DATA130011 PJ2

Yuquan Zhou 21307100050

June 1, 2025

1 Introduction

This project conducts image classification on the CIFAR-10 dataset. The project implements ResNet and DenseNet using PyTorch, and investigates the effect of model size, regularization, activation functions, and optimizers. The project also investigates the impact of batch normalization on the smoothness of the loss landscape.

Code and model weights are available.

2 Train a Network on CIFAR-10

2.1 Model Architecture

ResNet and DenseNet are adopted. Both meet all the project requirements(FC layer, 2D-conv layer, 2D-pooling layer, Activations; BN layer, Dropout, Residual, Connection).

2.1.1 ResNet

ResNet-18 and ResNet-50 in the ResNet family are used. Model architecture follows the settings of He et al. [1](see table 1).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	32×32	7×7, 64, stride 2				
conv2_x	32×32	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	16×16	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	8×8	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	4×4	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 10-d fc, softmax				

Table 1: Architectures of ResNet for CIFAR-10. Building blocks are shown in brackets, with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2. **Dropout is applied at FC layer.**

2.1.2 DenseNet

DenseNet-201 and DenseNet-264 in the DenseNet family are used. Model architecture follows the settings of Huang et al. [2](see table 2).

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	32×32	7×7 conv, stride 2			
Pooling	32×32	3×3 max pool, stride 2			
Dense Block (1)	32×32	[1×1 conv; 3×3 conv] ×6	[1×1 conv; 3×3 conv] ×6	[1×1 conv; 3×3 conv] ×6	[1×1 conv; 3×3 conv] ×6
Transition Layer (1)	32×32	1×1 conv			
	16×16	2×2 average pool, stride 2			
Dense Block (2)	16×16	[1×1 conv; 3×3 conv] ×12	[1×1 conv; 3×3 conv] ×12	[1×1 conv; 3×3 conv] ×12	[1×1 conv; 3×3 conv] ×12
Transition Layer (2)	16×16	1×1 conv			
	8×8	2×2 average pool, stride 2			
Dense Block (3)	8×8	[1×1 conv; 3×3 conv] ×24	[1×1 conv; 3×3 conv] ×32	[1×1 conv; 3×3 conv] ×48	[1×1 conv; 3×3 conv] ×64
Transition Layer (3)	8×8	1×1 conv			
	4×4	2×2 average pool, stride 2			
Dense Block (4)	4×4	[1×1 conv; 3×3 conv] ×16	[1×1 conv; 3×3 conv] ×32	[1×1 conv; 3×3 conv] ×32	[1×1 conv; 3×3 conv] ×48
Classification Layer	1×1	4×4 global average pool			
		10-d fully-connected, softmax			

Table 2: DenseNet architectures for CIFAR-10. The growth rate for all networks is $k = 24$. Each "conv" layer corresponds to the sequence BN-ReLU-Conv. **Each dense layer adopts a dropout rate of 0.2.**

2.2 Experimental Settings & Results

Experiments are conducted to explore the following factors:

1. Number of neurons/filters (model width and depth);
2. Different loss functions with different regularization (L1 & L2);
3. Different activations;
4. Different optimizers.

The experimental settings and corresponding results are shown in table 3. All experiments adopt Cosine Annealing LR scheduler.

