

Experiment

This case study serves as an experiment in determining how approximate computation techniques may change the performance of a Convolutional Neural Network (CNN) classifier. We have chosen to use the CiFar-10 data set containing $32 \times 32 \times 3$ pixel RGB images of various animals and vehicles. Each pixel is given by an integer 0 to 255 (a byte) which encodes the brightness of that pixel. All images are labeled 0 through 9, encoding the object that appears within. In most cases, the subject of the image extends to the edges of the image, which makes this data set favorable in this experiment.

To apply an approximate computing technique to each image, we use a *mute-bit* function. In doing so, each approximated pixel has gone from being stored as a specific 8-bit object to 8-bits of all 0's. This is used as a method of approximation to simulate a This approximation method was applied to exterior pixels of each image sample, to effectively create a border of N -approximated pixels in depth. Models with $N = 2, 4, 6, 8$ were compared to an unaffected baseline model. Examples of this approximation can be seen in fig. ().

IMAGES!

As a way to compensate for the previously described approximate computing technique, we also explore a method of attempting to recover the information in the lost pixel border. We do this by using sets of the unchanged pixels from the center of the image and copy and pasting them to the outside of the image. In doing this, the previous N -pixel border of muted bits is now replaced with features remaining. Thus, key features that were preserved in the image's center have been duplicated to the previously muted bits on the outside. This can also be seen in fig. ().

To test how this method works over different model architecture variants, we use a system inspired from the *VGG-16* CNN model. This model uses groups of layers that work together to produce feature maps which then pass that map on to the next set of layers. We define a *layer group* to be three immediately adjacent layers in a CNN where the first two layers are 2D convolution layers with identical number of output filters and kernel sizes followed by a Max-Pooling layer or an Average Pooling layer. The output of that layer can be fed into another similar layer group or a set of flattening and sense layers moving directly to output.

We test models with one and two layer groups, each containing a 64 feature map output, with a $k \times k$ kernel. We test each number of layer groups with $k = 2, 3, 4, 5, 6$ as well to analyze behavior of 10 unique, but related model variants.

RESULTS!

Conclusion

To mean any changes in performances of the neural-network classifier, we track the (1) average validation loss value, (2) average validation precision score and (3) average validation recall scores across all 10 classes.