Optimizing Data Curation through Spectral Analysis and Joint Batch Selection (SALN)

Mohammadreza Sharifi*

Abstract

In modern deep learning models, long training times and large datasets present significant challenges to both efficiency and scalability. Effective data curation and sample selection are crucial for optimizing the training process of deep neural networks. This paper introduces SALN, a method designed to prioritize and select samples within each batch rather than from the entire dataset. By utilizing jointly selected batches, SALN enhances training efficiency compared to independent batch selection. The proposed method applies a spectral analysis-based heuristic to identify the most informative data points within each batch, improving both training speed and accuracy. The SALN algorithm significantly reduces training time and enhances accuracy when compared to traditional batch prioritization or standard training procedures. It demonstrates up to an 8x reduction in training time and up to a 5% increase in accuracy over standard training methods. Moreover, SALN achieves better performance and shorter training times compared to Google's JEST method developed by DeepMind. The code and Jupyter notebooks are available at `github.com/rezasharifi82/SALN`.

Keywords: Spectral Analysis, Data Curation, Joint Batch Selection

1 Introduction

The effectiveness of a deep learning model's training process largely depends on the quality of the data to which the model is exposed. High-quality data enables the model to learn accurately and generalize well to new, unseen data, improving its overall performance.[5]. Training a deep neural network model on a curated dataset can significantly improve both accuracy and efficiency [8].

There are many well-known methods that apply a kind of data point selection heuristic on the individual manner[7]. However, batches of data can have interdependencies [1] that may lead to much better results than processing them individually [23]. Moreover, there are some approaches for selecting well-informed batches of data and using them in the training process. These methods represent algorithms or criteria for selecting batches of data that may gain the most information from our dataset [8].

In this paper, a heuristic is proposed to select the most informative batch of data, which can lead to better accuracy as well as shorter training time. This method is based on the principles of eigenvalues and eigenvectors, leveraging these mathematical tools to identify the core structure and the most informative parts of the data [16, 20], rather than using random selection improved by loss control [8].

2 Related Work

JEST: Selecting an informative batch of data can lead to a dramatic improvement in training time and computational resource usage of the model. Recent research conducted by Google DeepMind has shown that selecting an informative batch of data can lead the model to significantly reduce iterations and computations [8]. Moreover, it can achieve the same or even better performance compared to using all of the data.

Spectral approaches:Past research has explored various methods for enhancing the efficiency of machine learning models, particularly through the use of spectral techniques. Eigenvectors and eigenvalues have been central to Principal Component Analysis (PCA) and Spectral Clustering, where they are used to identify the most significant dimensions in the data. These methods have proven effective in reducing the complexity of data and uncovering latent structures, which can improve model performance [16, 20]. In the context of data selection, prior work has focused primarily on filtering or pruning individual samples based on heuristics such as importance sampling or noise reduction [24]. Additionally, methods such as Core-set selection have been introduced to identify a small but representative subset of data for training large-scale models, reducing computational costs while maintaining accuracy as good as possible [2]. These approaches, however, often operate on individual samples and do not take significant advantage of the relationships between batches of data points. Recent works have also explored spectral learning techniques to guide batch selection. Methods like SpectralNet [25] utilize the spectral properties of data to group data points into batches that better capture the under-

*Department of Computer Science, Ferdowsi University of Mashhad, `sharifi.mohammadreza@mail.um.ac.ir`

lying structure of the dataset, resulting in more efficient training.

Batch Selection: Batch selection in machine learning has gained considerable attention in recent years [8] as a strategy to improve the efficiency of training models, particularly deep neural networks. Unlike methods that focus on individual data point selection, batch selection methods aim to identify groups of data that are most informative or representative together, thereby accelerating the learning process and improving model performance [8]. Early work on mini-batch gradient descent [4] demonstrated that training on small batches of data could significantly reduce computational time while retaining much of the accuracy of full-batch methods. Since then, several approaches have been developed to enhance the effectiveness of batch selection. For instance, Core-set selection [24] selects small but representative subsets of data that approximate the original dataset's distribution, leading to faster convergence and lower resource usage.

Active Learning: In the context of active learning, batch active learning approaches such as BatchBALD [17] seek to select batches of data that maximize the information gain during training. This contrasts with traditional active learning techniques that focus on one data point at a time, demonstrating the benefits of jointly selecting data in batches for training efficiency.

