ID: 1968990
Cheng-Yen Yang

Homework 2 (Colab)

EE599: Deep Learning
November 7, 2019

1 CIFAR

What design that you tried worked the best? This includes things like network design, learning rate, batch size, number of epochs, and other optimization parameters, data augmentation etc. What was the final train loss? Test loss? Test Accuracy?

I use VGG16(Simonyan et al.) as the CNN framework which reach **91.73% Test Accuracy** after 60 epochs using SGD optimizer.

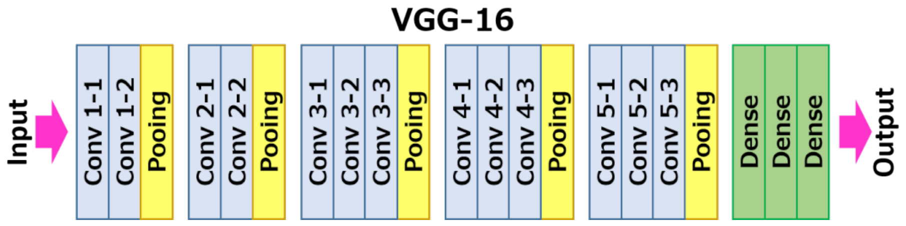

Figure 1: Architecture of VGG16

Other hyperparamters are:

BATCH_SIZE	EPOCHS	LR	MOMETUM	WEIGHT_DECAY
512	60	0.01 for epoch [0,30) 0.001 for epoch [30,60)	0.9	0.0005

And beside the default data augmentation setting given by TAs, I play around with various data augmentations and add **transforms.RandomRotation()** to increase the performance of Cifar-Net().

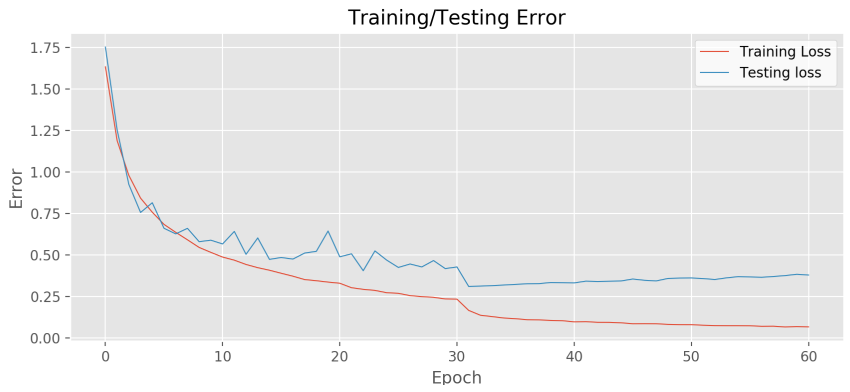

Figure 2: Training and Testing Error of the best model

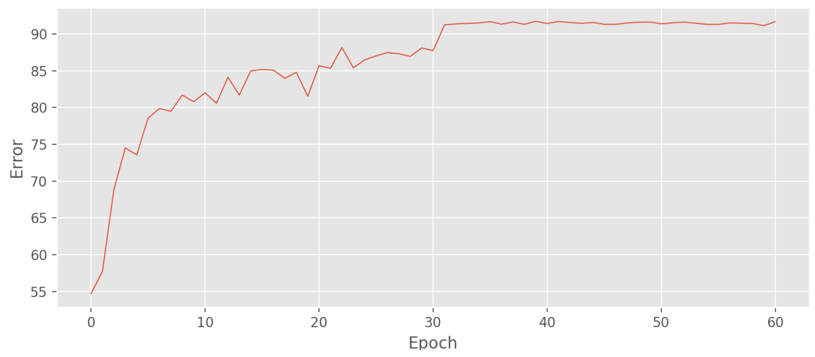

Figure 3: Testing Accuracy of the best model of **91.73%** accuracy

What design worked the worst (but still performed better than random chance)?

Simply **change the CNN framework from VGG16 to VGG11** and **remove the batch normalization layers**, a much shallower convolutional neural network resulted in much lower **Test Accuracy being 88.23%** after 60 epochs. All of the hyperparameters setting are identical as the model in the question 1.

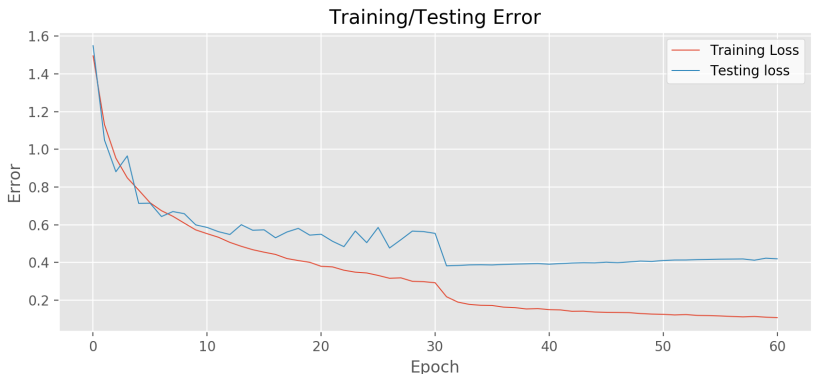

Figure 4: Training and Testing Error of the bad model

Why do you think the best one worked well and the worst one worked poorly.

