Name: ZHANG XINJIE **Student ID Number: 20M14457**
E-mail: zhang.x.az@m.titech.ac.jp

Final Report for Image Recognition

Outline of the Train1000 dataset

The train1000 dataset contains 1000 samples from CIFAR10 dataset, one of the most widely used datasets for machine learning and deep learning research. The original CIFAR10 dataset contains 60,000 32×32 RGB images in 10 different classes. The 10 different classes represent airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships, and trucks. There are 6,000 images of each class. The train1000 dataset extract 100 images from the CIFAR10 dataset for each corresponding class. The samples from CIFAR10 is shown in Fig.1.

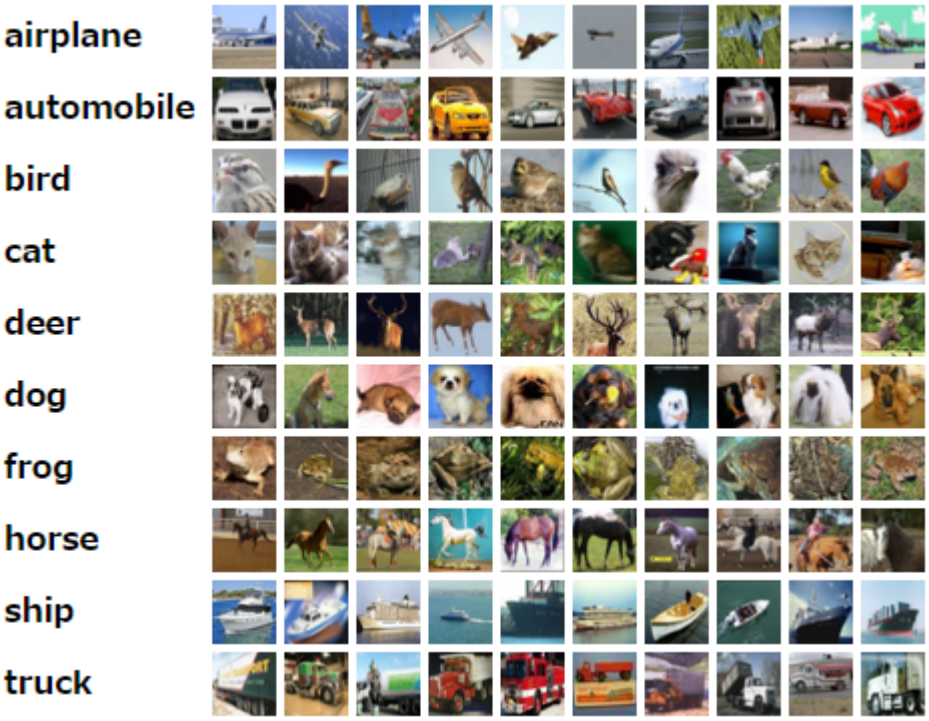

Fig.1 Samples from CIFAR10 dataset

Network architectures and the reasons

The architectures of convolutional neural networks are shown in the Table.1.

Table.1 Architectures of CNNs

Architecture	Architecture	Architecture	Architecture	Architecture
Input (32, 32,3)	Input (32,32,3)	Input (32, 32, 3)	Input (32, 32, 3) (normalization: “rescale-zero-one”)	Input (32, 32, 3) (normalization: “zerocenter ”)

conv2d (3,32)	conv2d (3,32)	conv2d (3, 32)	conv2d (3, 32)	conv2d (3, 32)
Activation	Activation	BN layer	BN layer	BN layer
maxpool (2,2)	maxpool (2,2)	Activation	Activation	Activation
conv2d (3,32)	conv2d (3,32)	maxpool (2,2)	maxpool (2,2)	maxpool (2,2)
Activation	Activation	conv2d (3, 32)	dropout (0.3)	dropout (0.3)
maxpool (2,2)	maxpool (2,2)	BN layer	conv2d (3, 32)	conv2d (3, 32)
conv2d (3,64)	conv2d (3,64)	Activation	BN layer	BN layer
Activation	Activation	maxpool (2,2)	Activation	Activation
maxpool (2,2)	maxpool (2,2)	conv2d (3, 64)	maxpool (2,2)	maxpool (2,2)
	conv2d (3,64)	BN layer	dropout (0.3)	dropout (0.3)
	Activation	Activation	conv2d (3, 64)	conv2d (3, 64)
	maxpool (2,2)	maxpool (2,)	BN layer	BN layer
		conv2d (3, 64)	Activation	Activation
		BN layer	maxpool (2,2)	maxpool (2,2)
		Activation	dropout (0.3)	dropout (0.3)
		maxpool (2,2)	conv2d (3, 64)	conv2d (3, 64)
		FC (128)	BN layer	BN layer
		Activation function	Activation	Activation
		FC (nb_classes)	maxpool (2,2)	maxpool (2,2)
		softmax	dropout (0.3)	dropout (0.3)
		classificationLayer	FC (128)	FC (128)
			Activation	Activation
			dropout (0.3)	dropout (0.3)
			FC (nb_classes)	FC (nb_classes)
			softmaxLayer	softmaxLayer
			classificationLayer	classificationLayer

The architecture of network 1 is the simplest. It only contains 2 convolutional layers with 3×3 filter and 32 output channels and 1 convolutional layer with 3×3 filter and 64 output channels, and each con layer following a max-pooling layer with 2×2 stride. Based on the architecture 1, the architecture 2 increases 1 convolutional layer with 3×3 filter and 64 output channels. The training accuracy is improved by 12% and the test accuracy increases about 2%. **The increase of convolutional layer can be considered efficient,** and this is proved with superior performance using well-known deep neural networks such as VGG16, VGG19 and ResNet.

