MLP Coursework 2: Exploring Convolutional Networks

s1882930

Abstract

Dilated convolution is becoming more and more popular as it maintains the resolution and receptive field of the network by in inserting “holes” in the convolution kernels. It combines the advantages of max pooling and strding. So the method to choose dilation factor and the way it change must influence the performance of the whole CNN. In this work, we test three different method to change the dialtion factor: constant, linear growth and exponential growth. The experiment results tell us that in small image data set the constant dialtion factor has a better classification effect and the growing dilation factor may sparse the local information and can’t reach to our expectation.

1. Introduction

Recently, convolutional neural networks (CNNs) have made a great success in computer vision tasks, which accerlateds the development speed of many industrial applications. Modern networks of image classification combine multi-scale contextual information by continous layers of pooling and subsampling , reducing resolution until a global prediction is obtained(Krizhevsky et al., 2012). Dilated convolutions are effectively convolutions with expanded receptive fields, allowing a model to learn higher order abstractions without dimensionality reduction. Also, they are usually inplemented in segmentation networks(Yu & Koltun, 2015), and audio generation(Van Den Oord et al., 2016) among other things.

Our work is motivated by that the previous work of using dilated convolution network in the semantic segmentation achieved successfully high accuracy in prediction due to due to its ability to expand the receptive field without losing resolution or coverage(Yu & Koltun, 2015). However, dilated convolution network caused the several problems that only locations which contain non-zero values can be sampled. It resulted in losing some feature information continously. That is to say, when we increase the dilation rate, using a larger pad containing more zeros, to implement covonlution and to finish classification tasks. Worse results may occur in this circumstance. Related issues are defined as "gridding", which is as zeros are padded between two pixels in a convolutional kernel, the receptive field of this kernel only covers an area with checkerboard patterns - only locations with non-zero values are sampled, losing some neighboring information.(Wiyatno & Orchard, 2018). Based on this, we raised the question about relationship between the rate of dilation and the performance of the model we use. Dilation covolution usually fit for the long-context. when applying it to the small picture classification task, will it get the satisfied performance as before.

We design several CNN structures to compare the model perforemance under different situation of different dilation rate. In this work, we design a convolution network which picks the different features, using the kernels, from the input data , pass the intermediate production through different convolutional layers and get a final vector with the classification lables in probability form to do the prediction. In our structure, we set several parameters, such as the way of padding, the number of filter, the type of convolutional layers and so on. Standard convolution corresponds to dilated convolution with rate = 1. Employing large value of dilated convolution rate expends the model’s field-of-view.

Initially, also the first stage of the experiment, we implement four baselines and compare the different performance with different dimention reduction types in the same setting of other parameters. Four basic layers are built to do the front propagation and back propagation, which enable the loss values to pass through, calculate the deviation, and upgrade the kernel matrixs’ values. Then, the dialted rate and the padding rate are changed in each experiement state. The dilated convolutional rates are set in the three differnt way: constant, linear growth and exponential growth.In order to tune with the problem generated by discordance between the kernel size and feature map size in each layer. The padding rate also is also tested in order to let the feater map accomodate the kernel.In the second stage of experiement, the CNN only contians the dilated convolutional layers without any maxpooling and average pooling.

Using CNN module, we evaluate the network through controlled experiments on the EMNIST dataset, a standard dataset for learning, classification and computer vision systems. In our model, each input is small picture with 28 * 28 pixel representations. The whole data set is separated to training set(100000), validation set(15800), and test set(15800). we set 100 as the batch size and do 100 epochs each experiment. The detailed procedure will be discussed in section 4.

Section 2 will focus on the features of each models and introuduce the technique in the experiment context. The section 3 will focus on the experiment context and show the setting. The detailed description of the experiment will be discussed in section 4 and following results and reflection will be shown in section 5.

2. Implementing convolutional networks

Different from a fully connected neural network, CNNs don't have every unit in one layer connected to other units in the next layer. What's more, the whole network share the same weight matrix and bias, and it helps to learn the different features at different parts of the image.