3 Methods

3.1 Data Preparation

Data preprocessing plays a crucial role in the training process of machine learning models [11]. Proper preprocessing can mitigate issues such as overfitting [19] and ensure the model generalizes better to unseen data [3]. In this study, I have used the Oxford-IIIT Pet Dataset [22] as the primary dataset and the CIFAR-10 as the secondary dataset for training the model. The Oxford-IIIT Pet Dataset contains images of 37 different breeds of cats and dogs, which are labeled and categorized. Meanwhile, the CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class and training and test sets of 50,000 and 10,000 images, respectively.

The data augmentation techniques used in this study are as follows:

- **Horizontal Flip**: Random horizontal flipping helps the model improve its robustness to horizontal orientation variations [19].

- **Rotation**: Random rotation by $\alpha = 15$ can significantly help the model become invariant to rotational changes [26].

- **Change on jitter**: To prevent overfitting, one of the most significant augmentations is to adjust the jitter parameters [13]. That involves the following configurations:
  - Brightness: 0.2
  - Contrast: 0.2
  - Saturation: 0.2
  - Hue: 0.1

- **Resize**: This ensures a uniform input size for all images, which is essential for deep neural networks [27].

- **Normalization**: Normalizing the pixel values to standardize the input data. The parameters of mean and standard deviation has been set to an appropriate number due to ImageNet statistics. [15, 19].
  - *Mean*: $[0.485, 0.456, 0.406]$
  - *Std.*: $[0.229, 0.224, 0.225]$

These augmentations were applied to the training data. However, for the test data, since data augmentation is not appropriate [10, 26] , I have just used resizing, cropping, and normalization. Also, I split my dataset into training, validation, and test sets.

3.2 The model

In this study, I used the pre-trained ResNet-18 model as my main classifier for several reasons:

- **Vanishing Gradients problem**: The residual connections in ResNet-18 efficiently prevent the vanishing gradient problem by allowing information to bypass layers, facilitating the training of deeper networks [12].

- **Strong Transfer Learning Performance**: ResNet-18 is strong enough to capture complex features while still being lightweight, which is enhanced by its pre-trained configuration [18, 12].

- **Accessability**: Resnet-18 is available in PyTorch and there is no need to be built from scratch.

So, because of all the reasons listed above, I decided to use the pre-trained version of ResNet-18 as a lightweight, reliable model that is generalized enough to perform significant classification tasks. As the final layer of this model, I modified the output of the fully-connected layer to have appropriate units, which is corresponds to the number of classes in each dataset.

3.3 Loss function and Optimizer

For all of the experiments, I have used *CrossEntropyLoss* as the ideal loss function for multi-class classification tasks. It combines *LogSoftMax* and negative log likelihood into one function [10].

Meanwhile, for the optimization task, I have used *Stochastic Gradient Descent (SGD)* with $momentum = 0.9$ and $learning\ rate = 0.001$.

3.4 Joint example selection

Joint example selection with sigmoid loss is a new method introduced by Google DeepMind [8]. It is an intuitive procedure for selecting data in a meaningful way from a batch, rather than relying on random selection, to feed into the model. This algorithm is based on several important principles [8]:

- **Loss calculation**: This algorithm needs both reference loss and learner loss.
  - *Reference loss*: The model's current performance on each example.
  - *Learner loss*: A reference loss that could represent a baseline(performance of a reference model).
- **Filter ratio**: This ratio is a hyperparameter which is used to select the amount of data.
- **Interaction**: This algorithm uses an intuitive procedure to prioritize those data from a batch which are most informative rather than others. In fact, it calculates the impression of selected data on the remaining ones, as well as the remaining ones on the selected data. In the other word, it can measure how the selected data could interact with a remaining one.
- **Not using old ones**: This algorithm uses a probability distribution which assigns a very-low weight to the previously selected samples. This action would guarantee the algorithm will not choose the previously-selected samples.
- **Size of chunks**: This algorithm has a parameter that adjusts the number of data points representable in each chunk, associated with the filter ratio.

