# Investigation of Parallel and Hyperparameters Strategy on Performance of Image Classification Training

Yannan Cao[1,*], Weiran Shen[2]

[1] Software Engineering, Dalian University of Technology, Shenyang, 116081, China

[2] Software Engineering, Beijing University of Technology, Beijing, 100124, China

*natascha1203@mail.dlut.edu.cn

**Abstract.** Convolutional Neural Networks (CNNs) have witnessed widespread adoption in the domain of image classification, while deep neural networks have been developed to tackle more intricate tasks. In the experimental investigation, a remarkable downward trend in GPU utilization was observed as the batch size of the LeNet model was increased, regardless of the parallel or non-parallel mechanism employed. The research findings establish that this phenomenon can be ascribed to a constraint in data loading speed, which in turn diminishes the efficiency of training when dealing with larger batch sizes, ultimately resulting in reduced GPU utilization. To mitigate this issue, the data loading thread are enhanced by adjusting the "num_worker" parameter in the dataloader, thereby investigating its impact on GPU utilization. Moreover, a series of comprehensive experiments are conducted to ascertain the appropriate learning rates required for maintaining satisfactory classification accuracy when utilizing large batch sizes. This paper contributes to the field in two primary ways. Firstly, it identifies the cause of decreased GPU utilization when the batch size is increased and proposes a solution to enhance efficiency. Secondly, it verifies the adjustment of learning rates when adopting large batch sizes to achieve comparable loss curves and classification accuracies.

**Keywords:** Convolutional Neural Networks, Image classification, GPU utilization, Data parallel

## 1 Introduction

In recent years, Convolutional Neural Networks (CNNs) have gained widespread adoption for image classification. The common CNN network includes LeNet [1], VGG [2], ResNet [3], etc. Notably, VGG and ResNet have achieved higher classification accuracy by virtue of their deeper architectures.

To deploy DNNs to multiple devices, numerous parallel mechanisms can be considered. To be more specific, data parallel replicate the model on multiple devices with each device

© The Author(s) 2023

P. Kar et al. (eds.), *Proceedings of the 2023 International Conference on Image, Algorithms and Artificial Intelligence (ICIAAI 2023)*, Advances in Computer Science Research 108,
https://doi.org/10.2991/978-94-6463-300-9_105

receives different training data [4]. Model parallel split the network onto multiple devices to deploy larger models [5]. Evaluation of these mechanisms is based on three key factors including training time, GPU utilization and accuracy.

In addition to the strategic aspects of model development, hyperparameters such as batch size also have significant influence on GPU utilization and training time. A larger batch size can potentially lead to increased GPU utilization but simply augmenting the batch size to maximize GPU usage may adversely impact training accuracy. To address this challenge, adjusting the learning rate can yield improved performance. Related study has indicated there is some relationship between batch size and learning rate. Smith et al. [6] proposed the "Super-Convergence" approach, which involves using very large learning rates with small batch sizes to achieve state-of-the-art results on image classification tasks. In addition, Goyal P proposed Liner Scaling rule which indicate that batch size and learning rate should be Proportionally increased to reach a similar accuracy [7].

In this work, several parallel mechanisms are implemented and hyperparameters including batch size, epoch, device number are adjusted to investigate the change of the models' performance. During these experiments, result indicate that the GPU utilization decrease when adapting a larger batch size in both data parallel, model parallel and nonparallel strategy adopting Lenet, this anomalous phenomenon did not happen when experiment with VGG and Resnet. To further analysis this phenomenon, the speed of data loading is changed by alter the child process number of dataloader, the result shows that when improve thread number from 2 to 5, the GPU utilization become positive correlation with batch size, indicating it is the data prepare speed that limit the GPU utilization. Furthermore, an explanation for the observed data-loading restriction exclusively affecting the LeNet model is provided by monitoring the percentage of data loading time. This is due to the fact that LeNet finish it forward, backward and optimizing at a shorter time compared to deep neural network like VGG so it requires a faster data loading speed to fit with its faster training or it will be a drawback for training efficiency. When it comes to network with training time, it can overlap the data loading time thus the tread number no longer limit GPU utilization.

Following the identification of the underlying cause, the learning rate is adjusted to obtain higher accuracy when training in larger batches in subsequent experiments. Typically, accuracy decreases as batch size increases due to fewer iterations. Therefore, larger learning rates are adopteds to fit larger batch sizes and record the loss curve for training. The results demonstrated that a similar loss curve can be observed when following the "pad rule" (i.e., scaling the learning rate and batch size proportionally).

