Part 1:

1. The name under which you submitted on Kaggle.
   houhao

2. Best accuracy (should match your accuracy on Kaggle).
   0.54200

3. Table defining your final architecture similar to the image above.

	Layer No.	Layer Type	Kernel size	Input | Output dimension	Input | Output Channels
VGG - Layer1	1	conv2d	3	32 | 32	3 | 64
	2	Relu	-	32 | 32	-
	3	normalization	-	32 | 32	-
	4	conv2d	3	32 | 32	64 | 64
	5	Relu	-	32 | 32	-
	6	normalization	-	32 | 32	-
	7	maxpool2d	2	32 | 16	-
VGG - Layer2					
	8	conv2d	3	16 | 16	64 | 128
	9	Relu	-	16 | 16	-
	10	normalization	-	16 | 16	-
	11	conv2d	3	16 | 16	128 | 128
	12	relu	-	16 | 16	-
	13	normalization	-	16 | 16	-
	14	maxpool2d	2	16 | 8	-
VGG - Layer3					
	15	conv2d	3	8 |8	128 | 256
	16	Relu	-	8 | 8	-
	17	Normalization	-	8| 8	-
	18	conv2d	3	8| 8	256 | 256
	19	Relu	-	8 | 8	-
	20	Normalization	-	8 | 8	-
	21	conv2d	3	8| 8	256 | 256
	22	Relu	-	8 | 8	-
	23	normalization	-	8 | 8	-
	24	maxpool2d	2	8 | 4	-
VGG - Layer4					
	25	conv2d	3	4 | 4	256 | 512
	26	Relu	-	4 | 4	-
	27	Normalization	-	4 | 4	-
	28	conv2d	3	4 | 4	512 | 512
	29	Relu	-	4 | 4	-
	30	Normalization	-	4 | 4	-

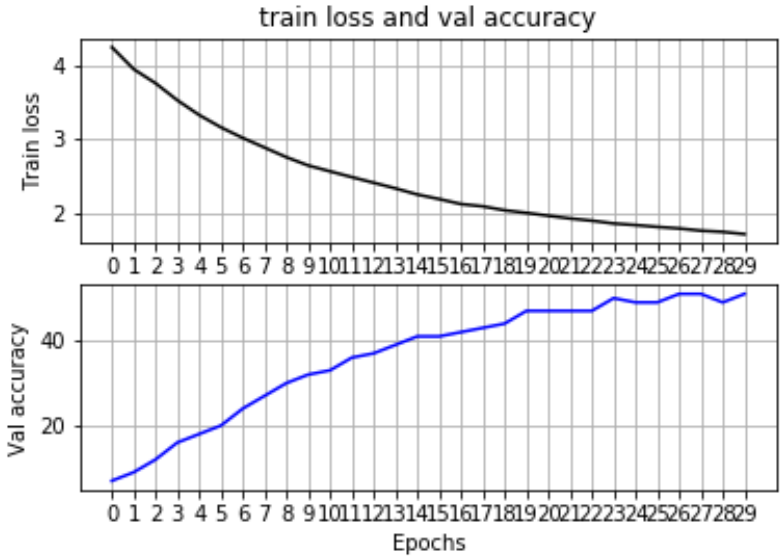
	31	conv2d	3	4 | 4	512 | 512
	32	Relu	-	4 | 4	-
	33	Normalization	-	4 | 4	-
	34	maxpool2d	2	4 | 2	-
VGG - Layer5					
	35	conv2d	3	2 | 2	512 | 512
	36	Relu	-	2 | 2	-
	37	Normalization	-	2 | 2	-
	38	conv2d	3	2 | 2	512 | 512
	39	Relu	-	2 | 2	-
	40	Normalization	-	2 | 2	-
	41	conv2d	3	2 | 2	512 | 512
	42	Relu	-	2 | 2	-
	43	Normalization	-	2 | 2	-
	44	maxpool2d	2	2 | 1	-
FC - Layer					
	45	dropout	-	512 | 512	-
	46	linear	-	512 | 256	-
	47	normalization	-	256 | 256	-
	48	relu	-	256 | 256	-
	49	dropout	-	256 | 256	-
	50	linear	-	256 | 128	-
	51	normalization	-	128 | 128	-
	52	relu	-	128 | 128	-
	53	linear	-	128 | 100	-

