MLP Coursework 2: Exploring Convolutional Networks

s1836694

Abstract

The report decribes the implementation of convolutional networks and the exploration of 4 dimensional reduction methods including Max Pooling, Average Pooling, Strided Convolution and Dilated Convolution. The key research questions in this work are: 1) which dimensional reduction method works better on EMNIST dataset according to accuracy, runtime, and memory requirement; 2) further investigate how Max Pooling and Dilated Convolution influence the performance; 3) Is there any improvement that can be achieved. Pooling is able to reduce the parameters and avoid overfitting. Our experiment implies that the advantage outweighs the drawback of losing resolution. Surprisingly, we also find that instead of inserting one Max Pooling Layer after each Conv Layer, only one Max Pooling Layer after the second Conv Layer performs better. Strided Conv and Dilated Conv can be alternatives of a standard Conv Layer followed by a Max Pooling Layer. Continuous rising dilate rate performs better than constant one. More filters leads to higher accuracy and faster convergence.

1. Introduction

Convolutional Neural Networks is powerful architecture used to extract features and make prediction. The overall motivation of the work is to understand how Convolutional Layer and Dimensional Reduction Layers (including Max Pooling, Average Pooling, Strided Convolution and Dilated Convolution) work and how it influences a CNN's performance. The report decribes the implementation of convolutional networks and the exploration of how context is handled in convolutional networks. The key research questions in this work are: 1) which dimensional reduction method works better on EMNIST dataset according to accuracy, runtime and memory requirement; 2) further investigate how Max Pooling and Dilated Convolution influence the performance; 3) Is there any improvement that can be made (e.g. increasing the number of filters, changing the hyperparameters or architecture of CNN). To study these questions, the experiments are carried out and our measurement considers multiple aspects including accuracy, memory requirement, size of receptive field, computational cost and convergence speed.

All the experiments are conducted on the EMNIST dataset (Cohen et al., 2017). There are 47 classes containing digits as well as both upper and lower-case letters. The whole dataset is split into three parts: training set (100,000 instances), validation set (15,800 instances) and test set (15,800 instances).

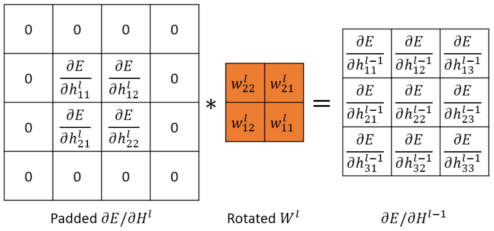

Figure 1. A simple example of backpropagation in convolutional layer. 0-padding is applied to 2D matrix of gradients w.r.t. outputs (with padding size equaling to $kernel\_size - 1$). The kernel is also rotated both vertically and horizontally.

The report contains two main parts: the implementation of CNN (Section 2) and experiments carried out to investigate 4 dimensional methods (Section 3-5). To be specific, Section 3 introduces the basic idea of 4 dimensional reduction methods. Why and how we carried out the experiments are covered in Section 4, followed by deeper explanation of the results (Section 5). Last section is conclusion which is drawn based on the experiments.

2. Implementing convolutional networks

This section briefly introduces the main ideas of Convolutional Layer as well as Pooling Layer, describes how we implemented them and compares two approaches of implementation.

Convolutional Layer is the most important component of CNNs, performing the majority of computations. The operation of convolution is rather expensive because the filter needs to compute dot products with one region of the inputs and move to next region of the inputs to operate dot products. This process repeats many times until the filter moves across the whole input volume. The advantages of convolutional layers are keeping context (or spatial) information of inputs and sharing the parameters (Stanford, 2016).

The implementation of Convolutional Layer contains three parts: forward propagation, back propagation and gradients with regards to weights. We use 'scipy.signal.correlate2d' function to implement convolving a 2-dimension image with a 2-dimension kernel, which is repeated until all the inputs channels, output channels and

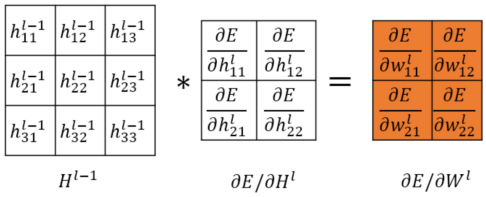

Figure 2. An illustration on how to compute the array of gradients w.r.t. weights.

