Cross-block Dense Connection in Densely Connected Convolutional Networks

Boyu Jiang
Minerva Schools at KGI
boyu.jiang@minerva.kgi.edu

Abstract

Recent advancements in neural networks can be largely attributed to the new waves of network architecture characterized by shortcut connections that can overcome the vanishing gradient problem. These new architectures provide the ability to train deeper networks and fit more complex features. Among them, Dense Convolutional Network (DenseNet) proposes an architecture that connects each layer to all of its subsequent layers called dense connections. However, DenseNet inherits the blocking architecture commonly used in convolutional network architecture, and the dense connection only applies to layers within the same block. In this paper, I will perform an empirical study on the effect of adding cross-block dense connections to the DenseNet model and analyze the cost and benefit of adding such connections.

1. Background

This section provides a general background in the field of machine learning. Covering from the overall motivation for machine learning algorithms to the detailed methodology of the convolutional neural network used in the computer vision tasks. It serves as a bridge to readers with limited experiences in machine learning to quickly understand the fundamentals incorporated in this paper.

1.1. Machine Learning

Machine learning is a data-driven way of solving complex problems. In traditional programming, we define rules for machines to follow. This works well in situations like programing a calculator to do matrix multiplication or sending a push notification whenever your bank account balance drops below zero. However, in situations when the rule is difficult to design, like determining if there is a cat in a given image, we cannot use traditional programming anymore because we do not know what rule to put in. This is when machine learning becomes useful. Machine learning uses data along with answers we already know to come up with the rule. In the cat identification example, we collect a lot of different images with and without cats along with the correct labeling of if the image contains a cat, and use machine learning to find the rule on how to answer the question if there is a cat in the image. We can define machine learning as the process of having the machine to find the rule to get from data to result, in other words, the mapping between input and output.

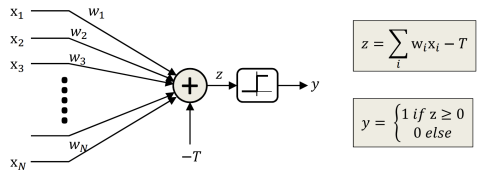

Figure 1. A threshold logic unit (TLU), it "fires" when the weighted sum of the input vector x and the bias factor T is positive. [8]

1.2. Threshold Logic Unit

Threshold Logic Unit (TLU) is the basic unit for machine learning. It took inspiration from neurons in the human brain and reproduced it using a mathematical model, so TLU is also commonly referred to as perceptron or artificial neuron. A basic TLU is a boolean function that takes a vector as input and returns a boolean output of 0 or 1. Formally, TLU takes the affine combination of the input vector then puts the result through a step function to obtain the output.1 In practice, a bias factor with value 1 is commonly added to the input vector so that the affine combination can be calculated as a linear combination, commonly known as the weighted sum. Then the threshold function is applied to the weighted sum, and if the weighted sum is greater than or equal to the threshold value, the TLU will return 1. Otherwise, if the weighted sum is less than the threshold value, the TLU will return 0. In a general TLU model, which is more commonly referred to as perceptron, the threshold step function is replaced by a "soft" activation function. Instead of having a "hard" cutoff point at the threshold value, a smoother version is used like the sigmoid function.

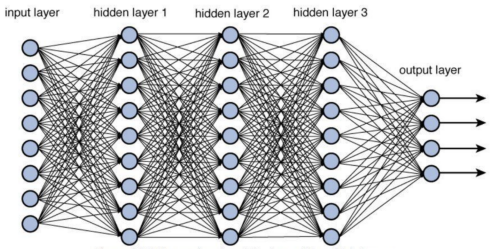

Figure 2. A multilayer perceptron with three hidden layer. [8]

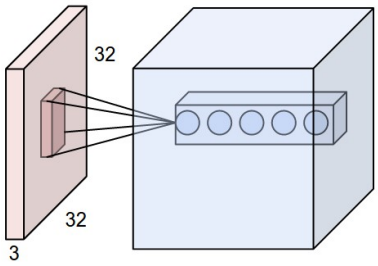

Figure 3. A 32x32x3 image input (red) connected to a convolutional layer with 5 neurons (blue). Each neuron in the convolutional layer is connected only to a local region in the input. [3]

1.3. Multilayer Perceptron

To make individual perceptrons into a useful machine learning model, the most straightforward way is to stack multiple perceptrons into layers, which is called a multilayer perceptron.2 This architecture came directly from the structure of the human brain, that a neuron sends information to its subsequent neurons and forms an aggregated effect known as human intelligence. In a multilayer perceptron, the first layer of perceptrons takes input directly from the input data, which is called the input layer. All the subsequent intermediate layers take the output of the previous layer as input and are called hidden layers. The final layer that maps the output of the last hidden layer to the model output is called the output layer. The total number of layers in a multilayer perceptron is called the depth of the multilayer perceptron. A multilayer perceptron is commonly referred to as a fully connected neural network in machine learning, and the perceptrons are referred to as neurons.

