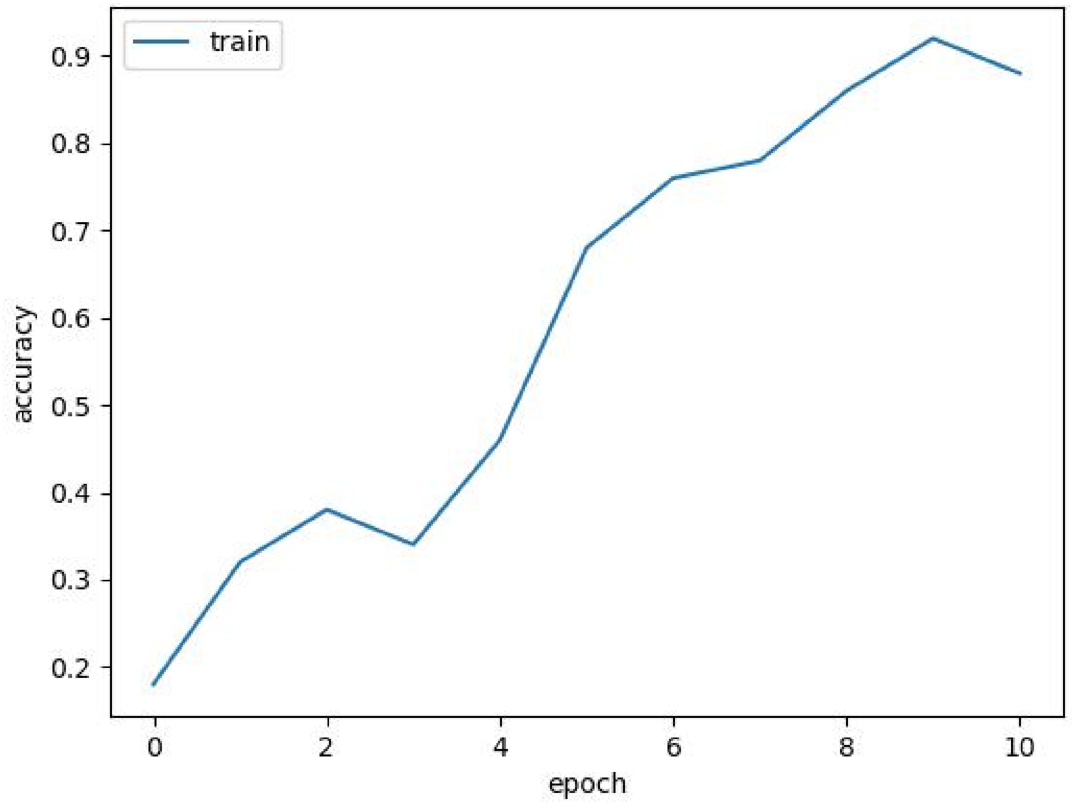
Convolutional Neural Network on CIFAR-10 Image Classification

Part-1 ConvNet

learning curve of accuracy:



My CNN Model

Model architecture and model design reasoning:

- My model has 3 convolutional layers one after another. The output channel is increased incrementally 3->32->64->128
- All convolutional layers have kernel of size 3, stride=1, and padding=1.
- Each convolutional layer uses a ReLU activation. Then the activation applied with a Batch Normalization.
- A max pooling layer is applied after the 3rd convolutional layer and it’s batch normalization. The max pooling layer has the stride o 2.
- After the max pooling layer, 2 fully connected layers are applied, finally reduce the dimension to 10 classe.

Reasoning:

1. Multiple convolutional layers extract useful information from the data. It can increase the model's ability to learn hierarchical representations of the input data.
2. Add padding of size 1 to maintain the size of the original image.
3. By having a deep network, it can increase the representation power and improve feature extraction and generalization.
4. Use ReLU activation for the non-linearity, which is proven to perform better than others
5. Use Max pooling to down-sample and reduce the spatial dimensions of the feature maps.
6. Use batch normalization to improve the stability and speed of training. It normalizes the activations of the neurons in a network, which can have significant benefits for the optimization process.

Hyper-parameters choices and reasoning:

Hyper-parameter:

batch_size: 128	learning_rate: 0.01	reg: 0.0005
epochs: 20	steps: [12, 18]	warmup: 0
momentum: 0.94		

Reasoning:

- if the learning rate is too big can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck. So the learning rate of 0.01 can best serve the learning process.
- Regularization of 0.0005 is not too big or too small. If too high, the model might be underfitting the data. If too low, the model might be overfitting the data. 0.0005 was chosen because it generates a model with good accuracy results.
- The batch size of 128 best suits the current learning rate and regularization, it proves enough invariance for generalization.
- Epochs of 20 prove good validation accuracy and let the model converges
- Momentum of 0.94 provides a good learning result.