	settings										Experimental Result			
	Model	Size	Activation	Dropout	Regularization	Optimizer	LR	Batch Size	Epochs	Augment	Train. Loss	Val. Loss	Train. Acc.	Val. Acc.
1	ResNet-18	11.17M	ReLU	0.2	None	Adam	1e-3	256	50	weak	0.0099	0.3709	99.69%	92.64%
2	ResNet-18	11.17M	GELU	0.2	None	Adam	1e-3	256	50	weak	0.0072	0.3742	99.79%	93.35%
3	ResNet-50	23.52M	ReLU	0.2	None	Adam	1e-3	256	50	weak	0.0125	0.3399	99.64%	93.12%
4	ResNet-50	23.52M	GELU	0.2	None	Adam	1e-3	256	50	weak	0.0112	0.3524	99.66%	92.76%
5	DenseNet-121	3.97M	ReLU	0.2	None	Adam	1e-3	256	50	weak	0.0214	0.2950	99.40%	93.32%
6	DenseNet-201	10.29M	ReLU	0.2	None	Adam	1e-3	256	200	weak	**0.0063***	0.3524	**99.82%***	94.25%
7	DenseNet-264	17.40M	ReLU	0.2	None	Adam	1e-3	256	200	weak	0.0066	0.3494	99.81%	94.27%
8	DenseNet-201	10.29M	ReLU	0.2	None	Adam	1e-3	256	200	strong	0.0720	0.2340	97.50%	94.99%
9	DenseNet-201	10.29M	GELU	0.2	None	Adam	1e-3	256	200	strong	0.0847	0.2378	97.09%	**95.03%***
10	DenseNet-201	10.29M	ReLU	0.2	None	SGD	1e-3	256	200	strong	1.6442	1.6688	38.93%	37.87%
11	DenseNet-201	10.29M	ReLU	0.2	L1, 1e-4	Adam	1e-3	256	200	strong	0.5837	0.2882	88.10%	91.05%
12	DenseNet-201	10.29M	ReLU	0.2	L2, 1e-4	Adam	1e-3	256	200	strong	0.1473	**0.2221***	97.31%	94.93%

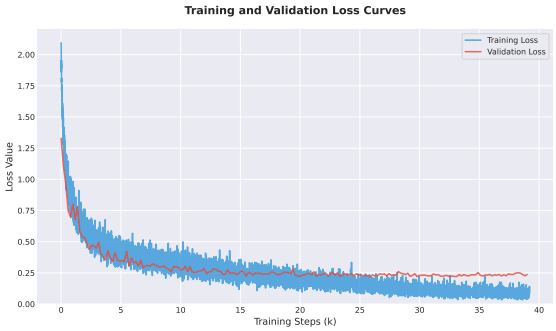
Table 3: Experimental settings and corresponding results. For Augmentation, "week" is with random crop and random horizontal flip, while "strong" further includes color jitter, random rotation, random affine and random earasing. Val Acc. is selected to be the highest value among all epochs instead of the last one epoch.

Among all tested settings, the best model is DenseNet-201 with GeLU and Strong augmentation(configuration 9). The training history (loss curves and accuracy curves) is visualized in fig. 1.

2.3 Results Analysis

By comparing the results of different configurations, we have the following conclusions (numbers in brackets refer to the configurations in table 3):

1. ResNet achieves favorable accuracy. (1,2,3,4)



Loss-DenseNet201

Accuracy-DenseNet-201

Figure 1: Training history of DenseNet-201(configuration 9).

2. Increasing the depth of the models can lower the loss and increase the accuracy(1-3, 2-4). However, with model architecture unchanged, the effect becomes marginal with increasing depth(6-7), and leads to risks of over-fitting.

3. Models with better architecture can achieve better or comparable performance with less parameters. (3-5)

4. Extensive augmentation increases training loss, but mitigates over-fitting, finally leading to a higher accuracy rate. (6-8)

5. Activation functions has an influence on the models' performance, although it is not as critical as model architecture(1-2,3-4). Also, more complicate activation function does not guarantee better performance(3-4).

6. Inappropriate regularization sometimes hurts the model. (8-9)

In addition, it is interesting that DenseNet achieves a similar score with ResNet with 1/3 to 1/6 parameters. It can be that ResNet has only local shortcut connections, and convolution is only done on neighboring layers. Therefore, ResNets need large channel number for intermediate layers, composing most of its parameters. However, in DenseNet, layers are fully connected in blocks, thus the channel number of a single layer can be small, saving a significant amount of parameters.

2.4 Visualization of the Network

In this section, the network is visualized. To be specific, the weight kernels of the first convolution layer are plotted. Also, a random image is picked from the CIFAR-10 dataset, and its feature map after the first convolution layer is plotted(see fig. 2).

