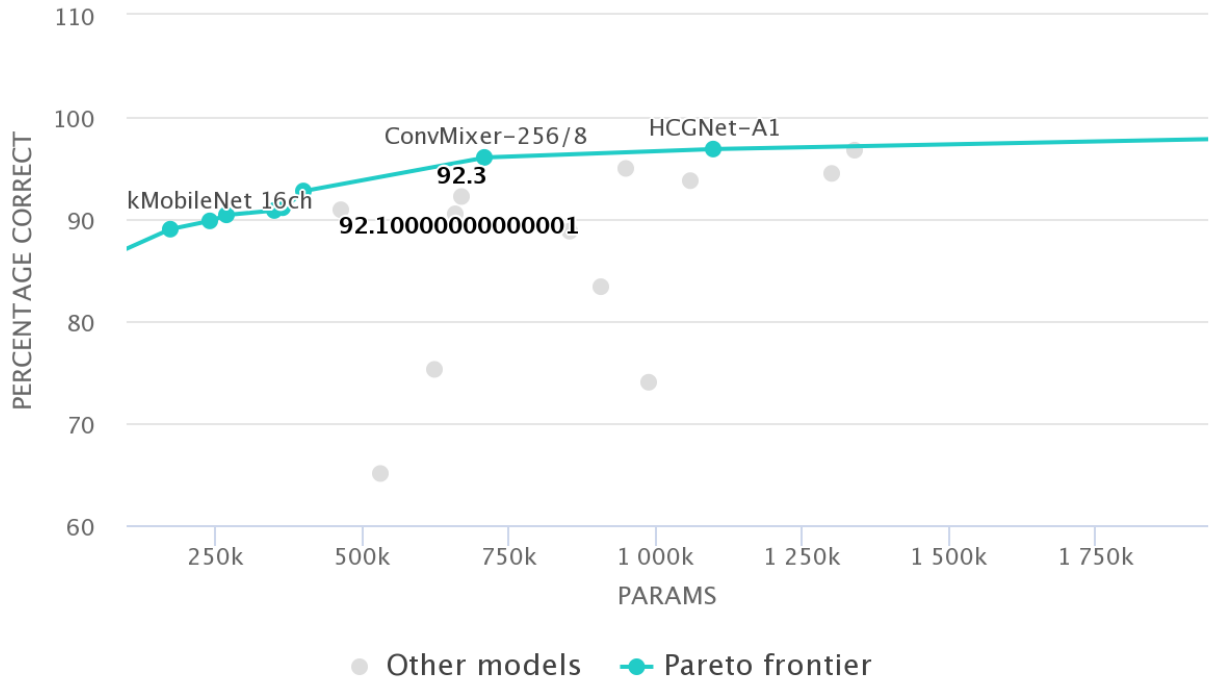
In this project, I implemented a residual neural network and used it to classify images in both the CIFAR-10 and CIFAR-100 datasets. For the CIFAR-100 dataset, our performance target was to get our model to achieve a 90% top-5 accuracy.

As for the CIFAR-10 dataset, the goal was to achieve a performance at the same level as what's considered "state of the art". If we take state of the art to mean the best performing classifiers which are able to get as high as 99.6% our goal for this project becomes impractical. Such classifiers can have tens of millions of parameters and be trained on several GPUs for countless hours. It would be too difficult to achieve similar results on our own personal computers which are much less powerful. Instead we need to determine what is a good and reasonable performance goal given the resources that we have.

To determine what a good performance goal would be, I looked at the performance of classifiers that used similar resources as to what I had available. The graph in the figure below shows accuracy as a function of the number of parameters used. Experimentation with different architectures showed me that my model would likely fall into this range of parameters. I took the average of the accuracy of all of these models to around 85% which I thought seemed like a reasonable goal to have for this project.



