Министерство просвещения Российской Федерации
МБОУ Аэрокосмический лицей имени Ю.В. Кондратюка

Проект: «Создание мобильного приложения “Решение химических уравнений”»

Ученика 10А класса
Киваева Ярослава
Научный руководитель:
Марковская Л.А.
учитель информатики в.к.к.

Новосибирск, 2019

Содержание

Введение

Актуальность

В последние десятилетия происходит стремительная информатизация всех сторон жизни общества. Появилась возможность обучаться, используя различные программы на телефонах, компьютерах и других электронных устройствах. Очень популярны приложения, процесс использования которых быстр и прост. В ходе работы было создано мобильное приложение, которое решает химические уравнения, изображённые на фотографиях. При разработке данного приложения были использованы методы глубокого обучения. На данный момент данное мобильное приложение не имеет аналогов в магазине мобильных приложений "Play Market".

Цель – создание мобильного приложения, которое способно решать химические уравнения, изображённые на фотографии

Задачи:

1. спроектировать архитектуру мобильного приложения;
2. разработать модуль, отвечающий за решение химических уравнений;
3. разработать модуль, отвечающий за обработку изображения, получаемого с камеры телефона;
4. объединить модули в единое приложение.

Практическая работа

Архитектура мобильного приложения:

• CalculatorActivity – активность, отвечающая за решение химических уравнений (является стартовой активностью)
• CameraActivity – активность, отвечающая за получение изображения с камеры и его обработки.
• SolutionChemicalEquations – класс, отвечающий за решение химического уравнения.
• Equation – класс химического уравнения.
• Compound – класс химического соединения.

Обработка изображения

• Монохромный фильтр
• Фильтр размытия
• Бинаризация изображения
• Нахождение связных контуров
• Деление исходного изображения на отдельные контуры

• Пропуск изображения каждого контура через свёрточную нейронную сеть и определение класса каждого контура.

Архетиктура свёрточной нейронной сети:
Модель состоит из 3 свёрточных слоёв и 1 полносвязного. Скорость обучения равна 1 стотысячной. Все гиперпараметры был подобраны экспериментальным способом.

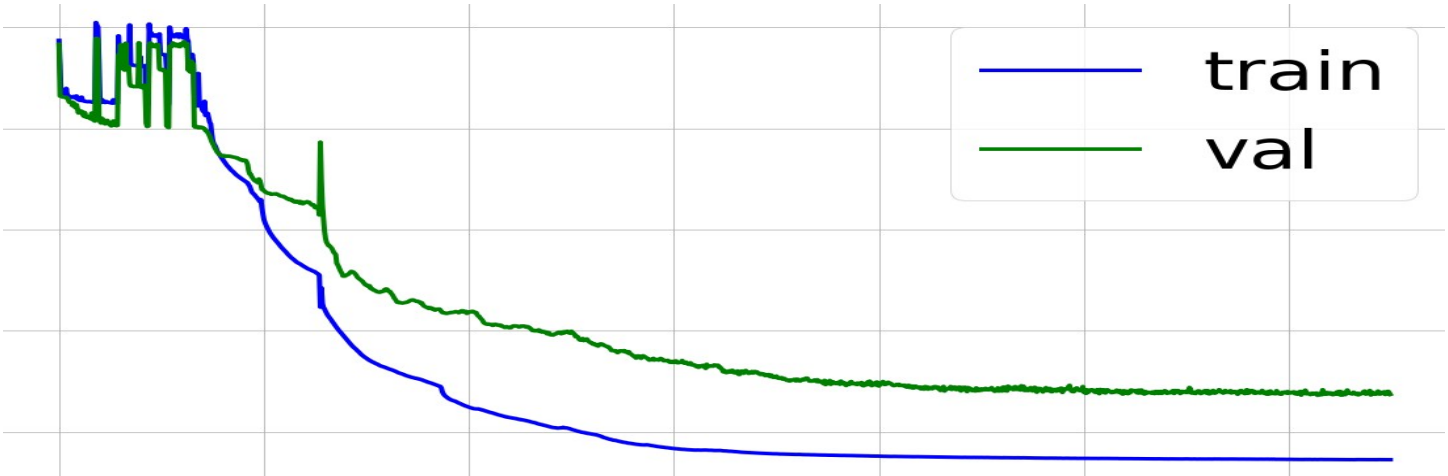
```
In [18]: model = NNClassifier(conv_net, lr=1e-5)
```

```
conv_net = torch.nn.Sequential(torch.nn.Conv2d(1, 16, 3, stride=2, padding=3),
                               torch.nn.ReLU(),
                               # 16x16x16

                               torch.nn.Conv2d(16, 32, 3, stride=2, padding=1),
                               torch.nn.ReLU(),
                               # 8x8x32

                               torch.nn.Conv2d(32, 64, 3, stride=2, padding=1),
                               torch.nn.ReLU(),
                               # 4x4x64

                               Flatten(),
                               torch.nn.Linear(1024, 36),
                               Softmax_layer())
```

