Reasonable SOTA: My architecture was very similar to the usual ResNet model. However I decided to make some slight changes based on different research papers.
- Usually in the residual block for the Resnet the combination of the input and residual function are put through a relu function. Based on the results in the following paper (https://arxiv.org/pdf/1603.05027.pdf) I decided to not use a relu function. So the next output of the residual block would be the input plus the residual function. Using this technique in the paper would allow me to reduce overfitting and allow me to tune my layer depth in order to make my model deep without running into issues.
-Weight initilization for filters for convolutional layers. Based on the following paper (https://arxiv.org/abs/1502.01852) they made a big discovery that the weight initialization used was much better than xavier weights.
-Additionally the batch norm that is typically seen after each convolution layer in ResNet was replaced with a group norm layer. The following paper (https://arxiv.org/pdf/1803.08494.pdf) shows through experimentation that group norm leads to smaller error curves when training on models like ResNet50.
-Data Augmentation was utilized specifically cropping and flipping. A combination of data augmentation techniques proved to produce better results compared to just choosing one. The data augmentation techniques I decided to peruse was inspired by these two papers (https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0) (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9966095/)
- The only form of regularization I decided to use was dropout.
-Dropout had a significant impact on the loss of the function. The loss before implementing droupout was very "jittery" or was prone to oscillating between several different values. While the loss did decrease it wasn't at a satisfactory rate. Implementing dropout decreased the oscillation between different loss values.
- Several experiments of the CIFAR10 training set were attempted in order to figure out the best possible hyper parameters. When using the Adam optimizer I decided to increase the learning rate from the default value because after 500 plus iterations the loss and accuracy would stagnate and become consistent (around 20%)
- The amount of experiments were limited as when increasing my batch size I would get a memory issue on my computer. Also all experiments were trained on a CPU and not a GPU. As my previous test showed increasing the number of iterations would increase accuracy the CPU would take longer to train the model.