## 2 Method

### 2.1 Dataset Description and Preprocessing

The model is trained with CIFAR10 [8]. The dataset contains 60,000 RGB images from 10

different categories, each with 6,000 images, of which 50,000 are used as the training set and 10,000 for the test set. The images in the dataset are 32×32 pixels in size and are divided into the following 10 categories. The data augmentation including the rotation, scaling, panning and colour changes was employed based on the dataset.

### 2.2 Deep Learning Model

Several CNNs are adopted for the experiments including LeNet, VGG and Resnet. More detailed information about them can be found as follows.

**Convolutional Neural Networks.** CNNs are a class of deep neural networks that are commonly used for image classification and computer vision tasks [9, 10]. A typical CNN consists of multiple convolutional layers, followed by pooling layers, fully connected layers, and sometimes normalization layers. Commonly used CNN architectures include LeNet, AlexNet, VGG, Resnet etc. These architectures differ in their depth, filter sizes, activation functions, regularization techniques, and other design choices. In the experiment, LeNet, VGG and Resnet are implemented. Input of the net is a batch of picture with size 32×32 and channel 3 while the fully connected layer includes 10 neurons representing for ten diverse classes.

### 2.3 GPU-based Parallel

Parallel mechanisms are widely used to improve the efficiency of deep learning training. In the experiment, data parallel and model parallel are implemented. The detailed introduction of the two-parallelism strategy can be found as follows.

**Data Parallel.** Data parallelism is achieved by replicating the neural network across multiple devices, with each copy of the network processing a different subset of the data. The data is typically divided into equal-sized batches, with each batch assigned to a different device.

**Model Parallel.** Model parallelism is achieved by splitting the model into smaller sub-models, which are assigned to various devices. During training, each device computes the forward and backward passes for its assigned sub-model, using the activations from neighboring devices as inputs. The gradients are then accumulated across all devices, and the model parameters are updated based on the combined gradients.

### 2.4 Implementation Details

The experiments are carried out with Pytorch framework using device 2080Ti and 3080Ti. Cross-entropy is adopted cross-entropy as loss function which can be expresses as the formula below:

$$L = \frac{1}{N} \sum_{i=1}^{N} \left( - \sum_{c=1}^{C} y_{ic} \log (p_{ic}) \right) \tag{1}$$

$N$: $samplesize$, $C$: $numberofclasses$ $y_{ic}$: probability that the true label of the ith sample belongs to the cth category, $p_{ic}$ : the predicted probability that the model belongs to the cth category for the ith sample.

The evaluation metric is classification accuracy which is the rate of correct classification on test sample.Typically, The ratio of training set to test set is 5 to 1.

In preliminary tests, LeNet, VGG and ResNet are adapted in data parallel and model parallel mechanism. Hyperparameter such as learning rate, batch sizes and device number are adjusted to investigate the change of model performance including classification accuracy and GPU utilization. Noteworthy, uncommon downward trend of GPU utilization was observed when increasing batch size of LeNet (i.e., 32,64,128,256) in both parallel mechanism (i.e., data and model, using 2 cards) and nonparallel mechanism whereas VGG and Resnet show common upward trend.

To further analysis this anomaly, the data loading processes is altered by setting the parameter num_worker. In Pytorch, dataloader is rebuilt at the beginning of every epoch to load batch data from disk to memory. Typically, the num_workers parameter specifies the number of child processes to be used for data loading. A larger worker number can result in faster data loading speed as more data will be loaded at a time, which can avoid reloading data during an epoch. Further, the perfecting strategy allows workers to load data required in following epochs in advance thereby saving time consuming for subsequent epochs. The following is a specific description of the experiments.

- Training was conducted with different numbers of workers (2, 4, 8, and 16) using the Lenet model. The batch size was multiplied from 8 to 256 in order to investigate GPU utilization.
- The total data loading time of the Lenet and VGG models was recorded, and the data loading time was calculated as a percentage of the total training time. The proportion indicates the influence of data loading efficiency on training efficiency which is positive correlation with GPU utilization.
- In order to obtain better accuracy for large batch sizes, learning rate is adjusted respectively under differently batch size. In detail, learning rate (i.e., 0.001,0.002,0.004,0.008) and batch size (i.e., 32,64,128,256) are adopted.

## 3 Results and Discussion

### 3.1 Training Results of Different Parallel Modes

VGG16 and LeNet were compared in several sets of experiments in model parallel and data parallel shown in Table 1.

