MLP Coursework 2: Exploring Convolutional Networks

s1853160

Abstract

Convolutional neural network (CNN) can be used to deal with many tasks of computer vision and natural language process. The experimental practice is an appropriate way to check if you really understand the knowledge and also to deepen your comprehension for the theory. This report tries to implement some import process on convolutional layer and the max-pooling layer of CNN and also come up with several exploratory questions, including the difference between max pooling and average pooling, effects of the number of layers, etc., and explore it through experiment. All the experiments are carried out on the EMNIST dataset which classic hand-writing digits dataset. The experiments show both max-pooling and average-pooling can work well on this data set and the number of layers has a little effect on the performances, but if the number of layers is set to a large number, it is easy to cause the model fail. In addition, this report also does some analysis about dilation, striding, and the results are detailed in section 4 and 5.

1. Introduction

Convolutional Neural Network (CNN, or ConvNet) is a kind of feedforward Neural networks, within range of its artificial neuron can be part of the response of the surrounding units, great performance for large image processing. A Convolutional neural network includes many parts, including Convolutional layer, ReLU layer, Pooling layer, Fully connected layer, Loss of layer.

The EMNIST is a set of hand-writing character digits including a huge number of datapoints. In this experiment, the data set was divided into three part: the training set(100000 data points), the validation set(15800 data points) and the testing set(15800 data points). Because there are 15 letters for which it is confusing to discriminate between uppercase and lower-case versions so we use a reduced label set of 47 different labels

The experiment is one of the best ways to check whether you have understood the knowledge. It's also one of the best ways to impress and reinforce your knowledge. For this purpose, this report would like to implement some important processes of the convolutional neural network, including forward propagation, back propagation, updating gradients with respected to parameters of convolutional layers and max pooling layer. This report also tries to come up with several research questions based on references and design some experiment to explore these questions.

There are four key research questions this report is going to explore. The first question is does the Max-pooling perform better than Average-pooling? The second question is how does the number of layers affect the model? And whether the overlapping pooling will improve the performances? What happens if the stride increases? These questions will be discussed in detail in section 4 and 5.

2. Implementing convolutional networks

2.1. Convolutional layers

Conventional neural networks do not scale well to the full image. In this experiment, the input of this report is four-dimensional input (100, 1, 28, 28), a batch contains 100 data, and each data is a picture of 28 by 28 by 1 (channel). If the experiment needs to use larger color images as input, such as 200 x 200 x 3 (RGB color channel), this will result in neurons with 200 x 200 x 3+1 (bias) = 120001 weights. Obviously, this is a waste of computing resources, and too many parameters can easily lead to overfitting. An important property of convolutional neural network is the weight sharing of convolution layer, which greatly reduces the number of weights.

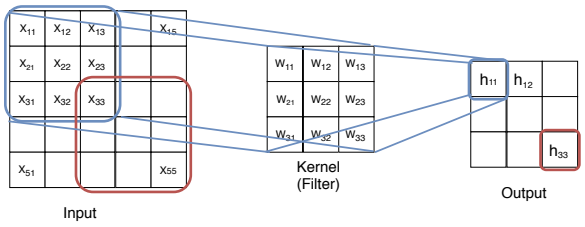

Figure 1. Convolutional layer process

Figure 1 illustrates an example of a convolution operation on a two-dimensional tensor. For the sake of brevity, here is a convolution of a channel, in fact the depth of the kernel should be the same as the depth of the input (Goodfellow et al., 2016).

$$h_{11} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j} + b$$

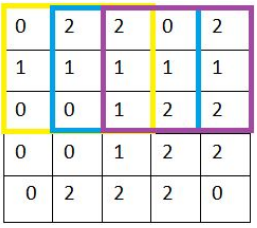

Figure 2. convolution 1-3

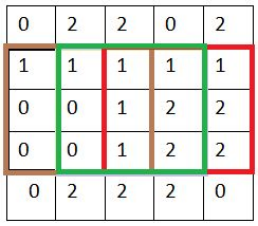

Figure 3. convolution 4-6

Figure 2 & 3 show how the kernel moves on the original input. If we set stride=1, when calculating h_{11}, the convolution kernel is convoluted with the contents of the yellow box. The next step is to take the blue box and do the convolution to get h_{12}.When the operation of Figure 2 is completed, the convolution kernel moves down 1 stride and repeats the previous operation, as shown in Figure 3. And the rest can be done in the same manner.

When doing **implementation** of convolutional layers, actually, in Python, the packages, ***scipy***, already provides functions (**scipy.signal.convolve2d()** or **scipy.signal.correlate2d()**) for convolution, and the difference between two is whether kernels flip. Because this function is only able to operate the convolution of 2-D tensors, for the case of 4-D tensors, we should use for loop to over each datapoint of minibatches, each output channel and each input channel.

