MLP Coursework 2: Exploring Convolutional Networks

s1678478

Abstract

In this coursework, we further explore the classification of images of handwritten digits using convolutional neural network(CNN). An extended version of MNIST database (EMNIST) is used here. The database contains handwritten letters, both upper and lower case, and handwritten digits. Several different dimension reduction methods are implemented on CNN and used to classify the handwritten letters and digits, including max-pooling, average-pooling, dilated convolution and strided convolution. In the first part of the coursework, the convolution and max-pooling layers are implemented within the framework. In the second part, three groups of experiments are designed and carried out to further explore the difference between the 4 dimension reduction methods. The highest accuracy achieved on testing set is reported by model with dilated convolution method applied which is 88.53%.

1. Introduction

All the experiments are carried out on the EMNIST Balanced dataset which contains 47 potential classes. There are 10 digits, 26 lower case letters, 26 upper case letters where the lower-case and upper-case labels are merged for the following letters: C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z. The conversion process used results in centred images of dimension 28*28. The EMNIST dataset is split into training set, validation set and test set which contain 100,000, 15,800 and 15,800 images respectively.

The traditional neural network layers use matrix multiplication by a matrix of parameters with a separate parameter discribing the interaction between each input unit and each output unit. Now we use convolution instead of general matrix multiplication and this structure is called convolutional neural network(CNN). (Goodfellow et al., 2016) The motivation for the coursework is to explore the method and advantages by using CNN to classify images of handwritten digits.

First of all, the implementation of convolutional network and pooling is addressed in section 2. Then in section 3, four different dimension reduction methods for CNN: max-pooling, average-pooling, dilated convolution and strided convolution are explained and discussed. Followed by section 4 where three sets of experiments are designed and carried out to explore the performance of using four different dimension reduction method with CNN. The general aim is to explore the difference and compare the performance among these four different methods.

The first research question is about how different number of filters affect the CNN with four different methods applied and how their performance are. The first set of experiments adjust the number of filters in the structure and left the other parameters as default. The second research question is about whether it is true that more layers used the better the performance of CNN. The second set of experiments are conducted by adjusting the number of layers with different method applied to CNN. The last question address the overfitting of the models, how the overfitting affect the performance of CNN. So in the third set of experiments, the weight decay coefficient is adjusted. In all experiments, only one parameter is altered, and others are left as default. The validation errors are plotted in graphs and the testing accuracy is reported in tables.

2. Implementing convolutional networks

2.1. convolution layer

The convolution layer perform convolution operations between the input matrix and the kernel matrix. For forward propagation, the input is a matrix of shape ($batch\_size(b)$, $num\_input\_channel(c_{in})$, $image\_height$, $image\_width$), the kernel has the shape of ($num\_output\_channel(c_{out})$, $num\_input\_channel(c_{in})$, $kernel\_height$,$kernel\_width$) and the returned output has the shape of ($batch\_size(b)$, $num\_output\_channel(c_{out})$, $output\_height$, $output\_width$). For example, for input shape of (2,4,7,7) which contains 2 batches of four 7×7 images, applying a kernel of shape (2,4,3,3), we can obtain the feature map of shape (2,2,5,5). If input $X \in R^{M \times N}$ and kernel $W \in R^{F \times F}$, the output H will have the size of

$$(M - F + 1) \times (N - F + 1)$$

as illustrated in Figure 1. The convolution is calculated by

$$H_{ij} = \Sigma_{k=1}^{F} \Sigma_{l=1}^{F} W_{k,n} X_{k+i-1,l+j-1} + b$$

In order to implement this convolution operation, we have two approaches. One is a serialisation approach uses a function called `im2col` and its reverse `col2im`, both are used to serialize the local receptive fields, turning the convolution operation into a matrix mlutiplication. Each 2D images in 4-dimensional input can be reshaped as a matrix with shape ($c_{in} \times kernel\_height \times kernel\_width, b \times output\_height \times output\_width$) using the function `im2col`.