However, considering the small scale of the dataset and poor computation environment with single CPU, very deep neural network is very difficult to implement. The baseline network contains two convolutional layers with 3×3 filter and 32 output channels and two convolutional layers with 3×3 filter and 64 output channels, each following a max-pooling layer with 2×2 stride. The configurations of convolutional layer and max-pooling layer is adopted because the configuration of filters, channels, and stride

has been proved the most efficient in many well-known classification tasks using deep learning neural networks.

Architecture 3 adds the batch normalization layer following each convolutional layer in comparison of architecture 1 & 2. Batch normalization is based on the assumption that mini-batch of training data should have same distribution like the overall dataset. When training convolutional neural network using large scale dataset, batch normalization can prevent the small changes in weights of one layer will not get propagated to other layers, thus enhancing the gradient propagation and bringing possibility to set higher learning rates for optimizers. **The implementation of batch normalization and increase of training epochs improves the performance with 4% in test accuracy.**

Architecture 4 adds dropout layers following each of max-pooling layer, with dropout rate 0.3. Dropout is one of the common regularization techniques, which was first proposedd in AlexNet. Dropout performs random invalid activations on neurons during the training to prevent overfitting. **The implementation of dropout layer and change of batch size significantly improves the performance with 4% in test accuracy.**

Architecture 5 only changes the way of normalization of the input images based on architecture 4, from [0,1] to [-1,1]. The performance is improved with 1% in test accuracy.

Hyperparameters and the reasons

The training hyperparameters are shown in Table.2.

Table.2 Different configurations of hyperparameters in training phase.

Architecture 1	Optimizer	SGD
	Initial learning rate	0.01
	Epochs	10
	Batch size	100
	Validation Frequency	10
Architecture 2 & 3	Optimizer	Adam
	Initial learning rate	0.01
	Learning rate drop rate	0.1
	Learning rate drop period	10
	Epochs	20
	Batch size	100
	Validation Frequency	10
Architecture 4 & 5	Optimizer	Adam
	Initial learning rate	0.01
	Learning rate drop rate	0.1
	Learning rate drop period	10

	Epochs	100
	Batch size	32
	ValidationFrequency	100

Three configurations of hyperparameters in training phase is implemented. The hyperparameters include the choice of optimizers, the initial learning rate, the learning rate schedule, the learning rate drop rate, the learning rate drop period, the number of training epochs, the number of mini-batch size, the number of validation frequency.

The optimizer in training network using architecture 1 is Stochastic Gradient Descent (SGD). SGD is based on Gradient Descent, which is an optimization algorithm to find the local minima for loss function, to train deep neural networks. The Adam optimization algorithm use adaptive estimates of low-order moments than momentum algorithm. **Adam is much more prevalent** because of it is computationally efficient and simple to implement; thus, the Adam is adopted in training networks using architecture 2, 3, 4, 5.

The learning rate is a hyper-parameter that controls the adjust of the weights of network during training, with respect of loss gradient. The learning rate scheduling is to adjust the learning rate during training by reducing the learning rate according to defined schedule, including time-based decay, step decay, and exponential decay. The step decay is adopted in training networks in all architectures, which is to drop the learning rate by every few epochs (defined by learning rate drop period) by timing learning rate drop rate. The default initial learning rate is as same as the initial learning rate given in VGG16 training strategy, and **it seems that drop rate of 0.1 and drop period of 10 is the best choice for convolutional neural network training in this case**, since I tried to change the drop rate to 0.2 and drop period to 5 but did not get very good performance.

One epoch is when an entire dataset is passed forward and backward through the neural network only once. For training deep neural network, it is not enough to update the weights in the network with the dataset passing only one time.

Batch size is the total number of training samples in single batch training and the number of iterations is the number of batches for completing one epoch. **The performance of networks in architecture 3 is significantly improved when the batch size changes from 100 to 32.** 32 is a commonly adopted parameter for mini-batch training.

Validation frequency is the frequency of network validation in number of iterations, which seems not affect the training accuracy.

Key techniques to improve the test accuracy

In summary of the explorations above, several techniques can be considered as key techniques of improving the test accuracy:

(1) Convolutional layers Convolutional layer is the key component to deepen the neural network. Kernel or filter is a small grid of parameters, being applied at each pixel position in an image and the output is fed into the next layer. The features of this image can be extracted hierarchically and progressively. Theoretically, the more convolutional layers, the more complex features, and the higher classification accuracy. But a very deep neural network may be very hard to train, optimize, and further to deepen.

