COMPSCI 273A Machine Learning Final Project Report
Group Name: NoIdea_Fashion-CIFAR10

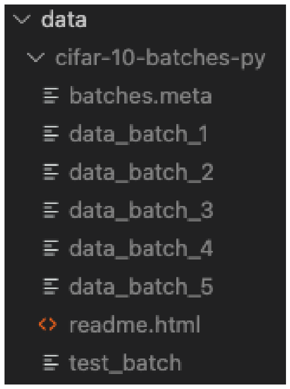
Ping-Xiang Chen [*1] **Dongjoo Seo** [*1] **Keonho Lee** [*1]

Abstract

This is the final project report of COMPSCI 273A. CIFAR-10 is the dataset in this project. We use PyTorch to implement various known deep neural networks for image classification. The source code of this project has been published on GitHub (Chen et al., 2022). The rest of this report is organized as follows. Section 1 presents the data exploration process, and Section 2 details the implementation and architecture of various deep neural network. Section 3 describes the process of data preprocessing and feature engineering. Section 4 evaluates the performance and compare implemented deep neural networks with CIFAR-10 dataset. Section 5 concludes this report.

1. Data Exploration

As shown in Figure 1, when we downloaded CIFAR-10 dataset, there are 5 data batches acting as the training set and 1 test batch can be used as testset. We further calculate the number of total images and classes in these batches, we found out that the CIFAR-10 dataset consists of 60000 colour images with size 32×32 in 10 classes, with 6000 images per class. These 10 classes in CIFAR-10 are: plane, car, bird, cat, deer, dog, frog, horse, ship and truck. Moreover, there are 50000 training images and 10000 test images. The sample visualization images and labels are shown in Figure 2. CIFAR-10 dataset is already included in PyTorch vision package. We can download the dataset by using `torchvision.datasets.CIFAR10`.



```
data
  cifar-10-batches-py
    batches.meta
    data_batch_1
    data_batch_2
    data_batch_3
    data_batch_4
    data_batch_5
    readme.html
    test_batch
```

Figure 1. Downloaded CIFAR10 dataset

Figure 2. Visualization results on CIFAR10 dataset

2. Model Exploration

Deep convolutional neural networks have already led to significant breakthroughs for image classification tasks. As a result, we decide to start this project by building simple convolutional neural networks (CNN) (PyTorch, 2021). The architecture of basic CNN is shown in Figure 3.

After building our first CNN using PyTorch, we found that the image classification accuracy of the trained CNN is just 56%, which is close to random guessing. We must do more to improve the performance of the baseline model. We can blindly add more layers or exploring all possible hyperparameters setting for this baseline convolutional neural network. However, such approach is inefficient and cumbersome. After carefully discussion, we decide to first refer to existing known CNN, and slightly adjust existing model's hyperparameters and architecture that could achieve high classification accuracy on CIFAR-10 dataset. The models that we use as reference in this final project are: AlexNet (Krizhevsky et al., 2012), VGG (Simonyan & Zisserman, 2014), GoogleNet (Szegedy et al., 2015). The adjusted architecture of AlexNet, VGG are shown in Figure 4, 5. Since

[*]Equal contribution [1]Department of Computer Science, University of California, Irvine. Correspondence to: Ping-Xiang Chen <p.x.chen@uci.edu>, Dongjoo Seo <dseo3@uci.edu>, Keonho Lee <keonhol1@uci.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