1.4. Universal Approximation Theorem

Universal approximation theorem [2] states that a fully-connected neural network with only one hidden layer and nonlinear activation function can approximate any continuous function given there are sufficient neurons in the hidden layer. This can be achieved because a perceptron can be used to produce the exact mapping between input and output at a given location in the data space without interfering with the output at other locations. So, assuming we have a single layer neural network with an astronomically large number of neurons, the single-layer network can approximate any continuous function by defining a mapping between input and output at all possible points. Single hidden-layer neural networks are indeed capable of approximating any function, however, the number of neurons required to fit a function grows exponentially with the number of statistically independent features. As a result, deeper networks can be more efficient than shallower networks to the point that the depth of the network exceeds the number of statistically independent features.

1.5. Neural Network

The term neural network is used in machine learning to describe a general kind of model that uses a network of functions to produce the mapping between input and output. For example, a multilayer perceptron is a type of neural network, where the functions are the perceptrons and the network structure is formed by the layer-by-layer connections between the perceptrons. This layer-by-layer network structure is commonly referred to as a sequential network and the structure of networks is called network architecture. There are a variety of options to form a neural network using different functions and connection architecture to build a neural network for a specific task, and the study of different ways to form a neural network is an important area in machine learning research.

1.6. Convolutional Neural Network

Convolutional neural networks [9] implement the concept of a filter. It can be thought of as applying a transformation on the input data. Convolution filter normally takes the shape of a 3-by-3 square, but it can be configured as a filter of any shape with any number of dimensions.3 It is commonly used in computer vision tasks because of its shift-invariance property. Assuming a task to identify which family a flower in a given image belongs to, a good model should produce the same output regardless of where the flower is located in the given image. In other words, moving the position of a flower from the top left of the input image to the bottom right of the input image should not change the model output. This property is called shift-invariance.4 In a simple multilayer perceptron, because perceptrons take the weighted sum of the input, the position of a data in the input vector will determine the weight it gets, making the model susceptible to shifts in position. In a convolutional neural network, convolutional filters shift around the entire input space. Information gathered by a convolutional filter is the relational data in the local space within the size of the convolutional filter. The spacial position of a data point in the input vector is not considered by convolutional filters,

Figure 4. The two image shows the same flower but on locations of the image. A shift-invariance model should identify the two flowers as the same. [8]

which guarantees shift-invariance. How convolution filters are used in the convolutional neural network is by defining multiple convolutional filters at the same time so that each filter will be identifying some aspects of information in the input data. By stacking convolutional filters on top of each other, the latter filters can aggregate the information in the previous filters and obtain higher-level understanding that can be used for good prediction. In practice, convolutional filters are often organized in layers with a convolutional layer taking the output of the previous convolutional layer as input. This aggregation of filters results in the ability to recognize more complex features. The convolutional layers closer to the input data tend to identify lower-level features like lines or edges, and the convolutional layers closer to the output tend to identify higher-level features like objects or parts. After all the convolutional layers, a fully connected layer is generally used to map the output features of the convolutional layers to prediction labels or other objectives.

2. Introduction

Convolutional Neural Network (CNN) has long been the standard method for image recognition tasks. The first version of CNN architecture LeNet-5 [9] contains only convolution 5 layers and was only able to process 32*32 pixel greyscale input images. It was until computational power became more accessible, we start to see larger networks like AlexNet [1] and VGGNet [7] that demonstrated the capability of CNN architecture. However, these networks contain no more than 20 convolution layers and it was impossible to grow the network deeper because of the vanishing gradient problem in the backpropagation process. In theory, more layers in a neural network should produce a more complex model which leads to better prediction. However, the gradient used to train a neural network decrease dramatically after each layer it passes through and after only a few layers, the gradient approaches 0, making it impossible to update the network. This will result in a poorly trained deep network that performs worth than a well trained shallow counterpart. In practice, we start to observe the vanishing gradient problem when the network is deeper than 5 layers.