3.5 SALN(My method)

In this approach, I've utilized spectral analysis to identify and prioritize the most informative data based on their structural significance within a batch. Since using spectral approaches can leverage the core structure of the data [20, 21], my methodology consists of the following steps:

1. **Feature extraction**: Using a pre-trained model without fine-tuning to extract features from the whole dataset can dramatically reduce time and computational resources [10]. This step is crucial in this study in order to have a comprehensive features for each data point. I have used ResNet-50 as a reference pre-trained model to extract the features of each data and convert them to a vector.
2. **Compute similarity matrix**: For each batch of data,computing a similarity matrix S using *cosine similarity* can significantly improve the algorithm's understanding of meaningful structures, which is useful for further spectral analysis. [28, 20].
3. **Degree matrix**: The degree matrix D is a diagonal matrix where each element D_{ii} represents the sum of the similarities of all relevant data point i in a dataset. The degree matrix plays a key role in defining the Laplacian matrix [6].
4. **Laplacian matrix**: The Laplacian matrix L is derived from the degree matrix D and the similarity matrix S, typically computed as $L = D - S$. The Laplacian matrix encodes the structure of the dataset by capturing the relationships between data points.
5. **Eigen decomposition**: To score the data and their relationships, the second smallest eigenvalue of the Laplacian matrix is calculated, which corresponds to the Fiedler value [9]. This approach utilizes the spectral properties of the Laplacian to identify samples that are essential for preserving the structural integrity of the current batch [9, 20, 21].
6. **Informative indices**: The resulting vector consists of indices corresponding to the most informative data points in the current batch. The size of this vector is determined by the filter ratio.
7. **Joint batch sampling**: Similar to the *JointExample Selection* algorithm [8], the introduced method implicitly uses the similar criteria, with a slight difference in the scoring mechanism, which is handled through spectral analysis.

3.6 Algorithm

The following algorithm describes the procedure for *Joint batch sampling* using Laplacian matrix and spectral analysis. The algorithm takes as input a batch of feature vectors corresponding to current batch's data. These vectors had been extracted in the manner which has described in previous section. Now the algorithm selects data points which are the most informative in this batch relative to filter ratio parameter. The *filter*

ratio parameter, is a hyper-parameter which can specify how many data should be selected from this batch. At the end, the algorithm returns the most informative indices, corresponding to extracted properties from Fiedler vector. These indices are going to be used further in the training loop with an specific intention to reduce number of processing input data.

```
def Joint_batch_sampling(current_batch_data, filter_ratio=0.8):

    # Number of images (samples) in this batch
    n_images = current_batch_data.size(0)

    batch_data = current_batch_data.detach().cpu()

    # Flatten the batch data to 2D
    batch_data_flat = batch_data.view(n_images, -1).numpy()

    # Number of samples to select
    n_draws = int(n_images * (1 - filter_ratio))

    # Compute cosine similarity matrix
    similarity_matrix = cosine_similarity(batch_data_flat)

    # Compute the degree matrix
    degree_matrix = np.diag(similarity_matrix.sum(axis=1))

    # Compute the Laplacian matrix
    laplacian_matrix = degree_matrix - similarity_matrix

    # Find eigens of Laplacian matrix
    eigenvalues, eigenvectors = np.linalg.eigh(laplacian_matrix)

    # Extract the Fiedler vector
    fiedler_vector = eigenvectors[:, 1]

    # Sort the absolute values of the Fiedler
    informative_indices = np.argsort(np.abs(fiedler_vector))[-n_draws:]

    return informative_indices
```

Algorithm 1: Joint batch sampling using spectral analysis(SALN)

4 Experiments

All the experiments were conducted on those datasets which have described in the previous section. The experiments were conducted on a Google-Colab service wit T4-GPU. Number of epochs and other important factors were kept the same. The experiments have been conducted on the same model, ResNet-18, with the same hyper-parameters and configurations. The only difference between the experiments is the method of selecting the data points in each batch during the 25 epochs.

4.1 Primary Dataset: Oxford-IIIT Pet

The primary dataset used in this study is the Oxford-IIIT Pet Dataset, which contains images of 37 different breeds of cats and dogs. The dataset is labeled and categorized, making it suitable for classification tasks.

4.1.1 Standard training and SALN

The following experiment has been conducted to compare the performance of the standard training method as the baseline, with the proposed SALN method. The standard training method uses no specific selection criteria in order to select the data, while SALN employs spectral analysis to identify the most informative data points in each batch. The standard training method processes the entire dataset without any form of selective sampling or data reduction. Every data point is used during training, and no prioritization or filtering is applied to optimize the learning process. The results of this experiment consisted from several reports:

- Comparison between training and validation accuracy(%) of SALN and standard training, which has represented in Table 1.

