_8888);

        sourceForFG.eraseColor(Color.CYAN);


        Canvas canvas = new Canvas(sourceForFG);   //220

        canvas.drawBitmap(temporaryBM, 0, 220, null); // ВХОДНОЙ РАЗМЕР ДЛЯ
МОДЕЛИ 960X1280 - кидаем квадратную фотку

        Log.d(TAGTENSOR, "sourceForFG  " + sourceForFG.getWidth() + " " +
sourceForFG.getHeight() + "  " + sourceForFG.getConfig());

        final int[] sourceArrForFG = new int[(sourceForFG.getWidth() *
sourceForFG.getHeight())];

        sourceForFG.getPixels(sourceArrForFG, 0, sourceForFG.getWidth(), 0,
0, sourceForFG.getWidth(), sourceForFG.getHeight());

        temporaryBM.recycle();

        temporaryBM = null;

        sourceForFG.recycle();

        sourceForFG = null;

         new Thread(new Runnable() {

            @Override

            public void run() {

                Bitmap maskFG = (MF.getInstance().getMatrix(tf,
sourceArrForFG)); //out - 960x960

                Log.d(TAGTENSOR, "maskFG config = " + maskFG.getConfig());
```

```
                Log.d(TAGTENSOR, " width = " + maskFG.getWidth() + " height
= " + maskFG.getHeight());


                final float scaleHeight = ((float) 2448) / 960;

                float scaleWidth = scaleHeight;

                Matrix matrix = new Matrix();

                matrix.postScale(scaleWidth, scaleHeight);


                maskFG = Bitmap.createBitmap(maskFG, 0, 0, 960, 960,
matrix, false); // 2448x2448

                Log.d(TAGTENSOR, " width = " + maskFG.getWidth() + " height
= " + maskFG.getHeight());


// совмещение первоначального фото и полученной маски 2448x2448

                final int[] fgArr = new int[(maskFG.getWidth() *
maskFG.getHeight())];

                maskFG.getPixels(fgArr, 0, maskFG.getWidth(), 0, 0,
maskFG.getWidth(), maskFG.getHeight());


                int[] bitmapArr = new int[(bitmap.getWidth() *
bitmap.getHeight())];

                bitmap.getPixels(bitmapArr, 0, bitmap.getWidth(), 0, 0,
bitmap.getWidth(), bitmap.getHeight());

                for (int i = bitmapArr.length - 1; i >= 48960; i--) {

                    fgArr[i] = Color.argb(Color.alpha(fgArr[i - 48960]),
Color.red(bitmapArr[i - 48960]), Color.green(bitmapArr[i - 48960]),
Color.blue(bitmapArr[i - 48960]));

                }

                for (int i = 48960 - 1; i >= 0; i--) {

                    fgArr[i] = Color.argb(0, 0, 0, 0);

                }

                final int maskFGWidth = maskFG.getWidth();

                final int maskFGHeight = maskFG.getHeight();

                maskFG.recycle();
```

```
                maskFG = null;