There are two ways to implent convolution operation.One is using "SciPy" convolution function "scipy.signal.convolve2d " and " scipy.signal.correlate2d". The other one is "Serialisation" in which the the convolution operation is turned into a single matrix multiplication. This method just use a a single large matrix multiplication to do *fprop* and *bprop*, which outcomes the scipy funtions in view of computationally efficient in a single large matrix multiplication. However, it has a disadvantage that the resultant matrix has repeated elements.

We can pad the input matrix around the border by symmetrically adding zeroes. Padding the input makes computation convenient by preserving input volume and prevents fast data loss at border. This padding process is controlled by a hyperparameter.

A convolutional layer and a max pooling layer are designed initially by completing the two corresponding classes in *layers.py*.

2.1. Convolutional Layer

Three functions are designed in the convolutional layer, *fprop*, *bprop* and *grads_wrt_params*.

Front Propagation we use H_{l-1} to describe an input layer and H_l is for an output layer, W_l for kernel, b_l for biases, E for error function. $\frac{\partial E}{\partial H_l}$ presents the gradients of E with regard to output, $\frac{\partial E}{\partial H_{l-1}}$ for gradients of E with regard to input, $\frac{\partial E}{\partial W_l}$ for gradients of E with regard to kernel, $\frac{\partial E}{\partial b}$ for gradients of E with regard to bias.

Then for function *fprop*, we can calculate the output with the following formula:

$$H_l = H_{l-1} * W_l + b_l$$

Back Propagation in the process of back propagation, gradients of E with regard to input is computed :

$$\frac{\partial E}{\partial H_{l-1}} = \frac{\partial E}{\partial H_l} * W_l$$

Grads_wrt_Params The gradient of the weights and bias are computed with the following formula:

$$\frac{\partial E}{\partial W_l} = H_{l-1} * \frac{\partial E}{\partial H_l}$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial H_l}$$

2.2. Pooling Layer

After convolutional layers, pooling layers are implemented to simplify or summarize the information from the convolution layer by doing a statistical aggregate like average or max via taking each feature map and generating a down sampled feature map. To reduce variance, reduce computation complexity and extract low level features from neighbourhood, pooling layers usually plays an important part in many tasks.

2.2.1. Max pooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions.This is to help overfitting by providing an abstracted form of the representation.Besides it reduces the computational cost by reducing the number of parameters to learn. Also, Max pooling is particularly well suited to the separa- tion of features that are very sparse.(Boureau et al., 2010). Max pooling (?) operation reports the maximum output within a rectangular neighbourhood. Figure 1 shows the the process of max pooling.It consists in splitting the input in (usually non-overlapping) patches and outputting the maximum value of each patch(Dumoulin & Visin, 2016) .

Front Propagation It outputs the maximum value of each patch decided by the kernel size. For each of the patch represented by the filter, we will take the maximun value of each patch and rebuild a new output matrix where each unit is the maxi of a patch from the previous layer. $\mathbf{v}_i$ presents the values in each patch or region.

$$f_m(\mathbf{v}) = \max_i \mathbf{v}_i$$

Back Propagation When we do back propagation, we need to get the production generated from last layer and comput the gredient. Through judging that if the value$a$ or$b$in H^{l-1} is the max one in the patch or region, one or zero will be put in the new matrix. Then we can get the $\frac{\partial H^l}{\partial H^{l-1}}$ matrix and $\frac{\partial E}{\partial H^{l-1}}$is the Kronecker product.

$$m = \max(a, b)$$

$$\frac{\partial m}{\partial a} = \begin{cases} 1 & \text{if a>b} \\ 0 & \text{else} \end{cases}$$

$$\frac{\partial m}{\partial b} = \begin{cases} 1 & \text{if b>a} \\ 0 & \text{else} \end{cases}$$

$$\frac{\partial E}{\partial H^{l-1}} = \frac{\partial E}{\partial H^l} * \frac{\partial H^l}{\partial H^{l-1}}$$