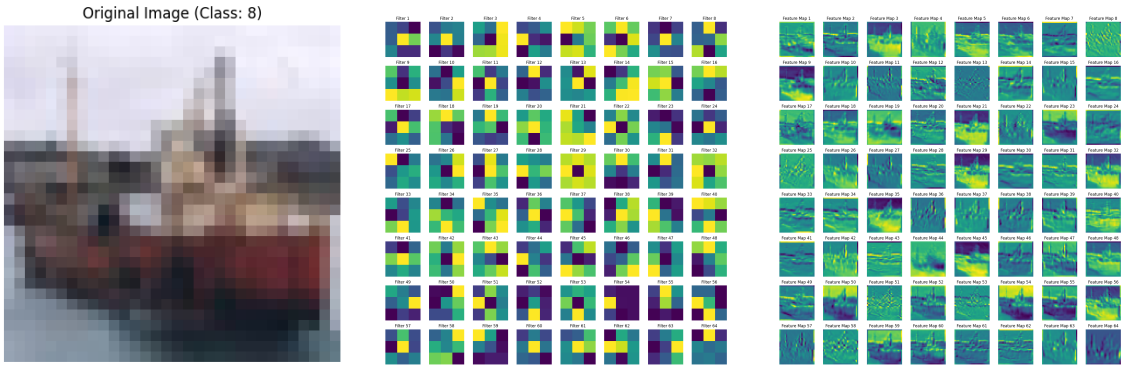
3 Batch Normalization

In this section, we explore the effect of batch normalization.

3.1 Performance Comparison

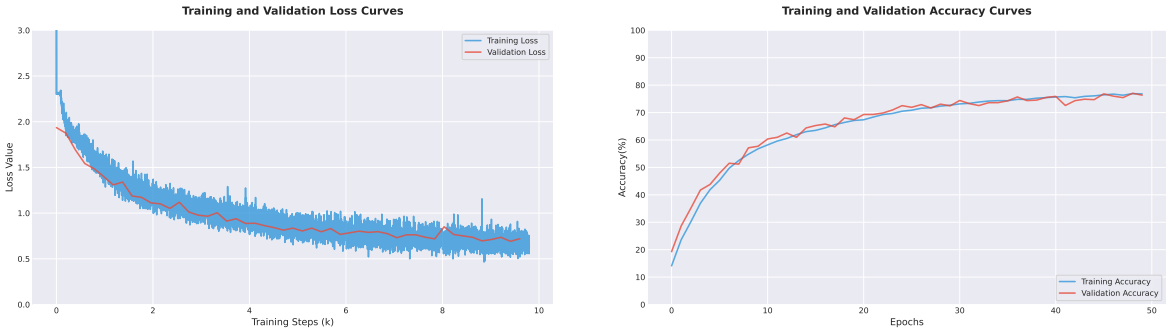
Firstly, batch normalization is added to the original VGG-A network. Networks with and without BN are trained and evaluated. The results are shown in table 4. We also plotted the training history(see fig. 3 and fig. 4).

To investigate the effect of batch normalization on the smoothness of the loss landscape, we adopt different learning rates(3e-3, 1e-3, 5e-4, 1e-4) while training the network with and without batch normalization, and plot the loss curves. If the BN layer can smooth the loss landscape as expected, the training should be less sensitive to the learning rate, narrowing the gap of loss value between the best and worst learning rate.The results in fig. 5 validate the assumption.



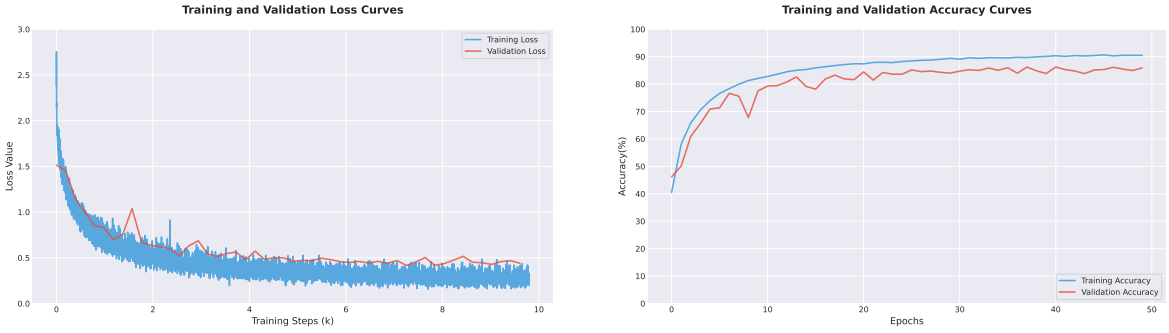
Original image Convolution filters Feature maps

Figure 2: Visualization of first layer convolution filters and there effects. Note that kernels are taken average across channels, instead of showcasing them in original RGB color.