                bitmapFG = Bitmap.createBitmap(fgArr, maskFGWidth,
maskFGHeight, Bitmap.Config.ARGB_8888);
                imageResult.setImageBitmap(bitmapFG);
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        new Thread(new Runnable() {
                            @Override
                            public void run() {
                                savePhotoFile(bitmapFG, "_fg.png",
Bitmap.CompressFormat.PNG, 100);
                                //    savePhotoFile(bitmapPhoto,
"_orig.jpg", Bitmap.CompressFormat.JPEG, 100);
                            }
                        }).start();
                    }
                });
            }
        }).start();
        Log.d(TAGTENSOR, "FG END");
    }
```

ДОДАТОК Б
СЛАЙДИ ПРЕЗЕНТАЦІЇ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження методів ідентифікації при розпізнаванні візуальних образів

Науковий Керівник: доцент — Назаров О.С.

Виконав: студент групи ІПЗмзд-18-1 — Кірсанов А.С.

2020

1

Постановка задачі

- Маємо в наявності навчену нейронну мережу, здатну визначати об'єкти, що знаходяться на передньому плані. Об'єктами виступають люди і предмети, які знаходяться в контакті з людьми.
- Необхідно, скориставшись методами фільтрації, поліпшити якість роботи даної нейронної мережі.

2

Актуальність завдання