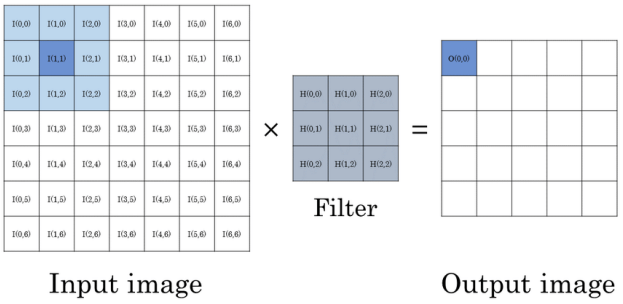

Figure 1. convolution with an input image of size 7×7 and a filter kernel of size- 3×3, the resulted output size is 5×5

The kernel is reshaped into 2D matrix with shape of $(c_{out}, c_{in} \times kernel\_height \times kernel\_width)$. Then the output matrix which is the dot product of the two 2D matrix can be computed. The corresponding feature map can be obtained by using `col2im` to transform the output matrix with biases added.

Another approach is to use the convolution function from `scipy.signal.convolved2d` which convolve a 2D image with a 2D kernel. This is the method implemented in the code. Here we do not transform the 4-dimensional input, we use the b and c_{in} as index of the 4-dimensional matrix to select the 2D matrix with the shape of $(input\_height, input\_width)$ and we use c_{out} and c_{in} to select the 2D kernel matrix with shape of $(kernel\_height, kernel\_width)$. Then we use `convolved2d` to compute the convolution between these two matrix directly. The results we get is filled in the corresponding output matrix of shape $(output\_height, output\_width)$ which is selected from the 4-dimensional output use index b and c_{out}. And we repeat doing this for each batch(b), input channels(c_{in}) and output channels(c_{out}) until we get all the corresponding output.

The foundation principle for the two approach is the same. The difference is that in `im2col` the dot product is computed and then transformed into 4-dimensional output using `col2im`. The disadvantage is that the output matrix has repeated elements and could be large. However in `convolved2d`, the convolution between two 2D matrix is computed directly.

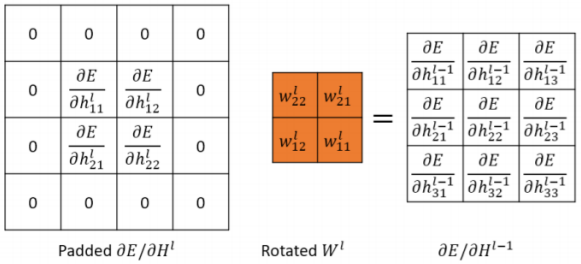

Figure 2. back propagation of convolution layer

As for the back propagation, similar convolution operation can be applied to the gradient of loss function w.r.t outputs and the kernel. However, before we perform convolution, we should pad the matrix of $\frac{\partial E}{\partial H^l}$ by a rate of $kernel\_size - 1$ and inverting the kernel. Then we can compute the convolution between the padded matrix and the inverted kernel which gives us the gradient of loss function w.r.t the previous layer(inputs) as shown in Figure 2. The convolution is conducted such as: $\frac{\partial E}{\partial h^{l-1}_{22}} = \frac{\partial E}{\partial h^l_{11}} w^l_{22} + \frac{\partial E}{\partial h^l_{12}} w^l_{21} + \frac{\partial E}{\partial h^l_{21}} w^l_{12} + \frac{\partial E}{\partial h^l_{22}} w^l_{11}$.

As for the gradients of E w.r.t parameters, kernel $W^l \in R^{R \times S}$ and output $H^l \in R^{M \times N}$, the equations are given as follow:

$$\frac{\partial E}{\partial w^l_{r,s}} = \Sigma^{M^l}_{m=1} \Sigma^{N^l}_{n=1} \frac{\partial E}{\partial H^l_{m,n}} h^{l-1}_{r+m-1,s+n-1}$$

$$\frac{\partial E}{\partial b^l} = \Sigma^{M^l}_{m=1} \Sigma^{N^l}_{n=1} \frac{\partial E}{\partial H^l_{m,n}}$$

Here the convolution is conducted between the input and the gradient w.r.t layer outputs to obtain the gradient w.r.t weights. The gradient w.r.t biases is the sum of the gradient w.r.t layer outputs.

