Final Project: Reproduce One Result of ”Understanding Deep Learning Requires Rethinking Generalization”

In this project, I attempted to reproduce and confirm the zero training loss with random labels given other neural network architectures and other datasets that inspired by Understanding Deep Learning Requires Rethinking Generalization. Zhang et.

Summary of the Results From Original Paper

In the paper, Understanding Deep Learning Requires Rethinking Generalization, Chiyuan Zhang and his colleagues deployed a number of systematic experiments to exhibit the traditional view of generalization, i.e., difference between training error and test error, is incapable to distinguish between different neural networks that have remarkably different generalization performance. The author established the convolutional networks and trained that on a copy of the data (image set) in which the true labels were replaced by random labels. Although there was no relationship between the data and the class labels, training on random labeling of the true data exhibited zero training error. Indeed, a varying level of label corruption from no corruption to complete random labels on the datasets was deployed on the neural network training, the networks still fitted the corrupted training set perfectly at all cases. Such a result implies that the increase of the generalization error of the model can be done by randomizing the labels without making any other change.

Moreover, the experiment of replacing the images by random pixels like Gaussian noise with different levels of the randomization would achieve the same results. However, increasing the noise level resulted in a steady deterioration of the generalization error without explicit regularizers. Even though it didnt reach the perfect 100% top-1 accuracy, the network still managed to reach around 90% top-1 accuracy. In contrast to the neural networks with random labels, the training with random pixels and Gaussian converges to the solution faster. The reason behind this phenomenon might be the input with random pixels are more separated and easier to build a network for arbitrary label assignments.

Through the extensive experiment, the authors also showed that the explicit forms of regularization, such as weight decay, and dropout, may improve generalization error of neural networks. In details, the use of data augmentation and weight decay help to improve the generalization performance of Inception, Alexnet, and MLPs, but these models still generalize very well with all of the regularizers turned off. Although there is a 10 accuracy drop when turning off all the regularizers, the authors supposed that bigger gains can be achieved by changing the model architecture. So, the regularizer is either unnecessary or insufficient for controlling generalization error. Furthermore, the explicit regularization appears to be more of a tuning parameter that often helps improve the final testing error.

The authors empirical observation with a theoretical construction also showed that generically large neural networks are capable to express any labeling of training data. Indeed, a very two-layer ReLu network with p=2n+d parameters is capable to express any labeling of any sample of size n in d dimensions. The result of this observation even exhibited that the depth-2 networks of linear size are capable to express any labeling of the training data.

In contrast to the role of explicit regularization, the stochastic gradient descent played an important role in the generalization of the models that didnt fit the data well. Out of all models that exactly fitted the data, the experiments showed that the SGD would always converge to a solution with a small norm for a linear model. Without preprocessing, although the model could still fit the data, the testing error was remarkably larger than the one with preprocessing. Hence, the linear model that was trained using SGD performed a better generalization error compared to other algorithms. However, the authors revealed that Gaussian kernel methods can generalize well with no regularization on small data sets. As a result, the author concluded that the notion of the minimum norm is not predictive of generalization performance. Even though the minimum-norm intuition may provide some guidance to new algorithm design, it still cant cover the full story of the generalization.

In conclusion, the authors emphasize the effective capacity of several successful neural network architectures and these models are capable to memorize the training data. The authors also excluded the reasons for why optimization is empirically easy from the true cause of generalization.

Procedure for Reproducing the Result

To achieve the purpose mentioned at the beginning, reproducing the result of fitting the random labels and pixels, I used a commonly used successfully model, Wide Resnet, and train that on random labels on the CIFAR10 dataset. The features of Wide Resnet took me some time to overview. The code I used is adapted from https://github.com/pluskid/fitting-random-labels. However, because the last commit the author made was 2 years ago, so I solved several problems in the process of running the code. Additionally, because the author only provided CIFAR10 dataset in his code, I also made some modifications in his codes and implement other python scripts to allow the Wide Resnet model to train on other different datasets. Since every dataset under torchvision.dataset has different parameters and attributes, I also investigated their characteristics and made the corresponding changes to attempt to fit them in the codes. The final repository of the code I used to reproduce the result of the original below is `https://github.com/ShuuTsubaki/fitting-random-labels`.