**Table 1.** The training results under model parallel and data parallel about VGG16 and LeNet

| Model | GPU Type, Number (Parallel Mode) | Epoch | Lr | Bs | ACC | Time | GPU Utilization |
|---|---|---|---|---|---|---|---|
| LeNet | 2080Ti,1 | 20 | 0.001 | 64 | 60% | 133.63 | 8% |
| LeNet | 2080Ti, 2(model) | 20 | 0.001 | 64 | 60% | 115 | 6%, 4% |
| LeNet | 3080Ti,1 | 10 | 0.001 | 64 | 53% | 61.9 | 7% |
| LeNet | 3080Ti,2(model) | 10 | 0.001 | 64 | 52% | 63.5 | 6%, 2% |
| LeNet | 3080Ti,2(data) | 10 | 0.001 | 64 | 51% | 66.3 | 7%, 6% |
| VGG | 2080Ti,1 | 10 | 0.001 | 64 | 82% | 127.38 | 90% |
| VGG | 2080Ti,2(model) | 10 | 0.001 | 64 | 82% | 111.01 | 90%, 16% |
| VGG | 2080Ti,2(data) | 10 | 0.001 | 64 | 79% | 173.76 | 69%, 61% |

In data parallelism, the time increases significantly, due to the communication overhead between the two GPUs. In the model parallelism, the LeNet convolutional layer is placed on GPU0, and the fully connected layer is placed on GPU1. Training time is reduced without loss of accuracy. Similarly, the same effect can be achieved by training the 13 convolutional layers and 3 fully connected layers of VGG on 2 graphics cards.

In addition to the parallel approach, the influence of batch size on training in hyperparameters is also crucial. As shown in Table 2 and Table 3, LeNet and VGG were trained in two ways, data parallel and model parallelism, respectively. In all experiments, the larger the batch size, the lower the accuracy and the shorter the training time.

**Table 2.** The results of training accuracy, test time, and GPU utilization under different batch sizes (data parallel, 10 epochs, lr_0.001)

| Model (data parallel) | Acc | Time(s) | GPU Utl(RTX3080ti) |
|---|---|---|---|
| LeNet/bs=32 | 60% | 84.09 | 10%,8% |
| LeNet/bs=64 | 51% | 66.35 | 7%,6% |
| LeNet/bs=128 | 45% | 63.23 | 5%,4% |
| VGG/bs=32 | 81% | 689.64 | 91%,85% |
| VGG/bs=64 | 79% | 276.38 | 92%,85% |
| VGG/bs=128 | 77% | 201.71 | 91%,87% |

**Table 3.** The results of training accuracy, test time, and GPU utilization under different batch sizes (model parallel, 10 epochs, lr_0.001)

| Model (model parallel) | Acc | Time(s) | GPU Utl |
|---|---|---|---|
| LeNet/bs=32 | 60% | 69.69 | 7%,3% |
| LeNet/bs=64 | 52% | 63.20 | 6%,2% |
| LeNet/bs=128 | 41% | 58.48 | 4%,1% |
| VGG/bs=32 | 81% | 142.17 | 79%,17% |
| VGG/bs=64 | 79% | 94.51 | 92%,13% |
| VGG/bs=128 | 76% | 79.31 | 93%,10% |

Table 4 shows the training results of ResNet under different batch sizes. During ResNet training, GPU utilization increases with batch size, and forward, backward, and optimize time also increases slightly, and the accuracy decreases.

**Table 4.** The training results of ResNet under different batch sizes

| Model | Acc | Time | GPUutl(RTX3080ti) | FBOtime |
|---|---|---|---|---|
| ResNet/bs=32/bn=1562 | 77% | 101.46 | 46% | 0.0044 |
| ResNet/bs=64/bn=781 | 76% | 65.70 | 58% | 0.0044 |
| ResNet/bs=128/bn=390 | 69% | 58.18 | 68% | 0.0055 |

When the batch size increases, the amount of data processed by the network at the same time increases, which increases the GPU utilization, and the accuracy rate decreases slightly due to the decrease in the number of iterations.

After completing the above experiments, some unconventional phenomena appeared in these experimental data: (1) The larger the LeNet runtime batch size, the lower the GPU utilization. When the batch size is between 8 and 256, the larger the batch size, the smaller the GPU utilization, which defies the related rules. (2) Training with large batch sizes enhances efficiency, but due to faster convergence associated with smaller batches that necessitate complete traversal of training data for weight updates, the omission of full traversal in larger batch sizes leads to diminished final accuracy.