4. Factors which helped improve your model performance. Explain each factor in 2-3 lines.
   (a) Data normalization. I calculated the mean and std for this CIFAR dataset and fed them into transforms.Normalize(). After normalization, the range of distributions of feature values would be the same. If not, the learning process would cause corrections in each dimension. Sometimes, it would be over compensating while under compensating in another.
   (b) Data augmentation. I randomly crop my image with four padding step and then horizon flip the image. This step improves my model performance by artificially creating new training data from existing training data. It expands the training dataset with new examples.
   (c) By referring VGG-19 architecture, I followed CNN structure but modified the fully connected layers. The structure of my model is shown above. Adding more conv layers with increasing output channels and also adding more linear (fc) layers can make network deeper. Thus, it can learn features at various levels of abstraction.
   (d) Normalization and relu layer. One normalization and relu layer follow convolution layer. It helps reduce overfitting and improve training of the model.

(e) Parameters selections. The below table shows my parameter selections. After several trials, these selections can meet the requirements.

Parameters	
epoch	30
lr	0.01
momentum	0.9
weight_decay	5.00E-04

5. Final architecture's plot for training loss and validation accuracy.

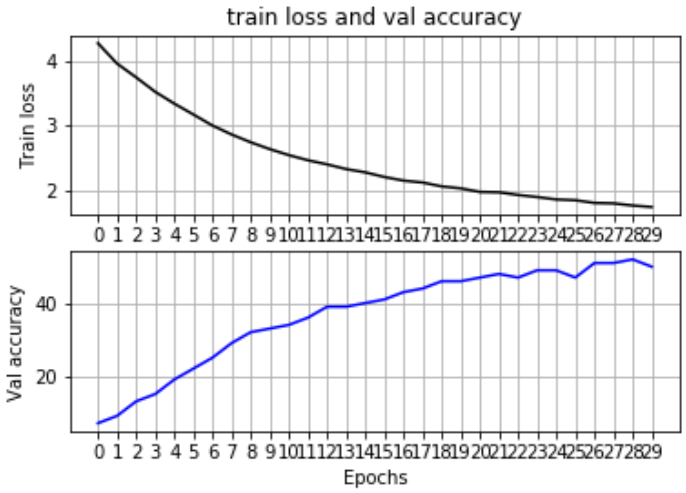


6. Ablation study to validate the above choices, i.e., a comparison of performance for two variants of a model, one with and one without a certain feature or implementation choice
(1) Data normalization.

With normalization: The test accuracy is 0.54200

Without normalization: The test accuracy is 0.53300

The explanation can be found in question4 (1)

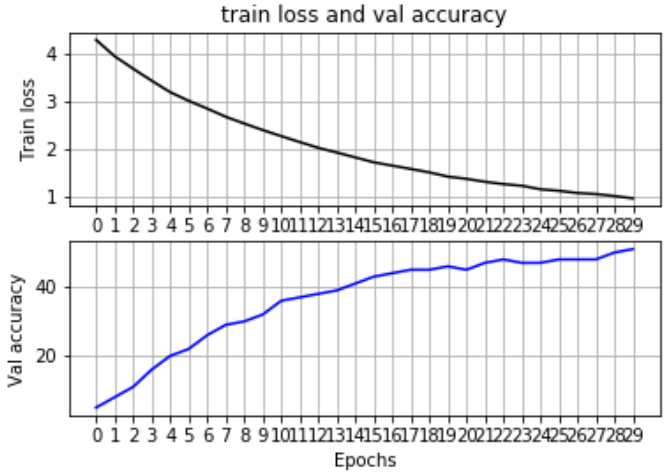


(2) Data augmentation.

With augmentation: The test accuracy is 0.54200

Without augmentation: The test accuracy is 0.50500

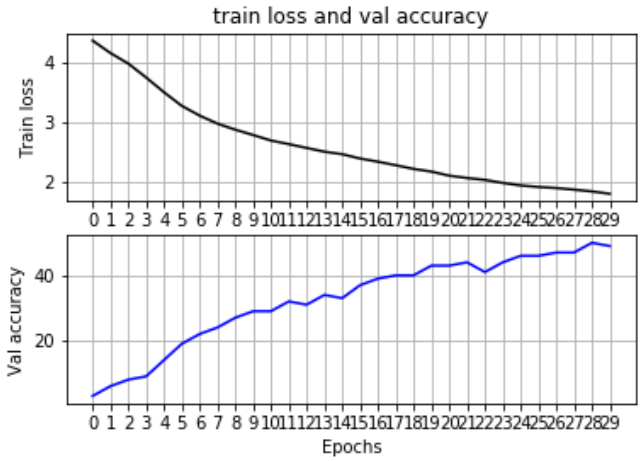
The explanation can be found in question4 (2)



(3) Normalization layer after relu.
After: The test accuracy is 0.54200

Before: The test accuracy is 0.52800

In convolution structure, it can generate better results when setting relu before normalization. The probably reason would be the input of next layer can be controlled by normalization layer instead of activation layer.