2.2. maxpooling layer

The shape for the input, kernel and output are all the same as in the convolution layer. In forward propagation, instead of compute convolution ,we use `max` function $m = max(a, b)$. We simply take the largest value from the region sepecified by the kernel size within the input and form a condense output matrix. An example is shown in Figure 3 where the maxpooling kernel size is 2×2 and the stride is 2. In back propagation, a binary "mask" is created based on the derivative of max function:

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

As a result, in the mask, the maximum value in each subsection is replaced by "1" and everywhere else is "0". The the gradient w.r.t input can be computed which is the dot product of mask and the gradient w.r.t output.

3. Context in convolutional networks

In this coursework we explore four dimension reduction approaches which are `max-pooling, average-pooling, dilated convolution and strided convolution`. The pseudocode for pooling and convolution is given in Algorithm 1 and Algorithm 2.

The `strided convolution` is the foundation of all convolution networks. Stride(s) is the distance between spatial locations where the convolution kernel is applied. In the default scenario, the distance is 1 in each dimension. The s can be set as different values which controls how many

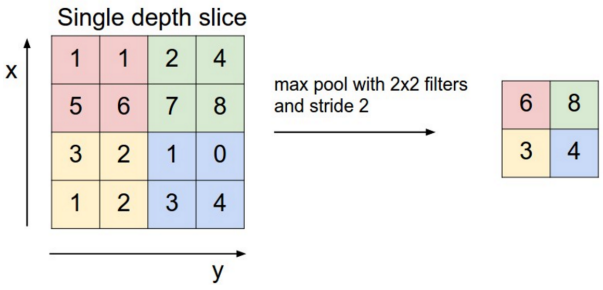

Figure 3. example of maxpooling

Algorithm 1 Forward Propagation of Convolution

given: input x, kernel k, padding p, stride s, biases b
initialize: output y of size: $\frac{size(x) - size(k) + 2 \times p}{s} + 1$

$x_0 = pad(x)$
if $size(x_0) > size(k)$ **then**
  repeat
    $\hat{x_0} = subsection(x_0, k)$
    $\hat{k} = Transform(k)$
    $y = Convolve2D(\hat{x_0}, \hat{k}) + b$
  until all $\hat{x}$ in x has been processed
end if
return y

steps the kernel needs to move until the next convolution is performed. The larger the s, the smaller the size of the output. The formula to calculated the output size is given in Algorithm 1.

In `dilated convolution` we use stride=1, however the kernel is expanded by a dilated rate(D). As shown in Figure 4. The original kernel size in (a) is 3×3, after the kernel expand by D=2 in (b) and D=4 in (c), the kernel still takes 9 inputs shown as red points in the figure. The dilated rate can be seen as the distance between the kernel input, it is obvious that by expanding the kernel we have a larger reception field. Compare with the traditional convolution where the reception field grows at a linear rate, the reception field here grows at an exponential rate. The advantage of `strided convolution` is that the reception field is expanded and each convolution output contains a wider range of original information. (Fisher Yu)

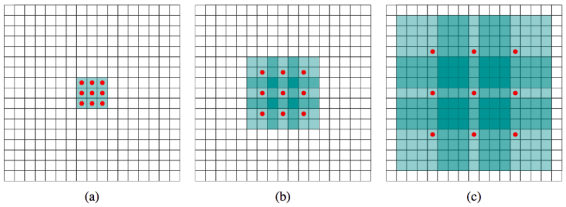

Figure 4. example of strided convolution

Algorithm 2 pooling

given: input x, kernel k, padding p, stride s, pooling-Function F
initialize: output y of size: $\frac{size(x) - size(k) + 2 \times p}{s} + 1$

$x_0 = pad(x)$
if $size(x_0) > size(k)$ **then**
  repeat
    $\hat{x_0} = subsection(x_0, k)$.
    $y = F(\hat{x_0})$
  until all $\hat{x}$ in x has been processed
end if
return y