Coding Experiences

C1. When I tried to run the command to train the Wide Resnet model with true labels, python train,py, the following problem occurred,

```
    losses.update(loss.data[0], input.size(0))
    top1.update(prec1[0], input.size(0))
```

```
IndexError: invalid index of a 0-dim tensor. Use tensor.item() to convert a
   0-dim tensor to a Python number
```

That is because in PyTorch $>= 0.5$, the index of 0-dim tensor is invalid. So, I changed to

```
    top1.update(prec1, input.size(0))
    losses.update(loss.data, input.size(0))
```

C2. When I tried to train the Wide Resnet model with random labels, the following problem occurred,

```
    labels = np.array(self.train_labels if self.train else self.test_labels)
AttributeError: 'CIFAR10RandomLabels' object has no attribute 'train_labels'
```

I investigated the source code for torchvision.dataset.cifar, it seemed like the torchvision removed the attributes named train_labels and test_labels in its past updates, so I made the following changes in cifar10_data.py,

```
    if self.train:
        self.train_labels = self.targets
    else:
        self.test_labels = self.targets
```

C3. In order to train the Wide Resnet on a CIFAR100 dataset, I added a script named cifar100_data.py and made some modification on the cmd_args.py and train.py. In contrast to CIFAR10 dataset, it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class. So the number of the classes in the argument has to set to 100. Otherwise, the following error would occur,

```
RuntimeError: cuda runtime error (59) : device-side assert triggered at
   C:/w/1/s/tmp_conda_3.7_055306/conda/conda-bld/pytorch_1556690124416/
work/aten/src/ATen/native/cuda/SoftMax.cu:620
```

C5. Because I didn't have enough GPU computing resource to run my proposed experiments, I used Google Colaboratory to run these experiments, but I frequently experienced the problem of network error and reconnection request. The running time of every experiment also took longer than I expected. However, I wrote a ipython notebook for running these scripts, the copy of this notebook and its outputs are attached in the repository, `https://github.com/ShuuTsubaki/fitting-random-labels/blob/master/final_project.ipynb`.
C4. To take the advantage of data visualization, I also implemented the script that can plot the training loss and testing loss over the number of iterations with a varying level of label corruption from no corruption to complete random labels on the datasets. `https://github.com/ShuuTsubaki/fitting-random-labels/blob/master/plot.py`

Measurements and Analysis of the Reproduced Results

The following outputs are from my colab run for Wide Resnet on the original CIFAR-10 : (the outputs are partially hidden due to space constraint, if there is a need, the reader can be refer to the ipython notebook in Github mentioned above)

```
000: Acc-tr: 61.60, Acc-val: 61.05, L-tr: 1.0634, L-val: 1.0958
001: Acc-tr: 69.36, Acc-val: 66.62, L-tr: 0.8749, L-val: 0.9485
002: Acc-tr: 71.29, Acc-val: 68.66, L-tr: 0.8570, L-val: 0.9382
003: Acc-tr: 76.68, Acc-val: 72.94, L-tr: 0.6839, L-val: 0.8149
004: Acc-tr: 82.17, Acc-val: 77.22, L-tr: 0.5168, L-val: 0.6744
005: Acc-tr: 83.92, Acc-val: 77.87, L-tr: 0.4650, L-val: 0.6476
...
145: Acc-tr: 95.87, Acc-val: 80.69, L-tr: 0.1193, L-val: 0.7876
146: Acc-tr: 96.07, Acc-val: 80.67, L-tr: 0.1145, L-val: 0.8287
147: Acc-tr: 94.66, Acc-val: 80.08, L-tr: 0.1553, L-val: 0.9099
...
200: Acc-tr: 100.00, Acc-val: 85.93, L-tr: 0.0005, L-val: 0.5978
201: Acc-tr: 100.00, Acc-val: 85.94, L-tr: 0.0004, L-val: 0.5970
...
299: Acc-tr: 100.00, Acc-val: 86.11, L-tr: 0.0004, L-val: 0.5787
```