This is where the idea of a shorter and more direct pass between the input and output layer emerges and the introduction of a skip or bypass connection became useful. In Residual Network (ResNet) [6], a special identity shortcut was defined to mitigate the vanishing gradient problem. Densely ConnectedConvolutional Networks (DenseNet) [5] develops another approach to the vanishing gradient problem. Instead of having a uniform identity shortcut for each layer, DenseNet proposes a dense connection between all layers. The benefit brought by this method, besides solving the vanishing gradient problem, dense connection means we can build a more complex model with smaller perimeter size and stronger supervision compared to the sequential model. DenseNet uses a blocking structure, which is a common network architecture for computer vision tasks. Within the same block, all layers have the same height and width dimension, making it easier to build and manage the network. This benefit can be observed in the DenseNet architecture because all layers in the same dense block have the same height and width, DenseNet was able to simply stack the old layers on top of the newly generated layers. It makes a simple approach, however, it also limits the amount of dense connection it is possible to make. CondenseNet [4] was introduced to take the concept of dense connection one step further, adding dense connections between the dense block. This network architecture saturated the dense connection in the network, making a layer in the network taking outputs from all of its previous layers. However, along with the improved accuracy, the added connection along with the complex architecture makes CondenseNet more difficult to train. There is also the concern that dense connections and feature refuse can add too much redundancy to the network, which might harm the performance.

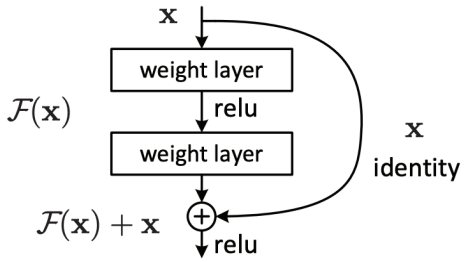

Figure 5. A residual block. [6]

2.1. Thesis

In this paper, an empirical study is conducted to determine if adding too much cross-block dense connection can cause redundancy in the network and cripple the network performance. Experiments are conducted to find the balance between the benefit of shortcut connection and the disadvantage of the added connections.

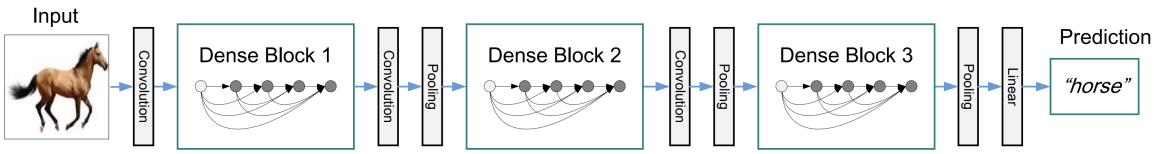

Figure 6. A DenseNet with three dense blocks. The blocks are connected sequentially and the connection layers are referred to as transition layers, which reduce the feature-map sizes through convolution and pooling. [5]

2.2. Literature Review

Consider an image network with L layers. At each layer, there is operation $H_l(\cdot)$ that will produce outcome x_l. The operation is a composite function perform a convolution. A typical setup of the composite function is batch normalization followed with 1-by-1 convolution and ends with 3-by-3 convolution. In traditional image network, H_l takes the output from the last layer x_{l-1} as input:

$$x_l = H_l(x_{l-1}) \tag{1}$$

ResNets. [6] In addition to the feed-forward connection in traditional networks, ResNet introduces identity shortcut to the transformation, the relation in ResNet looks like:5

$$x_l = H_l(x_{l-1}) + x_{l-1} \tag{2}$$

The $+x_{l-1}$ operation is called a skip connection. It can provide a stable gradient in the training process as in the optimization phase of the training, the error is backpropagated as the partial derivative:

$$\frac{\partial H_l(x_{l-1}) + x_{l-1}}{\partial x_{l-1}} = H'_l(x_{l-1}) + 1 \tag{3}$$

So the skip connection part of the residual block became 1 after the partial derivative. This means even when $H'_l(x_{l-1})$ vanishes, the identity is still passing to the deeper layers, which supports the flow of gradients and helps the network converge.