### 3.2 LeNet GPU Utilization Training Abnormal Results

As shown in Table 5, from various parallel training experiments on VGG16 and ResNet models, as the batch size increases, the utilization of the GPU will also increase. Changes in GPU utilization are positively correlated with changes in batch size.

**Table 5.** The GPU utilization of VGG16 and ResNet under different batch sizes

| GPU utl\Batch size | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| VGG16 | 68% | 78% | 89% | 92% |
| ResNet | - | 46% | 58% | 68% |

Table 6 shows the change in GPU utilization when running LeNet on one and two RTX3080 with different batch sizes. This set of data, whether it is single-card training or parallel training of 2-card models, shows that as the batch size increases, the GPU utilization gradually decreases. The downward trend in GPU utilization can be clearly seen in Fig. 1 and Fig. 2.

**Table 6.** Changes of GPU utilization under different batch sizes under single GPU and two GPUs (Model Parallel)

| Batch Size | 8 | | 16 | | 32 | | 64 | | 128 | | 256 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GPU Number | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| GPU utl (%) | 11 | 8 5 | 10 | 9 5 | 10 | 7 4 | 7 | 6 2 | 5 | 4 1 | 4 | 4 1 |

Through subsequent experiments, the influence of hardware environment and some hyperparameters on this abnormal phenomenon was eliminated one by one. This includes factors such as graphics memory, bit width, frequency, epoch, learning rate, data format and size. With the deepening of the experiment, the experimental method is set from the two aspects of training method and the characteristics of the model itself. Fig. 3 and Fig. 4 show the trend of GPU utilization with loss during VGG16 and LeNet training, respectively, showing that there is no obvious correlation between the two.