- Часто буває, що в наявності є навчена нейронна мережа для виконання певного завдання. При цьому можливий варіант, що з її допомогою необхідно виконати завдання, що трохи відрізняються від початкового або виявляється, що нейронна мережа недостатньо навчена для виконання свого завдання, наприклад, через малий обсяг навчальної бази.
- При цьому перенавчити або довчити нейросеть немає можливості через відсутність часу, фінансів або просто відсутність початкових кодів.
- В этом случае может помочь предварительная обработка исходного изображения, в частности фильтрация.

3

Програмна реалізація фільтрів (Matlab)

- ***Фільтр Винера***

При відсутності фокусу спотворююча система добре апроксимується циліндричної функцією розсіювання точки (ФРТ) радіуса r.

PSF = fspecial('disk',4);

J = deconvwnr (I, PSF);

- ***Фільтр контр***

Для підвищення контрастності зображень використовуємо високочастотну і низькочастотну фільтрацію одночасно. Додаємо вихідне зображення до обробленої за допомогою "top-hat" -фільтра і віднімаємо зображення, оброблене "bottom-hat" -фільтром

se=strel('disk', 3);

J=imsubtract(imadd(I, imtophat(I, se)), imbothat(I, se));

- ***Фільтр Гауса***

Центральний елемент маски цього фільтра має максимальне значення, відповідне піку розподілу Гауса (нормального розподілу).

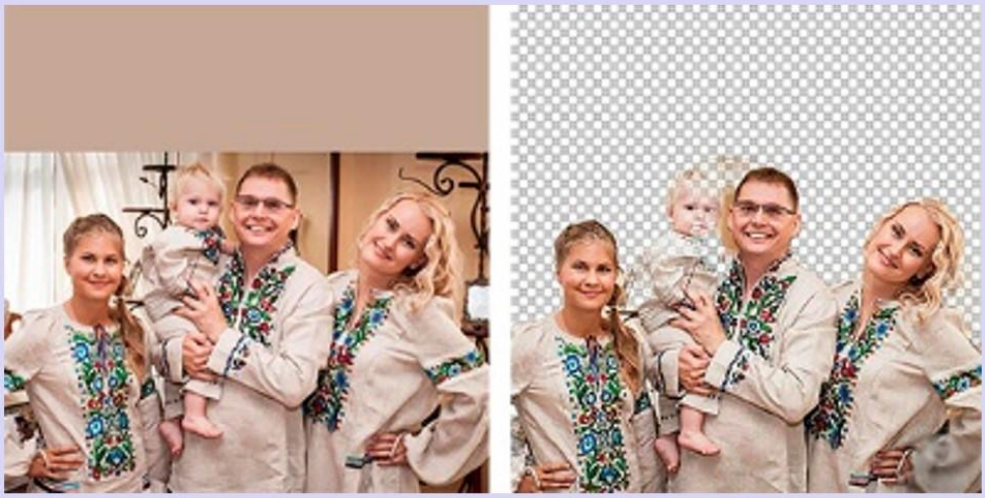
Фильтрация в координатной области

h1 = fspecial("gaussian", [nxn], Sigma);

J = imfilter(I, h1);

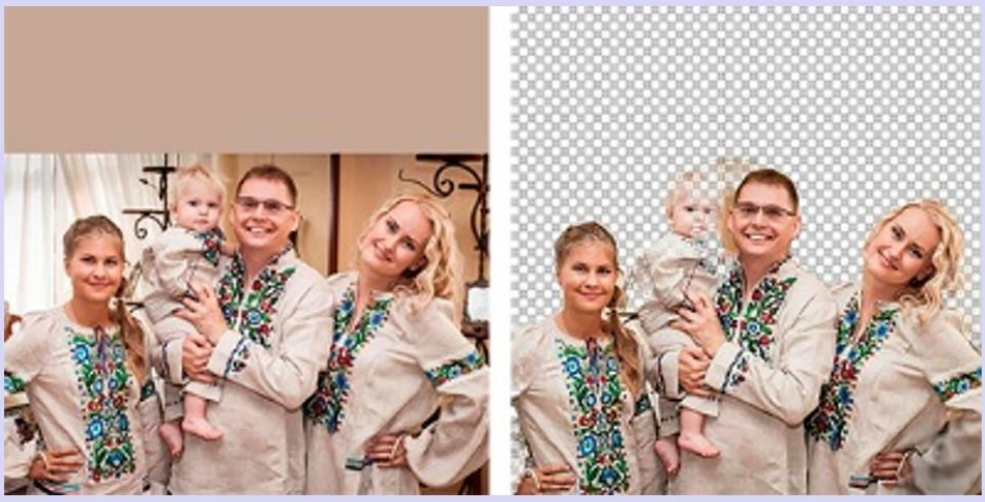
4

Результат без попередньої обробки вихідного зображення