DenseNet. [5] The addition of x_{l-1} shortcut provides the stable gradient for ResNet and makes it possible to train ResNet at over 100 layers. DenseNet pushes this information flow between layers further by introducing dense connection. Instead of taking the output of the immediate earlier layer as input, it takes the output of all the previous layers as input:7

$$x_l = H_l([x_0, x_1, ..., x_{l-1}]) \tag{4}$$

However, in DenseNet, dense connections only apply to layers within the same dense block. The connection between different dense blocks remains the traditional feed-forward fashion.6

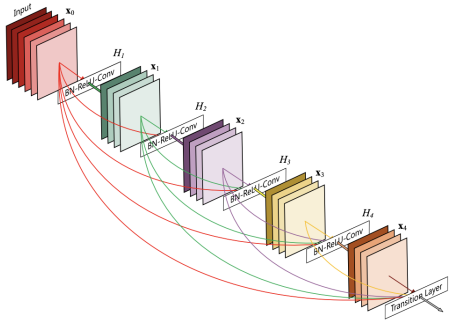

Figure 7. A 5-layer dense block. In addition to the traditional feed-forward connection, dense connections are added so that each layer takes all preceding feature-maps as input. [5]

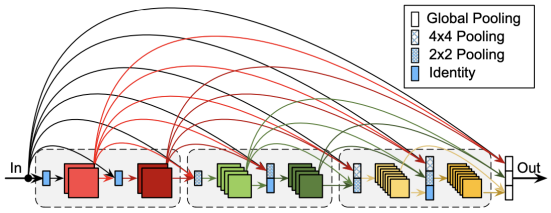

Figure 8. An illustration of the CondenseNet architecture. It differs from the original DenseNet that layers in different dense blocks with different resolution feature maps are also directly connected. [4]

CondenseNet. [4] To encourage feature re-use even more than the original DenseNet architecture, CondenseNet introduced dense connections between different dense blocks. To achieve these connections between layers with different feature sizes, the outputs with bigger feature sizes are downsampled into the same feature size as the smaller layers using by going through an average pooling layer.8

3. Methodology

The study follows DenseNet architecture as the base model architecture.9 This means in-block dense connections are always included in all of the tested models. On top of the base model, options are added for dense connec-

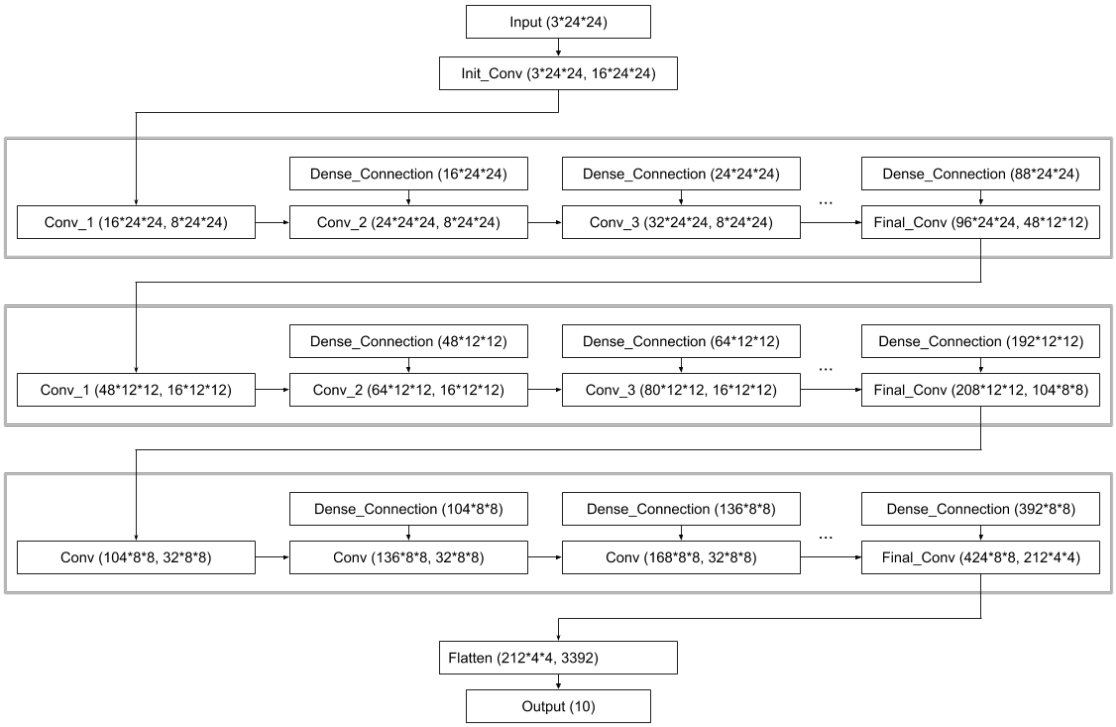

Figure 9. Modeled DenseNet architecture with 3*24*24 input size and output of 10 classes. The network has 3 dense blocks with 10 stages for each block and growth rate of 8, 16, 32 respectively.

tions between different dense blocks and the translation layers connecting different dense blocks. Adjustments of these two settings are made to create different variation models of DensNet and CondenseNet for this empirical study.