Batch Normalization Layers is the main reason that the models from question 1 and 2 worked differently. According to lectures, batch normalization reduces the amount of values shift around in weights and allows each layer of a network to learn by itself a little bit more independently of other layers, which resulted in better result for CIFAR-10 classification.

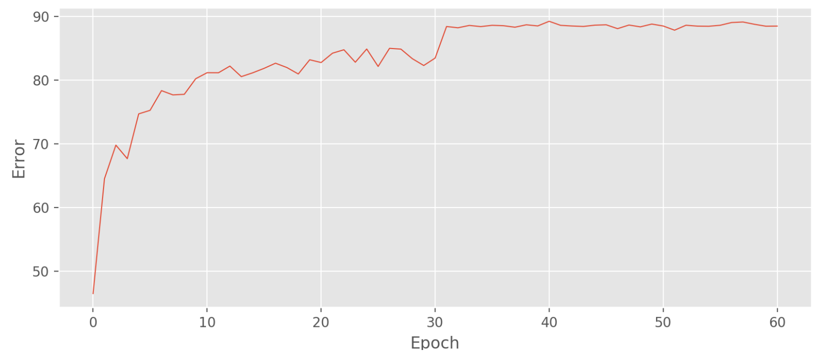

Figure 5: Testing Accuracy of the bad model of **88.23%** accuracy

2 TinyImageNet

What design that you tried worked the best? How many epochs were you able to run it for?

I use ResNet-50(He et al.) as the CNN framework which reach **57.125% Testing Accuracy** after 60 epochs using SGD optimizer.

Other hyperparamters are:

BATCH_SIZE	EPOCHS	LR	MOMETUM	WEIGHT__DECAY
128	60	0.01	0.9	0.0005

And beside the default data augmentation setting given by TAs, I play around with various data augmentations and add **transforms.RandomRotation()** to increase the performance of Cifar-Net().

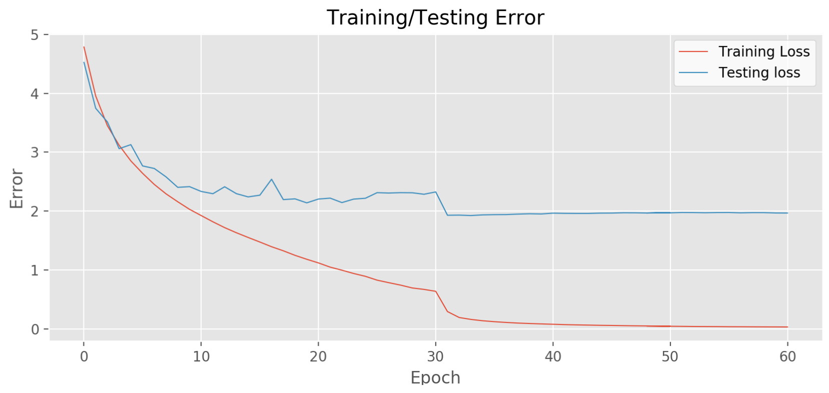

Figure 6: Training and Testing Error of the best model

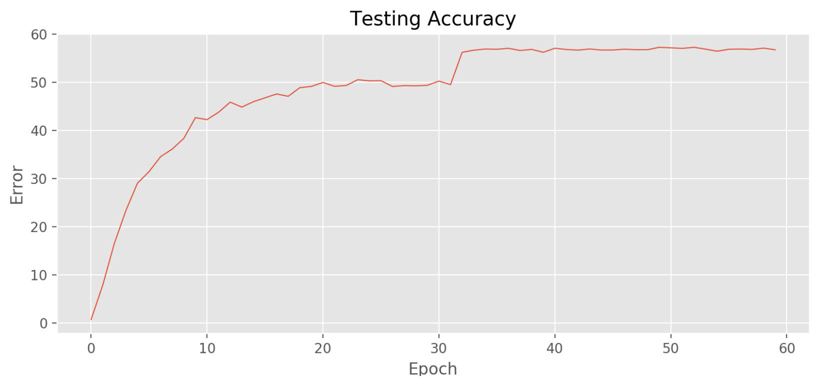

Figure 7: Testing Accuracy of the best model of **57.125%** accuracy

Were you able to use larger/deeper networks on TinyImageNet than you used on CIFAR and increase accuracy? If so, why? If not, why not?

I first try to train TinyImagenet with the same VGG16 network I did on CIFAR, however the training loss and testing accuracy goes around 35% only.

Later I tried out the ResNet34 and ResNet50, which probably solved the gradient problem as the testing accuracy goes up to around 55% which is pretty acceptable without fine tuning.

The real ImageNet dataset has significantly larger images. How would you change your network design if the images were twice as large? How about smaller than Tiny ImageNet (32x32)? How do you think your accuracy would change?

For larger images, it seems straight forward to just down-sampled or scaled the images into more appropriate size for CNN networks for training. However, this method may resulted in trade-off between training duration and test accuracy as we are literally throwing away the details and features of the high-resolution images.

If we want to deal with the whole image without down-sampling, we can remove some of the fully connected layers at the end of the networks since it is which normally contained large amount of weights or reduce the batch size to fit the memory capacity in CPUs or GPUS.

As for smaller images, instead of up-sampling or scaling the images, it seems more suitable to perform data augmentations on the training data. However, I think the accuracy outcome will depends on the attribute and distribution of the dataset gradually.