The parameters of the kernel should be initialed manually as first steps. The commonly used filter parameters are 3×3, which is the smallest size to capture the notion of left/right, up/down, center). 1×1 convolution filter is also usually utilized because it can be considered as a linear transformation of the input channels.

(2) Learning rate scheduling and training epochs Constant learning rate may not good for long-term training of deep neural network. Learning rate scheduling is to adjust the learning rate during training by reducing the learning rate according to a pre-defined schedule. Common learning rate schedules include time-based decay, step decay, exponential decay. The commonly adopted schedules are step decay and exponential decay. In training deep neural network, the well-scheduled decay of learning rate can improve the performance and prevent the optimization to be trapped in local minima.

(3) Batch normalization Batch normalization can prevent the small changes in weights of one layer will not get propagated to other layers, thus enhancing the gradient propagation and bringing possibility to set higher learning rates for optimizers.

(4) Activation Function The exploration of effects of different activation functions is shown in Table.3.

Table.3 Performance of network architectures using different activation functions

Activation function	Architecture 3		Architecture 4		Architecture 5	
	Training accuracy	Test accuracy	Training accuracy	Test accuracy	Training accuracy	Test accuracy
ReLU	99.5%	47.3%	88.3%	49.09%	88.1%	**50.08%**
tanhLayer	99.4%	41.75%	96.8%	46.32%	97.5%	**48.07%**
leakyReluLayer	98.9%	45.23%	82.8%	**47.09%**	87.8%	46.83%
preluLayer (number of channels)	99.6%	43.73%	98%	51.46%	99.5%	**52.40%**
eluLayer	N/A	N/A	N/A	N/A	98.3%	**50.02%**

In the architecture 3, the training accuracy of networks using different activation functions achieves over 99%, which may be regarded as overfitting. In this case, the network with ReLU activation function performs best. In architecture 4 and 5, with dropout layer, the overfitting problem is improved significantly, and the network with PReLU activation performed best and improved the most (from 43.72% to 51.46%). However, it worth noting that the training accuracy of network with ReLU activation function has not achieved over 90%, which may imply there is still some improving space. The improvement of networks with leakyReLU is not very significant compared with other activation functions. All activation functions achieved their own best performance in configuration 3 except leakyReLU function.

(5) Regularization and Dropout Generalization capability of a deep neural network is always an important problem, because the deep neural networks are not only expected to perform well on particular datasets or tasks. It is quite common that a deep neural network is overfitting, which means the neural network learns the distribution of training data set to specifically to generalize on the validation data set. Two common regularization techniques are proposed and widely applied to prevent the overfitting problems: L2 regularization and dropout. L2 regularization is to apply a term of penalty to the loss function. Dropout performs random invalid activations on neurons during the training.

Evaluation results and discussions

The evaluation results of networks in each architecture are shown in Table.4. The best performance 52.40% is achieved by network using architecture 5 with preluLayer for all activation functions and training under hyperparameters in configuration 3.

Table.4 Best performance of different network architectures

Architecture	**Training accuracy**	**Testing accuracy**
1	72.2%	42.57%
2	84.1%	43.97%
3	99.5%	47.3%
4	98.0%	51.46%
5	99.5%	52.40%

Data augmentation As it is shown in the Table. 4, the training accuracy of the model with best performance has achieved approximately to 100%, which can be considered as overfitting. Overfitting is very common in training small scale dataset, and data augmentation can be considered as an effective method to overcome overfitting.

However, data augmentation does not improve the performance with simple application. The training accuracy is drastically lowered after applying data augmentation, but the test accuracy is lowered as well. Data augmentation may have significant contribution in very deep neural network, with much more convolutional layers.

Hyperparameters and architectures There are numerous parameters in training deep neural networks and they seem to be very important. In this case, we need to build a convolutional neural network model from scratch; thus, it may be better to start from building the architectures of the neural networks, and then tune the hyperparameters. If given a pretrained model, there is no much need to design the architecture from scratch. It may be quicker to get a result to fine-tune the hyperparameters based on the pre-trained model.

Ensemble Learning and transfer learning Due to the poor computation environment, I have not tried the ensemble learning. Ensemble learning has been mentioned as one of the key techniques to get good performance of models for classification tasks. This approach is to train multiple models instead of a single model and to combine the predictions from multiple models, to reduce the variance of neural network models. The most commonly used ensemble learning approach for neural networks is to calculate the average of the predictions made by a collection of networks with the same configuration and different initial random weights that is trained on the same dataset. The number of models in the ensemble is often kept small because of the computational expense in training models and diminishing returns in performance from adding more ensemble members.

Though it is prohibited to use transfer learning in this lecture, but it is a really great technique to train neural networks. The transfer learning is based on the assumption that the knowledge obtained from source domain for source learning task would be applicable to a target learning task based on target domain, i.e., the performance of a pre-trained model on target learning task can be improved by transferring latent knowledge from source task. The most commonly applied transfer learning technique is network-based method, which directly applies the partial of pre-trained network in source domain to the target network with models in similar network architecture. The transferred contents include network structure and weights in the hidden layers. Currently, pre-trained models using transfer learning technique is quite prevalent because its great performance in generalization and convenience to increase computation efficiency.