3.1. Configuration

Cross-block Connection Rate configures the number of dense connections between dense blocks. It is a floating-point number between 0 and 1. A rate of 0 represents there is no cross-block dense connection and a rate of 1 represent there is a full cross-block dense connection (a layer takes all of the previous layers from its block and all of its previous blocks as input, output feature maps in the previous block are downsampled using average pooling). In DenseNet, this rate is 0 and in CondenseNet, this rate is 1. For cross-block connection rate between 0 and 1, a random sample is created from all of the layers in all of the previous blocks. The number of layers sampled is the rate times the total layers. Note the sample is created at the layer level so that each layer will select a different set of sampled layers to take as input by the cross-block connection.

Translation Reduction Rate determines how the number of output channels produced by the translation convolution at the end of each dense block except the last dense block. It is a floating-point number between 0 (not including 0) and 1. A rate of 0 represents all of the output channels produced by the translation is reduced, leaving no direct connection between this dense block to the next dense block, this is why translation reduction rate should not be set to 0. A rate of 1 represents no reduction in the translation channels and all of the output channels go to the next dense block. When the rate is between 0 and 1, a 1-by-1 convolutional filter is used to reduce the channel to the rate times the channel count. In DenseNet, this rate is 0.5 meaning half of the output by the end of block translation is reduced by the 1-by-1 convolutional filter. In CondenseNet, the reduction rate is 0.1, reducing most of the channels by the translation layer to make room for its abundant cross-block dense connections.

3.2. Implementation

The entire Python package along with detailed instruction on how to interact with the code can be found here: https://github.com/Boyu1997/cnn-dense-connection.

4. Experiment

Using the configurable model defined in section 3, two groups of experiments are carried out. A pre-trained model is also created for the best performing model is also created

in each experiment.

An example Python notebook providing a demo of a basic experiment process along with some pre-trained result showcase can be found here: https://github.com/Boyu1997/cnn-dense-connection/blob/master/example/example.ipynb.

4.1. Dataset

CIFAR-10 dataset, abbreviated as C10, is used to evaluate the performance of the different models in the experiment. The dataset consists of $60,000$ colored natural images with 3232 pixels containing an object belongs to one of the 10 classes. CIFAR-10 dataset came in two subsets, the training set contains $50,000$ images and the testing set contains $10,000$ images. The training set is further split into training and validation set, using the standard train validation split rate of 0.8, resulting in $40,000$ training images and $10,000$ validation images. Data augmentation schemes are used in the training set to obtain better performance and the dataset with augmentation is referred to as C10+.

4.2. Training

Two sets of experiments are conducted. The first experiment is based on the DenseNet model, where additional cross-block connections are introduced to use the output from previous convolutional layers output in previous dense blocks randomly for at each convolutional layer at the rate of 0.2, 0.4, and 0.6. The reduction factor at the end of each dense block is set at 0.5 which is the same rate in the DenseNet paper. The second experiment is based on CondenseNet. Following the implementation in CondenseNet, we set the reduction rate at the end of each block at 0.1, which for a network with 10 stages in each block, this reduction rate means the output of the final convolutional layers at the end of each block is the same shape as all the other convolutional layers. The final output is stacked onto the output of all the previous layers. For CondenseNet, each layer will take the entire stack of outputs as input. For the proposed variation networks, the output of the final convolutional layer in the previous block will always be kept, while for all the other convolutional layer output in the output stack, they will be selected randomly at the rate of 0.8, 0.6, and 0.4 to compare the impact of reducing cross-block connection on the network performance.

4.3. Results

Training and validation lost and accuracy12 13, average epoch training time10 11, as well as test accuracy1, are generated and used to compare model performance.

5. Discussion

Testing the model in two perimeter sizes at 1M and 0.3M. The 1M models can produce comparable results to the best performance in the CIFAR-10 dataset, which is commonly agreed at 95% accuracy or 5% error rate without overfitting. The 0.3M models are used to test for mutiple experiment setting and the results at both perimeter sizes points to a similar result.

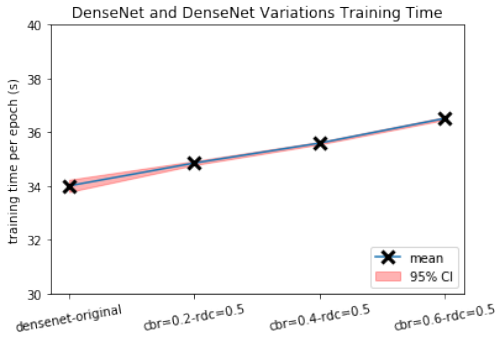

Figure 10. Average epoch training time for DenseNet and DenseNet variant experiments.

