The authors proposed a new cache placement scheme that can seemingly increase the spatial locality of the access pattern of a program. This scheme works is set up on the premise that the data accessed by the program is similar in storage space in a stride interval. For example, a 2D image's pixel value does not change much both horizontally and vertically. If we would tradeoff data accuracy of the image, we can map the pixel values of the "image strip" into one single function based on some mapping function, so when the program is accessing pixels under/above a former accessed location, the cache would hit again (even though it should not) hence reduce cache access frequency. This is especially powerful when doing CNN as CNN is very noise tolerant and convolution is very low on spatial locality most of the time.

There are several minor weaknesses in this paper:

1. The parameters such as stride value is user defined and require ISA extension support. This puts great responsibility on the programmer and the compiler.
2. We can see a "pixelized" effect on the output of some results. Depending on the application this image artifact may or may not be ideal.
3. The mapping function seems computationally expensive, but I am not sure how expensive it is compared to the hash function that normal cache uses.

To mitigate the 1st weakness proposed above. We can utilize the DSP resource (assume it is available) within the system, and tag image IO from storage to memory as an input byte stream. During each stream, performs a DFT on the input stream parallelly and marks the major DFT component after each transmission. The cache controller should access the DFT and estimate the stride from the DFT factors. This would make the programmer have a better day and let hardware handles the trouble. However, in a standard CNN, all the input training set have the same size, and each layer have standardized interface, so the parameters such as stride and window are rather static. So, this solution might not be ideal.

Q&A: Assume we want to perform a 3D convolution on a 3d array instead of image. How would you expect the access time delay changes using the proposed bunk cache?

Q: The access time will slightly increase as the computation for bunk address would compute both for x, y, and z stride.