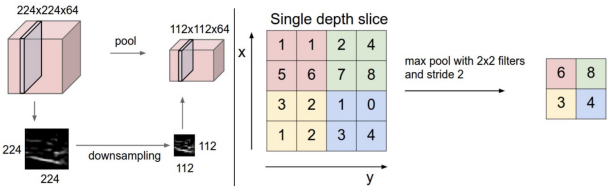

Figure 3. How Pooling Layer works. In the left example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. In the right example, the input slice of size [4x4] is pooled with filter size 2, stride 2 into output slice of size [2x2]. (Stanford, 2016)

instances in the batch are traversed. Similarly, in the back propagation, there are also three loops (i.e. traverse the whole batch size, each input channels and each output channels) within which we operate convolution with rotated kernels on the 2D array of gradients with respect to outputs. Before convolution, 0-padding with $padding = kernel\_size - 1$ is needed to applied to the 2D array of gradients with respect to outputs (see Figure 1). Figure 2 illustrates how to compute the array of gradients w.r.t. weights. The array of gradients w.r.t. weights has the same size of $num\_input\_channels \times num\_input\_channels$ kernels. This time, the array of gradients w.r.t. outputs is used as the convolutional kernels on the input of the layer. There are still three loops but batch iteration is inside, because we need to sum up the 2D gradients w.r.t. weights array for every instances to generate the final one. In terms of bias for each output channel in each instance (or image), it is the sum of every element in the array of 2D grads_wrt_output. Similarly, the bias for each image needs to be added up to get the final bias for the current output channel.

Pooling Layer aims to reduce the parameters and the computation of the whole CNNs. What the Pooling Layer does is downsampling. The downsampling process operates over each activation map independently (Stanford, 2016). Figure 3 (left) illustrates the general idea of Pooling layers. As we can see, the depth of the output volume is not changed but the 2D size of each depth slice is reduced significantly.

The implementation of Pooling Layer includes two parts: forward propagation and backward propagation. As for the forward propagation, we slide the kernel across each input image (in a batch) in non-overlapping way and pick the largest value as one element of output (shown in Figure 3 - right). Similarly, we implement the back propagation with the same four loops (including batch, output channels, rows and columns), but this time we need to use input of the layer to find the position of the largest value in the mask. The value in the array of gradients w.r.t. inputs is set to the corresponding gradient w.r.t. output if the position is the position of the largest value in the mask in forward propagation. Otherwise the value in the array of gradients w.r.t. inputs is set to 0.

PROCEDURES	CORRELATE2D	IM2COL
CONV-FPROP	1.036e-04 s	9.759e-05 s
CONV-BPROP	7.862e-05 s	6.352e-05 s
CONV-GRADS WRT PARAM	9.691e-05 s	8.429e-06 s
MAXPOOLING-FPROP	4.491e-04 s	9.676e-05 s
MAXPOOLING-BPROP	8.005e-04 s	1.794e-04 s

Table 1. Computational efficiency of two functions to implement convolution: 'correlate2d' and 'im2col'. The runtime is measured on the same machine to keep the same running speed. In order to make the runtime more stable and reliable, we pick the average value by running forward-propagation, backward-propagation and parameter update 20k times.

The experiments on the computational efficiency of different methods are carried on here. In addition to the method mentioned above (i.e. 'scipy.signal.correlate2d'), there is another method called 'serialisation' (or 'im2col') to implement convolutional layer. We recorded the running time of each forward propagation, backward propagation and gradients w.r.t. parameters calculation (see Table 1). As shown in Table 1, the method of 'serialisation' requires less runtime than 'correlate2d', especially the computation of gradients w.r.t. parameters in convolutional layer. This is because 'serialisation' method stretches each convolutional operation. Therefore, the convolution operation is turned into a single matrix multiplication, which can achieve higher computational efficiency than the many small matrix multiplications that a naive implementation would have (Vasudevan et al., 2017). However, the disadvantage of this approach is that the resultant matrix has repeated elements, and be large (dependent on the number of feature map, batch size, and image size).

We borrow this idea and apply 'im2col' function to the Max-Pooling Layer as well. As shown in the last two rows in Table 1, it does also dramatically improve the computational efficiency of Pooling Layer.

3. Context in convolutional networks

This section introduces how four dimensional reduction methods work: Max Pooling, Average Pooling, Strided Convolution and Dilated Convolution. we explore the difference among these methods by comparing the way of modelling context.