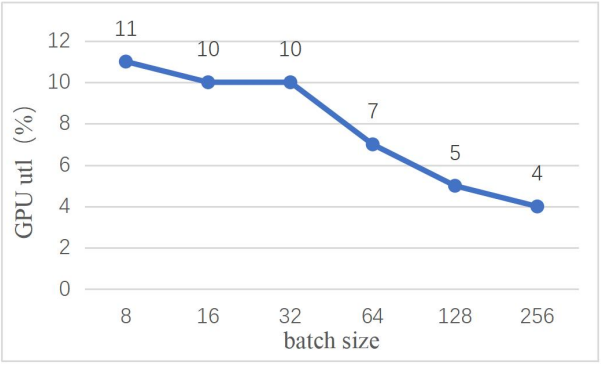


**Fig. 1.** The trend of GPU utilization under different batch sizes (Photo/Picture credit: Original).

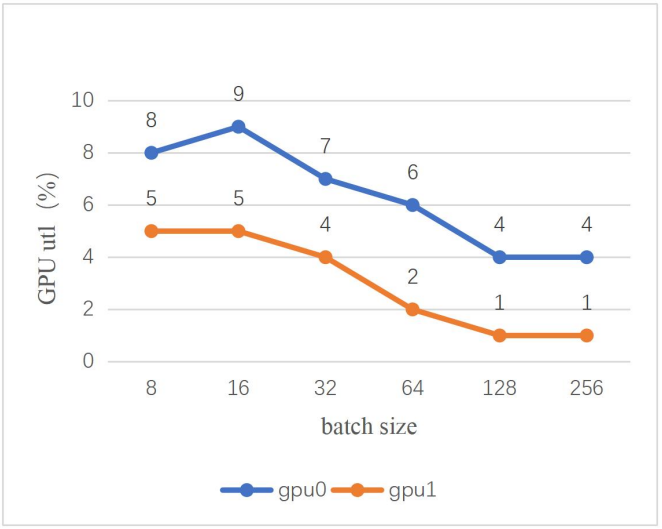


**Fig. 2.** The trend of utilization of two GPUs under different batch sizes (Photo/Picture credit: Original).

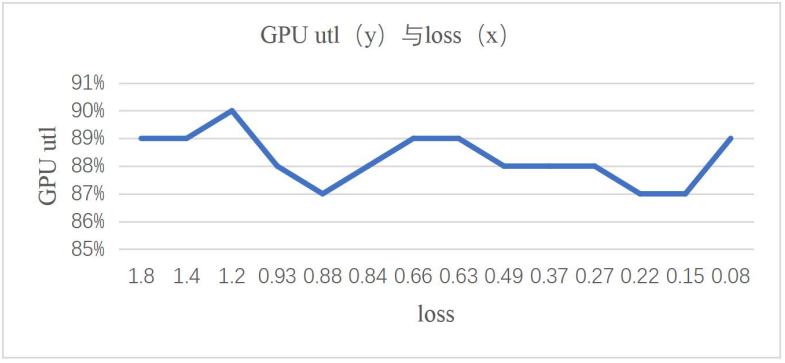


**Fig. 3.** VGG16 training process loss and GPU utilization relationship (Photo/Picture credit: Original).

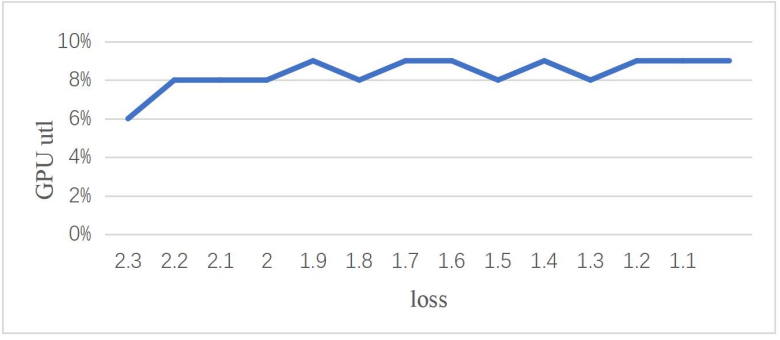


**Fig. 4.** LeNet training process loss and GPU utilization relationship (Photo/Picture credit: Original).

In this experiment, the method of varying the number of child processes for data loading

and preprocessing is used. It is better to set the quantity in a relatively large range, but not the larger the better. Because the larger the number, although there are more threads, the consumption of splitting to each thread is also large, so it will increase the load on the CPU, thereby reducing the utilization of the GPU. If with the increase in the number of child processes, that is, the more num_workers, the data preparation speed is accelerated, the time the GPU waits for data is reduced, and the GPU utilization is positively correlated with the change of batch size, it means that the GPU will have a lot of waiting time when training with a larger batch size.

When the number of child processes is set to 2, 4, 8 and 16, the batch size of different sizes changes, which can be found in Table 7.

**Table 7.** GPU utilization changes with batch size and num_workers

| Num_workers\Batch size | 4 | 8 | 16 | 32 | 64 | 128 | 256 |
|---|---|---|---|---|---|---|---|
| 2 | 12% | 11% | 10% | 10% | 7% | 5% | 4% |
| 4 | - | 12% | 13% | 15% | 14% | 10% | 8% |
| 8 | - | - | 13% | 15% | 19% | 21% | 17% |
| 16 | - | - | - | 15% | 18% | 23% | 27% |

The first line in Table 7 has the same parameters as the initial test results (Table 6), and a numerical reference comparison with a batch size of 4 is added here. When the num_workers is 2, GPU utilization still maintains a downward trend as the batch size increases. Especially when the batch size is between 32 and 256, the num_workers is too small currently. Different changes have occurred in the situation when num_workers are 4 showed on the second line in Table 7. When the batch size is 8, 16, 32, the GPU utilization also increases. When the batch size is 64, 128, 256, the situation is the same as the first line in Table 7. When the batch size of the 4 child processes is above 64, the data preparation speed still limits the operation speed of the GPU. The situation when num_workers is 8 basically conforms to the law of GPU utilization changing with batch size in other models. It also coincides with the situation described in the second line in Table 7, where a num_workers that is too small will affect GPU computing due to slow data transfer speed. The num_workers of 16 is fully in line with expectations.

As can be seen from Fig. 5, the overall trend of GPU occupancy at num_workers 2 and 4 is still declining, especially when the batch size is large. When the num_workers is adjusted to a larger 8 and 16, the data preparation rate finally no longer limits the operation rate of the GPU.

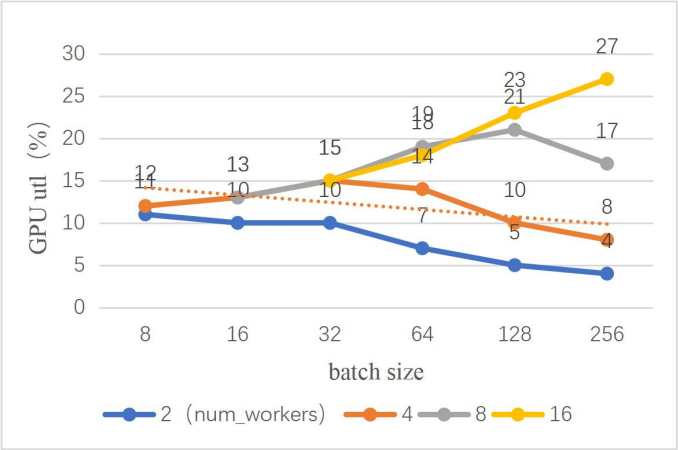


**Fig. 5.** Trend of GPU utilization increasing with the increase of batch size under different num_workers (Photo/Picture credit: Original).