Обучение нейронной сети:

```
In [65]: model.fit(X_train, y_train, epochs=500, batch_size=256,
                   valid_data=(X_valid, y_valid))
```

Модель нейронной сети обучалась 1500 эпох. Точность на тестовой выборке составила 72 процента.
Решение химического уравнения:



Решение химических уравнений осуществляется за счёт базы данных, в которой имеются 25 тысяч уравнений, каждой из которых соответствует 3 колонки: идентификатор уравнения, левая часть уравнения и правая часть уравнения. Вывод формируется путём вывода полей объекта уравнения.

Вывод: Создано приложение, решающие химические уравнения, которое апробировано на практике.

Список литературы

http://habrahabr.ru
http://stackoverflow.com

Приложение 1:

CalculatorActivity:

```
import android.Manifest;
import android.content.Intent;
import android.support.v4.app.ActivityCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import java.security.Permission;

public class CalculatorActivity extends AppCompatActivity {
    static{ System.loadLibrary("opencv_java"); }
    EditText input_text;
    TextView output_text;
    Button solve_button;
    Button image_button;
    String input;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_calculator);
        setPermissions();
        input_text = findViewById(R.id.input);
        output_text = findViewById(R.id.output);
        solve_button = findViewById(R.id.solve_button);
        image_button = findViewById(R.id.image_button);

        View.OnClickListener solveTask = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                input = input_text.getText().toString(); //ввод
                input = input.replaceAll("[,.\\ ]", "");
                SolutionChemicalEquations test = new SolutionChemicalEquations(input.split("\\+"),
getApplicationContext()); // соединения
                String output = "";
                for (Equation eq : test.getSolutionEquations()) {
                    output += eq.id + ": ";
                    for (Integer el: eq.getRight()) {
                        output += test.IntToCompound(el, (new
DbCursors(getApplicationContext())).getCursor("compound")) + " ";
                        //Log.i(TAG, el + " " + "+");
                    }
                    output += " :" + eq.frequency;
                    output += "\n";
                    //Log.i(TAG, (eq.getRight()).size() + "");
                }
                output_text.setText(output);
            }
        };

        View.OnClickListener takePhoto = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(CalculatorActivity.this, CameraActivity.class);
```

```
            startActivity(intent);
        }
    };

    solve_button.setOnClickListener(solveTask);
    image_button.setOnClickListener(takePhoto);
  }

  private void setPermissions() {
    ActivityCompat.requestPermissions(this, new String[]{
        Manifest.permission.WRITE_EXTERNAL_STORAGE,
        Manifest.permission.CAMERA,
    }, 1);
  }
}
```