By using pooling method, we can down-sample an input representation (image, hidden-layer output matrix, etc.) by applying a specific filter to non-overlapping sub-regions within the input. This allow us to reduce the computational cost by reducing the number of parameters to learn and also reduce the probability for overfitting. The pooling layers generalize the results from a convolutional filter - making the detection of features invariant to scale or orientation changes. The max-pooling method applies a max filter to the input which pick the maximum value in the specific sub-region and the average-pooling return the average of the specific sub-region. Max-pooling can select the most distinguishable feature from the feature map and thus preserve the information of image structure. Average-pooling perform the subsampling based on all of the features and preserve the image background information better. In the image classification application, we prefer max-pooling because it can learn more about the texture structure and edges of the image where average-pooling can not. Max-pooling is used to minimize the estimate variance while average-pooling minimize the shift of estimate variance.

In the following section, three research questions are explored by altering only one parameter in each set of experiments. The aim is to compare the difference among the 4 different dimension reduction method and observe how each parameter affect the performance of CNN with 4 different method applied. The parameters being investigated in 3 sets of experiment are listed as follow:

- number of filter
- number of layer
- weight decay coefficient

4. Experiments

For all the experiment conducted, only the specified parameter is altered and others are left as default. The baseline model for all experiments uses the following parameters: $num\_filter = 64$, $num\_layer = 4$, $dim\_reduction\_type =$ 'maxpooling', $batch\_size = 100$, $weight\_decay\_coefficient = 1 \times 10^{-5}$. The accuracy of the baseline model on testing data is 87.68%.