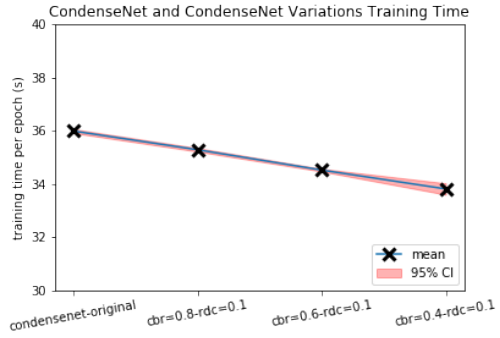

Figure 11. Average epoch training time for CondenseNet and CondenseNet variant experiments.

Method	Depth	Params	C10	C10+
DenseNet	30	0.24M	-	15.45
DenseNet (cbr=0.2)	30	0.26M	-	14.18
DenseNet (cbr=0.4)	30	0.29M	-	14.80
DenseNet (cbr=0.6)	30	0.31M	-	15.65
CondenseNet	30	0.32M	-	13.95
CondenseNet (cbr=0.8)	30	0.29M	-	13.98
CondenseNet (cbr=0.6)	30	0.27M	-	15.06
CondenseNet (cbr=0.4)	30	0.25M	-	14.78
DenseNet	78	1.09M	-	8.08
DenseNet (cbr=0.2)	72	1.08M	-	7.99
CondenseNet (cbr=0.8)	66	1.07M	-	7.74

Table 1. Testset top 1 error rate for experiment models.

5.1. DenseNet Experiment

Showing by the result in figure 12, with the cross-block connection at 0.2 (second column from the left) the network gained a boost in performance. However, if we keep in-

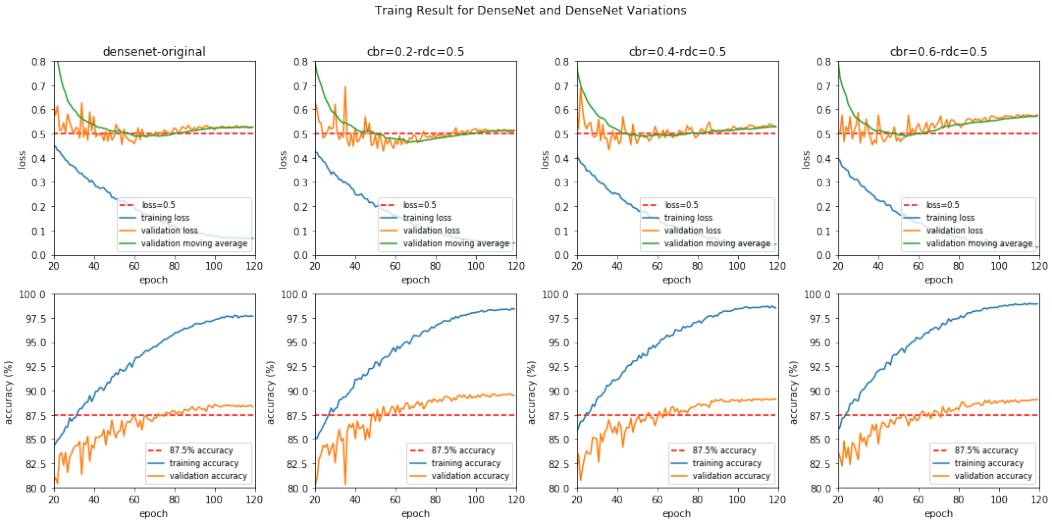

Figure 12. Training and validation loss and accuracy for DenseNet and DenseNet variant experiments.