We have talked about Pooling in Section 2. Max Pooling is

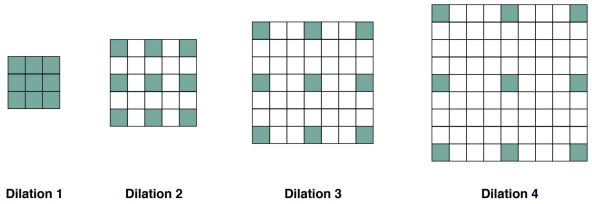

Figure 4. An illustration showing the effect of increasing the dilation of a 3 x 3 filter. For a given image patch, each coloured square corresponds to the locations at which the filter is placed. (Yu & Koltun, 2015)

the most common strategy which downsamples the input by only maintaining the largest value in the mask of filters (see Figure 3). Average Pooling has the exactly same mask but computes the output value by the mean of values in the mask. The drawback of pooling is losing resolution during downsampling. Therefore, Springenberg et al. (2014) suggests that instead of pooling layer, simply using convolutional layers with larger stride can also achieve the purpose of dimension reduction. The method is called Strided Convolution (SC) here. SC has the same receptive field size (kernel size) as standard convolution. With appropriate stride size, SC can cover all the information from last layer. However, SC requires parameters which Pooling does not require. With more memory cost, SC can only achieve the competitive performance compared to Max Pooling, rather than see a significant increase (Springenberg et al., 2014). Later, Yu & Koltun (2015) proposes another more powerful method call Dilated Convolution (DC) with larger receptive field keeping the number of parameters. As shown in Figure 4, this has the effect of increasing the receptive field of the filter as dilation is incremented. The stride is set to 1, so the operation does not lose any resolution.

The key research questions in this work are: 1) which dimensional reduction method works better on EMNIST dataset according to accuracy, runtime and memory requirement; 2) further investigate how Max Pooling and Dilated Convolution influence the performance; 3) Is there any improvement that can be made.

4. Experiments

This section covers all the experiments that we carried out to address the research questions mentioned in Section 3. The interpretation of experiment results is discussed in next section.

The baseline model in the experiments is a convolutional network with 4 convolutional layers with 64 filters and each convolutional layer (3x3 kernel, stride=1, padding=1) is followed by a ReLU layer and a Dimensional reduction layer. In each Convolutional layer, the kernel size is 3x3, stride is 1 and padding is 1. The hyperparameter setting of Convolutional layer keeps the output size the same as input size. For instance, in the first layer, the output size is (28-3+2x1)/1+1=28.

METHODS	OUTPUT SIZE	RUNTIME	ACCURACY
MAX POOLING	28-25-8-5-3	17.14s	87.63%
AVG POOLING	28-25-8-5-3	17.50s	88.05%
STRIDED CONV	28-14-7-4-2	20.60s	87.19%
DILATED CONV	28-26-22-16-8	62.23s	88.39%

Table 2. Comparison across 4 dimensional reduction methods. Column 2 represents the size of output of each dimensional reduction layer. Note that the runtime is the average time of 5 epochs which is measured on the same machine to make comparison reliable.

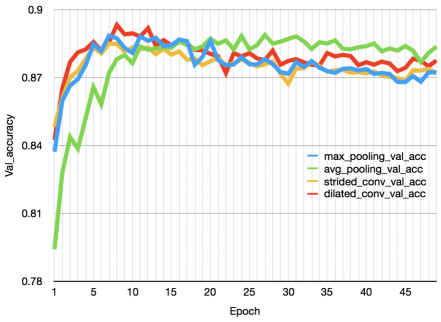

Figure 5. Learning Curve of the CNN with different dimensional reduction methods.

4.1. Comparison across 4 dimensional reduction methods

This experiment aims to explore how each dimensional reduction method influences the performance of a CNN by measuring and comparing their accuracy, runtime, memory requirement as well as feature map.

Hyperparameter settings of each method are as followed: Max Pooling (2x2 kernel, stride=2, padding=1), Average Pooling (2x2 kernel, stride=2, padding=1), Strided Convolution (3x3 kernel, stride=2, padding=1) and Dilated Convolution (3x3 kernel, stride=1, padding=1, dilate=i+1, where i refers to the *i*th layer). Note that due to the small size of input image (28x28), the kernel size, the stride parameter and the number of layers cannot be large. As illustrated in Table 2, Pooling requires less runtime with competitive accuracy and Dilated Convolution achieves the highest accuracy (88.39%). In addition, Figure 5 compares the speed of convergence, where Average Pooling is hardest to optimised and other three methods have similar convergence speed.