Although the blue and green parts in Fig. 6 have high GPU utilization, the batch size is generally large enough to significantly affect the accuracy. The yellow and gray parts are the more suitable intervals. Obviously, the loading, preprocessing, post-processing, etc. of data are placed on the CPU, that is, other IO tasks of reading and writing. It is a good idea to adjust GPU utilization by adjusting the number of num_workers. It is better to set the quantity in a relatively large range (2-16), but not the larger the better. Because the larger the number, although there are more threads, the consumption of splitting to each thread is also large, so it will increase the load on the CPU, thereby reducing the utilization of the GPU. The number of num_workers is generally used in conjunction with the number of batch sizes.

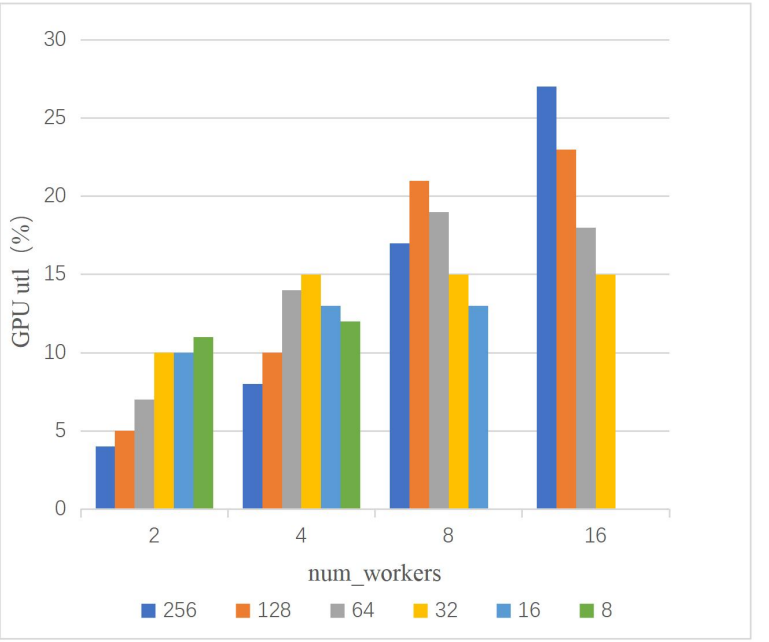


**Fig. 6.** Under the influence of num_workers and batch size, GPU utilization is in the right range (Photo/Picture credit: Original).

For the diametrically opposed performance of VGG and LeNet in Batch Size increase, in addition to the above effects, it is judged that LeNet is idle due to the increase in batch size training data loading time, and VGG obviously takes longer to load data when bs increases, but due to its long training time, the loading time is relatively low, and the overall GPU utilization still shows an upward trend.

In the following experiment, percentage of dataloading time for LeNet, VGG are recorded and GPU utilization is investigated.

**Table 8.** Data loading time proportion and GPU utilization of LeNet on different batch size(2080Ti)

| Batch size | Num_worker | Percentage_of dataloading time | GPU utilization |
|---|---|---|---|
| 32 | 0,2 | 80%,58% | 5%,10% |
| 64 | 0,2 | 87%,71% | 4%,8% |
| 128 | 0,2 | 91%,79% | 4%,7% |
| 256 | 0,2 | 93%,85% | 3%,5% |

**Table 9.** Data loading time proportion and GPU utilization of VGG on different batch size(2080Ti)

| Batch size | Num_worker | Percentage_of dataloading time | GPU utilization |
|---|---|---|---|
| 32 | 0,2 | 37%,8% | 60%89% |
| 64 | 0,2 | 46%,7% | 52%,91% |
| 128 | 0,2 | 49%,6% | 48%,94% |
| 256 | 0,2 | 53%,5% | 45%,95% |

Table 8 and Table 9 illustrate the percentage of data loading time and GPU utilization of LeNet and VGG under various batch sizes and num_worker. Typically, when adapting to no num_worker, both LeNet and VGG shows downward trend when increasing batch size. However, when adapting two processes for loading data, GPU utilization of VGG becomes positive correlation with batch size while LeNet still shows a downward trend.

