In recent years, deeper and wider neural networks have shown excellent performance in computer vision tasks, while their enormous amount of parameters results in increased computational cost and overfitting. Several methods have been proposed to compress the size of the networks without reducing network performance. We present a novel incremental training algorithm for deep neural networks called planting. Our planting can search the optimal network architecture with smaller number of parameters for improving the network performance by augmenting channels incrementally to layers of the initial networks while keeping the earlier trained parameters fixed. We propose using the knowledge distillation method for training the channels planted. By transferring the knowledge of deeper and wider networks, we can grow the networks effectively and efficiently.

We evaluate the effectiveness of the proposed method on different datasets such as CIFAR-10/100 and STL-10. For the STL-10 dataset, we show that we are able to achieve comparable performance with only 7% parameters compared to the larger network and reduce the overfitting caused by a small amount of the data.

Our planting approach consists of the following training processes (show in Fig.1)

(0) Training a large network as the teacher network.
(1) Training a small network with fewer channels of each layer by a standard classification training method.
(2) Adding channels to a layer on the small network by using a knowledge distillation method with the teacher network while keeping the earlier trained parameters fixed.
(3) Repeat (2) the number of layers times
(4) Selecting a planted network with the smallest validation loss as the next base network for planting
(5) Repeating (4) while reducing the classification loss than the previous network

We have performed experiments using CIFAR-10/100 and STL-10. The structure of networks are shown in Table.1.

### Table 1 The Structure of Networks

| Network for CIFAR-10/100 | Network for STL-10 |
|--------------------------|-------------------|
| ReLU(conv1(kernel=3))    | ReLU(conv1(kernel=3)) |
| max pooling(2x2)         | max pooling(2x2)   |
| ReLU(conv2(kernel=3))    | ReLU(conv2(kernel=3)) |
| max pooling(2x2)         | max pooling(2x2)   |
| ReLU(conv3(kernel=3))    | ReLU(conv3(kernel=3)) |
| max pooling(2x2)         | max pooling(2x2)   |
| ReLU(conv4(kernel=3))    | ReLU(conv4(kernel=3)) |
| max pooling(2x2)         | max pooling(2x2)   |
| ReLU(conv5(kernel=3))    | ReLU(conv5(kernel=3)) |
| max pooling(2x2)         | max pooling(2x2)   |
| ReLU(fc1)                | ReLU(fc1)          |
| output=fc2)              | output=fc2)        |

### Table 2 The Results on CIFAR-10

| Network | Params | Test Err. | Test Acc. | Loss func |
|---------|--------|-----------|-----------|-----------|
| Teacher | 128K   | 0.5007    | 88.10%    | CELoss    |
| Student | 128K   | 0.3823    | 88.51%    | KL-Loss   |
| Initial Network (Student[8]) | 20.4K | 0.8300 | 71.35% | CELoss |
| Student[16] | 43.9K | 0.6671 | 79.42% | CELoss |
| Student[32] | 104.8K | 0.4898 | 84.03% | CELoss |
| Student[64] | 282.0K | 0.4431 | 86.83% | CELoss |
| Ours     | 40.6K  | 0.4825    | 84.35%    | KL-Loss   |

### Table 3 The Results on CIFAR-100

| Network | Params | Test Err. | Test Acc. | Loss func |
|---------|--------|-----------|-----------|-----------|
| Teacher | 128K   | 0.5100    | 76.57%    | CELoss    |
| Student | 128K   | 1.6232    | 60.05%    | KL-Loss   |
| Student[8] | 32.0K | 2.5280 | 36.13% | CELoss |
| Initial Network (Student[16]) | 55.5K | 2.1370 | 45.45% | CELoss |
| Student[32] | 116.5K | 1.9022 | 52.15% | CELoss |
| Student[64] | 293.6K | 1.9510 | 55.74% | CELoss |
| Ours     | 78.5K  | 1.7584    | 53.31%    | KL-Loss   |

### Table 4 The Results on STL-10

| Network | Params | Test Err. | Test Acc. | Loss func |
|---------|--------|-----------|-----------|-----------|
| Teacher[64] | 445.8K | 1.1807 | 66.47% | CELoss |
| Initial Network (Student[8]) | 40.8K | 1.2776 | 55.55% | CELoss |
| Student[16] | 84.9K | 1.2924 | 59.34% | CELoss |
| Student[32] | 186.8K | 1.9998 | 61.10% | CELoss |
| Ours     | 82.6K  | 1.0772    | 67.12%    | KL-Loss   |

### Conclusion

We proposed a novel incremental training method called planting using knowledge transfer, that can train smaller network with excellent performance and find the optimal network architecture automatically. We confirmed that the proposed approach was able to achieve comparable performance with smaller parameters compare to the larger network and reduce the over-fitting.