4.2. Influence of losing resolution of Max Pooling

Yu & Koltun (2015) mentions that Max Pooling has a significant drawback on losing the resolution. In the experiment, we aim to investigate how this disadvantage influence the performance of CNN.

# OF MP LAYERS	OUTPUT SIZE	RUNTIME	ACCURACY
NO MP LAYER	28-28-28-28-28	31.91s	88.27%
ONLY FIRST LAYER	28-15-15-15-15	27.59s	88.51%
FIRST 2 LAYERS	28-15-8-8-8	17.82s	88.34%
FIRST 3 LAYERS	28-15-8-5-5	17.04s	88.50%
FIRST 4 LAYERS	28-15-8-5-3	17.14s	87.63%

Table 3. Comparison across different number of Max Pooling Layers. Column 1 represents how many MP layers inserted in the CNN.

The simplest way to do the research is changing the kernel size of MP layer and comparing the performance, but the input image is too small to set kernel size to a large value (i.e. only [2x2] and [3x3] can be implemented). MP with 3x3 kernel size requires less runtime (11.32s) but achieves lower accuracy (87.14%), which means that losing resolution makes accuracy decrease and speed increase. However, only two candidates used to compare are not enough to draw a reliable conclusion, so we also devised another method: removing the different number of MP layers (see Table 3). The interpretation of the result is discussed in Section 5.2.

In addition to the influence on the whole CNN performance, we also investigate the influence on the feature map in each layer. We conduct the experiment by inserting only one MP layer at every position: after 1st, 2nd, 3rd and 4th Convolution layer respectively (see Table 4). The experiment is able to answer the question - which layer's feature map is influenced by Max Pooling the most. Furthermore, the question can be interpreted as a more high-level research topic - in the 4-layer CNN which layer's feature map is the most important (i.e. pixels/resolutions are indispensable in that layer). The motivation of the experiment derives from the uncertainty of whether the feature map close to the input or that far from the input matters more. The further discussion of the result is in Section 5.2.

POSITON OF MP LAYER	OUTPUT SIZE	ACCURACY
1ST LAYER	28-15-15-15-15	88.51%
2ND LAYER	28-28-15-15-15	88.77%
3RD LAYER	28-28-28-15-15	88.36%
4TH LAYER	28-28-28-28-15	88.15%
NO MP LAYER	28-28-28-28-28	88.27%

Table 4. Comparison CNNs with a Max Pooling layer at different position. For example, '1st layer' refers that the MP layer is inserted after the first Convolutional Layer.

4.3. Influence of dilate rate

In terms of the default hyperparameter settings of Dilated Convolution in the provided code, it is surprised that the dilate rate is set to i+2 rather than i+1 and a constant value, where i refers to the $(i+1)$th layer. Therefore, the purpose of the experiment is to investigate what the benefit of the setting and whether the constant dilate rate is better. Table 5 compares the performances of the CNN containing 4 DC layers with dilate rate 2, 3, 4, i+1, i+2. The interpretation is displayed in Section 5.3.

DILATE RATE	OUTPUT SIZE	RUNTIME	ACCURACY
[2, 3, 4, 5]	28-26-22-16-8	62.2s	88.39%
[1, 2, 3, 4]	28-28-26-22-16	70.5s	88.30%
[2, 2, 2, 2]	28-26-24-22-20	78.86s	88.92%
[3, 3, 3, 3]	28-24-20-16-12	61.97s	88.15%
[4, 4, 4, 4]	28-22-16-10-4	42.17s	87.58%

Table 5. Comparison the performances of Dilated Convolution with different dilate rate. The ith element in the bracket refers to the dilate rate in the ith layer.

4.4. Strided Convolution and Dilated Convolution used as feature extractors

Springenberg et al. (2014) mentions that instead of using Pooling Layer where some resolutions are lost, a standard Conv layer with larger stride (Strided Conv) is preferred to employed to extract features and reduce the dimension at the same time and a comparable performance can be achieved. The idea inspires the experiment, where we remove 4 standard Conv Layers from the original CNN, only remaining 4 Strided Conv layers or 4 Dilated Conv Layers. The aim is to see whether they can achieve the competitive performance. The result is shown in Table 6 and more explanation is delivered in Section 5.4.