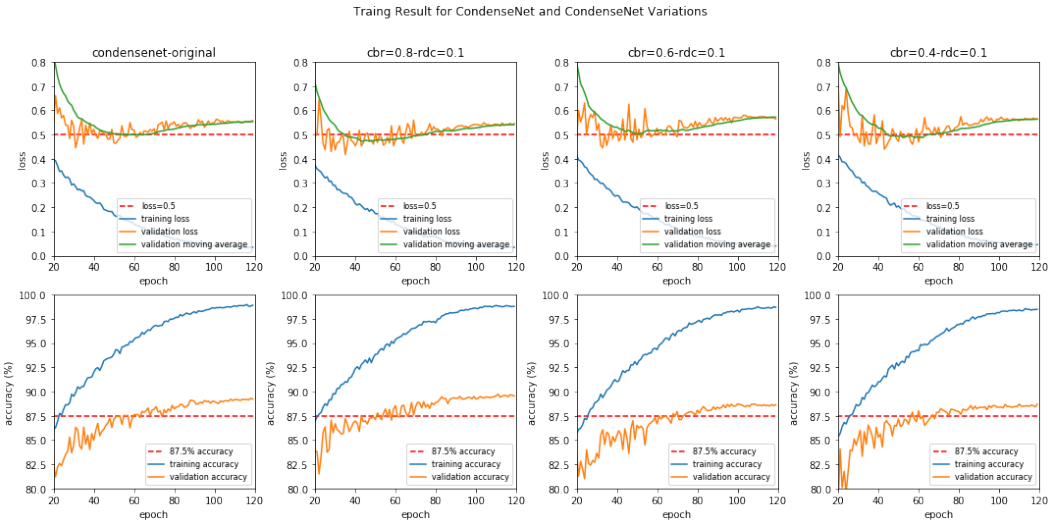

Figure 13. Training and validation loss and accuracy for CondenseNet and CondenseNet variant experiments.

creasing the cross block connection rate to 0.4 and 0.6 (the right two columns), we see the minimum validation loss starts to increase, going back to the 0.5 minimum validation loss of the DenseNet model. Similar to the validation accuracy, we see very similar validation accuracy for DenseNet and DenseNet with the cross-block connection rate at 0.4 and 0.6, but DenseNet with 0.2 cross-block connection rate performs noticeably better. This result confirms the theory that though cross-block dense connections can improve performance, the result shows that it is not the case that more cross-block dense connection can directly translate to better performance. The benefit of having cross-block dense connections diminishes after passing the rate of 0.2.

5.2. CondenseNet Experiment

From result figure 13, the variation CondenseNet with the cross-block connection rate at 0.8 (second column from the left) performs better than the other three networks in terms of validation loss. Looking at validation accuracy, the 0.8 cross-block connection network also performs the best. However, the validation accuracy for CondenseNet is very similar despite its higher loss. The other two CondenseNet variation networks (right two columns) have a similar and lower accuracy among the four networks. It is a bit unclear why the result looks like this. A possible explanation is that because of the high reduction rate at the end of each block, the initial input toward the next block is not sufficient to

contain all the necessary information. This is why when removing a larger portion of the cross block connections, we start to see a drop in performance despite the initial gain of network performance. Nevertheless, the observation that reducing the cross block connection rate from 1.0 in CondenseNet to 0.8 results in an improvement in performance shows evidence that the full cross-block dense connection in CondenseNet is redundant.

5.3. Conclusion

The experiment results confirm both positions in the thesis. First, the introduction of cross-block dense connection can improve the model performance by providing shortcut connections similar to the ResNet shortcut connection. Second, network architecture like CondenseNet has too much redundancy and a fully cross-block dense connection can wreck network performance. The result shows that by reducing the cross block dense connection in CondenseNet from 100% to 80%, the network can retain the same performance while reducing the perimeter size by over 10%.

5.4. Future Work

The empirical findings in this study show that cross-block dense connections can be beneficial to network performance but the relation is not simply more connections means better performance. It appears to the performance benefit starts to degrade after a certain connection rate. The two experiments with different configurations in the translation reduction rate give two different optimum cross-block dense connection rate, at 0.2 and 0.8. The finding raises three questions that need further research. First, is there an optimum cross-block dense connection rate and what are the factors affecting this rate. Second, if the redundancy produced by having too much cross-block dense connections applies also to dense connection within the same dense block. If so, removing some dense connections should boost the performance of DenseNet. Finally, the possibility that the redundancy is a problem for all shortcut connections, and the possibility of having a similar improvement on ResNet.

References

[1] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. Imagenet classification with deep convolutional neural networks, 2012. 3

[2] George Cybenko. Approximation by superpositions of a sigmoidal function, 1989. 2

[3] Serena Yeung Fei-Fei Li, Justin Johnson. Cs231n convolutional neural networks for visual recognition, 2017. Stanford University. 2

[4] Laurens van der Maaten Kilian Q. Weinberger Gao Huang, Shichen Liu. Condensenet: An efficient densenet using learned group convolutions, 2018. 3, 4

[5] Laurens van der Maaten Kilian Q. Weinberger Gao Huang, Zhuang Liu. Densely connected convolutional networks, 2017. 3, 4

[6] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2016. 3, 4

[7] Andrew Zisserman Karen Simonyan. Very deep convolutional networks for large-scale image recognition, 2015. 3

[8] Bhiksha Raj. 11-785 introduction to deep learning, 2019. CMU School of Computer Science. 1, 2, 3

[9] Yoshua Bengio Patrick Haffner Yann LeCun, Léon Bottou. Gradientbased learning applied to document recognition, 1998. 2, 3

#MC-professionalism: Following the CVPR styling guideline, one of the standard formatting in computer vision research, a professional work product.