The following outputs are from my colab run for Wide Resnet on the CIFAR-10 with partially corrupted labels (0.5):

```
000: Acc-tr: 58.58, Acc-val: 57.12, L-tr: 1.1676, L-val: 1.2036
001: Acc-tr: 66.53, Acc-val: 64.70, L-tr: 0.9566, L-val: 1.0202
002: Acc-tr: 70.45, Acc-val: 68.72, L-tr: 0.8370, L-val: 0.9127
003: Acc-tr: 77.64, Acc-val: 73.70, L-tr: 0.6330, L-val: 0.7729
004: Acc-tr: 78.71, Acc-val: 74.04, L-tr: 0.5942, L-val: 0.7472
005: Acc-tr: 84.93, Acc-val: 79.04, L-tr: 0.4287, L-val: 0.6256
...
145: Acc-tr: 95.70, Acc-val: 80.39, L-tr: 0.1224, L-val: 0.8589
146: Acc-tr: 97.42, Acc-val: 82.07, L-tr: 0.0744, L-val: 0.7459
147: Acc-tr: 94.01, Acc-val: 78.87, L-tr: 0.1819, L-val: 0.9050
...
200: Acc-tr: 100.00, Acc-val: 86.73, L-tr: 0.0005, L-val: 0.5697
201: Acc-tr: 100.00, Acc-val: 86.81, L-tr: 0.0005, L-val: 0.5672
...
299: Acc-tr: 100.00, Acc-val: 86.95, L-tr: 0.0004, L-val: 0.5524
```

The following outputs are from my colab run for Wide Resnet on the CIFAR-10 with random labels:

```
000: Acc-tr: 59.89, Acc-val: 59.01, L-tr: 1.1491, L-val: 1.1794
001: Acc-tr: 70.86, Acc-val: 68.70, L-tr: 0.8266, L-val: 0.8940
```

```
002: Acc-tr: 73.00, Acc-val: 71.16, L-tr: 0.7782, L-val: 0.8674
003: Acc-tr: 78.26, Acc-val: 74.35, L-tr: 0.6163, L-val: 0.7423
004: Acc-tr: 80.07, Acc-val: 75.50, L-tr: 0.5830, L-val: 0.7591
005: Acc-tr: 83.93, Acc-val: 78.48, L-tr: 0.4606, L-val: 0.6633
...
145: Acc-tr: 95.96, Acc-val: 80.80, L-tr: 0.1164, L-val: 0.8132
146: Acc-tr: 97.01, Acc-val: 81.85, L-tr: 0.0854, L-val: 0.7278
147: Acc-tr: 95.85, Acc-val: 81.02, L-tr: 0.1245, L-val: 0.7798
...
200: Acc-tr: 100.00, Acc-val: 86.37, L-tr: 0.0005, L-val: 0.5744
201: Acc-tr: 100.00, Acc-val: 86.31, L-tr: 0.0004, L-val: 0.5744
...
299: Acc-tr: 100.00, Acc-val: 86.34, L-tr: 0.0004, L-val: 0.5579
```

Based on the output I got from above, I used plot.py to visualize the measurements.