It is noticed that percentage of dataloading time (PD) is negative correlation with GPU utilization. Furthermore, when PD becomes lower, GPU utilization increases as enlarging batch size. Specifically, large PD means considerable time has to be spent on waiting for training data which can result in longer idle time, hence why it causes lower GPU utl. Both data loading speed (decided by Num_worker) and network passing time (i.e. time spent on forward, backward and optimize) can impact PD. Further, smaller batch size and deep network structure can cause longer data passing time, resulting in larger PD. VGG has larger PD than Lenet because it includes many more layers.

### 3.3 Results of Improving Performance During Larger Batch Size Training

In general, the higher the GPU utilization during training, the higher the efficiency of training. Therefore, when training parameters, it is more inclined to train under the condition of high GPU utilization. However, in the above experiments, high GPU utilization is often generated when the batch size is too high, such as 128, 256. Currently, the accuracy of the model tends to be low and has no training value. In this experiment, it was decided to determine the change of large-batch training performance by monitoring the change of loss during training and the accuracy.

**Table 10.** Variation of accuracy of different batch sizes with learning rate under RTX2080 (LeNet, 20 epochs)

| Batch size | Learning Rate | Accuracy | Time |
|---|---|---|---|
| 32 | 0.0005/0.001/0.002 | 60%/63%/64% | 149/144/154 |
| 64 | 0.0005/0.001/0.002 | 52%/60%/63% | 135/140/131 |
| 128 | 0.0005/0.001/0.002 | 43%/54%/60% | 127/132/135 |

Table 10 shows how increasing the learning rate can maintain accuracy when increasing the batch size. Stochastic gradient descent is continuous and uses minibatches, so it is not easy to

parallelize. Using a larger batch size allows for more parallel computation because the training example can be split between different worker nodes. This, in turn, can significantly speed up model training [11]. For further experimentation, batch size and epoch are added to the parameters to find the optimal learning rate under different batch sizes.

By monitoring the changes and accuracy of loss during training, the change of large-batch training performance is determined. When the batch size is 128 and 256, the change in accuracy rate with learning rate is shown in Tables 11 and 12.

**Table 11.** Accuracy as a function of learning rate (batch_128, epoch_60)

| Learning Rate | 0.001 | 0.002 | 0.004 | 0.008 |
|---|---|---|---|---|
| Accuracy | 63% | 62% | 61% | 57% |

**Table 12.** Accuracy as a function of learning rate (batch_256, epoch_60)

| Learning Rate | 0.001 | 0.002 | 0.004 | 0.008 |
|---|---|---|---|---|
| Accuracy | 58% | 62% | 64% | 60% |

From Table 11 and Table 12, the optimal learning rate is about 0.001 when the batch size is 128, and the optimal learning rate is about 0.004 when the batch size is 256, which indicates that the learning rate should be increased at the same time to achieve the optimal accuracy rate when increasing the batch size.

When the batch size is 128 and 256, the downward trend of loss at different learning rates is shown in Fig. 7. However, loss does not fully reflect the accuracy due to overfitting of the learning of the training set. Therefore, it is necessary to further look at the training results with minimum loss and accuracy.

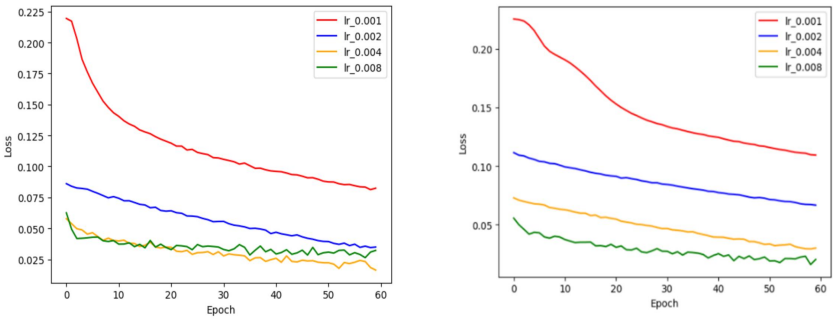


**Fig. 7.** The downward trend of loss at different learning rates at 128 (left) and 256 (right) for batch size (Photo/Picture credit: Original).

Table 13 shows where loss converges at 60 epochs under different conditions. The numeric

value of loss represents the distance between the model output and the real result. As can be seen from the data in Table 13, the loss has converged very closely in the above four cases. The trend can be seen more intuitively from Fig. 8.