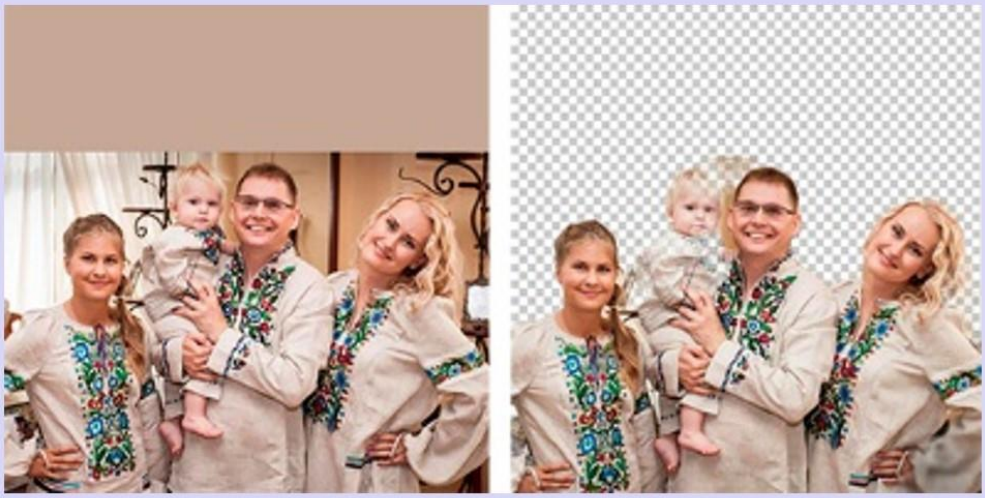
5

Результат з попередньою обробкою фільтром Гауса 0.7



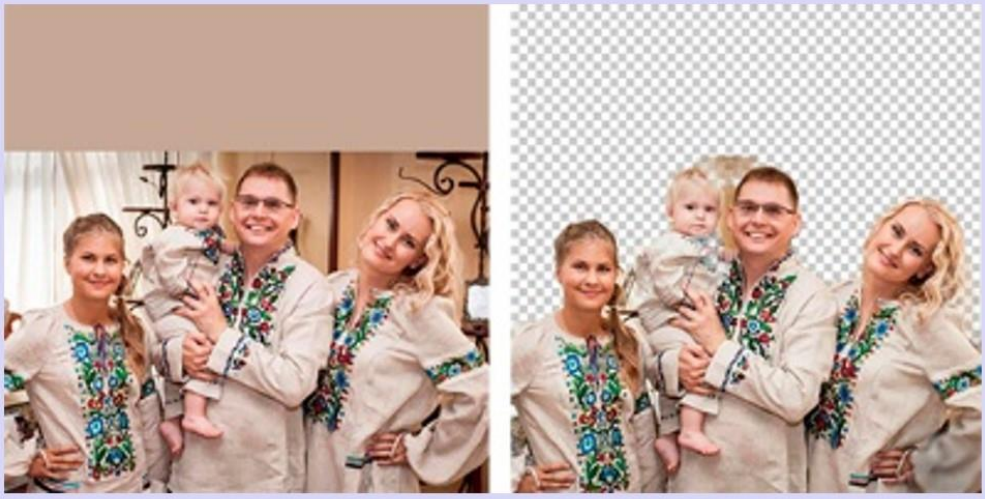
6

Результат з попередньою обробкою фільтром Гауса 2



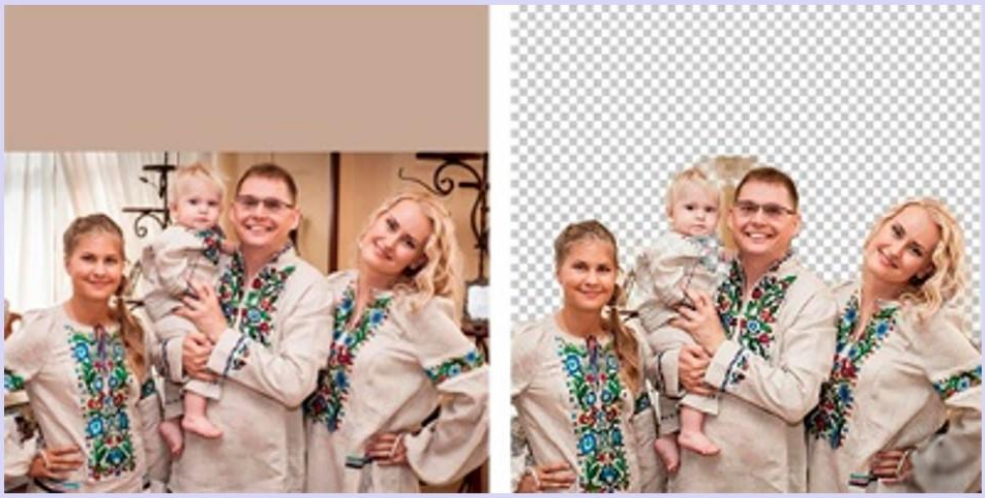
7

Результат з попередньою обробкою фільтром Гауса 3



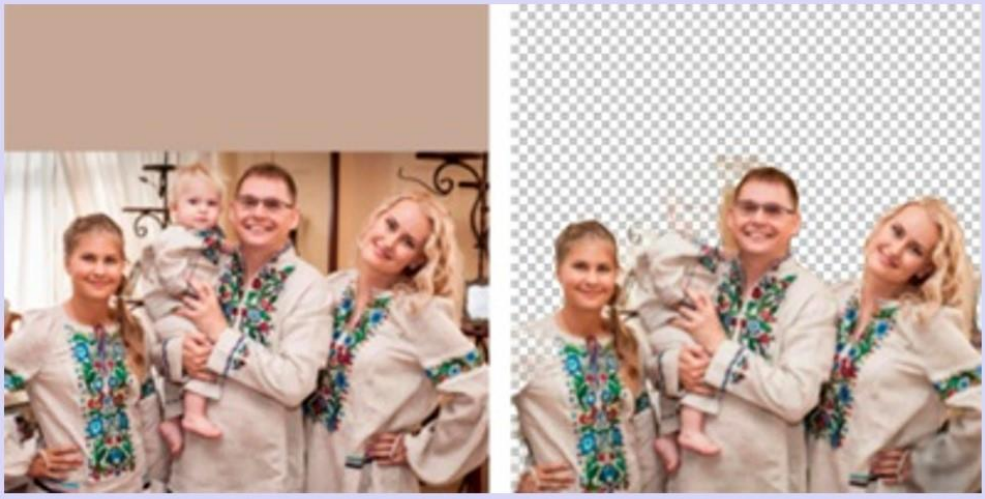
8

Результат з попередньою обробкою фільтром Гауса 4



9

Результат з попередньою обробкою фільтром Гауса 7



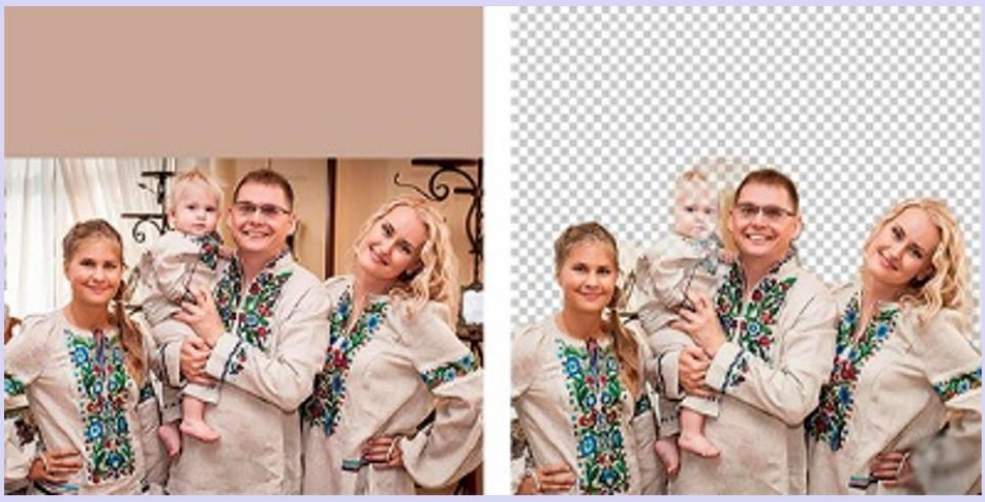
10

Аналіз попередньої обробки фільтром Гауса

- При збільшенні коефіцієнта фільтра Гауса поступово проявляються все більше нових деталей, які раніше не розпізнавати нейронною мережею як частина об'єкта.
- Але це поліпшення спостерігається до певного значення коефіцієнта фільтра Гауса, після якого якість обробки починає падати. Як видно з результатів, піком поліпшення якості обробки є значення коефіцієнта 4
- Починаючи зі значення 5 якість обробки незначно починає погіршуватися. Проявляється це в зникненні з зображення деталей, що належать об'єкту. На слайді ми бачимо вже значне погіршення якості обробки при значенні коефіцієнта 7.

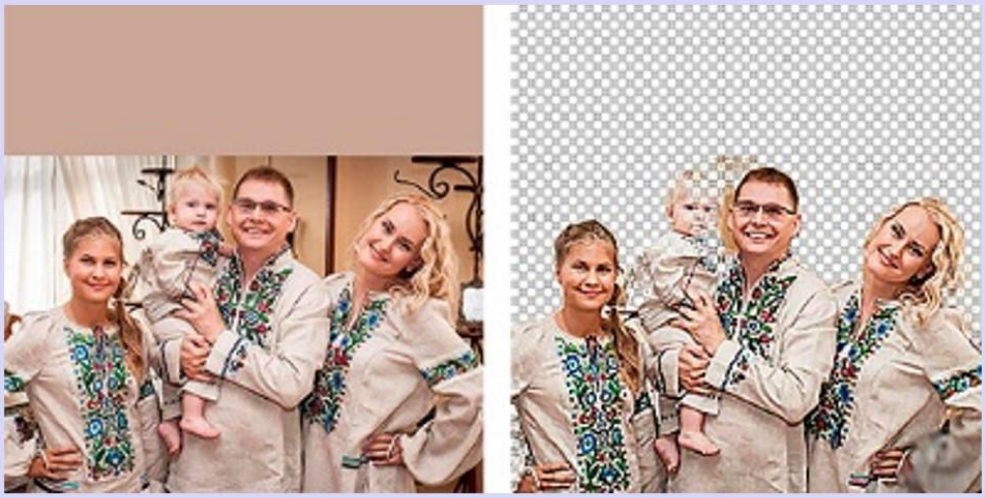
11

Результат з попередньою обробкою фільтром Вінера



12

Результат з попередньою обробкою підвищення контрастності



13

Аналіз попередньої обробки фільтром Вінера та фільтром підвищення контрасту