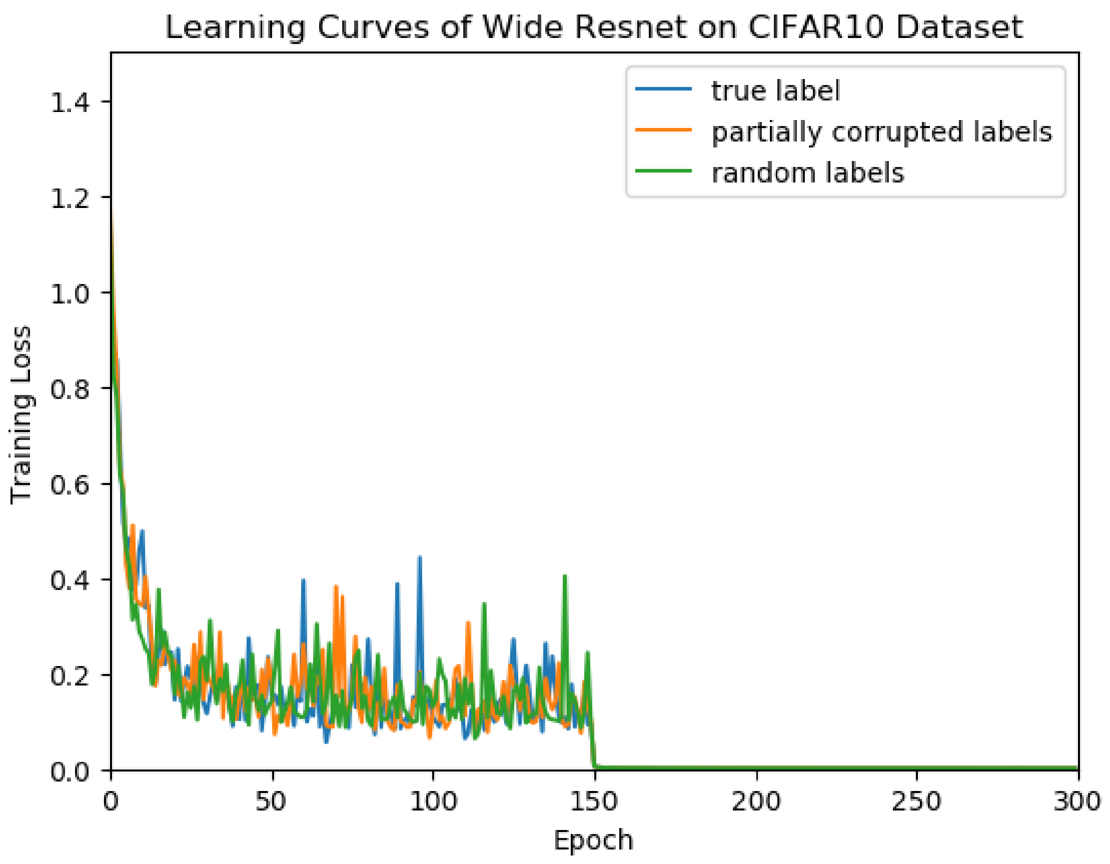
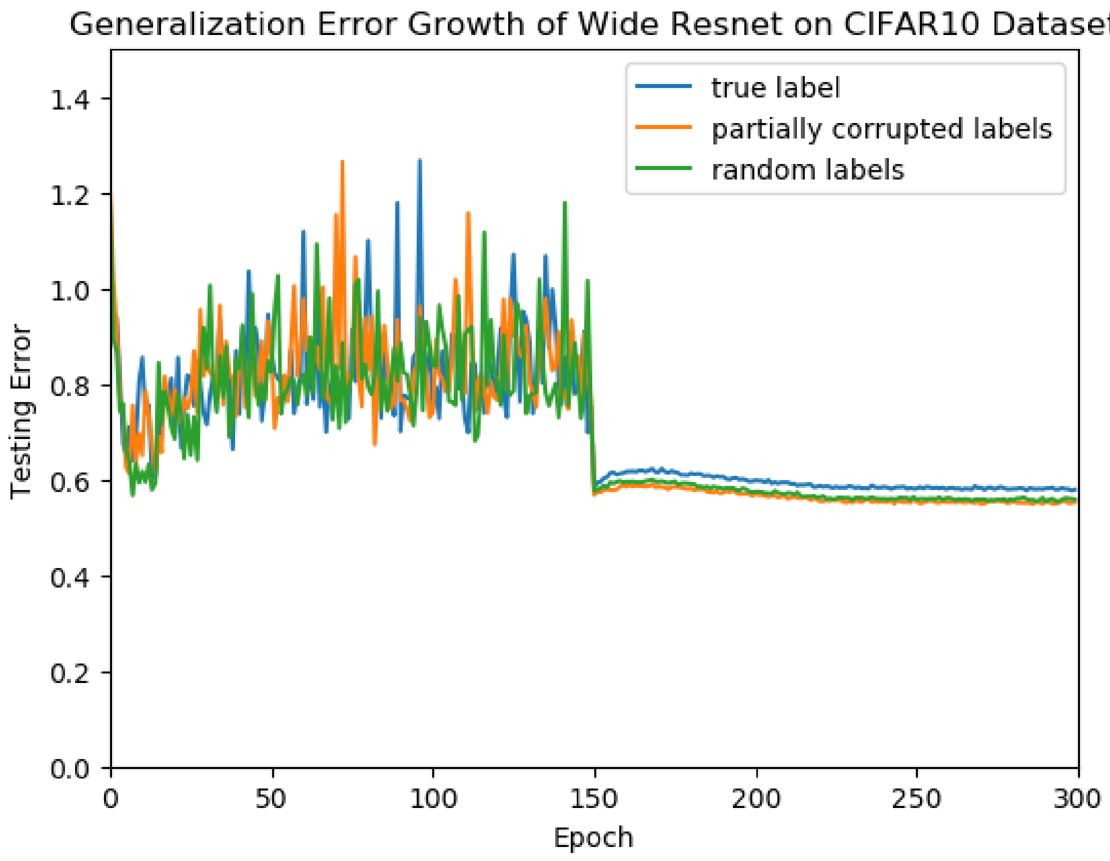


Figure above shows the training loss of various experiment settings decaying with the epoch iteration. Similar to the result Zhang and his colleagues found in the original paper, we see that Wide Resnet still converges to zero loss on the CIFAR10 training set. Although the label assignments for every training sample is uncorrelated, the predictions errors are still back-propagated to make the gradients for parameter update. We can find that once the model starts to fit the data, it converges very quickly, and it converges to overfit the training set extremely well.