METHODS	RUNTIME	ACCURACY
4 CONV LAYERS WITH MAX POOLING	17.12s	87.63%
4 STRIDED CONV LAYERS	7.51s	86.53%
4 DILATED CONV LAYERS	22.79s	87.92%

Table 6. Computational costs and classification accuracies of 4 Conv Layers with Max Pooling, 4 Strided Conv Layers and 4 Dilated Conv Layers.

4.5. Influence of increasing the number of filters

Increasing the number of filters in Conv Layers enables a CNN to extract more features so that the accuracy can be improved. However, there may be some problems such as time-consuming and overfitting. The motivation of the experiment is to investigate whether the performance of the models used in the previous experiments can be further improved by increasing the number of filters. Therefore, the experiment aims to investigate the impact of increasing the number of filters. Table 7 and Figure 7 display the performance and learning curves of 4 baseline systems with 128 filters instead. Further discussion shows in Section 5.5.

METHODS	RUNTIME	ACCURACY
MAX POOLING 128 FILTERS	44.14S	88.08%
AVG POOLING 128 FILTERS	45.11S	88.05%
STRIDED CONV 128 FILTERS	50.24S	87.77%
DILATED CONV 128 FILTERS	173.19S	88.49%

Table 7. Performances of 4 CNNs with increased number of filters (128).

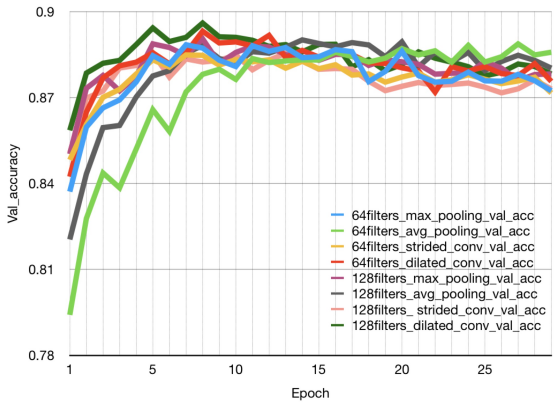

Figure 6. An illustration showing the effect of increasing the number of filters by learning curves.

5. Discussion

This section interprets the results of the experiments carried out in Section 4. Each subsection corresponds to one experiment in Section 4 with same subtitle.

5.1. Interpretation: comparison across 4 dimensional reduction methods

The reason why Pooling requires less time is that he operation of Pooling (max or average) is simpler and computationally cheaper than Convolution (discrete value integration). Another advantage of Pooling Layer is that it does not require parameters which means that it requires less memory cost. However, Max Pooling loses resolution which desires to be improved. Strided Convolution is one of the solution, but there is no increase in the accuracy (see Table 2), which is also mentioned by Springenberg et al.(2014). Therefore, for the 4-layer CNN on EMNIST dataset in the experiment, we think it is not worthy using Strided Convolution because it requires more computational and memory cost but does not give an significant improvement on accuracy. Another solution called Dilated Convolution achieves the highest accuracy with the same number of parameters (3x3 kernel) (i.e. the same memory cost). The reason is that DC enlarges the receptive fields in each layer, which means that each convolutional operation takes into account more context (Yu & Koltun, 2015). However, the speed of DC is 3 times slower than others. One reason is that the method does not reduce the size of feature map dramatically at the first several layers (because dilate rate is small at the beginning), which means that more computations are operated (shown in Column 2 in Table 2). Another reason is the special implementation in Pytorch, which can be probably improved in the future. When it comes to the difference between two pooling methods, Max Pooling seems to work as a feature selector which selects the most recognizable features while Average Pooling combines all features by mean value (which is similar to a linear transformation). Therefore, AP reserves all the information from previous layers but slightly increases the complexity of the model (Yu et al., 2014). Although MP loses resolution, it also helps to improve the problem of overfitting. As shown in Table 2, AP achieves higher accuracy than MP because AP contains more information, and the architecture is rather simple so we cannot see an obvious benefit of MP on combating overfitting.

