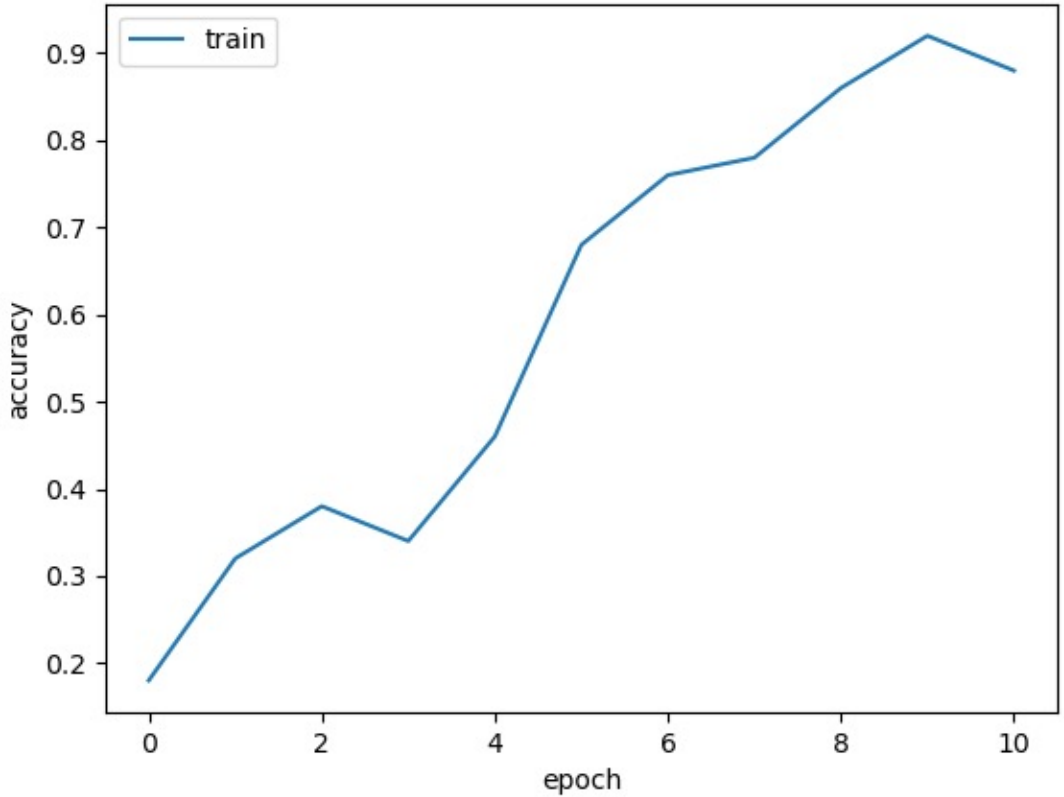
Assignment 2 Writeup

Name: Steven Rivadeneira
GT Email: stevenrr@gatech.edu

Part-1 ConvNet

Part1 Learning curve:



My CNN Model

Describe your model design in plain text here:

My model design description:

- My model has a total of 9 convolutional layers in succession. The primary building block of my model is “3 convolutional layers” in series, each set of 3 convolution layers outputs the same channel size. The output channel is increased incrementally in later layers. The first 3 convolutional layers’ output channel is 32, then the next 3 convolutional layers output channel is 64, then the final 3 convolutional layers output channel is 128.
- Each convolutional layer has kernel of dimension 3, stride=1, and padding=1.
- Each convolutional layer has a ReLU activation afterwards.
- Each convolutional layer has a Batch Normalization applied after the ReLU activation.
- Max pooling is done after each set of 3 convolutional layers. Max pooling occurs after the batch normalization. It is applied after the 3rd convolutional layer, after the 6th, and after the 9th
- There is also a residual that was added after the 3rd layer, and 6th layer. To downsample and match dimensions of residual (in shallow layer) with the hidden layers (in deeper layer), there is a convolution and a batch normalization applied to the residual.

Describe your model design in plain text here:

My model design reasoning:

- Convolutional layers are added for the layers to extract useful patterns within the images at various scales. By having layers at different depths in the network, the intent is to allow the NN flexibility in learning various patterns at different scales. This is done in combination with max pooling (explained below). In addition, output channel size is increased in order to compensate for height and width reduction as the layers get deeper.
- Kernel dimension 3 was trial and error, and was deemed to provide great results. A standard stride=1, and padding=1 was chosen and worked well.
- ReLU activation was used to inject nonlinearities. ReLU was chosen over other activation functions to avoid a vanishing gradient.
- Batch normalization was applied after every layer to make it easier for the algorithm to learn by normalizing the distributions of numbers across all dimensions.
- Max pooling was used to downsample, and increase the receptive field of the kernels as the layers get deeper, allowing weights to learn abstract features.
- Residual was added make it easier for the algorithm, by having the architecture set up to learn appropriate perturbations from an “input”. In practice, marginal improvement in performance was observed.