Method	Training	Validation
SALN	96.5%	94.02%
Standard Training	86.1%	76.49%

Table 1: Accuracy Comparison of SALN and Standard Training

- Comparison between training and validation loss of SALN and standard training, which has represented in Table 2.

Method	Training	Validation
SALN	0.1606	0.2102
Standard Training	0.4983	0.6546

Table 2: Loss Comparison of SALN and Standard Training

- Comparison between test set accuracy of SALN and standard training, which has represented in Table 3.

Method	Test Accuracy
SALN	86.8%
Standard Training	82.02%

Table 3: Comparison of SALN and Standard Training Test results

- Comparison between training time of SALN and standard training, which has represented in Table 4.

Method	Training Time (minute)
SALN	6.31
Standard Training	24.48

Table 4: Training time Comparison of SALN and Standard Training

- Standard Training accuracy and loss curves over 25 epochs, which has represented in Figure 1.

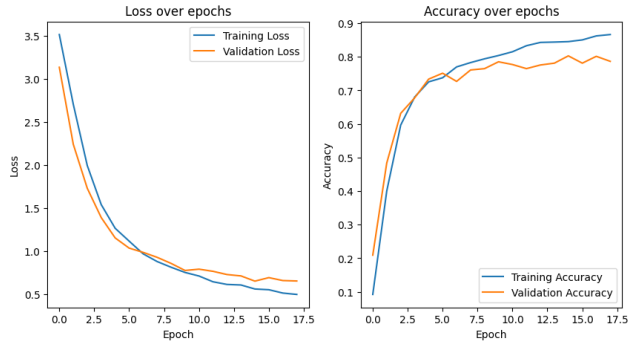

Figure 1: Accuracy and Loss Curves of **Standard-Training** Method over epochs

- SALN accuracy and loss curves over 25 epochs, which has represented in Figure 2.

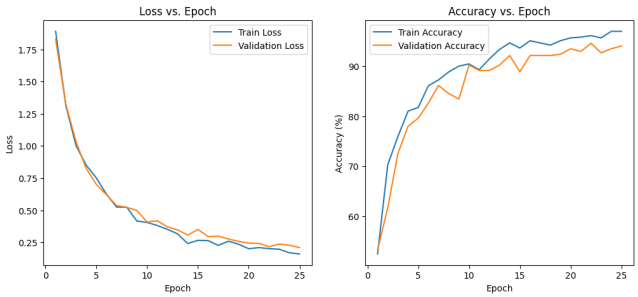

Figure 2: Accuracy and Loss Curves of **SALN** Method over epochs

4.1.2 JEST and SALN

The following experiment has been conducted to compare the performance of the JEST method which has proposed by Google DeepMind, and the SALN method. The JEST method uses a specific selection criteria in order to select the informative data. This criteria were discussed in the previous section. The results of this experiment consisted from several reports:

- Comparison between training and validation accuracy(%) of SALN and JEST, which has represented in Table 5.

Method	Training	Validation
SALN	96.5%	94.02%
JEST	79.59%	75.54%

Table 5: Accuracy Comparison of SALN and JEST

- Comparison between training and validation loss of SALN and JEST, which has represented in Table 6.

Method	Training	Validation
SALN	0.1606	0.2102
JEST	0.7309	0.8060

Table 6: Loss Comparison of SALN and JEST

- Comparison between test set accuracy of SALN and JEST, which has represented in Table 7.

Method	Test Accuracy
SALN	86.8%
JEST	87.55%

Table 7: Loss Comparison of SALN and JEST

- Comparison between training time of SALN and JEST, which has represented in Table 16.

Method	Training Time (minute)
SALN	6.31
JEST	14.87

Table 8: Training time Comparison of SALN and JEST

- JEST accuracy and loss curves over 25 epochs, which has represented in Figure 3.

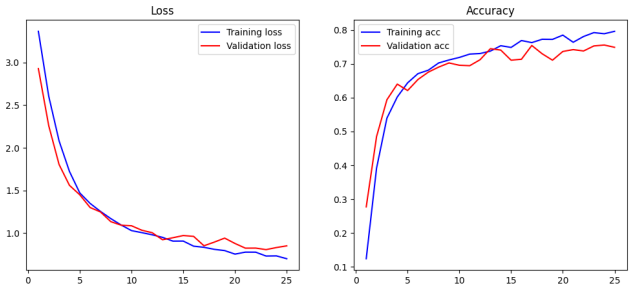

Figure 3: Accuracy and Loss Curves of **JEST** Method over epochs

- SALN accuracy and loss curves over 25 epochs, which has represented in Figure 2.