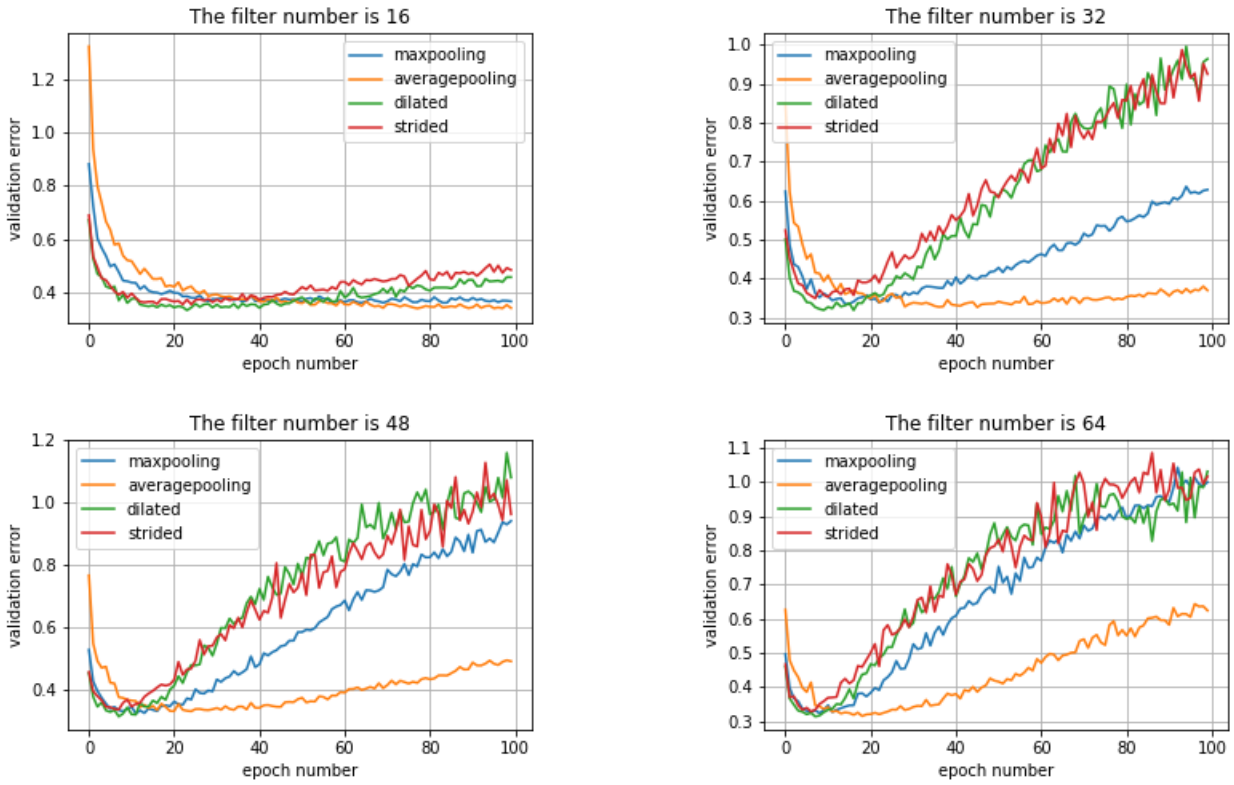

Figure 5. Different number of filters for four dimension reduction method

4.1. number of filter

In the first part, total of 16 experiments are conducted for 4 different methods each with 4 different value of *num_filters*. The *num_filters* we choose is 16, 32, 48 and 64. The validation error for all experiments are plotted in Figure 5. The number of filters determines the number of output channels. The more filters we use the faster the model learns and longer the running time. Both strided and dilated convolution finish the learning process around 15 epochs and the error keeps increasing in the following epochs which indicate severe overfitting. The CNN with average-pooling is the slowest learner. The test accuracies are listed in Table 1. For all values of *num_filters*, the accuracy of `dilated convolution` CNN is the highest and the highest accuracy among all 16 experiments is 88.41% which occurs when dilated convolution CNN uses 64 filters.

NUM_FILTER	MAX	AVERAGE	DILATED	STRIDED
16	86.85%	87.24%	87.40%	86.62%
32	87.63%	87.37%	88.10%	87.15%
48	87.48%	87.64%	88.15%	87.41%
64	87.68%	87.38%	88.41%	87.18%

Table 1. Classification accuracy on testing sets for 4 models each with 4 different filter number

4.2. number of layer

In the second part, total of 16 experiments are conducted. The different settings we choose are 1, 2, 3 and 4 layers. The number of layers determines how many times of convolution are conducted. The more layers in the model, the longer the running time and there are obvious increase in accuracy until number is 3. For all 4 method, the highest accuracy occurs at number of layers is 3, then there is a drop in accuracy when number of layers increases to 4. The validation error is plotted in Figure 2. We can see as the number of layer grows, the faster the model learns. When we use only one layer, for all methods, the learning processes are not completed and When we have 2 layers all models finish learning within 100 epochs except for `average-pooling`, for which the error is still decreasing. From Table 2, there is a big impact on accuracy for each method when the number of layers increase from 1 to 2, especially for `maxpooling` and `average-pooling`. For dilated convolution the inmprovement in accuracy is the least obvious since the model starts with a relative high accuracy. And model provides the highest accuracy is still the dilated convolution which is 88.53% when 3 layers are used.

4.3. weight decay coefficient

The value we choose for the weight decay coefficients are 1×10^{-4}, 1×10^{-5} and 1×10^{-6}. Total of 12 experiments are conducted. Weight decay specifies regularization in the

NUM_LAYER	MAX	AVERAGE	DILATED	STRIDED
1	76.92%	75.36%	88.32%	86.92%
2	87.07%	87.50%	88.51%	87.63%
3	87.88%	88.34%	88.53%	87.56%
4	87.68%	87.38%	88.41%	87.18%

Table 2. Classification accuracy on testing sets for 4 models each with 4 different layer number

neural network. During training, a regularization term is added to the network's loss to compute the back propagation gradient. The weight decay value determines how dominant this regularization term will be in the gradient computation. The more training examples provided, the weaker this term should be. The more parameters the net work contains the higher this term should be. So, Weight decay is a regularization term that penalizes big weights. When the weight decay coefficient is big, the penalty for big weights is also big, when it is small weights can freely grow. The Figure 7 shows the validation error of each different dimension reduction method with 3 different weight decay coefficients. It is clear that when the coefficient is the smallest, 1×10^{-6}, the error is the highest and the overfitting issue is the most severe. When the coefficient is the largest, all model shows significant improvement regarding the error compared to the other two value of weight decay coefficients. From Table 3 we can see the model gives the highest accuracy is still dilated convolution.

