A2 – Ami Zou

Once you've finished implementing and testing your scheme, upload your code to a GitHub repository. Submit a link to your repo along with a 1-page write-up answering the following questions about your scheme.

Github repo link: https://github.com/ami-zou/COMP590-A2

1. What scheme or schemes did you try? If you came up your own idea, describe it here.

My idea is a combination both **2.prior-value context-adaptive** and **3.differential coding.** Essentially, I store the differences between each pixel at a certain position in the current frame and its value in the previous frame. This is how I implemented it:

- I have an array of length 64*64 = 4096 of all the pixel values in the last frame
- For each pixel position, I create a model for it (4096 models in total)
- For each model, I have a differences array with value ranging from -255 to +255 to store the difference between this pixel value to its previous value at the same positon. Also, I have a counter array (with length 255 + 1 + 255 = 511) to store the frequencies of each difference.

Then, I start to read the file:

- Firstly, I stored and encoded the total number of symbols and the bit range width
- Then, I read the initial 4096 pixels and store them in the lastFrame array
- After that, I read the rest of the (total_pixels - 4096) pixels:
    - For each index, I mod it by 4096 to get its absolute position in the frame
    - Then I retrieve the pixel model at that positon
    - I calculate the difference by subtracting the pixel value at the same positon in lastFrame, and then encode the difference
    - Lastly, I update both the model and the lastFrame pixel value

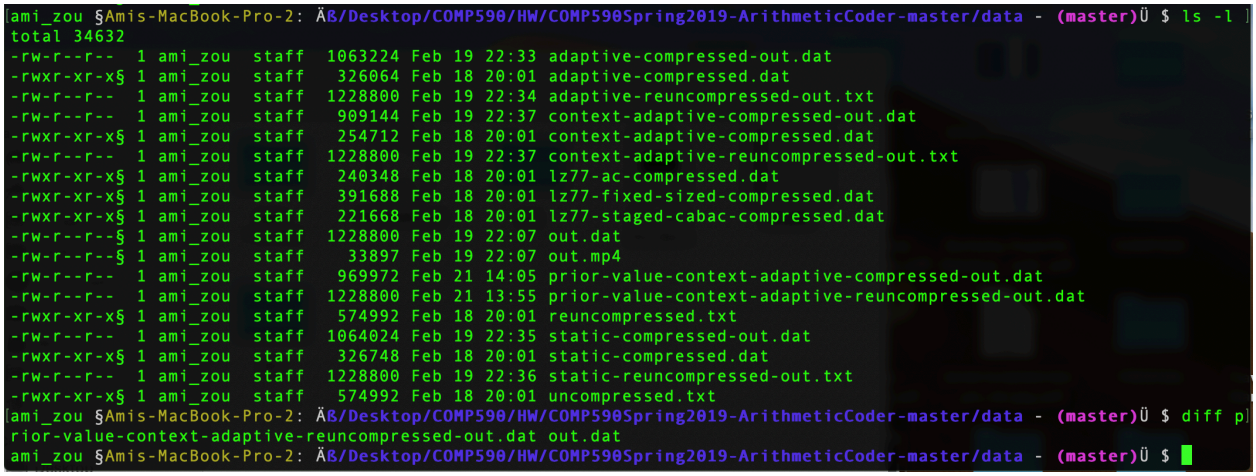
The decoder works the opposite way:

- Firstly, it reads the first 32 and 8 bits for the total_pixels and bit_range_width
- Then it reads the following 4096 pixels and decoded it directly
- It constructs the models and differences similar to the way in the encoder
- However, it needs to calculate the current pixel value by adding the decoded difference to the saved pixel value in lastFrame

2. Why do you think your scheme would do a good job predicting pixel values? How does your scheme exploit temporal and/or spatial coherence?

Yes, my scheme does a good job predicting pixel values because it exploits the temporal coherence. Each pixel doesn't change much from the last frame, and since I'm storing the difference instead of the actual value, it is very likely that some differences (such as 0, -1, -2, +1) will have a much higher value than others, which makes it highly efficient using arithmetic encoding

3. When applying the English text-based models (static, adaptive, and context-adaptive) to the video data, which scheme performed best?



Here is a screenshot of all the compressed and uncompressed video data of “out.dat” using the three models:

- Static: 1,064,024 bytes
- Adaptive: 1,063,224 bytes
- Context-adaptive: 909, 144 bytes

As we can see, context-adaptive arithmetic encoding provides a much better compression.

Does the scheme you developed compress better or worse than the English text-based models when applied to video data?

As you can see, my model compressed the raw video file to 969, 972 bytes in “prior-value-context-adaptive-compressed-out.dat”, which beats both the static and adaptive English text-based models but does worse than the context-adaptive English text-based model.

My understanding is that my initialization takes too much space. For the English text-based models, it only has 256 symbols and models, and each symbol stores the same array length of counters. However, I have 4096 models for the 64*64 frame, and each model has arrays of length 511 to store all the possible differences and their frequencies. Together, even though theoretically my scheme should be much more efficient than the English text-based models, storing all the extra values takes a lot of space.

If you weren't able to finish and test your own scheme, how do you think your scheme would fare in comparison to the English text-based models?

I finished my own scheme :D. It compresses and decompresses the file. As you can see in the screenshot, ‘$ diff prior-value-context-adaptive-reuncompressed-out.dat out.dat’ doesn’t give any difference, meaning that my encoder and decoder work correctly.

4. What is one change you could make to your scheme that might improve its results?

My scheme takes too much space. One thing I could do is to have an escape operator. Because of the temporal coherence, many differences values will not be used at all. However, in order to make CDF and arithmetic encoding work, I have to maintain the differences array with length 511 and set all the counts with a base value 1 to start with. In fact, if I have an escaper, I don’t

need to store the unused differences at all. I just need to encode the previously used counts and differences, and when I encounter a new difference:

- encode an escaper
- Then encode a fixed length representation of the new difference (9 bits for range 511)
- Lastly, update the differences and count array so we can use this difference in the future

The decoder will work the opposite way: updating the lastFrame and constructing the differences array when reading the bits. When reads an escaper, add the next 9 bits as a new difference to the differences array.