- *Фільтр Вінера*. При відсутності фокусу спотворююча система добре апроксимується циліндричної функцією розсіювання точки (ФРТ) радіуса r. На слайді представлений результат з попередньою обробкою фільтром Вінера з ФРТ радіусом 4. Як видно з результату, фільтрація даним типом фільтру не привела до позитивного результату.
- *Фільтр підвищення контрастності*. У нашому випадку він ймовірно міг би дати позитивний результат, так як одним з його ефектів є більш чітке виділення меж об'єктів. На слайді представлений результат обробки при використанні фільтра Вінера. Як видно з результату - очікуваного поліпшення обробки нами не досягнуто.

14

ВИСНОВКИ

- Результатом атестаційної магістерської роботи є аналіз результатів роботи нейронної мережі з використанням попередньої обробки вхідних даних, на підставі яких став зрозумілий подальший план робіт і досліджень. Встановлено, що найкращим чином на роботу використовуваної нейронної мережі за визначенням переднього плану впливає низькочастотний фільтр Гауса, проте ідеального результату добитися не вдалося. Подальшим шляхом досліджень може бути комбінація різних фільтрів, але для цього необхідно вже автоматичний перебір комбінацій і як наслідок більш потужні апаратні засоби.
- В цілому розглянутий метод поліпшення роботи нейронної мережі заслуговує на увагу, проте необхідно враховувати, що для кожної окремо взятої нейронної мережі і кола завдань необхідно підбирати індивідуальний пакет фільтрів.

15