CameraActivity:

```
import android.content.Intent;
import android.content.res.Resources;
import android.graphics.Bitmap;
import android.graphics.Color;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.StrictMode;
import android.provider.MediaStore;
import android.support.v7.app.AppCompatActivity;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.MediaController;
import android.widget.Toast;
import android.widget.VideoView;
import java.io.IOException;

import static android.graphics.ImageDecoder.*;
import static android.provider.MediaStore.Images.Media.*;

public class CameraActivity extends AppCompatActivity {
  static{ System.loadLibrary("opencv_java"); }
  public static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;

  private Button bt_image;
  private FrameLayout fl_camera;
  private Uri fileUri;
  private Bitmap bitmap;


  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_camera);

    takePicture();
    bt_image = findViewById(R.id.lab04_bt_imagem_capturar);
    fl_camera = findViewById(R.id.lab04_fl_camera);

    View.OnClickListener listener = new View.OnClickListener() {
      @Override
      public void onClick(View v) {
```

```
            takePicture();
        }
    };

    fl_camera.setOnClickListener(listener);

}

private void takePicture(){
    Intent i = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    fileUri = MediaHelper.getOutputMediaImageFileUri(getBaseContext());
    i.putExtra(MediaStore.EXTRA_OUTPUT, fileUri);
    startActivityForResult(i, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}

@SuppressWarnings("deprecation")
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    fl_camera.removeAllViews();


    switch (resultCode) {

        case RESULT_OK:

            ImageView ivDATA = new ImageView(this);
            try {
                if (Build.VERSION.SDK_INT < 28) {
                    bitmap = getBitmap(this.getContentResolver(), fileUri);
                } else {
                    Source source = createSource(this.getContentResolver(), fileUri);
                    bitmap = decodeBitmap(source);
                }
                ivDATA.setImageBitmap(bitmap);
            } catch (IOException e) {
                e.printStackTrace();
            }


            fl_camera.addView(ivDATA);
            break;


    }
}
}
```

Compound:

```
import android.content.Context;
import android.database.Cursor;
import android.util.Log;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class Compound {
    int id, frequency;
    String formula;
```

```
    List<String> oxidationNumbers = new ArrayList();
    boolean verified;
    private static final String TAG = "MyApp";
    public Compound(Integer input, Context context) {
        Cursor cursor = (new DbCursors(context)).getCursor("compound");

        cursor.moveToFirst();
        do {
            //Log.i(TAG, cursor.getInt(cursor.getColumnIndex("_id")) + "==" + Integer.parseInt(input));
            if (cursor.getInt(cursor.getColumnIndex("_id")) == input) {
                this.id = cursor.getInt(cursor.getColumnIndex("_id"));
                this.frequency = cursor.getInt(cursor.getColumnIndex("frequency"));
                this.formula = cursor.getString(cursor.getColumnIndex("formula"));
                this.oxidationNumbers =
Arrays.asList((cursor.getString(cursor.getColumnIndex("oxidation_numbers"))).split(","));
                this.verified = (cursor.getInt(cursor.getColumnIndex("verified")) == 1);
                //Log.i(TAG, cursor.getString(cursor.getColumnIndex("formula")));
                break;
            }
        } while (cursor.moveToNext());
    }


}
```

DataBaseHelper:

```
import android.content.Context;

import com.readystatesoftware.sqliteasset.SQLiteAssetHelper;

public class DataBaseHelper extends SQLiteAssetHelper {

    private static final String DATABASE_NAME = "equations.db";
    private static final int DATABASE_VERSION = 2;

    public DataBaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
        setForcedUpgrade();
    }
}
```

DBCursor:

```
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;

public class DbCursors {
    SQLiteDatabase db;

    public DbCursors(Context context) {
        DataBaseHelper DbHelper = new DataBaseHelper(context);
        db = DbHelper.getReadableDatabase();
    }

    public Cursor getCursor (String name) {
        return db.query(name, null, null, null, null, null, null);
    }
}
```

Equations:

```
import java.util.ArrayList;
import java.util.List;

public class Equation {
    int id, frequency;
    List<Integer> left;
    List<Integer> right;

    public Equation(int id, int frequency, List<Integer> left, List<Integer> right) {
        this.id = id;
        this.frequency = frequency;
        this.left = left;
        this.right = right;
    }

    public int getFrequency() {
        return frequency;
    }

    public List<Integer> getRight() {
        return right;
    }
}
```