2.2.2. Average pooling

Average pooling extracts features smoothly, that is to say implementing average pooling bring all information to the

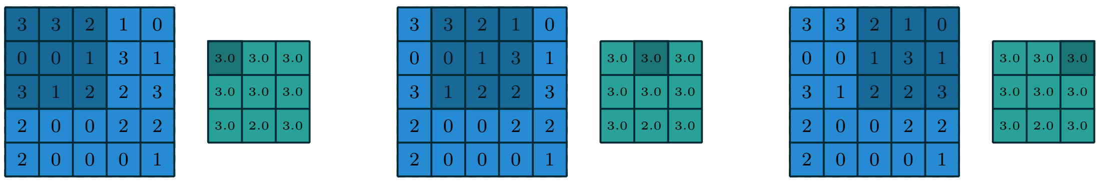

Figure 1. Computing the output values of a 3×3 max pooling operation on a 5×5 input using 1×1 strides (Dumoulin & Visin, 2016)

next layer which is a generalized computation. P is the number of the unit number in each patch overlapped by kernels. v_i is the value in each patch we want to calculate. Just like the max pooling precess, we can get the average matrix through each layer and let the product to pass though next layer.

$$f_a(\mathbf{v}) = \frac{1}{P}\sum_{i=1}^{P}\mathbf{v}_i$$

Sometimes, average pooling can get bad result compared with max pooling because it take the global information into consideration and get the average value. And max poolin can get some specific features such as edge information which can be essential to classification task.

2.3. Strided Convolution

Another way to reduce dimension is striding, which is to generate a strided layer to reduce the dimension and extract features. Figure 2 shows that it is different from max pooling operation. And for simple models, strided convolution is applicable and have proved to outperform many previous complex architecture.

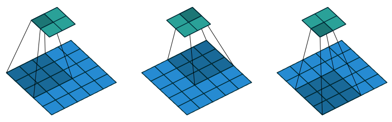

Figure 2. a 3×3 kernel over a 5×5 input with a stride factor of 2 (Dumoulin & Visin, 2016)

However, max pooling is only related to comparison and dropping, while strided convolution is involved with convolution, which indicates that strided convolution would cost more running time as well as storage place to generate the output.

2.4. Dilated Convolution

Based on the fact that dilated convolutions enable exponentially expanding receptive fields without lost on original information (Yu & Koltun, 2015). The filters in dilated convolution are simply stretched and multiplied by corresponding input pointwisely, shown in Figure 3. The receptive field of output units would be expanded. The different dilation factor could give different view(shown in the Figure 3).

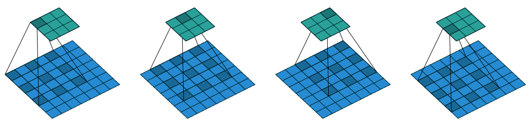

Figure 3. Convolving a 3×3 kernel over a 7×7 input with a dilation factor of 2 (Dumoulin & Visin, 2016)

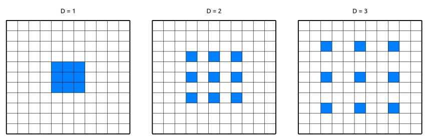

Figure 4. different dilation factors for 1,2,3

Runing the Dilated convolution need more time and storage than strided convolution. Dilated convolution(a.k.a. atrous convolution), use downsamlping filters instead of feature maps, by inserting zeros between non-zero unit in each patcg. Strided convolutions are operations which just shrink the feature map from previous layer to the next.

3. Context in convolutional networks

The common conext of the network consists of convolutional layer, convolutional layer, dimension reduction layers and the final regression function layer, shown in the Figure 5. The major difference is a convolution filter is extracting features from the matrix of data, whereas the pooling layer is only downsampling the matrix of data and, at the same time, reduce the size of data.

The context is designed to test the performance of the classificaiton effect on EMNIST dataset via using different dimension reduction layers. The model takes N images as input and produces a vector K containing the true label of each character.