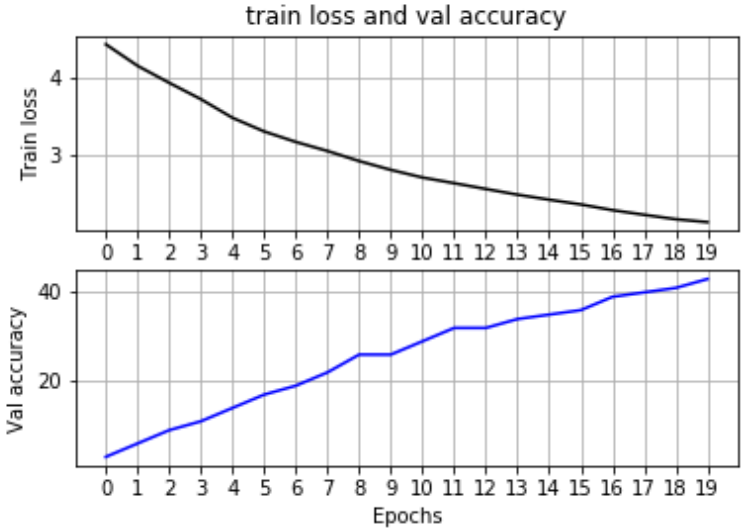


(4) Parameters (i.e. epoch number).

30 epoch: The test accuracy is 0.54200

20 epoch: The test accuracy is 0.43900

The model has not learned well with less epoch. But, by using more epoch, it would cause overfitting.

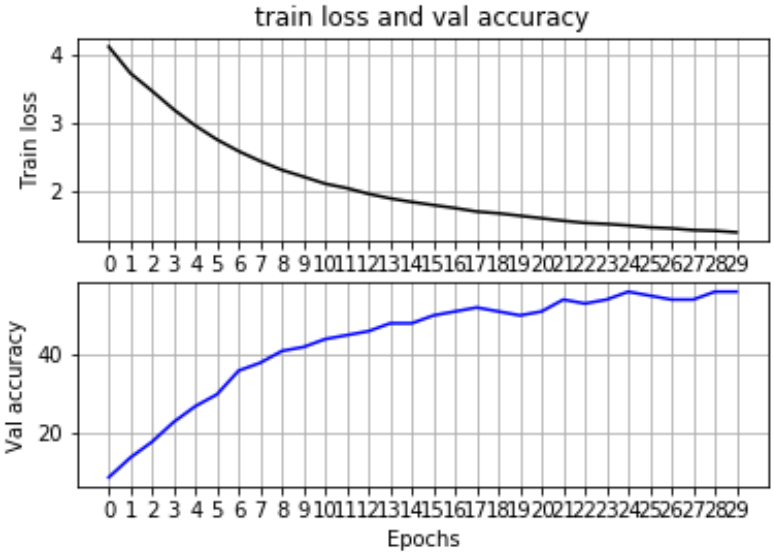


(5) Drop-out later

With drop out later: The test accuracy is 0.54200

Without: The test accuracy is 0.56699

Drop-out would avoid overfitting. But my model has very complicated structure. By using drop-out, it would increase my training time.


train loss and val accuracy
Train loss
4
3
2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Val accuracy
40
20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
Epochs

Part 2:

1. Report the train and test accuracy achieved by using the ResNet as a fixed feature extractor vs. fine-tuning the whole network.

Fine-tuning (Whole network):

Train loss: 0.0906, Train accuracy: 0.8670;

Test loss: 0.2559 Test accuracy: 0.5041

Fixed feature (Last layer):

Train loss: 0.2464 Train accuracy: 0.6227

Test loss: 0.3449 Test accuracy: 0.3615

	Training loss	Training accuracy	Test loss	Test accuracy
Fine-tuning	0.0906	0.8670	0.2559	0.5041
Fixed feature	0.2464	0.6227	0.3449	0.3615

2. Report any hyperparameter settings you used (batch_size, learning_rate, resnet_last_only, num_epochs)

Batch_size: 8

Learning_rate: 0.0005

Resnet_last_only: False

Num_epochs: 50

Weight_decay: 5e-4

Random resize crop : 256, scale = (0.08, 0.7), ratio = (0.75, 1)

.