Indian Institute of Technology, Jodhpur
AY 2021-22 Semester 1

CSL 7530 - Advanced Machine Learning

Presentation on Working Demo of the Code

An Empirical Study of Example Forgetting During Deep Neural Network Learning

reproduced by Sanyam Jain (P20QC001)

Original paper: Toneva, M., Alessandro, S., Tachet des Combes, R., Trischler, A., Bengio, Y., & Gordon, G. (2019). An Empirical Study of Example Forgetting During Deep Neural Network Learning. ICLR.

For MNIST Dataset

- Step 1: Setup arguments and fix the parameters.

```
args = {'dataset': 'mnist',
        'batch_size': 64,
        'epochs':200,
        'lr':0.01,
        'momentum':0.5,
        'no_cuda':False,
        'seed':2,
        'sorting_file':"none",
        'remove_n':0,
        'keep_lowest_n':0,
        'no_dropout':False,
        'input_dir':'mnist_results/',
        'output_dir':'mnist_results/'

        }
```

For MNIST Dataset

- Step 2: Load the train_dataset and test_dataset

```
# Load the appropriate train and test datasets
trainset = datasets.MNIST(
    root='/tmp/data', train=True, download=True,
    transform=transform)
testset = datasets.MNIST(
    root='/tmp/data', train=False, download=True,
    transform=transform)
```

For MNIST Dataset

- Step 3: Train and Test for specified Epochs.

```
for epoch in range(args['epochs']):
    start_time = time.time()

    train(args, model, device, trainset, optimizer, epoch,
            example_stats)
    test(args, model, device, testset, example_stats)

    epoch_time = time.time() - start_time
    elapsed_time += epoch_time
    print('| Elapsed time : %d:%02d:%02d' %
                (get_hms(elapsed_time)))

  # Save the stats dictionary

  # Log the best train and test accuracy so far
```

For MNIST Dataset

- Step 4: in train() preserve the indices for the batch

```
for batch_idx, batch_start_ind in enumerate(
            range(0, len(trainset.train_labels), batch_size)):

        # Get trainset indices for batch
        batch_inds = trainset_permutation_inds[batch_start_ind:
                                               batch_start_ind + batch_size]

        # Get batch inputs and targets, transform them appropriately
        transformed_trainset = []
        for ind in batch_inds:
            transformed_trainset.append(trainset.__getitem__(ind)[0])
        inputs = torch.stack(transformed_trainset)
        targets = torch.LongTensor(
            np.array(trainset.train_labels)[batch_inds].tolist())
```

For MNIST Dataset

- Step 5: Output, loss and predicted class. Forward propagation, compute loss, and get predictions.

```
# Forward propagation, compute loss, get predictions
        optimizer.zero_grad()
# Output is prediction
        outputs = model(inputs)
# Cross entropy loss
        loss = criterion(outputs, targets)
# get the index of maximum value in the data
        _, predicted = torch.max(outputs.data, 1)
```

For MNIST Dataset

- Step 7: Calculate accuracy and statistics for examples in the mini-batch.

```
# Update statistics and loss
       acc = predicted == targets
       for j, index in enumerate(batch_inds):
           index_in_original_dataset = train_indx[index]
            output_correct_class = outputs.data[
                j, targets[j].item()]
```

For MNIST Dataset

- Step 8: Combine all statistics and save to a file.

```
# Add the statistics of the current training example to dictionary
            index_stats = example_stats.get(index_in_original_dataset,
                                            [[], [], []])
            index_stats[0].append(loss[j].item())
            index_stats[1].append(acc[j].sum().item())
            index_stats[2].append(margin)
            example_stats[index_in_original_dataset] = index_stats
```

For CIFAR Dataset

- Process 1:
  - Step 1: Train the CIFAR dataset without any additional effort. Keep all parameters 0 and flags as False.
  - Step 2: Loss and predict.
  - Step 3: Record the accuracy and loss statistics.

```
args = {'dataset': 'cifar10',
        'model':'resnet18',
        'batch_size': 128,
        'epochs':20,
        'learning_rate':0.01,
        'data_augmentation':False,
        'cutout':False,
        'n_holes':1,
        'length':16,
        'no_cuda':False,
        'seed':2,
        'sorting_file':"none",
        'remove_n':0,
        'keep_lowest_n':0,
        'no_dropout':False,
        'remove_subsample':0,
        'noise_percent_labels':0,
        'noise_percent_pixels':0,
        'noise_std_pixels':0,
        'optimizer':'sgd',
        'input_dir':'cifar10_results/',
        'output_dir':'cifar10_results/'

        }
```