5.2. Interpretation: influence of losing resolution of Max Pooling

In Table 3, the **runtime** becomes lower as the number of MP layers increases in general. This can be explained clearly by comparing the output size of each layer. More MP layers make the output size smaller, which leads to less computational cost. However, when the forth/last MP layer is added, the runtime increases surprisingly. This is because the last MP layer is redundant in the CNN architecture with adaptive pooling which is used to make sure output of conv layers is always (2, 2) spacially (helps with comparisons). There is no convolutional operation after the last MP layer, so the speed cannot be increased but decrease due to one more max pooling operation. In terms of **accuracy**, more MP layers make the model lose more information/resolution and as a result the accuracy decreases (shown in Table 3). However, the CNN without MP layers has lower accuracy. This is perhaps because CNN without MP layers are so complex that the model overfits the data; MP layer can help to simplify the model and avoid overfitting by reducing the parameters of CNN (Scherer et al., 2010).

In Table 4, MP makes the model become worse only when the MP layer inserted at the last layer, whereas other insertion positions lead to an overall improvement on the accuracy, which means that MP's benefit of combating overfitting beats the disadvantage of losing resolution. Inserting MP after the second Conv Layer is the best model to avoid overfitting and reduce the bad influence of losing resolution. The experiment implies that the influence of Max Pooling includes 2 aspects (Scherer et al., 2010): **1) losing resolution** (If a CNN loses the resolution of feature maps which are closer to the input (at the beginning), this will lead to the loss of the resolution in all the following layers; On the other hand, if a CNN downsamples the last feature map which has the most semantic information and the largest receptive field on the input image, the model also seems to lose too much information.) **2) improving overfitting** (If MP inserted to the last layer, the model may be still complex, which leads to overfitting.) In most case, the benefit of MP outweighs its drawback.

5.3. Interpretation: influence of dilate rate

As shown in Table 5 (Column 'output size'), DC layer with dilate rate [1,2,3,4] does not reduce dimensions/parameters. With more computational cost (or runtime), it has lower accuracy than the model with dilate rate [2,3,4,5]. Therefore, the latter one is chosen in provided code. When dilate rate is constant, dilate rate 2 gives the best accuracy but requires much high computational cost than the default setting (illustrated in Table 5). It is more reasonable if we comparing default setting (dilate rate = i+1) with the one whose dilate rate is 3, because they have similar runtimes. In this case, dilate rate $i + 2$ brings higher accuracy (88.39%). Overall, dilate rate $i + 2$ is better than constant values when their computational costs are similar. This is because constant values leads to an issue which is usually called 'gridding problem'. As depicted in Figure 7 (a), the pixels (marked in blue) contributing to the calculation of the center pixel (marked in red) through convolution layers are not consecutive (Wang et al., 2017). In order words, not all adjacent pixels contribute to the convolutional operation of the output pixel value (red). However, continuously increasing dilate rate from 1 to 4 is able to solve the problem.

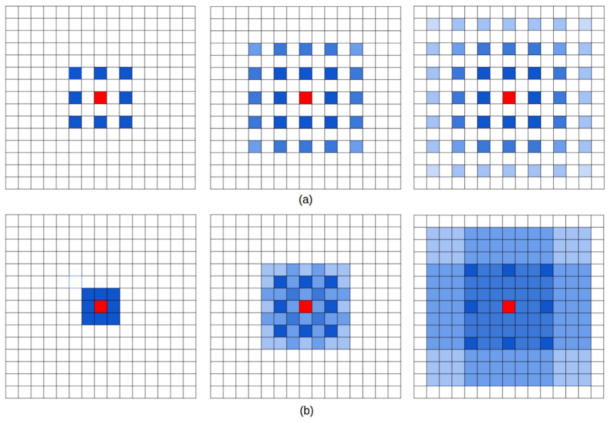

Figure 7. Illustration of the gridding problem. Left to right: the pixels (marked in blue) contributes to the calculation of the center pixel (marked in red) through three convolution layers with kernel size 3 x 3. (a) all convolutional layers have a dilation rate r = 2. (b) subsequent convolutional layers have dilation rates of r = 1, 2, 3, respectively. (Wang et al., 2017)

5.4. Interpretation: Strided Convolution and Dilated Convolution used as feature extractors

As depicted in Table 6, pure Strided Conv Layers has more than twice faster speed with slightly lower accuracy than standard Conv Layers with Max Pooling (baseline). The reason is rather simple: the former has less computation (output size of after each layer: [28-14-7-4-2]) than the latter [28-28-15-15-8-8-5-5-3]. Pure 4-layer Dilated Conv achieves higher accuracy (87.92%) with longer runtime than baseline (87.63%). Therefore, the pattern of CNN (Conv Layers followed by Max Pooling Layers) can be replaced by a simpler pattern which is standard Conv Layers with larger stride or dilate rate (Springenberg et al., 2014).