We can observe the test errors after convergence from figure above . With a varying level of label corruptions from 0 (no corruption) to 0.5 (partially corrupted) and then to 1 (complete random labels) on the CIFAR10 dataset, the test errors converge to around 0.6%. Since the training errors are zero as previous figure showed, the test errors can treat as the generalization errors. These errors refer the accuracy of the model in different setting are approached to around 100%.
I also trained the Wide Resnet on CIFAR100. The following outputs are from my colab run for Wide Resnet on the original CIFAR-100 :

```
000: Acc-tr: 13.67, Acc-val: 13.76, L-tr: 3.5994, L-val: 3.6218
001: Acc-tr: 24.23, Acc-val: 23.52, L-tr: 2.9807, L-val: 3.0470
002: Acc-tr: 34.88, Acc-val: 33.03, L-tr: 2.4689, L-val: 2.5765
003: Acc-tr: 39.30, Acc-val: 35.72, L-tr: 2.2313, L-val: 2.3935
004: Acc-tr: 46.16, Acc-val: 42.21, L-tr: 1.9556, L-val: 2.1615
```

```
005: Acc-tr: 43.94, Acc-val: 39.20, L-tr: 2.0877, L-val: 2.3665
...
145: Acc-tr: 85.59, Acc-val: 49.44, L-tr: 0.4463, L-val: 2.5971
146: Acc-tr: 86.78, Acc-val: 48.92, L-tr: 0.4044, L-val: 2.5993
147: Acc-tr: 80.33, Acc-val: 47.41, L-tr: 0.6356, L-val: 2.9195
...
200: Acc-tr: 99.98, Acc-val: 56.77, L-tr: 0.0043, L-val: 2.2184
201: Acc-tr: 99.98, Acc-val: 56.85, L-tr: 0.0044, L-val: 2.2044
...
299: Acc-tr: 99.98, Acc-val: 56.58, L-tr: 0.0042, L-val: 2.1633
```

The following outputs are from my colab run for Wide Resnet on the CIFAR-100 with partially corrupted labels (0.5):