WD_COEFFICIENT	MAX	AVERAGE	DILATED	STRIDED
1E-4	87.66%	88.40%	88.09%	87.59%
1E-5	87.68%	87.38%	88.41%	87.18%
1E-6	87.72%	88.20%	88.19%	87.77%

Table 3. Classification accuracy on testing sets for 4 models each with 4 different weight decay coefficients

5. Discussion

Regarding the first research question, the more filters used, longer the running time. The slope of the increase in accuracy is slowing down while the number of layers increases. `Maxpooling` and `dilated convolution` reported the highest accuracy compared to themselves when 64 filters are used. `Averagepooling` and `strided convolution` reported the highest accuracy compared to themselves when 48 filters are used. The overall highest accuracy is reported by model with `dilated convolution` applied which achieves 88.41%.

Regarding the second research question, the more layers used, the longer the running time. When we change the number of layers from 1 to 3, the more layers we use, the higher the accuracy. `Maxpooling` and `Averagepooling` shows a near 10% increase increase when the number of

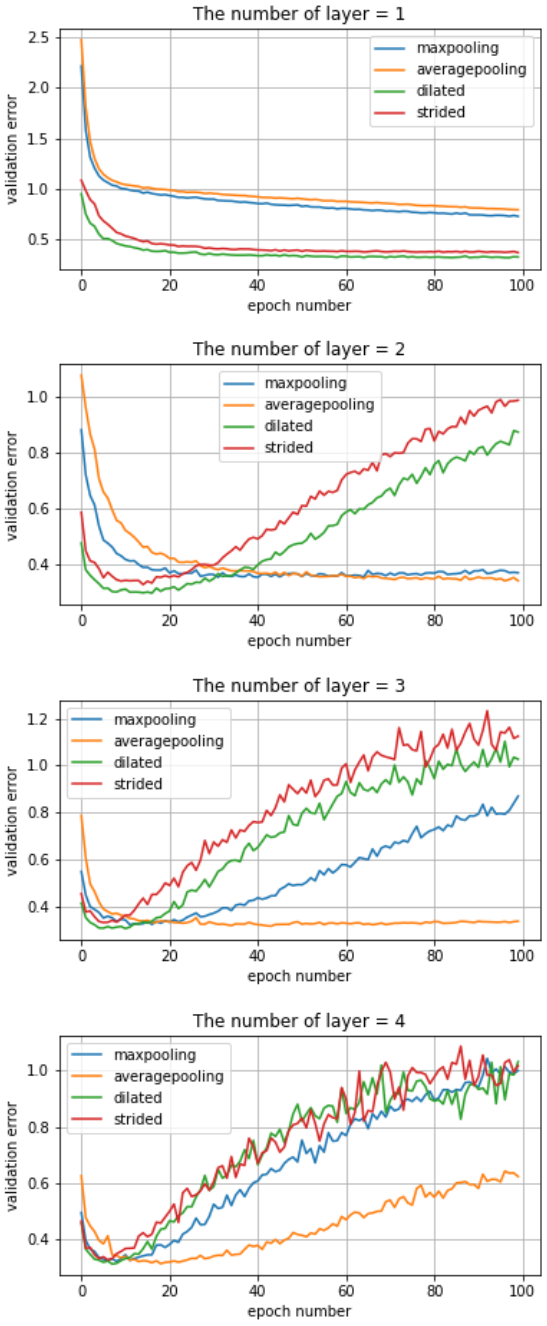

Figure 6. Different number of layers for CNN with four different dimension reduction method