However, after discussing the performances of our models with other classmates, I found that very few people were able to get above 80% test accuracy. I think my performance goal was too high and the reason is because it didn't account for training time. As I will discuss later on, training time severely limited the amount of experimentation I could do with my model. To see if one architecture was better than another, I would have to let the two models run for a very long time before I could actually compare it. Otherwise, I would potentially be creating a bias for models that can converge quickly but to a high loss. I couldn't tune my hyperparameters as

much as I would have liked either. Taking this into account, I believe that a better performance goal for the CIFAR-10 dataset would be something from 75% to 80%.

The first obstacle I ran into in training my model was that my model's training loss would change rapidly and become very large and would never go below 2.3. At first, I believed that this was because my architecture was too simple for CIFAR-10. So I increased the size of my model significantly to around 7 million parameters but still the training loss never went below 2.3.

After trying out different things, I realized that the reason why this happened was because I had set the standard deviation of the initialization of the weights of the kernels for my convolutional layer to be sqrt(2/(length + width)). After reading the PReLu paper, I realized that my standard deviation was too large and should instead be sqrt(2/(length*width)). After doing this, my training loss for the cifar-10 dataset began to decrease much more quickly.

After that, I hit my next obstacle when my validation accuracy stagnated at 65%. When I got here, I tried out several things. I first tried resizing the images to be 128x128 so that I could increase the stride of my convolution layers. By increasing the stride, the kernel would cover more area and as a result, the receptive field of the network would increase. However, I found that this made training my model very slow and didn't really seem to improve accuracy either.

I also tried normalizing the inputs images to my model so that they would have 0 mean and unit standard deviation. However, I found that it was slightly better to use a min-max scaling instead so I decided to use that. I then tried gray scaling the images by averaging them along the three channels. The idea is that by averaging, we can get rid of information that might not be necessary for classifying the objects which will make it easier for the model to learn. Doing this made my model evaluate batches slightly more quickly but at the same time, the validation loss decreased much more slowly. This likely means that averaging removed key important features that were necessary for the model to generalize outside of its training set.

I increased the initial population of my genetic hyperparameter tuner in the hopes that the algorithm explores a new region of hyperparameters and finds a better combination. This somewhat helped but I found that the genetic tuning was just too slow due to how long training took. Instead, I was forced to manually experiment with different parameter values and see how it affected my validation loss. This did help increase my validation accuracy which was now at 72%.

Finally, I realized that the reason why I was having so much difficulty increasing my validation accuracy was because I wasn't using enough regularization techniques. My model was fairly large but using such a size for classifying the CIFAR-10 dataset wasn't unheard of. A lot of the leading models in CIFAR-10 use tens of millions of parameters. In order to ensure that my model didn't overfit, I decided to use data augmentation to add noise to the training set. I randomly flipped images, randomly changed the brightness, and randomly cropped images to a 24x24 size and then padded them to their original 32x32 size. All of this added noise to the training set which helped prevent the model from overfitting on it and instead encouraged it to learn a more "general solution".

In the end, the model using around 7.1 million parameters was able to achieve a test accuracy of 70.4% and an accuracy of 74.9% on the validation set for CIFAR-10. I unfortunately wasn't able to meet my performance goal but I believe that this was largely because my

computer wasn't powerful enough to train the model quickly. By training more quickly, I can more easily experiment with different techniques.

For the CIFAR-100 dataset, I actually ended up using the same architecture but I didn't use the random cropping and padding in my data augmentation. When I used that, my training loss never dropped below 4.3.

Beyond this however, no matter what I tried, the architecture that I had used for CIFAR-10 ended up being the one that gave me the best performance. Much of what I had tried, I had tried before with CIFAR-10 as well so I won't go into it. With it, I was able to achieve a top-5 accuracy of 76.1% on the validation set and 73.2% on the test set. The table below summarizes my models' performances on CIFAR-10 and CIFAR-100 as well as the number of parameters used by each model.

Dataset	Test Accuracy (Top-5 for CIFAR-100)	Validation Accuracy (Top-5 for CIFAR-100)	Number of Parameters
CIFAR-10	70.4%	74.9%	~7.1 million
CIFAR-100	73.2%	76.1%	~ 7.1 million