What's your final accuracy on validation set?

0.8206

Data Wrangling

Result of training with regular CE loss on imbalanced CIFAR-10

Best per-class accuracy after tuning:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
CE Loss	0.9270	0.8320	0.4410	0.3180	0.0100	0.0000	0.0760	0.0000	0.0000	0.0000

Result of training with CB-Focal loss on imbalanced CIFAR-10

Per-class accuracy on different beta value:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
beta= 0.99	0.9350	0.8410	0.4970	0.4310	0.0490	0.0000	0.0130	0.0000	0.0000	0.0000
beta= 0.999	0.8930	0.8140	0.3320	0.2840	0.1000	0.0940	0.2780	0.2520	0.0130	0.0230
beta= 0.9999	0.6120	0.7160	0.1440	0.1460	0.2740	0.3310	0.3200	0.3710	0.3450	0.1100
beta= 0.99999	0.6000	0.6980	0.1870	0.2820	0.2370	0.1910	0.3280	0.3850	0.3930	0.2270

Results of CE loss and CB-Focal Loss(best) comparison:

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
CE Loss	0.9270	0.8320	0.4410	0.3180	0.0100	0.0000	0.0760	0.0000	0.0000	0.0000
CB-Focal	0.6000	0.6980	0.1870	0.2820	0.2370	0.1910	0.3280	0.3850	0.3930	0.2270

Verification the correctness of the focal loss solution:.

I tested whether the reweight process is correct, I printed the sample of the imbalanced data:
number of sample = [5000, 2997, 1796, 1077, 645, 387, 232, 139, 83, 50]
Then I calculated the correct adjust weight by the updating rule under different beta value, and I tested that the weights from the program are the same as I calculated by the updating rule:

Beta = 0.99999, adjusted weight = [0.04120926, 0.06807029, 0.11291209, 0.18761758, 0.31260274, 0.52033348, 0.86729799, 1.44690375 2.4224497, 4.02060312]

Beta = 0.9999, adjusted weight =[0.05061136, 0.07689979, 0.12113387, 0.19503667, 0.31880537, 0.52458985, 0.86834894, 1.44262727, 2.40922305, 3.99272382]

Beta = 0.999, adjusted weight =[0.05061136, 0.07689979, 0.12113387, 0.19503667, 0.31880537, 0.52458985, 0.86834894, 1.44262727, 2.40922305, 3.99272382]

Beta = 0.99, adjusted weight =[0.78383248, 0.78383248, 0.78383249, 0.78384809, 0.78503361, 0.80020139, 0.86815906, 1.0414146, 1.38542928, 1.98441651]

Describe and explain the observation on the result:

Observation:

- The sample size is decending as class number increases. With the cross-entropy loss, the final acuracy is high in the first few accuracy, but the accuracy is decreasing rapidly as the class sample size decreases. Class 1 accuracy is above 0.9, but class 4 accuracy is as low as 0.01, the classes after have accuracy of 0.
- With the Focal Loss, accuracy accross classes is less imbalanced, no class has accuracy of 0, most classes are flucturate around 0.4 accuracy.
- With the Focal Loss, the average performace accross classes increase as the beta value increases.
- When beta = 0.99, the classes with more sample have high accuracy, but classes with less sample have 0 accuracy, which is very imbalanced.
- When beta increases to 0.99999, the classes with more sample have lower accuracy and the classes with less sample have higher accuray compares to when beta = 0.99. The accuracy distribution is more balanced with higher beta value
- Overall. the cross-entropy loss generate more imbalanced accuracy than the Focal Loss. And Focal Loss trained model has higher average accuracy
- Focal Loss model with lower beta value generates more imbalanced accuracy and lower average accuracy, which is almost the same as cross-entropy loss model

Describe and explain the observation on the result

Explanation:

The definition of the class-balanced focal loss is as below:

$$CB_{focal}(z, y) = -\frac{1-\beta}{1-\beta^{n_y}} \sum_{i=1}^{C} (1 - p_i^t)^\gamma \, log(p_i^t)$$

We can see that the reweight depends on both beta and sample size. With the same beta value, the larger the sample size is, the smaller the focal loss will be. This is why with focal length the classes with different sample sizes perform more balanced.

Also, with the same sample size, the bigger beta value will reweight the loss according to the sample size more drastically, which means the big beta value will have a stronger reweight effect. This is the reason why the accuracy distribution across the classes is more balanced when beta = 0.99999.

A more balanced accuracy distribution can improve the performance across the whole set, that’s why more balanced data will have higher average accuracy.