Loss-DenseNet201 Accuracy-DenseNet-201

Figure 3: Training history of VGG-A without batch normalization.



Loss-DenseNet201 Accuracy-DenseNet-201

Figure 4: Training history of VGG-A with batch normalization.

		settings									Experimental Result			
	Model	Size	Activation	Dropout	Regularization	Optimizer	LR	Batch Size	Epochs	Augment	Train. Loss	Val. Loss	Train. Acc.	Val. Acc.
1	VGG-A	9.76M	ReLU	0	None	Adam	3e-3	256	50	weak	0.6817	0.7201	76.80%	77.02%
2	VGG-A	9.76M	ReLU	0	None	Adam	1e-3	256	50	weak	0.2535	0.4730	91.36%	85.45%
3	VGG-A	9.76M	ReLU	0	None	Adam	5e-4	256	50	weak	0.1168	0.5243	96.05%	87.16%
4	VGG-A	9.76M	ReLU	0	None	Adam	1e-4	256	50	weak	0.1537	0.5769	94.65%	84.17%
5	VGG-A(BN)	9.76M	ReLU	0	None	Adam	3e-3	256	50	weak	0.2759	0.4409	90.51%	86.24%
6	VGG-A(BN)	9.76M	ReLU	0	None	Adam	1e-3	256	50	weak	0.1308	0.4331	95.49%	88.65%
7	VGG-A(BN)	9.76M	ReLU	0	None	Adam	5e-4	256	50	weak	0.1038	0.4352	96.48%	88.33%
8	VGG-A(BN)	9.76M	ReLU	0	None	Adam	1e-4	256	50	weak	0.1227	0.5103	95.59%	86.23%

Table 4: VGG-A with and without batch normalization trained with different learning rates.

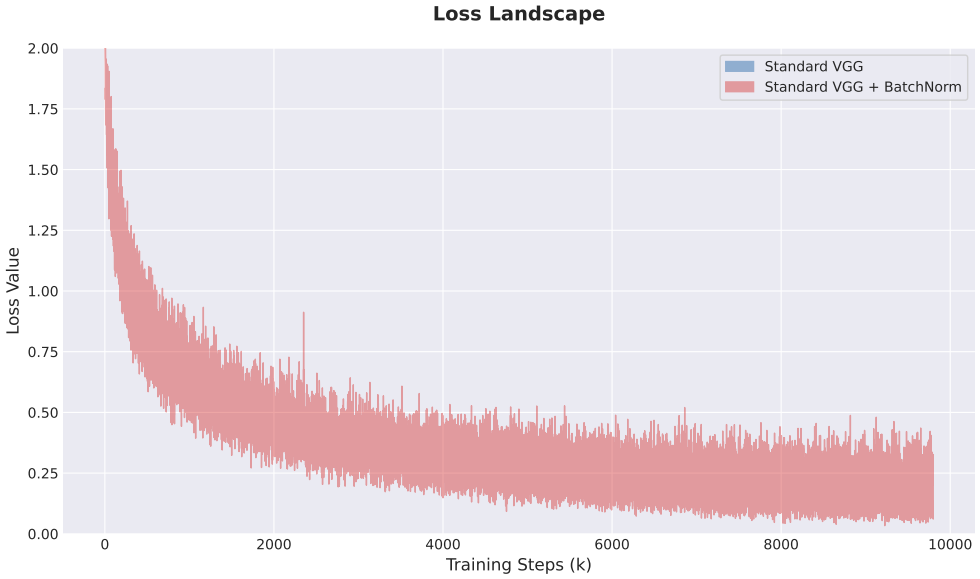


Figure 5: Loss landscape experiment. The colored area marks the gap between best and worst learning rate.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] 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*, pages 4700–4708, 2017.