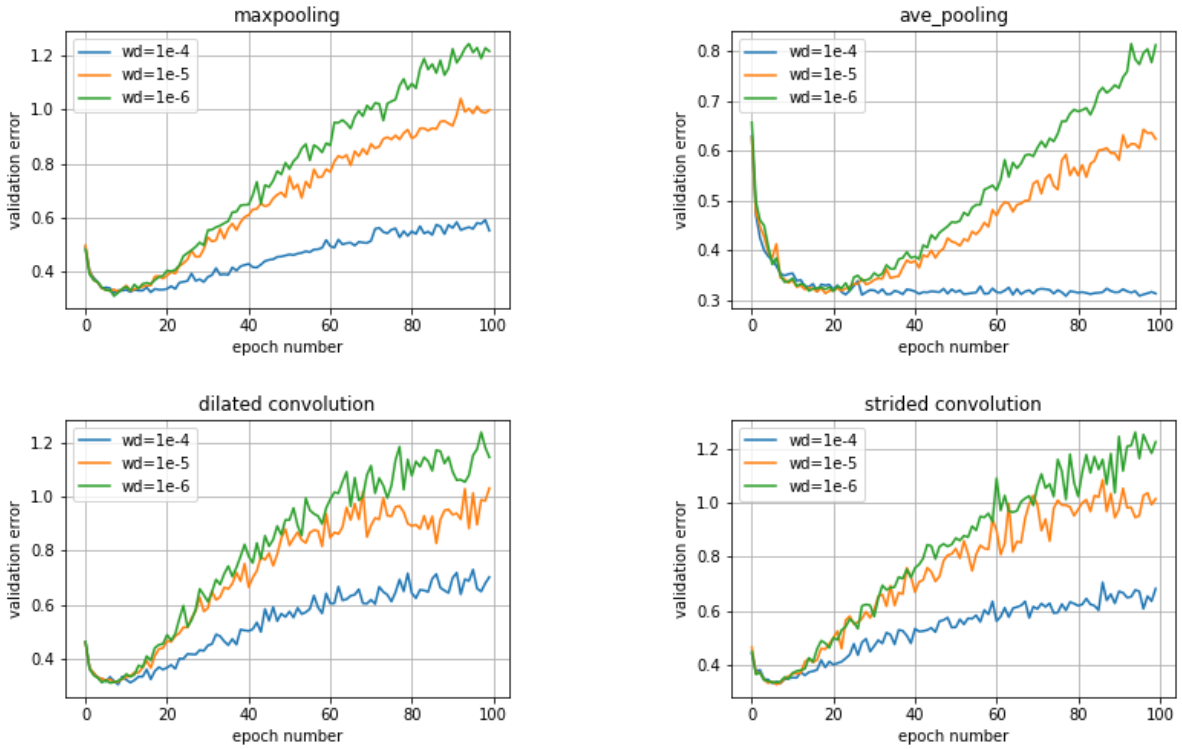

Figure 7. Different weight decay coefficient for four dimension reduction method

layers is raised from 1 to 2. However, when the number of layers is 4, all models witness a drop in accuracy. This shows that not the more layers we use the better. When more convolution operations are conducted at the beginning of a learning process the more accurate the models are and when enough layers are used, the additional convolution layer might cause extra information loss during the convolution process and thus result in decrease in accuracy. The highest accuracy reported among all models is `dilated convolution`, 88.53%, when umber of layers is 3.

Regarding the third research question, the change in weight decay coefficient has an obvious influence on the error curve. The larger the value, the lower the error rate. As for the accuracy of different models, there is no big changes, this might because that the model with the best performance during the training process is selected and then tested on the testing set, so the overfitting afterwards does not affect the best model and its accuracy. The highest accuracy reported among all models is `dilated convolution`, 88.41%, when the default settings are used which is the same as the model in the first set of experiment.

6. Conclusions

Overall, the models with `dilated convolution` takes the longest time to run and also achieve the highest accuracy compared with other 3 methods. The performance of `dilated convolution` model is better than all other models within all experiments and the highest accuracy reported in this coursework is 88.53% where the settings are listed as follow: $num\_filter = 64$, $num\_layer = 3$, $dim\_reduction\_type$ = 'dilated convolution', $batch\_size = 100$, $weight\_decay\_coefficient = 1 \times 10^{-5}$. The baseline accuracy is 87.68% and for the majority experiments of `Averagepooling` and `strided convolution`, the performance are below the baseline. Thus, we might say, in most cases, models using `Averagepooling` and `dilated convolution` are better options.

The future research work can investigate what kind of applications are the most suitable for each of the dimension reduction method and what would it be if the pooling methods are combined with the `dilated convolution`, would there be further increase in the accuracy achieved.

References

Fisher Yu, MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS, 2016.

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep learning*. The MIT Press, 2016.