Starting with describing a basic form of the context, where each layer has one input channel. The representation in each layer is the same , although the feature maps are not normalized. We use cross-entropy as loss function. Detailed descriptions are as follows:

Each convolution operates on all layers. Each of these convolutions is followed by a pointwise truncation$max(\Delta, 0)$. The architecture of baseline is summarized in Table 1. For a fair comparison, all methods use the similar neural network architecture, which is a multi-layered perceptron with eight hidden layers(contain the reduction layers) of 64 filters each, followed by a softmax output layer.

Input and output We use EMNIST dataset The basic formula for output size based on kernel and other parameters is

$$(D + 2P - F)/S + 1$$

where D is the input image size of one dimension, P is the padding rate, F is the kernel size, and S is the stride.

Loss function We use cross-entropy as loss function for classification in all methods and scenarios.

Optimiser All models are trained for 100 epochs per task using the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$, learning rate= 0.00001) as the default setting .Such a model is usually optimized with Adam (Kingma & Ba, 2014), which in general is a good choice for this task. In all experiments, the optimizer is all the same.

Batch size The mini-batch size is always a trade-off between computation efficiency –because GPU architecture prefers it large enough – and accuracy(Mishkin et al.). And in this work, we ch

Layer number we let the number of layer to be 4 unchanged in each experiment. Due to image size limitations(28*28), the number of layer is also restricted. The deeper the network, the better the performance we will get when there is no overfitting. In order to let the network have ability to fit our data, we set 4 layers for each task or experiment. It must have to tune with the padding factor, the kernel size, and the dilation factor. For each experiment we can calculate it by an small programme designed in the task by the writter.

Filter For CNN, we use four convolution layers (with 64 filters of size 3*3 for all convolution layers.).

Layer: The dimension reduction layers usually has four types: max pooling layer, average pooling layer, strided convolutional layer and dilated convolutional layer. They can reduce the dimension of features and improve the regression effect.

Activation function: Rectified Linear Unit (ReLU) is implemented as the activation function after a convolutional network. In this work, we ues a linear layer to do transformation from the final output of the layers to the one-hot lables, mapping the numerical outcome to each class.

4. Experiments

Our implementation is based on the pytorch and the Google Compute Engine.

4.1. Baseline

Now, we describe the experimental configuration. Here, we use the MNIST dataset with the splitting to generate the four context in page 5. The standard train split was used, with 10w images and 1.58w test/validation images . The preprocessing of images includes zero padding to 28*28 pixels and a standard initialisation for the weight metrix(kernel matirx) with zero mean with unit variance. No other data augmentation is applied.

We use four baseline strategies. First, we use the basic experiment setting to implement the experiment, the initial settings are shown in the Table 1, the initial motivation of the experiment setting comes from the idea that the different demention reductuion way will influence the run time and test accuracy result. Although dilated convolution has a great effect on the long-range context, it is uncertain that it will hava the same performance as the dense classification when we apply it in the EMNIST dataset.

The four experiments are implemented separately. The CNN with striding experiment is added a strided layer(of size 3x3) , followed by another convolution layer (with 64 filters of size 3x3). The second experiment change the dimention reduction type to dilation and use the kernel of 3x3 and the same setting like previous one. When it comes to the max pooling experiment, we add one max-pooling layer (of size 2x2), followed by another the same convolution layer. And the average pooling experiment is similiar to the max pooling experiment, just changing the dimension reduction layer type after all convolutional layer. We don't use drop out method to avoid overfitting because the initial setting has the similar funcion and the main purpose is to see the run time and the accuraty rate of four dimention reduction type.

