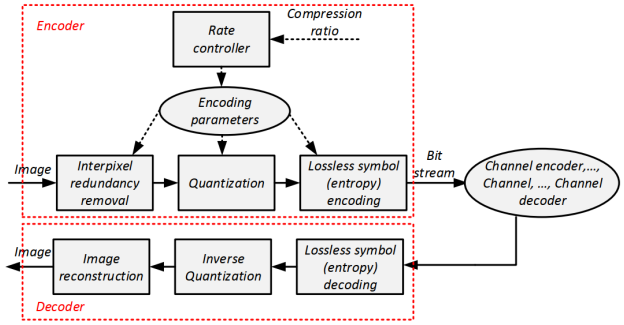
Кодирование изображений с предсказанием и квантованием

Общая схема сжатия изображений



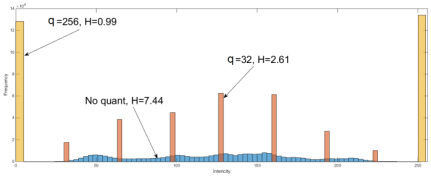
- Устранение избыточности, связанной со схожестью пикселей:
  - Преобразование цветового пространства;
  - Кодирование с предсказанием;
  - Кодирование с преобразованием.

  } 3 основных подхода
- Квантование (при сжатии с потерями);
- Адаптивное кодирование полученных данных:
  - Кодирование длин серий;
  - Побуквенное кодирование (Хаффман);
  - Контекстное адаптивное арифметическое кодирование.

Самый примитивный алгоритм кодирования изображений выглядит так:
На входе есть монохромное изображение
Кодер выполняет равномерное скалярное квантование (вносится искажение, причем это необратимое преобразование)
Далее выполняется энтропийное кодирование
(помним, что H(Z)<H(X), т.к. преобразование необратимое)

1. На входе монохромное изображение $X = \{x_1, x_2, ...\}$, $x_i \in \{0, 1, ..., 255\}$.
2. Кодер
   1. Квантование: $z_i = \left\lfloor \frac{x_i + \Delta/2}{\Delta} \right\rfloor$.
   2. Кодирование $\{z_1, z_2, ..\}$ (например, кодом Хаффмана).
   3. Битрейт R можно оценить как: $R \approx H(Z) = -\sum_j p_j \log_2(p_j)$, где p_j это вероятность $z_i = j$.
3. Декодер
   1. Декодирование $\{z_1, z_2, ..\}$.
   2. Восстановление $\hat{x}_i = \Delta \cdot z_i$.

Δ – шаг квантования



- Квантование – необратимая операция, поэтому $H(Q(X)) < H(X)$.

Чем больше шаг квантования, тем лучше сжатие и хуже качество

Как можно улучшить этот алгоритм:
Учесть то, что соседние пиксели зависимы друг от друга
Например, вместо обычного Хаффмана сделать Хаффмана "X при условии X" (но тогда надо будет работать с кучей таблиц Хаффмана) или адаптивное арифметическое кодирование с контекстом длины 1.
А можно реализовать кодирование с предсказанием, которое учитывает, что соседние пиксели зависят друг от друга.
Вычисляется ошибка предсказания, т.е. разность между текущим и предыдущим пикселем, квантуется и кодируется вместо самих пикселей (энтропия будет хуже, но не сильно - вообще это немного заметно, но зато так можно использовать Хаффмана с одной только таблицей)

"Неправильный" алгоритм кодирования

- Кодер
  1. Вычисление ошибки предсказания: $e_i = x_i - x_{i-1}$.
  2. Квантование: $z_i = \left\lfloor \frac{e_i + \Delta/2}{\Delta} \right\rfloor$.

Накопление ошибки в декодере:

1. Пусть $\hat{x}_0 = x_0$
2. $e_1 = x_1 - x_0, (x_0 = x_1 - e_1)$
3. $\hat{e}_1 = \Delta \cdot z_1 = e_1 + \delta_1$
4. $\hat{x}_1 = x_0 + \hat{e}_1 = x_0 + e_1 + \delta_1 = x_1 + \delta_1$

3. Кодирование z_i.

- Декодер
  1. Декодирование z_i.
  2. Деквантование: $\hat{e}_i = \Delta \cdot z_i$.
  3. Восстановление: $\hat{x}_i = \hat{x}_{i-1} + \hat{e}_i$.

5. $e_2 = x_2 - x_1, (x_1 = x_2 - e_2)$
6. $\hat{e}_2 = \Delta \cdot z_2 = e_2 + \delta_2$
7. $\hat{x}_2 = \hat{x}_1 + \hat{e}_2 = x_1 + \delta_1 + e_2 + \delta_2 = x_2 + \delta_1 + \delta_2$
8. $\hat{x}_n = x_n + \sum_{k=1}^{n} \delta_k$.

Но есть проблема - накопление ошибки. (Потому что при вычислении ошибки кодер вычитает настоящее значение, а декодер прибавляет искаженное), поэтому в кодер частично встраивается декодер - так кодер узнает искаженное значение, совпадающее потом со значением при декодировании.

В итоге алгоритм такой:

- Кодер
  1. Вычисление ошибки предсказания: $e_i = x_i - \hat{x}_{i-1}$.
  2. Квантование: $z_i = \left\lfloor \frac{e_i + \Delta/2}{\Delta} \right\rfloor$.
  3. Кодирование z_i.
  4. Восстановление: $\hat{x}_i = \hat{x}_{i-1} + \hat{e}_i = \hat{x}_{i-1} + z_i \cdot \Delta$.
- Декодер
  1. Декодирование z_i.
  2. Деквантование: $\hat{e}_i = \Delta \cdot z_i$.
  3. Восстановление: $\hat{x}_i = \hat{x}_{i-1} + \hat{e}_i$.

Ошибка в декодере:

1. $e_1 = x_1 - x_0$, $(x_0 = x_1 - e_1)$
2. $\hat{e}_1 = \Delta \cdot z_1 = e_1 + \delta_1$
3. $\hat{x}_1 = x_0 + \hat{e}_1 = x_0 + e_1 + \delta_1 = x_1 + \delta_1$
4. $e_2 = x_2 - \hat{x}_1$, $(\hat{x}_1 = x_2 - e_2)$
5. $\hat{e}_2 = \Delta \cdot z_2 = e_2 + \delta_2$
6. $\hat{x}_2 = \hat{x}_1 + \hat{e}_2 = \hat{x}_1 + e_2 + \delta_2 = x_2 + \delta_2$.
7. $\hat{x}_n = x_n + \delta_n$.

Эта идея заложена в стандарт JPEG-LS. Там используется предыдущий пиксель и еще те, что сверху над предыдущим и текущим. Находится разность между соседними парами из этих соседних пикселей. Если разность вертикальных пикселей больше, чем горизонтальных, значит, на изображении вертикальные полосы, и нужно из текущего пикселя вычитать тот, который сверху от него. Если по разности ничего не понятно (когда она одинаковая), тогда складываем пиксели слева и сверху от текущего и вычитаем третий, который над предыдущим. Так вычисляется предсказатель.