For CIFAR Dataset

- Process 2:
  - Step 4: Using forgetting statistics, apply sort on the dataset.
  - Step 5: Squash the accuracy values such that 0 means the example is not learned. 1 means example is learned and value from 1 to 0 means that the example is forgettable;

```python
def compute_forgetting_statistics(diag_stats, npresentations):

    presentations_needed_to_learn = {}
    unlearned_per_presentation = {}
    margins_per_presentation = {}
    first_learned = {}
```

```python
            presentation_acc = np.array(example_stats[1]
[:npresentations])
            transitions = presentation_acc[1:] -
presentation_acc[:-1]

            if len(np.where(transitions == -1)[0]) > 0:
                unlearned_per_presentation[example_id] =
np.where(
                    transitions == -1)[0] + 2
            else:
                unlearned_per_presentation[example_id] = []

            if len(np.where(presentation_acc == 0)[0]) > 0:
                presentations_needed_to_learn[example_id] =
np.where(
                    presentation_acc == 0)[0][-1] + 1
            else:
                presentations_needed_to_learn[example_id] = 0

            margins_per_presentation = np.array(
                example_stats[2][:npresentations])


            if len(np.where(presentation_acc == 1)[0]) > 0:
                first_learned[example_id] = np.where(
                    presentation_acc == 1)[0][0]
            else:
                first_learned[example_id] = np.nan
```

For CIFAR Dataset

- Process 2:
  - step 6: Sort the examples basis of forgetting.
  - Highest forgetting count to lowest forgetting count.

```python
def sort_examples_by_forgetting(unlearned_per_presentation_all,
                                first_learned_all, npresentations):

    # Initialize lists
    example_original_order = []
    example_stats = []

    for example_id in unlearned_per_presentation_all[0].keys():

        # Add current example to lists
        example_original_order.append(example_id)
        example_stats.append(0)

        # Iterate over all training runs to calculate the total forgetting count for current example
        for i in range(len(unlearned_per_presentation_all)):

            # Get all presentations when current example was forgotten during current training run
            stats = unlearned_per_presentation_all[i][example_id]

            # If example was never learned during current training run, add max forgetting counts
            if np.isnan(first_learned_all[i][example_id]):
                example_stats[-1] += npresentations
            else:
                example_stats[-1] += len(stats)

    print('Number of unforgettable examples: {}'.format(
        len(np.where(np.array(example_stats) == 0)[0])))
    return np.array(example_original_order)[np.argsort(
        example_stats)], np.sort(example_stats)
```

|| त्वं ज्ञानमयो विज्ञानमयोऽसि ||

For CIFAR Dataset

- Process 3: Train the learning algorithm again and repeat process 1 and 2 with random data removal of samples / examples

```
if args['keep_lowest_n'] < 0:
    # Remove remove_n number of examples from the train set at random
    train_indx = npr.permutation(np.arange(len(
        train_dataset.train_labels)))[:len(train_dataset.train_labels) -
                                      args['remove_n']]
```

```
elif args['remove_subsample']:
    # Remove remove_sample number of examples at random from the first keep_lowest_n examples
    # Useful when the first keep_lowest_n examples have equal forgetting counts
    lowest_n = np.array(ordered_indx)[0:args['keep_lowest_n']]
    train_indx = lowest_n[npr.permutation(np.arange(
        args['keep_lowest_n']))[:args['keep_lowest_n'] - args['remove_subsample']]]
    train_indx = np.hstack((train_indx,
                            np.array(ordered_indx)[args['keep_lowest_n']:]))
```

For CIFAR Dataset

- Process 4: Train the learning algorithm again and repeat process 1 and 2 with sorted removal of samples / examples

```
# Get indices of examples that should be used for training
if args['sorting_file'] == 'none':
    train_indx = np.array(range(len(train_dataset.train_labels)))
else:
    try:
        with open(
                os.path.join(args['input_dir'], args['sorting_file']) + '.pkl',
                'rb') as fin:
            ordered_indx = pickle.load(fin)['indices']
    except IOError:
        with open(os.path.join(args['input_dir'], args['sorting_file']),
                  'rb') as fin:
            ordered_indx = pickle.load(fin)['indices']

    # Get the indices to remove from training
    elements_to_remove = np.array(
        ordered_indx)[args['keep_lowest_n']:args['keep_lowest_n'] + args['remove_n']]

    # Remove the corresponding elements
    train_indx = np.setdiff1d(
        range(len(train_dataset.train_labels)), elements_to_remove)
```

References:

Madhu S. Advani and Andrew M. Saxe. High-dimensional dynamics of generalization error in neural networks. CoRR, abs/1710.03667, 2017.

Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards AI. In Large Scale Kernel Machines. MIT Press, 2007.

Yoshua Bengio, Je´roˆme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Proceedings of the 26th annual international conference on machine learning, pp. 41–48. ACM, 2009.

Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. Journal of artificial intelligence research, 11:131–167, 1999.

Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active Bias: Training More Accurate Neural Networks by Emphasizing High Variance Samples. In Advances in Neural In- formation Processing Systems, pp. 1002– 1012, 2017.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing Gradi- ent Descent Into Wide Valleys. ICLR ’17, 2016.

Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

Yang Fan, Fei Tian, Tao Qin, and Jiang Bian. Learning What Data to Learn. 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptationof deep networks. In Proc. of ICML, 2017.

S. Hochreiter and J. Schmidhuber. Flat minima. Neural Computation, 9(1):1–42, 1997.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4700–4708, 2017.

References:

Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data- driven curriculum for very deep neural networks on corrupted labels. In Proceedings of the 35th International Conference on Machine Learning. PMLR, 2018.
George H John. Robust decision trees: removing outliers from databases. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pp. 174–179. AAAI Press, 1995.
Angelos Katharopoulos and Franois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In Jennifer G. Dy and Andreas Krause (eds.), ICML, volume 80 of JMLR Workshop and Conference Proceedings, pp. 2530–2539. JMLR.org, 2018. URL http: //dblp.uni-trier.de/db/conf/icml/icml2018.html#KatharopoulosF18.
Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Pe- ter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. arXiv preprint arXiv:1609.04836, 2016.
Tae-Hoon Kim and Jonghyun Choi. Screenernet: Learning curriculum for neural networks. CoRR, abs/1801.00904, 2018. URL http://dblp.uni-trier.de/db/journals/ corr/ corr1801.html#abs-1801-00904.
Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL http: //arxiv.org/abs/1412.6980. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, and Others. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences, pp. 201611835, 2017.
Robert Kleinberg, Yuanzhi Li, and Yang Yuan. An alternative view: When does sgd escape lo- cal minima? CoRR, abs/1802.06175, 2018. URL http://dblp.uni-trier.de/db/ journals/corr/ corr1802.html#abs-1802-06175.
Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh (eds.), ICML, volume 70 of JMLR Workshop and Conference Proceedings, pp. 1885–1894. JMLR.org, 2017. URL http://dblp.uni-trier.de/db/ conf/icml/icml2017.html#KohL17.
Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https: //www.cs.toronto.edu/˜kriz/learning-features-2009-TR.pdf. M Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-Paced Learning for Latent Variable Models. In Proc. of NIPS, pp. 1–9, 2010. Y. LeCun, C. Cortes C., and C. Burges. The mnist database of handwritten digits. 1999. URL http://yann.lecun.com/exdb/mnist/.
Yong Jae Lee and Kristen Grauman. Learning the easy things first: Self-paced visual category discovery. In Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, pp. 1721–1728. IEEE, 2011.

References:

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. CoRR, abs/1804.08838, 2018. URL http:// dblp.uni-trier. de/db/journals/corr/corr1804.html#abs-1804- 08838.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Psychology of learning and motivation, volume 24, pp. 109–165. Elsevier, 1989.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. CoRR, abs/1412.6614, 2014. URL http: //dblp.unitrier.de/db/journals/corr/corr1412.html#NeyshaburTS14. Guillermo Valle Perez, Chico Q. Camargo, and Ard A.

Louis. Deep learning generalizes be- cause the parameter-function map is biased towards simple functions. CoRR, abs/1805.08522, 2018. URL http://dblp.unitrier.de/db/ journals/corr/corr1805.html# abs-1805-08522. Sachin Ravi and Hugo Larochelle.

Optimization as a model for few-shot learning. In Proc. of ICLR, 2017. Hippolyt Ritter, Aleksandar Botev, and David Barber. Online Structured Laplace Approximations For Overcoming Catastrophic Forgetting. 2018. URL http://arxiv.org/abs/1805. 07810.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015. Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The Im- plicit Bias of Gradient Descent on Separable Data. 2017. URL http://arxiv.org/abs/ 1710.10345. Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus.

Training convolutional networks with noisy labels. arXiv preprint arXiv:1406.2080, 2014. R. Tachet, M. Pezeshki, S. Shabanian, A. Courville, and Y. Bengio. On the learning dynamics of deep neural networks. 2018. doi: arXiv:1809.06848v1. URL https://arxiv.org/abs/ 1809.06848.

Huan Wang, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. Identifying Generalization Properties in Neural Networks. pp. 1–23, 2018. doi: arXiv:1809.07402v1. URL http:// arxiv.org/abs/1809.07402. Tengyu Xu, Yi Zhou, Kaiyi Ji, and Yingbin Liang.

Convergence of sgd in learning relu models with separable data. CoRR, abs/1806.04339, 2018. URL http://dblp.uni-trier.de/db/ journals/corr/corr1806.html#abs-1806-04339.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2016. URL http://arxiv. org/abs/1605.07146. cite arxiv:1605.07146.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. arXiv preprint arXiv:1611.03530, 2016.

Peilin Zhao and Tong Zhang. Stochastic Optimization with Importance Sampling for Regularized Loss Minimization. In Proc. of ICML, 2015.