5.5. Interpretation: influence of increasing the number of filters

As shown in Figure 6, models with 128 filters has faster convergence no matter which dimensional reduction method is employed. When Table 2 and Table 7 are compared, we find that all the models with more filters achieves higher accuracy, which means that increasing number of filters does not lead to overfitting but improvement in the experiment. The number of filters is double but the runtime becomes more than twice, which means that increasing the number of filters dramatically increases the computational cost with slight improvement on accuracy (Zhang et al., 2016).

6. Conclusions

We implemented Conv Layer and Max Pooling Layer by two functions: 'correlate2d' and 'im2col'. The latter one has higher efficiency because the process is tuned to a single matrix multiplication (Vasudevan et al., 2017). Due to the expensive computation of CNNs, dimensional reduction layers are needed to reduce the parameters of CNNs. Max/Average Pooling layer can reduce the size of feature maps and also does not contain parameters, which makes the process computationally cheaper. Furthermore, it also has positive impact on combating overfitting (Scherer et al., 2010). However, the usage of Max Pooling brings an issue of losing resolution. In the experiment, we found that the benefit of Max Pooling outweight this drawback; 4-layer CNN with only one MP Layer after the second Conv Layer trained on the small dataset (EMNIST), can achieve higher accuracy (88.77%) than that with one MP Layer inserted at other position or 4 MP Layers. Instead of plugging a MP Layer after each Conv Layer by default, the research topic - how many to insert and where to insert, should be investigated in the future.

Due to the drawback of Pooling Layer, Springenberg et al. (2014) suggests that a preferred alternative to reduce parameters and avoid losing resolution is standard Convolution with larger stride (Strided Conv) which achieves comparable accuracy with Max Pooling. In addition, Dilated Convolution is able to enlarge the receptive field to increase the accuracy. Constant dilate rate leads to 'griding problem', so a continuously increasing rate is preferred to achieve higher accuracy (Wang et al., 2017). As SC and DC can work as both feature extractors and dimensional reduction methods, pure SC Layers and pure DC Layers architecture without standard Conv Layers can also achieve a competitive accuracy with less time, which is also mentioned by Springenberg et al. (2014). Lastly, when the number of filters increases to 128, as stated by Zhang et al. (2016), the accuracy increases at the cost of huge computational requirement. One more benefit of more filters we found is faster convergence speed.

References

Cohen, Gregory, Afshar, Saeed, Tapson, Jonathan, and van Schaik, André. EMNIST: an extension of MNIST to handwritten letters. *CoRR*, abs/1702.05373, 2017. URL http://arxiv.org/abs/1702.05373.

Scherer, Dominik, Müller, Andreas, and Behnke, Sven. Evaluation of pooling operations in convolutional architectures for object recognition. In *Artificial Neural Networks–ICANN 2010*, pp. 92–101. Springer, 2010.

Springenberg, Jost Tobias, Dosovitskiy, Alexey, Brox, Thomas, and Riedmiller, Martin A. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL http://arxiv.org/abs/1412.6806.

Stanford. CS231N 2.1 Convolutional Neural Networks: Architectures, Convolution / Pooling Layers, 2016.

Vasudevan, Aravind, Anderson, Andrew, and Gregg, David. Parallel multi channel convolution using general matrix multiplication. *CoRR*, abs/1704.04428, 2017. URL http://arxiv.org/abs/1704.04428.

Wang, Panqu, Chen, Pengfei, Yuan, Ye, Liu, Ding, Huang, Zehua, Hou, Xiaodi, and Cottrell, Garrison W. Understanding convolution for semantic segmentation. *CoRR*, abs/1702.08502, 2017. URL http://arxiv.org/abs/1702.08502.

Yu, Dingjun, Wang, Hanli, Chen, Peiqiu, and Wei, Zhihua. Mixed pooling for convolutional neural networks. In Miao, Duoqian, Pedrycz, Witold, ŚlÈl'zak, Dominik, Peters, Georg, Hu, Qinghua, and Wang, Ruizhi (eds.), *Rough Sets and Knowledge Technology*, pp. 364–375, Cham, 2014. Springer International Publishing.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015. URL http://arxiv.org/abs/1511.07122.

Zhang, Chiyuan, Bengio, Samy, Hardt, Moritz, Recht, Benjamin, and Vinyals, Oriol. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.