#MC-organization: Dividing the report into five sections, background knowledge, introduction, methodology, experiment, and result discussion. Communicate the research project in an effective and organized way.

#MC-evidencebased: Research findings based on empirical experiment result, with normative claims pointing to the empirical finding to support it.

#MC-sourcequality: Literature reviews on some of the most important and most cited papers in computer vision research. Recent state of the art findings published between 2016 and 2018.

#CS-responsibility: Conducting experiments and following the project planning timeline. Spending extra time fixing issues in the code even though it was discussed in the last minutes. Progress can be found on GitHub repository commit-tree.

#CP194-qualitydeliverables: Implementation and self-developed experiment on state of the art computer vision research, report findings back by empirical findings.

#CP194-planningarchitecture: Prioritize implementation and experiment, using experiment results as milestones to check the project progress and adjust plan and expectation to ensure on-time delivery.

#CP194-feedback: Getting feedback from code review by Professor Sterne, discussing results with Ph.D. students at HKUST. Implementation changes to model architecture and implementation detail so that the model can execute correctly.

#CP194-research: Research on DenseNet and CondenseNet, compare the difference and learn from the authors how they attempt an improvement on a high-performance model architecture.

#CP194-connect: Use my Ph.D. classmate during my exchange semester at HKUST as knowledgeable people in the field, ask their view and share project ideas to get comments.

#CP194-metrics: Evaluating the performance of model and quality improvement made by the model using industry-standard measurement, like the train validation and test split.

#CP194-accountability: Self-direct the project, finding resources and communicating with knowledgeable people to ensure the final product is industry standard. At the last minute change in the codebase to fix an essential piece of implementation that is not doing what I intended, allocate more time to the project from my personal life to make an on-time delivery.

#CS156-neuralnetworks: Implementation of neural network in PyTorch with self defined function. Experiment with different variation of DenseNet and CondenseNet architecture and discuss cost and benefit of this kind of network structure.

#CS156-deeplearning: Discussion on the vanishing gradient problem of deep learning and how previous works (ResNet, DenseNet) has attempts solve this problem and explains why their method works. As well as making original contribution on the shortcut connections.

#MC-audience: Having the non-technical readers in mind, have an in-depth background knowledge section to provide preliminary information before the main part of the paper. In addition, a Python notebook with step by step instruction is created for beginners to use the package.

#EA-testability: Came up with two hypotheses on the benefit and disadvantage of cross-block dense connection, test the hypothesis by conducting an empirical experiment.

#EA-dataviz: Visualization used throughout the paper, from experiencing hard to companies concepts to the empirical result of the experiment.

#EA-modeling: Explain the multilayer perceptron as a modeling of a human brain to get the problem-solving ability like a human, though on a very specific task.

#EA-gapanalysis: Use the concept of gap analysis to explain the importance of convolutional neural networks. Show why shift-invariance is so important to computer vision tasks, and show why convolutional filters are the right way to solve the problem.

#EA-heuristics: The process of developing machine learning architecture relies strongly on heuristics. The thought "this might work" or "this does not look right" is a common starting point in machine learning research. In my example, when reading the CondenseNet paper and calculating how many times a feature gets reused in such network architecture, I had a thought that this architecture is redundant, so I decided to design an experiment to test my heuristic thought.

#FA-induction: Use the information that shortcut connection can be beneficial as it provides better gradient in training, assume even with just some shortcut connection, like the connection rate of 0.2, there will still be an observable benefit to the network performance.

#FA-deduction: Use empirical results obtained in experiment to deduct the validity of inductive claims and make deductive logical conclusions.

#FA-optimization: Explaining why optimization is difficult with the vanishing gradient problem, and showing how shortcut connections can help the optimization process in training neural networks.

#FA-algorithms: Analyze machine learning algorithms and came up with a novel improvement on the algorithm. Doing implementation and performance testing on the cammed up algorithm.

#CS-emergentproperties: Explain intuitively how a convolutional neural network finds features is though emergent properties, the initial layers find small patterns and layer by layer, a higher-level understanding is built and the final layers of a convolutional neural network can identify objects like a flower or a car.

#CS-networks: A study on the network structure, the research is about how the convolutional filters are connected in the network can affect the quality of the model and a change in model performance.