**Table 13.** Minimum loss for different batch sizes and learning rate.

| Batch, lr | 32,0.001 | 64,0.002 | 128,0.004 | 256,0.008 |
|---|---|---|---|---|
| Min_loss | 0.0285 | 0.0317 | 0.0311 | 0.0321 |

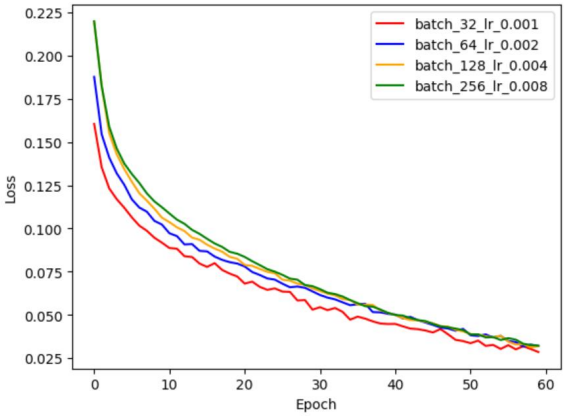


**Fig. 8.** Declining trend of loss under different batch sizes and learning rates (Photo/Picture credit: Original).

In fact, as shown in Table 14, adjusting the learning rate does eliminate most of the performance gap between small and large batches. As epoch increases, the value of loss gets closer and closer. The accuracy of large batch size training is also similar to that of small batch size.

**Table 14.** Accuracy of different batch sizes and learning rate

| Batch,lr | 32,0.001 | 64,0.002 | 128,0.004 | 256,0.008 |
|---|---|---|---|---|
| Accuracy | 60% | 61% | 61% | 60% |

When comparing large batches and small batches, generally default to the same epoch to compare, so that the number of large batch training iterations will be less. At this time, if the learning rate does not make any adjustments, the large batch trains fewer iterations, resulting in a lower degree of fit, and the accuracy will be low. So, the learning rate needs to be adjusted, a large batch contains more samples, avoiding the situation that small batches contain extreme samples, and the variance is smaller. This means that the gradient direction calculated using a large batch is more credible, so a larger learning rate can be used.

## 4 Conclusion

This research paper investigates the influence of batch size on GPU utilization during the process of image classification using LeNet, VGG, and ResNet architectures. Additionally, the optimal learning rate required for each architecture at different batch sizes was examined. Multiple parallelism mechanisms were implemented, and their performances were evaluated. The impact of device numbers on the efficiency of data parallelism was investigated. Through the experimental exploration, a significant decline in GPU utilization is observed when increasing the batch size of the LeNet model, irrespective of the employed parallel or non-parallel mechanism. The objective of the experiments is to provide valuable insights into the effects of batch size on GPU utilization efficiency and determine the optimal learning rate for achieving improved accuracy in image classification using state-of-the-art CNNs. Furthermore, a comprehensive comparison of the characteristics exhibited by various parallel approaches, including data parallelism and model parallelism is conducted.

## Acknowledgment

All the authors contributed equally and their names were listed in alphabetical order.

## References

1. LeCun, Y., Bottou, L., Bengio, Y., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324 (1998).
2. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
3. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778 (2016).
4. Hillis, W. D., Steele, J. G. L.: Data parallel algorithms. Communications of the ACM, 29(12): 1170-1183 (1986).
5. Gaunt, A. L., Johnson, M. A., Riechert, M., et al.: AMPNet: Asynchronous model-parallel training for dynamic neural networks. arXiv preprint arXiv:1705.09786 (2017).
6. Smith, L. N., Topin, N.: Super-convergence: Very fast training of neural networks using large learning rates. Artificial intelligence and machine learning for multi-domain operations applications. SPIE, 11006: 369-386 (2019).
7. Goyal, P., Dollár, P., Girshick, R., et al.: Accurate, large minibatch sgd: Training imagenet in 1 hour. arXiv preprint arXiv:1706.02677 (2017).
8. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. (2009).
9. Keskar, N., et al.: On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. arXiv preprint arXiv:1609.04836 (2016).
10. Yu, Q., Wang, J., Jin, Z., et al.: Pose-guided matching based on deep learning for assessing quality of action on rehabilitation training. Biomedical Signal Processing and Control, 72: 103323 (2022).

**Open Access** This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (http://creativecommons.org/licenses/by-nc/4.0/), which permits any noncommercial use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.