SolutionChemicalEquations:

```
import android.content.Context;
import android.database.Cursor;
import android.util.Log;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

public class SolutionChemicalEquations {
    private static final String TAG = "MyApp";
    List<Equation> SolutionEquations;

    public SolutionChemicalEquations(String[] input, Context context) {
        //for (int i = 0; i < input.length; i++){
            //Log.i(TAG, "!" + input[i]);
        //}
        List<Integer> inputInt = CompoundToInt(input, (new DbCursors(context)).getCursor("compound")); // id
соединений
        List<Compound> inputCompounds = new ArrayList<Compound>(); // объекты соединений
        for (Integer s: inputInt) {
            inputCompounds.add(new Compound(s, context));
        }
        List<Equation> possibleEquations = PossibleEquations(inputCompounds, context);
        //Log.i(TAG,  possibleEquations.size() + "");
        for (Equation eq : possibleEquations) {
            for (Integer el: eq.getRight()) {
                //Log.i(TAG,  Integer.toString(el));
            }
            //Log.i(TAG,  "+");
        }
        SolutionEquations = possibleEquations;
    }
```

```
    public List<Equation> PossibleEquations (List<Compound> left, Context context) {
        //Log.i(TAG,  left.size() + "");
        //for (Compound el:left) {
        //    Log.i(TAG,  el.id + "");
        //}
        List<Integer> leftId = new ArrayList<>();
        for (Compound s : left) {
            leftId.add(s.id);
        }
        Cursor cursor = (new DbCursors(context)).getCursor("equation");

        List<Equation> possibleEquations = new ArrayList<>();

        cursor.moveToFirst();
        do {
            List<Integer> leftHelp = new ArrayList<>(leftId);
            List<String> listTempStr = Arrays.asList((cursor.getString(cursor.getColumnIndex("left"))).split("-")); //
_id
            List<Integer> listTempInt = new ArrayList<>();
            Integer TempInt;
            for (String s : listTempStr) {
                Log.i(TAG, leftId.get(0) + "");
                TempInt = Integer.parseInt(s);
                //if (cursor.getInt(cursor.getColumnIndex("_id")) < 10) Log.i(TAG, TempInt + "+" + leftHelp.get(0) + "+"
+ cursor.getString(cursor.getColumnIndex("_id")));
                if (leftHelp.contains(TempInt)) {
                    //Log.i(TAG, "help");
                    leftHelp.remove(leftHelp.indexOf(TempInt));
                }
                listTempInt.add(TempInt);
            }

            if (leftHelp.isEmpty()) {
                //Log.i(TAG, "+");
                List<String> listTempStrRight =
Arrays.asList((cursor.getString(cursor.getColumnIndex("right"))).split("-")); // _id
                List<Integer> listTempIntRight = new ArrayList<>();
                for (String s : listTempStrRight) {
                    listTempIntRight.add(Integer.parseInt(s));
                    //Log.i(TAG, s);
                }
                Equation tempEquation = new Equation(cursor.getInt(cursor.getColumnIndex("_id")),
cursor.getInt(cursor.getColumnIndex("frequency")),
                        leftId, listTempIntRight);
                possibleEquations.add(tempEquation);
            }

        } while (cursor.moveToNext());
        Collections.sort(possibleEquations, new FrequencyComp());
        Log.i(TAG,  possibleEquations.size() + "");
        return possibleEquations;
    }

    class FrequencyComp implements Comparator<Equation> {

        @Override
        public int compare(Equation e1, Equation e2) {
            if(e1.getFrequency() < e2.getFrequency()){
                return 1;
            } else {
                return -1;
```

```
            }
        }
    }

    public List<Equation> getSolutionEquations() {
        return SolutionEquations;
    }

    private List<Integer> CompoundToInt (String[] input, Cursor cursor) {
        List<Integer> inputInt= new ArrayList<>();
        for (String s:input) {
            cursor.moveToFirst();
            do {
                if (cursor.getInt(cursor.getColumnIndex("_id")) < 10) {
                    Log.i(TAG, cursor.getString(cursor.getColumnIndex("formula")).toLowerCase() + " + " +
s.toLowerCase() + " + " + cursor.getString(cursor.getColumnIndex("formula")).equalsIgnoreCase(s));
                }
                if (cursor.getString(cursor.getColumnIndex("formula")).equalsIgnoreCase(s)) {
                    inputInt.add(cursor.getInt(cursor.getColumnIndex("_id")));
                    break;
                }
            } while (cursor.moveToNext());
        }
        return inputInt;
    }

    public String IntToCompound (Integer id, Cursor cursor) {
        cursor.moveToFirst();
        do {
            if (cursor.getInt(cursor.getColumnIndex("_id")) == id) {
                return cursor.getString(cursor.getColumnIndex("formula"));
            }
        } while (cursor.moveToNext());
        return null;
    }
}
```