**5** ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование приложения является неотъемлемой частью разработки, а также одним из важнейших шагов на пути реализации того или иного функционала в программе. Тестирование программного обеспечения позволяет установить соответствие между реальным поведением программы, и ожидаемым.

**5.1** Верификация нейронной сети

Для верификации нейронной сети необходимо определить критерии оценки результата колоризации. Среди различных критериев оценки можно выделить следующие:

* соответствие результата колоризации оригинальному изображению, до его преобразования в полутоновое (при наличии такового);
* субъективное мнение пользователей; результат колоризации может не соответствовать действительности, но при этом быть приемлемым для пользователя (пример на рисунке 5.1).



Рисунок 5.1 – Пример ошибочной, но приемлемой колоризации

Несмотря на то, что субъективное мнение пользователей, является неплохой оценкой для работы нейронной сети, этот критерий не подходит для данного дипломного проекта. Главным недостатком является тот факт, что для того, чтобы получить адекватную оценку, необходимо набрать достаточное количество уникальных пользователей клиентского приложения, что невозможно без его выпуска, на стадии тестирования. Также стоит заметить, что для использования данного критерия оценки необходимо дополнительно реализовать пользовательский интерфейс, для назначения оценки, что не было запланировано на этапе функционального проектирования.

Соответствие результата колоризации исходному изображению до преобразования в полутоновое больше подходит для этапа тестирования продукта. Для того, чтобы оценка приемлемой, вместо относительно большого количества уникальных пользователей приложения необходимо достаточное количество запросов на колоризацию изображений. Несмотря на тот факт, что для реализации верификации данным способом не требуется построение дополнительного интерфейса для пользователя, необходим дополнительный модуль для оценки колоризации на клиентском приложении. Дополнительный модуль должен реализовывать алгоритм, принимающий исходное изображение, не подверженное никакой обработке, результат колоризации. В качестве результата алгоритм должен возвращать оценку результату колоризации.

Для данной цели был создан размещённый в одноимённом файле класс ColorizationEvaluator, который является частью java-пакета utils клиентского приложения. Данный класс имеет только статические методы и поля, что упрощает его интеграцию в разные части приложения:

public class ColorizationEvaluator {  
  
 private static Bitmap original;  
 private static Bitmap result;  
  
 public static void setOriginal(Bitmap original) {  
 ColorizationEvaluator.original = original;  
 }  
  
 public static void setResult(Bitmap result) {  
 ColorizationEvaluator.result = result;  
 }  
  
 public static Float getEvaluation() {  
 if (original == null || result == null  
 || original.getHeight() != result.getHeight()  
 || original.getWidth() != result.getWidth())  
 return null;  
 int height = original.getHeight();  
 int width = original.getWidth();  
 float evaluationSum = 0;  
 for (int i = 0; i < width; i++)  
 for (int j = 0; j < height; j++) {  
 int originalPixel = original.getPixel(i, j);  
 int resultPixel = result.getPixel(i, j);  
 int originalR = getRChannel(originalPixel);  
 int resultR = getRChannel(resultPixel);  
 int originalG = getGChannel(originalPixel);  
 int resultG = getGChannel(resultPixel);  
 int originalB = getBChannel(originalPixel);  
 int resultB = getBChannel(resultPixel);  
 evaluationSum += getEvaluationForChannel(originalR, resultR);  
 evaluationSum += getEvaluationForChannel(originalG, resultG);  
 evaluationSum += getEvaluationForChannel(originalB, resultB);  
 }  
 return evaluationSum / (height \* width \* 3);  
 }  
  
 private static float getEvaluationForChannel(int original, int result) {  
 float maxDelta = original > 127 ? original : 255 - original;  
 float error = Math.abs(result - original);  
 return (maxDelta - error) / maxDelta;  
 }  
  
 private static int getRChannel(int pixel) {  
 return (pixel & 0x00FF0000) >> 16;  
 }  
  
 private static int getGChannel(int pixel) {  
 return (pixel & 0x0000FF00) >> 8;  
 }  
  
 private static int getBChannel(int pixel) {  
 return (pixel & 0x000000FF);  
 }  
  
 public static void clearBitmaps() {  
 original = null;  
 result = null;  
 }  
}

Данный класс содержит следующие методы:

* setOriginal – метод для сохранения исходного изображения до его преобразования в полутон; вызывается классом GetBitmapTask после изменения разрешения изображения;
* setResult – метод для сохранения результата колоризации; вызывается классом DownloadingTask после декодирования изображения из строки формата Base64;
* getEvaluation – метод, возвращающий среднюю оценку точности колоризации пикселя по всему изображению;
* getEvaluationForChannel – инкапсулированный метод, возвращающий оценку точности колоризации для отдельного канала определённого пикселя;
* getRChannel – инкапсулированный метод, возвращающий значение канала, ответственного за наличие красного цвета; использует бинарные операторы;
* getGChannel – инкапсулированный метод, возвращающий значение канала, ответственного за наличие зелёного цвета; использует бинарные операторы;
* getBChannel – инкапсулированный метод, возвращающий значение канала, ответственного за наличие зелёного цвета; использует бинарные операторы;
* clearBitmaps – метод для очистки ссылок на Bitmap-объекты; вызывается после получения оценки классом DownloadingTask.

В качестве оценки точности колоризации изображения используется среднее значение точности колоризации отдельно взятых каналов изображения. Для одного канала, ответственного за наличие цветовой составляющей пикселя, оценка вычисляется как отношение модуля разности оригинального значения и результата колоризации к максимально возможной разности двух каналов.