Describe your choice of hyper-parameters:

My hyper-parameter choices:

- Batch size: 128
- Learning rate: 0.01
- Regularization: 0.0003
- Epochs: 20
- Steps: [6, 8]
- Warmup: 0
- Momentum 0.95
- Imbalance: Regular
- Loss Type: Cross-Entropy

My hyper-parameter choices reasoning:

- Learning rate exploded losses above 0.1. Learning rate of 0.00001 caused losses to converge too slowly. A learning rate in between this range was chosen.
- Regularization of 0.0003 was chosen because this gave good enough accuracy results on the validation set after trial and error.
- Epochs=20 was chosen for safe measure, although losses converged after 10 epochs.
- Momentum of 0.95 was chosen arbitrarily and gave good enough results.

What’s your final accuracy on validation set?

- **0.8711 (87.11%)**

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	0.92	0.903	0.596	0.534	0.245	0.053	0.179	0.067	0.000	0.000

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

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
beta= 0.99	0.867	0.661	0.592	0.250	0.002	0.000	0.002	0.000	0.000	0.000
beta= 0.999	0.841	0.534	0.525	0.079	0.054	0.004	0.046	0.023	0.000	0.004
beta = 0.9999	0.425	0.185	0.365	0.142	0.255	0.202	0.142	0.138	0.142	0.233
beta = 0.99999	0.403	0.174	0.114	0.089	0.515	0.130	0.250	0.230	0.417	0.199

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

	Class 0	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7	Class 8	Class 9
CE Loss	0.92	0.903	0.596	0.534	0.245	0.053	0.179	0.067	0.000	0.000
CB-Focal (beta = 0.99999)	0.403	0.174	0.114	0.089	0.515	0.130	0.250	0.230	0.417	0.199

Describe and explain your observation on the result:

Description:

- NOTE: Sample size for classes 0 – 9 are, respectively:
  N = [5000, 2997, 1796, 1077, 645, 387, 232, 139, 83, 50]

- With the regular Cross-Entropy (CE) Loss in the imbalanced CIFAR-10 dataset, the model has less than 25% accuracy for class 4, 5, 6, 7, 8, 9, and it only has above 90% accuracy for class 1 and 2.

- When the Class-balanced Focal Loss (CB FL) is applied to the imbalanced CIFAR-10 dataset, accuracy changes depending on the beta value. For beta value= 0.99 , accuracy of all classes is reduced when compared to CE, without accuracy improvement in any class. However, as beta value increases to 0.99999, accuracy for minority classes begin to increase significantly, while accuracy for majority classes continue to decrease.

- Although accuracies for each class fluctuate a lot as beta changes from 0.99 to 0.99999, the average accuracy per class fluctuates significantly less, and stays between 0.2-0.25~ per class. Average accuracy per class was computed by adding up all accuracies and dividing by the total number of classes.

Describe and explain your observation on the result:

Explanation:

- Recall the definition of Class-balanced Focal Loss (CB FL):

The class-balanced (CB) focal loss is:

$$\mathrm{CB}_{\mathrm{focal}}(\mathbf{z}, y) = -\frac{1-\beta}{1-\beta^{n_y}} \sum_{i=1}^{C} (1-p_i^t)^{\gamma} \log(p_i^t). \quad (13)$$

- This means that focal losses get reweighed based on how much beta is, and also based on sample size. For a numerical explanation, the weights for different beta values look like this:

- Weights (beta = 0.99) = [0.08, 0.08, 0.08, 0.08, 0.08, 0.08, 0.09, 0.10, 0.14, 0.20]
- Weights (beta = 0.999) = [0.02, 0.02, 0.02, 0.03, 0.04, 0.06, 0.09, 0.14, 0.23, 0.37]
- Weights (beta = 0.9999) = [0.01, 0.01, 0.01, 0.02, 0.03, 0.05, 0.09, 0.14, 0.24, 0.40]
- Weights (beta = 0.99999) = [0.00, 0.01, 0.01, 0.02, 0.03, 0.05, 0.09, 0.14, 0.24, 0.40]

- NOTE: The weights above were calculated by applying the formula "weights = $\frac{1-\beta}{1-\beta^{n_y}}$" . Weights values were normalized to sum to 1.

Explanation (continued):

- To do direct comparison of CB FL versus CE loss wouldn't be completely apples to apples when various elements were changed between both methods at once. First, sigmoid was used for CB FL, and Softmax is being used for CE. In addition, CB FL implements focal loss, which downweighs relative loss for well classified samples, while vanilla CE does not. Third, beta is varied across different runs. However, patterns could still be seen via the trends – in particular, we see that CB FL does a significantly better job at increasing accuracy for minority classes when using certain beta values.

- We can see in previous slide that for beta = 0.99, accuracy for majority classes is reduced relative to CE. This may be for all the reasons mentioned in the above point but could be in part because classes 6-9 get relatively higher weighting on their loss due to their higher relative weight values (as shown in previous slide). This can cause the model to penalize mistakes on the majority class less, leading to less training of the model. However, the accuracy on class 6-9 is still very low, which indicates the model is not being penalized enough for being incorrect on the minority classes either. This suggests increasing beta may help. Increasing beta would increase relative loss on the minority class, which would help the model not ignore mistakes it makes on minority class samples.

- As beta approaches beta = 0.99999, there is a clear increase in accuracy in the minority class. This makes sense from the weights, as we see that the relative weighting for being incorrect on a minority class sample is higher and is more heavily penalized relative to a majority class sample. However, this also means that majority class losses may not be penalized enough. There is a stark decrease in accuracy for majority classes. It seems that striking a balance between relative weighting of majority and minority classes is very important when applying the CB FL method.