?? shows the accuracy learning rate lines on the training set and the validation set. And **??** shows that in each independent experiment when we find the best model and apply it on the test set, the accuracy and the loss values. The yellow lines represent the string experiment and the green ones for the dilation, the blue ones for the average pooling and the orange ones for max pooling.

	dilation	striding	max	average
accuracy	0.8756	0.8732	0.8792	0.8803
loss	0.4384	0.3698	0.3533	0.3579

Table 2. the accuracy values on the test set among baselines

4.2. Dilation factor study

This part is motivated by the study of using the dilated convolution to do dense prediction task in the field of semantic segmentation(Yu & Koltun, 2015). When we get two-dimensional image, for each location i on the output y and a kernel w, dilated convolution is applied over the input feature map x:

$$y[i] = \sum_k x[i + rk]w[k]$$

where the dilated rate r represents inserting $r - 1$ zeros between two continous filter values along each spatial dimension in each patch. When the dilated rate equals to 1, it

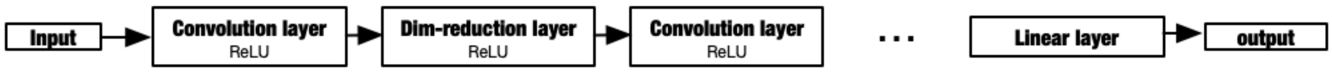

Figure 5. the context of CNN

Table 1. The intial setting for baseline

convolutional layer				dim_reduction			
layer	kernel size	padding	stride	type	pading factor	stride	dilation factor
1	3	0	0	striding	1	2	1
2	3	0	0	dilation	1	1	layer+2
3	3	0	0	max pooling	1	0	1
4	3	0	0	average pooling	1	0	1

is just the common convolutional layer. And we can modify the dilated rate *r* in each layer to increase the kernel's field-of-view(shown in the **??**).

In order to learn the relationship between dilation rate and the model performance. we use the same CNN net work as the previous experiment. **Four layers** are set between the input images and the output images. when we finish each convolution computation, the *ReLU* function as an activation to generate the outcome in each layer. We still use *Adam* with the default parameters and the *LinearRegression* function as the prediction method to finish the classification task.

The experiment setting is hown in the Table 3. Here we will give the detailed description about each experiment implement and the reason why we make such setting.

experiment.no	dilated rate	kernel size	padding rate
1	2,3,4,5	3	1
2	2,2,2,2	3	1
3	1,1,2,4	3	1
4	2,3,4,5	3	2
5	2,2,2,2	3	2
6	1,1,2,4	3	2
7	1,2,4,8	3	2

Table 3. different dilation rate setting in each experiment

We focus on how the dilated factor change along the CNN. That is to say, with the the index of the layer increase, how the dilate factor change tuned with sub-sampling. In the last part, we use the linear growth for the dialted factor as a baseline(dilated factor: 2,3,4,5). It can be seen in a linear growth way for expending the kernel size by inserting zeros between each unit the same kernel. From the result of the experiment in the baseline in the Table 2, it performs not as good as we think. It may resulted from the fact that when the dilated factor increase in the deeper layer, we may lose some feature information or the overlapping part may have a bad influence on the feature extraction. Based on it, if the dilated factor change in other way, can the performance of the model get improvement? Out of this question, we do the following experiment.

First, we want to compare the baseline which the dilated rate increases in a **linear way**, with **constant** and **exponential growth** . However, the dilated factor can not be too large because of the input of each layer and the exponential growth faces some difficulty to do. It can be fixed by change the padding rate, while, out of the think that controling variable padding rate in the experiment, we also do the experiment[4-6] to see the model performance.

4.2.1. The first group of dilation rate change experiment

At first, we do the experiment 1-3 shown in the Table 3, which pading rate equals to one. In order to compare with the baseline in the previous experiment, the pading rate as a hyperparameter needed to be controlled. For the baseline seen as a linear growth(dilated factor: 2,3,4,5 in each layer with 64 filters), we compare the model performance with the experiment results of dilated rate grown in a exponential growth(dilated factor: 1,1,2,4) and unchanged-keeping the dilated rate in a constant($r = 2$). Because $paddingrate = 1$ and the kernel size(3) setting constrain the dilated rate to grow, so that in experiment 2, we only set the the exponential growth in **the 3rd, 4th layer**.