4.2 Secondary Dataset: CIFAR-10

The secondary dataset used in this study is the CIFAR-10 dataset, which contains 60,000 32x32 color images in 10 classes, with 6,000 images per class and training and test sets of 50,000 and 10,000 images, respectively.

4.2.1 Standard training and SALN

Same as the primary dataset, the following experiment has been conducted to compare the performance of the standard training method as the baseline, with the proposed SALN method. The results of this experiment consisted from several reports:

- Comparison between training and validation accuracy(%) of SALN and standard training, which has represented in Table 9.

Method	Training	Validation
SALN	81.99%	84.08%
Standard Training	85.54%	86.82%

Table 9: Accuracy Comparison of SALN and Standard Training

- Comparison between training and validation loss of SALN and standard training, which has represented in Table 10.

Method	Training	Validation
SALN	0.5080	0.4607
Standard Training	0.4055	0.3664

Table 10: Loss Comparison of SALN and Standard Training

- Comparison between test set accuracy of SALN and standard training, which has represented in Table 11.

Method	Test Accuracy
SALN	82.46%
Standard Training	82.56%

Table 11: Comparison of SALN and Standard Training Test results

- Comparison between training time of SALN and standard training, which has represented in Table 12.

Method	Training Time (minute)
SALN	22.66
Standard Training	39.41

Table 12: Training time Comparison of SALN and Standard Training

- Standard Training accuracy and loss curves over 25 epochs, which has represented in Figure 4.

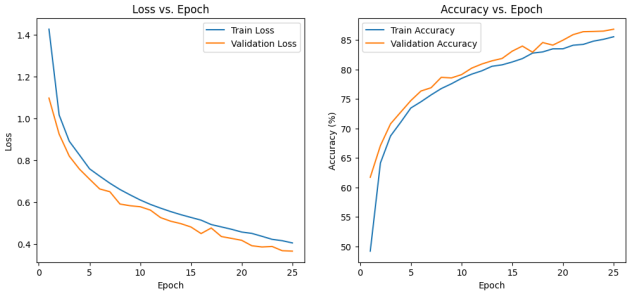

Figure 4: Accuracy and Loss Curves of **Standard-Training** Method over epochs

- SALN accuracy and loss curves over 25 epochs, which has represented in Figure 5.

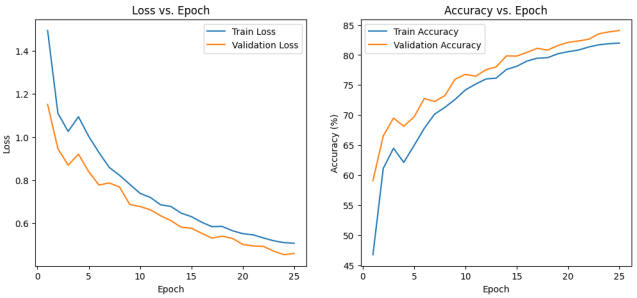

Figure 5: Accuracy and Loss Curves of **SALN** Method over epochs

4.2.2 JEST and SALN

Same as primary dataset, the following experiment has been conducted to compare the performance of the

JEST method which has proposed by Google DeepMind, and the SALN method. The results of this experiment consisted from several reports:

- Comparison between training and validation accuracy(%) of SALN and JEST, which has represented in Table 13.

Method	Training	Validation
SALN	81.99%	84.08%
JEST	72.81%	76.86%

Table 13: Accuracy Comparison of SALN and JEST

- Comparison between training and validation loss of SALN and JEST, which has represented in Table 14.

Method	Training	Validation
SALN	0.5080	0.4607
JEST	0.7832	0.6732

Table 14: Loss Comparison of SALN and JEST

- Comparison between test set accuracy of SALN and JEST, which has represented in Table 15.

Method	Test Accuracy
SALN	82.46%
JEST	77.72%

Table 15: Loss Comparison of SALN and JEST

- Comparison between training time of SALN and JEST, which has represented in Table 16.

Method	Training Time (minute)
SALN	22.66
JEST	37.68

Table 16: Training time Comparison of SALN and JEST

- JEST accuracy and loss curves over 25 epochs, which has represented in Figure 6.