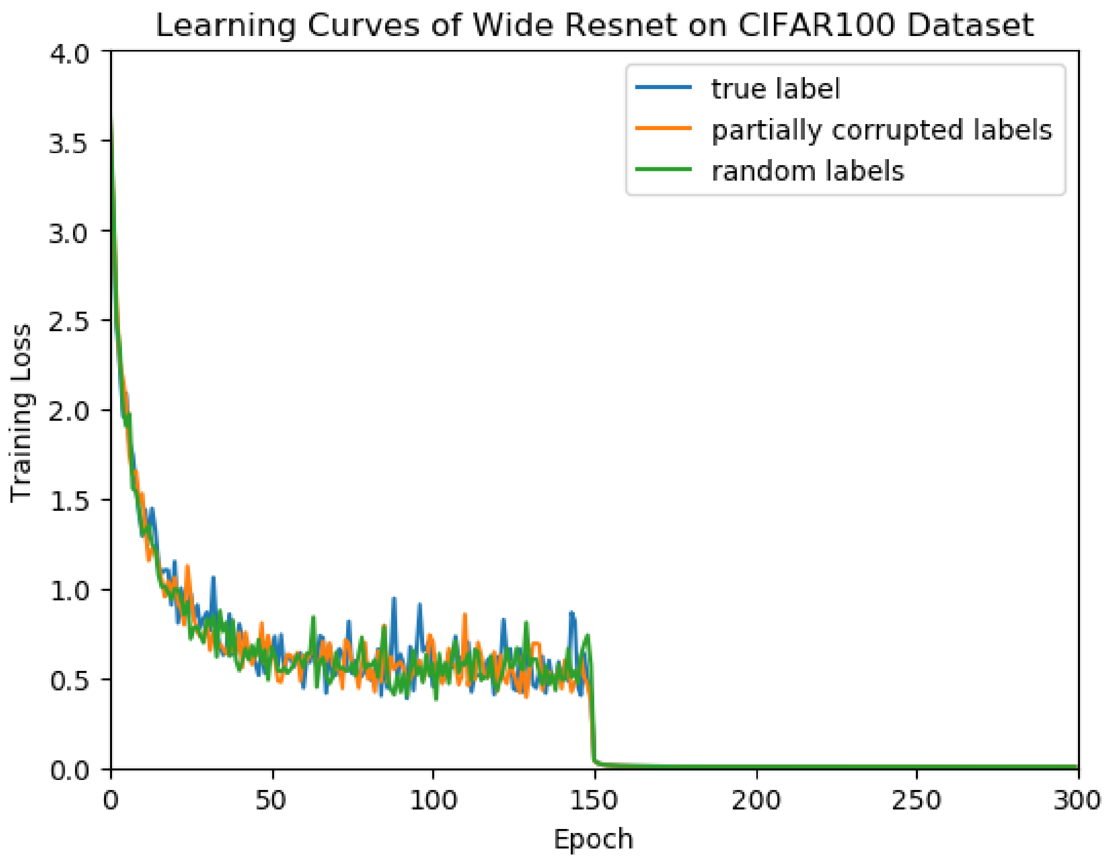
```
000: Acc-tr: 13.32, Acc-val: 12.92, L-tr: 3.6161, L-val: 3.6413
001: Acc-tr: 22.13, Acc-val: 21.40, L-tr: 3.0710, L-val: 3.1265
002: Acc-tr: 32.19, Acc-val: 30.62, L-tr: 2.5748, L-val: 2.6697
003: Acc-tr: 38.82, Acc-val: 35.94, L-tr: 2.2633, L-val: 2.4102
004: Acc-tr: 41.11, Acc-val: 37.56, L-tr: 2.1685, L-val: 2.3834
005: Acc-tr: 44.82, Acc-val: 40.05, L-tr: 2.0458, L-val: 2.3138
...
145: Acc-tr: 84.40, Acc-val: 49.30, L-tr: 0.4894, L-val: 2.8438
146: Acc-tr: 80.95, Acc-val: 47.19, L-tr: 0.6159, L-val: 2.8634
147: Acc-tr: 83.93, Acc-val: 47.92, L-tr: 0.5030, L-val: 2.7608
...
200: Acc-tr: 99.98, Acc-val: 56.26, L-tr: 0.0043, L-val: 2.2122
201: Acc-tr: 99.98, Acc-val: 56.33, L-tr: 0.0043, L-val: 2.2106
...
299: Acc-tr: 99.98, Acc-val: 55.66, L-tr: 0.0040, L-val: 2.1789
```

The following outputs are from my colab run for Wide Resnet on the CIFAR-100 with random labels:

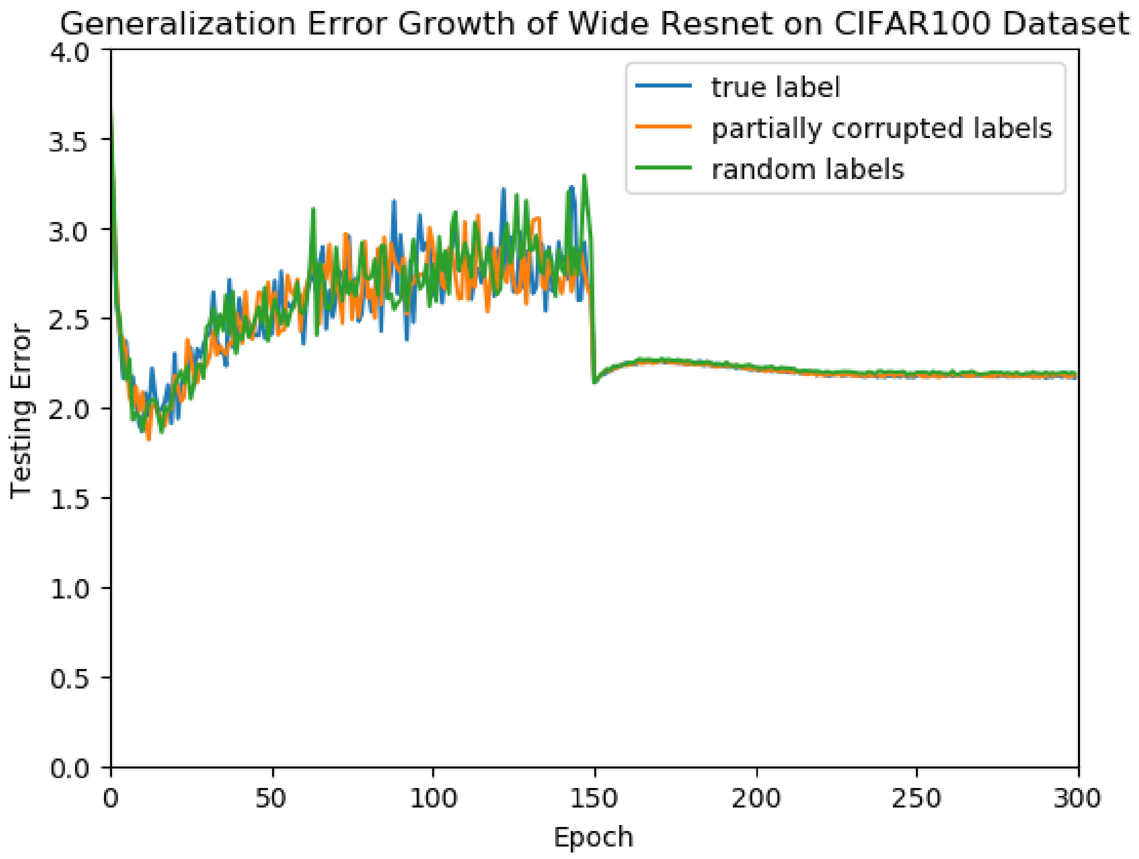
```
000: Acc-tr: 12.49, Acc-val: 11.81, L-tr: 3.6397, L-val: 3.6806
001: Acc-tr: 21.73, Acc-val: 21.02, L-tr: 3.1562, L-val: 3.2268
002: Acc-tr: 33.57, Acc-val: 31.69, L-tr: 2.4840, L-val: 2.5858
003: Acc-tr: 37.39, Acc-val: 35.16, L-tr: 2.3291, L-val: 2.4836
004: Acc-tr: 44.69, Acc-val: 40.47, L-tr: 2.0008, L-val: 2.2160
005: Acc-tr: 47.11, Acc-val: 42.46, L-tr: 1.9040, L-val: 2.1540
...
145: Acc-tr: 84.81, Acc-val: 48.82, L-tr: 0.4663, L-val: 2.7377
146: Acc-tr: 80.34, Acc-val: 46.15, L-tr: 0.6262, L-val: 2.9115
147: Acc-tr: 79.73, Acc-val: 45.39, L-tr: 0.6896, L-val: 3.2918
...
200: Acc-tr: 99.98, Acc-val: 56.22, L-tr: 0.0043, L-val: 2.2214
201: Acc-tr: 99.98, Acc-val: 56.19, L-tr: 0.0042, L-val: 2.2231
```

```
...
299: Acc-tr: 99.98, Acc-val: 55.58, L-tr: 0.0039, L-val: 2.1891
```

Based on the output I got from above, I used plot.py to visualize the measurements.



From the figure , we can see the behavior of the model trained on the CIFAR10 dataset also occur on the CIFAR100 dataset. The training errors with different levels of label corruptions dramatically decrease in first 50 epochs and begin to converge very quickly. Finally, they converge to zero and fit the training set perfectly.



From the figure , we can see that even though the testing errors are bumped up in a number of epochs, they still dramatically converge to around 2.0% with different settings. The performance of random guessing on CIFAR 100 is as good as the performance of the model with true labels.

In conclusion, given the Wide Resnet model with CIFAR100, we can still observe that the model achieves the zero training loss with different levels of label corruptions (0, 0.5, and 1). Hence, we successfully reproduced the result and can confirm that there is also zero training loss with random labels given other neural network architectures and other datasets.

Reference

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals. Understanding deep learning requires rethinking generalization. International Conference on Learning Representations (ICLR), 2017. `https://arxiv.org/abs/1611.03530`.

Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In ICML, 2016.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: risk bounds and structural results. Journal of Machine Learning Research, 3:463482, March 2003.