The accuracy and loss value results on the test set are shown in the Table 4, and the learning curves shown in the Figure 7.

dilated rate	accuracy	loss
2,3,4,5	0.8756851898734177	0.43845667
2,2,2,2	0.8859493670886075	0.31677052
1,1,2,4	0.8813924050632913	0.38239577

Table 4. the best models results of experiment 1-3 on test set

4.2.2. The second group of experiment

We set the $paddingrate = 2$ for the purpose that the dilated rate can grow in exponential way(dilated rate = 1,2,4,8). Out of controling variables, we implement the experiment 4-7 to observe the CNN regression effect on different dilated rate growth in the context of $paddingrate = 2$. Four experiment are implemented in the CNN context we mentioned before (see Table 3). Experiment 4 is contrast to the baseline, only changing the baseline, experiment 5 is set for the constant dilated rate($r = 2$) , experiment 6 coming from experiment 3 and experiment 7 is set for the dilated rate growing in a expontial way.

dilated rate	accuracy	loss
2,3,4,5	0.8806962025316458	0.348427
2,2,2,2	0.8872151898734177	0.32867286
1,1,2,4	0.8808227848101267	0.39694628
1,2,4,8	0.8808860759493672	0.4037839

Table 5. the best model results of experiment 4-7 on the test set

Figure 8 shows the the learning curves of 4 different experiment and each experiment use the same colors on the training set and validation set. Table 5 lists the best model results of experiment 4-7 on the test set. In the training process, the constant dilated rate converages slower and perform worse than others. The other groups have no obvious difference in the training process and in the test process.

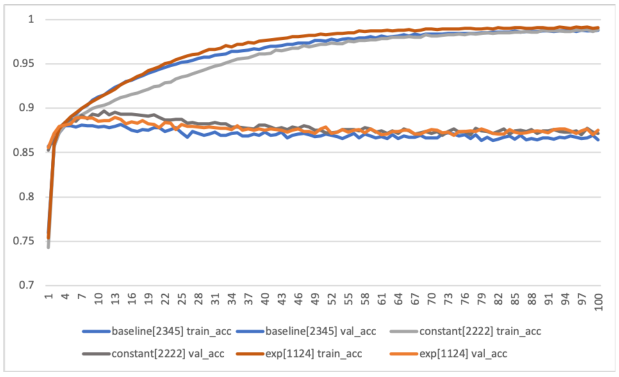

Figure 7. The accuracy values in experiment[1,2,3]

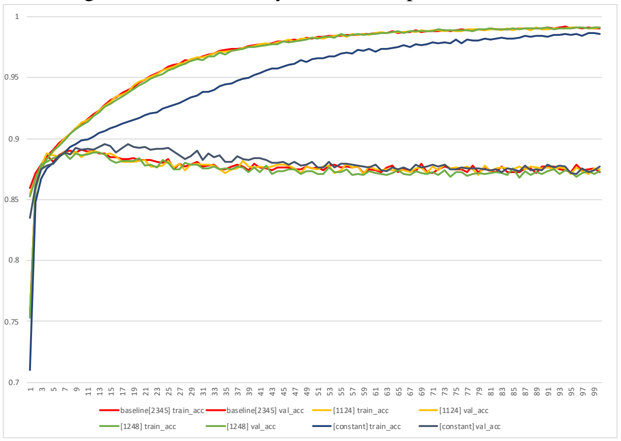

Figure 8. The accuracy values in experiment[4,5,6,7]

5. Discussion

From the experiment about baselines study, **??** and **??** tells us the accuracy learning rate lines and the loss and accuaracy in each independent experiment with the model found in each experiment. It is shown that dilation method has the similar performance as the strding method. And the max pooling can be trained faster and get a better result on the validation set. It can converge faster than the other experiment setting. And there is no apparent difference when performing the best model of each setting on the test set. They all get the accuracy rate atabout 88%.From the point of run time view, dilation method has the largest time cost. The computation complexity is higer than the others.