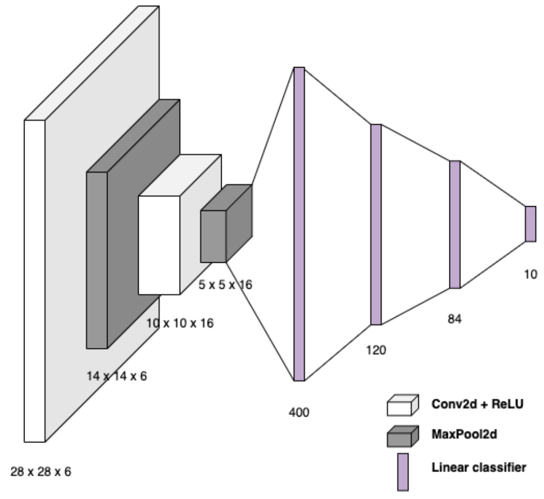


Figure 3. Model architecture of basic CNN

GoogleNet is much deeper than other CNNs, due to limited amount of pages, we do not show its architecture, please refer to the source code for detail implementation. After using existing CNN as backbone, we will have higher chance to obtain well-performed image classifier. We will evaluate the performance of each models in Section 4.

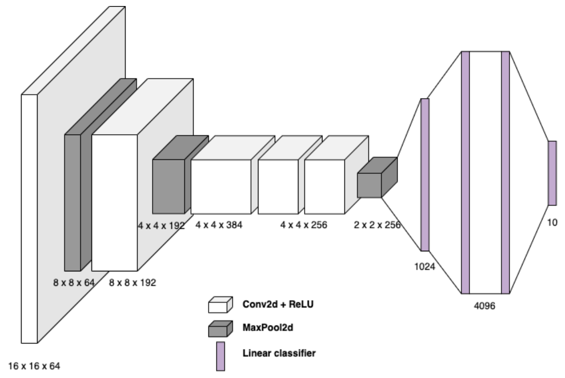


Figure 4. Model architecture of adjusted AlexNet

3. Data Preprocessing and Feature Design

Each image in CIFAR-10 dataset is 3-channeled which is typically used to represent RGB images. For the data preprocessing, we first load the CIFAR-10 testset and normalize each RGB values for of each images in the testset by using `transforms.Normalize` in PyTorch.

In order to prevent the deep neural network from overfitting, we use `torch.utils.data.random_split` to further split the dataset in to one testset and validation set with 70-30 split. Furthermore, we set `torch.Generator().manual_seed(0)` to enable fix split, which can allow us to do fair comparison between different CNN under fix experimental condition.

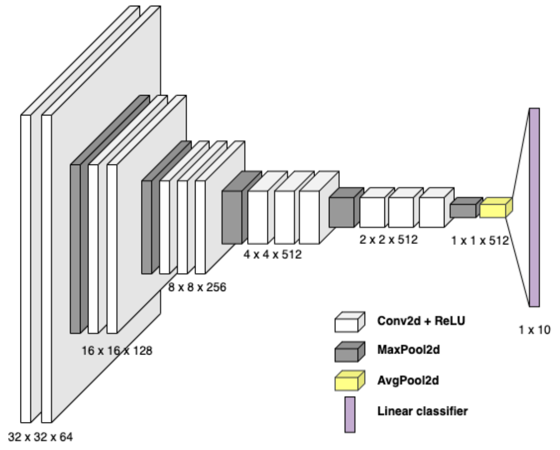


Figure 5. Model architecture of adjusted VGG

Now, the test data can be feed into different CNN models for training. Please note that during training, we do not use any data from the testset. We keep track the history of training and save the model which has the highest validation accuracy to do the classification. Then, we will evaluate the saved model with the testset which is also downloaded using `torchvision.datasets.CIFAR10`.

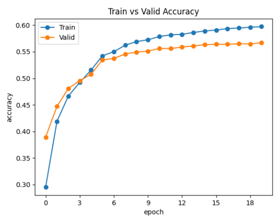
4. Performance Validation

Table 1 summarizes the training setting for all CNN models implemented by ourselves. In order to shorten the training time, we batch the dataset with size 200. Furthermore, we use learning rate scheduler to decrease the learning rate when the training phase pass certain epochs. This learning rate scheduler can be implemented by using `optim.lr_scheduler.MultiStepLR` in PyTorch. We set training milestone's epoch at 5, 7, 10, 15, 18. At each milestones, the learning rate will times with 0.5. All CNN models are trained with same setting for fair performance comparison.

