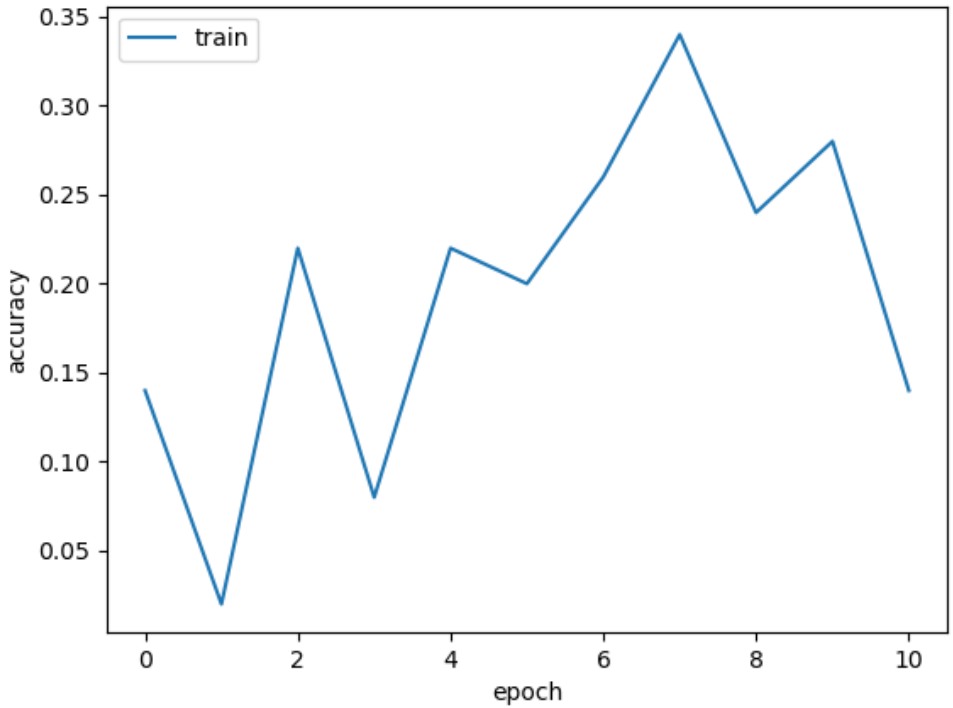
Part-1 ConvNet

Put your learning curve here:



My CNN Model

Describe your model design in plain text here:

I first added convolution layer and batch normalization layer with 3x3 kernel. This is to grab the features of the image and create a feature map. Relu is used for the convolution layers. Then, I applied max pooling layer. Max pooling layer right after convolution layer will select the max element from the feature map. I tried to add another set of convolution layer and batch normalization layer to get into more detail and increase accuracy. Then, I used a linear layer at the end.

Describe your choice of hyper-parameters:

Several parameters were tuned to get the highest final accuracy. For the batch size, I tried 64, 128, and 256. I was expecting better computational speed, but it didn't make a huge difference. At the end, I ended up using 64. Moreover, learning rate is tuned. 0.001, 0.0001 is tested and as a final, I used 0.001. It clearly had better learning when I applied 0.001 instead of 0.0001. Moreover, epochs was tested. I tried 10 (default) and 20. My initial thought was, higher epochs will make accuracy to constantly go up. It did but after 10 epochs, the difference of accuracy was not that much. When I ran my model, the accuracy difference between epochs 10 and 20 was around 0.1.

What's your final accuracy on validation set?

My final accuracy on validation set is 0.7220

Data Wrangling

What’s your result of training with regular CE loss on imbalanced CIFAR-10?

Fill in your per-class accuracy in the table

	Class 0	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**	**Class 6**	**Class 7**	**Class 8**	**Class 9**
CE Loss	0.8700	0.7840	0.4000	0.2970	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000

Best Accuracy: 0.2361 (with epochs: 10, learning_rate: 0.001 reg: 0.0005)

What's your result of training with CB-Focal loss on imbalanced CIFAR-10?

Tune the hyper-parameter beta and fill in your per-class accuracy in the table

	Class0	**Class1**	**Class2**	**Class3**	**Class4**	**Class5**	**Class6**	**Class7**	**Class8**	**Class9**
beta=0.9999	0.6190	0.6810	0.2480	0.1130	0.2530	0.3660	0.3340	0.3180	0.3250	0.2830
beta=0.9	0.8770	0.7960	0.4140	0.2480	0.0160	0.0020	0.0000	0.0000	0.0000	0.0000
beta=0.5	0.8770	0.7730	0.4250	0.2450	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Beta=0.1	0.8960	0.7320	0.3800	0.2680	0.0240	0.0010	0.0000	0.0000	0.0000	0.0000

Best Accuracy: 0.3540 (with epochs: 10, learning_rate: 0.001 reg: 0.0005, beta=0.9999)
Best Accuracy: 0.2353 (with epochs: 10, learning_rate: 0.001 reg: 0.0005, beta=0.9)
Best Accuracy: 0.2320 (with epochs: 10, learning_rate: 0.001 reg: 0.0005, beta=0.5)
Best Accuracy: 0.2301 (with epochs: 10, learning_rate: 0.001 reg: 0.0005, beta=0.1)

Put your results of CE loss and CB-Focal Loss(best) together:

	Class0	**Class1**	**Class2**	**Class3**	**Class4**	**Class5**	**Class6**	**Class7**	**Class8**	**Class9**
CE Loss	0.8700	0.7840	0.4000	0.2970	0.0100	0.0000	0.0000	0.0000	0.0000	0.0000
CB-Focal	0.6190	0.6810	0.2480	0.1130	0.2530	0.3660	0.3340	0.3180	0.3250	0.2830

Best Accuracy: 0.2361 (with epochs: 10, learning_rate: 0.001 reg: 0.0005)
Best Accuracy: 0.3540 (with epochs: 10, learning_rate: 0.001 reg: 0.0005, beta=0.9999)

Describe and explain your observation on the result:

Comparison between CE loss and Focal loss was quite successful. For CE loss, I got best accuracy of 0.2361. For Focal loss, I got tuned accuracy of 0.3540. It was quite interesting to see how the accuracy changes as I tuned the beta value. I tried 4 different beta values. (0.9999, 0.9, 0.5, and 0.1) Beta value of 0.9, 0.5, and 0.1 had similar accuracy with CE loss. One thing that I observed is that as beta value decreases, it put more weights toward the majority class. Which means, first 2 or 3 class had high accuracy but all the other classes had accuracy of 0.0000. As beta value increased, accuracy of majority class decreased but accuracy of minority class started to show up and increase. The result came out to be this way because of the class-balanced focal loss function. As beta value increases, the values of weights became more balanced compared to the low beta values. ((1-beta)/(1-beta^n)) By using effective number of samples, it was successful to balance out the alpha (traditional FL).