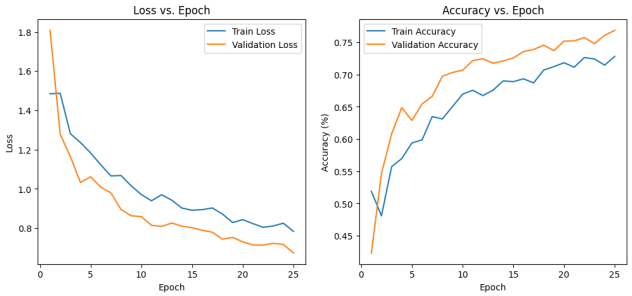

Figure 6: Accuracy and Loss Curves of **JEST** Method over epochs

- SALN accuracy and loss curves over 25 epochs, which has represented in Figure 5.

4.3 SALN data-selection visualization

In this section, the data selection process of the SALN algorithm will be visualized. The visualization will show the top 50% data points that were selected by the algorithm in each batch. Here, I have selected Batch No.0 and Batch No.1 from the Oxford-IIIT Pet dataset and, Batch No.50 and Batch No.51 from the CIFAR-10 dataset.

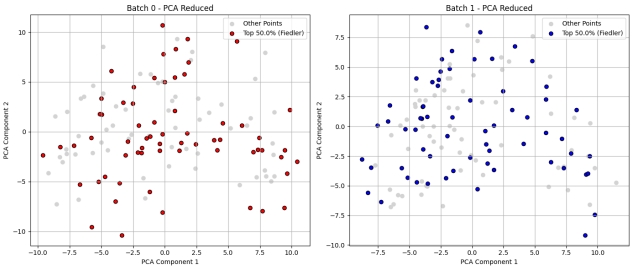

Figure 7: SALN Data Selection Visualization of Oxford-IIIT Pet Dataset

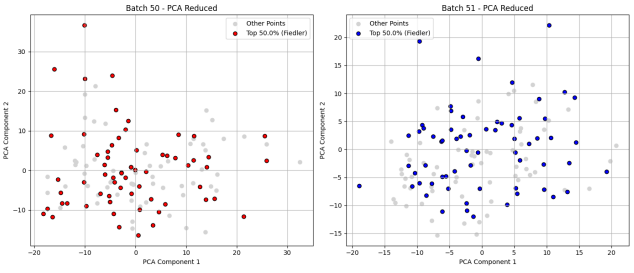

Figure 8: SALN Data Selection Visualization of CIFAR-10 Dataset

4.4 Analysis of SALN Weights and Filters

In this section, an analysis of the model which has trained on Oxford-IIIT Pet dataset will be presented.

Since ResNet-18 is well-known, the focus will be on the details of the final fully connected layer. The analysis would be in two parts:

- **Weights Heatmap of fully connected layer**: The heatmap in Figure 9 visualizes the weights of the final fully connected (fc) layer in the model. Each row represents a neuron in the fc layer, and each column corresponds to a weight connected to an input feature.

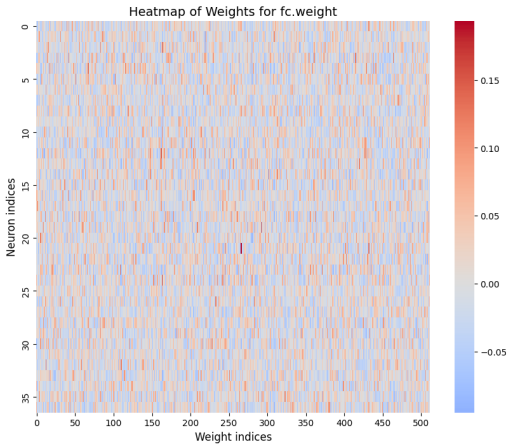

Figure 9: SALN Weights Heatmap of fully connected layer

- **Weights Distribution of fully connected layer**:Figure 10 shows the distribution of the weights in the final fully connected (fc) layer. The histogram represents how the weights are spread across all neurons in this layer. A large concentration of weights near zero suggests that many connections have small magnitudes, which is often seen in models that are well-regularized. The range of the weight values indicates how much emphasis the model places on different input features, with any outliers potentially showing connections that have a stronger influence on the final predictions.

5 Discussion

The proposed method, SALN, introduces a novel approach to jointly batch sampling through the application of spectral methodologies. This technique enhances the selection of data points during training, aiming to accelerate the learning process when compared to standard training methods.