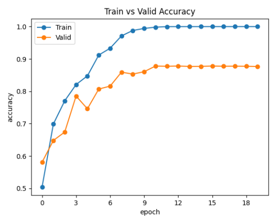
Table 1. Training Setting

Parameters Name	Setting
Epoch	20
Batch Size	200
Optimizer	Adam
Initial Learning rate	0.001
Loss Function	Cross Entropy
Activation Function	Relu

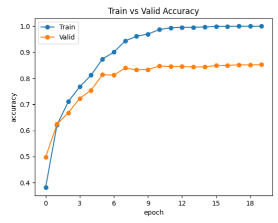
Figure 6, 7 shows training accuracy, validation accuracy, training loss, validation loss for each epoch during train-



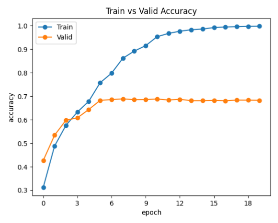
(a) Basic CNN



(b) GoogleNet

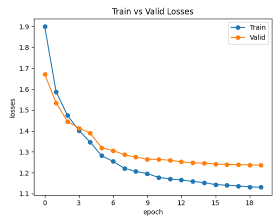


(c) VGG

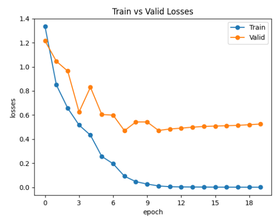


(d) AlexNet

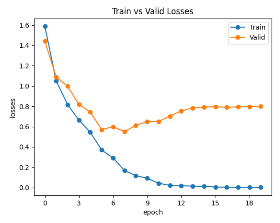
Figure 6. Training and validation accuracy of each implemented CNN models.



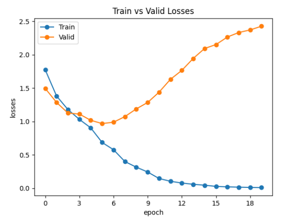
(a) Basic CNN



(b) GoogleNet



(c) VGG



(d) AlexNet

Figure 7. Training and validation loss of each implemented CNN models.

ing. We could see that in the beginning of training, all the models started to increase both training and validation accuracy. However, as shown in Figure 6(d) and 7(d), our implemented Alexnet start to suffer serious overfitting to the dataset with high training accuracy (low validation loss), low validation accuracy (high validation loss). Similarly, all models also have slightly overfitting issues roughly after 14 training epochs. From Figure 6, we could know that our implemented GoogleNet has the highest validation accuracy, which could implies that GoogleNet might have a highly chance to perform the best in classification accuracy on testset among other CNNs. The testset evaluation results are shown in Table 2.

Table 2. Experimental Results

	Accuracy	Inference Time (s)
GoogleNet	**87%**	7.657
VGG	84%	1.246
AlexNet	68%	**0.854**
Basic CNN	56%	2.236

As shown in Table 2, GoogleNet has the higher classification accuracy. VGG is the second best. By further investigating the architecture of different CNN model, we can find that the more complex the model is, the higher accuracy it can achieves. However, such complicate architecture also have drawbacks in terms of inference time of the model. As shown in Table 2, it takes almost 7 times time for GoogleNet to do the image classification than others.

5. Conclusion

In order to achieve higher classification performance on CIFAR-10, we referenced, adjusted and implemented existing CNN models (Chen et al., 2022). We can achieve up to 87% classification accuracy by using GoogleNet as our CNN backbone. We also found out that such high accuracy mainly comes from complex model nature.

References

Chen, P. X., Seo, D., and Lee, K. Final project. `https://github.com/pingxiang-chen/cifar10-uci-cs273a-final-project`, 2022.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

PyTorch. Deep learning with pytorch: A 60 minute blitz. `https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html`, 2021.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.