Then, in the dilation study for padding =1 (experiment[1-3]), We can have a clear view of the trend of how the accuray grows in each experiment Figure 7 . It is illustrated that, there is a apparent difference between experiment 2($r = 2$), where the dilated rate unchanged in each layer, and other experiment. The unvaried dilated rate($r = 2$) has a low accuary and converages slower than other groups in the training set but performs the best on the test set. The 3th experiment(dilated rate grows in a expontential way) has a faster learning speed and converge faster than others. At the same time, it can get a good result on the test set.

In the second stage for dilation factor in experiment[4-7] Figure 8 and Table 5 show us the result. In the training process, the constant dilated rate converages slower and perform worse than others. The other groups have no obvious difference in the training process and in the test process.

Although the dilated convolution operator is particularly suited to dense prediction due to its ability to expand the receptive field without losing resolution or coverage (Yu & Koltun, 2015), it isn't has a ideal effect on the letter classifiction. Instead, the model with constant dilation factor wins, which has a stable and effective classification ability on the test set, 88.59% accuracy in first part and 88.72% in the second part. When the dilation factor keep growing in linear way or exponential way, it has an accaurcy about 88.1%. However, we can't ignor the fact from the experiment result that the model with dilation factor growing in linear way and exponential way can converage faster and perform better on validation set. It may be worth studying in the future.

6. Conclusions

Dilated convolution allows us to explicitly control how densely to compute feature responses in fully convolutional networks (Chen et al., 2017) . And in image classification (Krizhevsky et al., 2012), using a neural network with dilated layers and max-pooling layers can have a image classification. However, from the experiment, the CNN with dilated layer may not perform ideally as we suppose.

It may come from the following reason:

The "gridding problem" The "gridding problem" mentioned in (Wang et al., 2018) may occur (shown in the Figure 9)when we use larger dilation factor in higher layers, the sampling could be sparse. This is because we lose the local information and the information can be irrelevant across large distances.

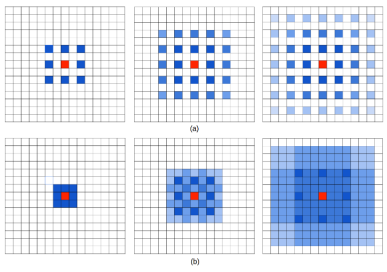

Figure 9. Ilustration of the gridding problem. Left to right: the pixels (marked in blue) contributes to the calculation of the center pixel (marked in red) through three convolution layers with kernel size 3*3.(a) all convolutional layers have a dilation rate r = 2.Ķ (b) subsequent convolutional layers have dilation rates of r = 1, 2, 3, respectively(Wang et al., 2018)

The kernel size We only use 3*3 as the kernel size to do regression and it may be too small to get the continous information from the image input.

Although through previous papers, dilated convolution has a good result on dense prediction, it truly performs worse than other structures. It may lie in that the EMNIST with 28*28 pixel can not offer dense information to the network and also the kernel size and the number of layer influence the dilation effect. In the future, we may need to change the kernel size with richer network structure to do image regression.

References

Boureau, Y-Lan, Ponce, Jean, and LeCun, Yann. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

Chen, Liang-Chieh, Papandreou, George, Schroff, Florian, and Adam, Hartwig. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. URL http://arxiv.org/abs/1706.05587.

Dumoulin, Vincent and Visin, Francesco. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, mar 2016.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Mishkin, D, Sergievskiy, N, and Matas, J. Systematic evaluation of cnn advances on the imagenet, june 2016.

Van Den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *SSW*, pp. 125, 2016.

Wang, Panqu, Chen, Pengfei, Yuan, Ye, Liu, Ding, Huang, Zehua, Hou, Xiaodi, and Cottrell, Garrison. Understanding convolution for semantic segmentation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1451–1460. IEEE, 2018.

Wiyatno, Rey and Orchard, Jeff. Style memory: Making a classifier network generative. *arXiv preprint arXiv:1803.01900*, 2018.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.