The results from our experiments indicate that SALN offers a substantial reduction in training time by utilizing an optimized jointly batch sampling mechanism. The spectral techniques employed in this method, enable the model to prioritize the most informative data.

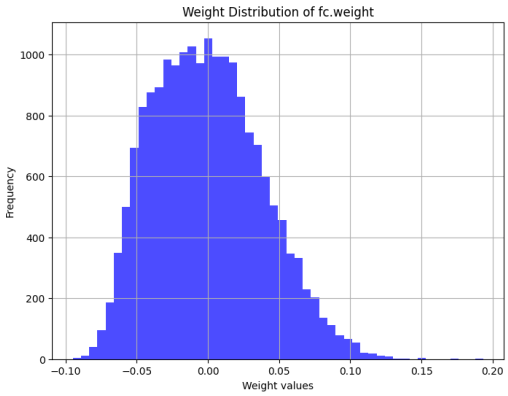

Figure 10: SALN Weights Heatmap of fully connected layer

And thereby, improving both the speed and effectiveness of training procedure.

Additionally, the experiments demonstrate the high potential of *data bootstrapping* in deep neural network training process. By focusing on batches that maximize the learnability, SALN ensures that the model is consistently exposed to the most challenging and relevant examples, which contributes to more efficient and robust learning.

In summary, SALN provides a promising advancement in batch sampling strategies, offering significant improvements in training efficiency, using less time than JEST, without sacrificing model performance.

References

[1] J. Ash, C. Zhang, A. Krishnamurthy, J. Langford, and A. Agarwal, Batch Active Learning at Scale. *arXiv preprint arXiv:2107.14263*, 2021.

[2] O. Bachem, M. Lucic, and A. Krause, Practical Coreset Constructions for Machine Learning. *NIPS*, 2017.

[3] C. M. Bishop, Pattern Recognition and Machine Learning. *Springer*, 2006.

[4] L. Bottou, Large-Scale Machine Learning with Stochastic Gradient Descent. *Proceedings of COMPSTAT*, 2010.

[5] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, The Effects of Data Quality on Machine Learning Performance. *arXiv preprint arXiv:2207.14529*, 2022.

[6] F. R. K. Chung, *Spectral Graph Theory*. American Mathematical Society, 1997.

[7] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia, Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[8] T. Evans, N. Parthasarathy, H. Merzic, and O. J. Henaff, Data curation via joint example selection further accelerates multimodal learning. *arXiv preprint arXiv:2406.17711*, 2024.

[9] M. Fiedler, Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2), 298–305, 1973.

[10] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[11] I. Guyon and A. Elisseeff, An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3:1157-1182, 2006.

[12] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778.

[13] A. G. Howard, Some Improvements on Deep Convolutional Neural Network Based Image Classification. *arXiv preprint arXiv:1312.5402*, 2013.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, Densely Connected Convolutional Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4700-4708.

[15] S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *International Conference on Machine Learning (ICML)*, 2015.

[16] I. T. Jolliffe, Principal Component Analysis, 2nd ed. Springer, 2002.

[17] A. Kirsch, J. Van Amersfoort, and Y. Gal, BatchBALD: Efficient and Diverse Batch Acquisition for Deep Bayesian Active Learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[18] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, Big Transfer (BiT): General Visual Representation Learning. *European Conference on Computer Vision (ECCV)*, 2020.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[20] U. von Luxburg, A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395-416, 2007.

[21] A. Y. Ng, M. I. Jordan, and Y. Weiss, On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems (NIPS)*, 14, 849–856, 2002.

[22] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, Cats and Dogs. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[23] J. Prall, Processing Paradigms: Stream vs Batch in the ML Era. *Airbyte*, 2023.

[24] O. Sener and S. Savarese, Active Learning for Convolutional Neural Networks: A Core-Set Approach. *International Conference on Learning Representations (ICLR)*, 2018.

[25] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger, SpectralNet: Spectral Clustering Using Deep Neural Networks. *International Conference on Learning Representations (ICLR)*, 2018.

[26] C. Shorten and T. M. Khoshgoftaar, A Survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(1):1-48, 2019.

[27] K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[28] A. Singhal, Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4), 35–43, 2001.

[29] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, On the Importance of Initialization and Momentum in Deep Learning. *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.