2.2. Pooling layers

The convolution layer is responsible for feature extraction and the pooling layer is responsible for feature selection. The common methods are Max-Pooling and Average-Pooling. "Because images have the "stationarity" property, which implies that features that are useful in one region are also likely to be useful for other regions. Thus, to describe a large image, one natural approach is to aggregate statistics of these features at various locations. For example, one could compute the mean (or max) value of a particular feature over a region of the image" (Ng et al., 2010).Pooling layers can also reduce over-fitting.

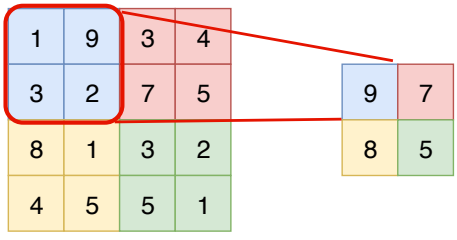

Figure 4. Pooling layer process (f=2, s=2)

Figure 4 shows an example of using maxpooling. In general, there are two parameters for the pooling layer, and f (usually taken as 2) denotes the size of the scanning area and s (usually taken as 2) denotes the stride. The parameter f in non-overlap pooling should be equal to s.

If we use for loop to implement the pooling layers, we need to loop over each datapoint, input channel, height and width of the image, because the numer of input channel is equal to the number of output channel. Then as shown in figure 4, find the max (MaxPooling) or mean (AveragePooling) of the corresponding region. For the MaxPooling, we can store the index corresponding to the maximum value so as to facilitate the calculation of back-propagation of the pooling layer.

2.3. Compare other approaches to implementing the convolution layer

Another approach, named "Serialisation", can also implement the convolution operation, and this method uses a function called **im2col** (reverse **col2im**). This approach completely uses matrix operations, and a common sense is that vectorised operations are generally much faster than loops (See the simple experiment of Appendix 1). However, this approach requires input all the matrix at the same time and and reconstruct them and kernels to two large matrix (dependent on the number of batch size, feature map and image size), then apply the matrix operation on these two matrices. Thus, this approach is more demanding on space. While the ordinary method only operate on the small matrices in each loop, so it requires less storage demands.

3. Context in convolutional networks

There are four approaches, MaxPooling, AveragePooling, Strided Convolution and Dilated Convolution, were explored in the section 4. This report has introduced the pooling in the previous section, so this section is more focused on the other two approaches, then outline research questions.

3.1. Pooling Layers

Pooling can be different types, as we have introduced before. Pooling can reduce the dimensionality (downsampling) of each feature map while maintaining important information of features. The reducing of parameters not only improves the running efficiency but reduces over-fitting. Pooling increase the receptive field of kernels. A very important property of pooling is invariance that means even after a small shift, the image still produces the same (pooled) features.In many tasks (such as object detection and sound recognition), we prefer to get the features with translation invariability, because the mark of the sample (image) remains the same even if the image is shifted.

3.2. Strided Convolutions

Basically, String approach (strided convolution) work a similar pattern as convolutional layers. But the stride of the strided convolution is set the parameter of stride can be set, 3 , etc., so this method works as downsampling.

With the increasing of stride, the size of feature ma decrease faster. Unlike pooling, if the stride is sma than the kernel size, there may be no information l The striding and the dilating are derived from the l convolution, so they also have weights need to lea thus, applying striding and dilating might lead to r complexity for the model.

3.3. Dilated Convolutions

In standard convolution operations, the elements o convolution kernel are adjacent to each other.But, dilated convolution, the elements of the convolutic nel are spaced.

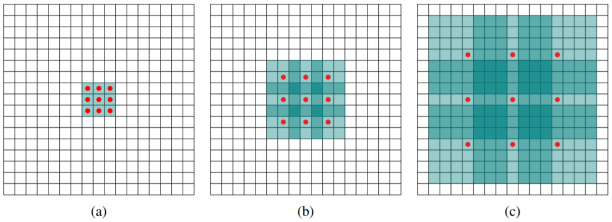

Figure 5. Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage.

The figure 5 shows how the dilated works. As the depth of the layer increases, the receptive field of the dilated convolution increases exponentially.Increased receptive field by inflating the kernel by inserting D - 1 spaces between the kernel elements (Such as dilation 2: inserting (2-1) space between elements). Unlike pooling layer, if the pooling layer is added to the network, it will lead to the loss of information and reduction of pixel-level accuracy. However, the dilated convolution integrates multi-scale contextual information without loss of resolution or the need to rescale images. This module is specially designed for dense prediction. (Yu & Koltun, 2015)

3.4. Key Research Questions

In the next section, we will firstly discuss the difference between max pooling and average pooling. Then this report will explore how the number of layers affect performance. And This report will discuss overlapping pooling. We also want to know what if the stride of models increases.

4. Experiments

The following experiments of this report is based on the **pytorch** framework on Python, which is one of the popular deep learning framework. To make it easy for the reader to get some of the same results, here are some of the default hyperparameter setting for this experiment (table 1). Other hyperparameters are explained in detail in each particular comparison experiment. The following experiments are carried out on the Google Cloud Platform with Nvidia Tesla P100 GPU.

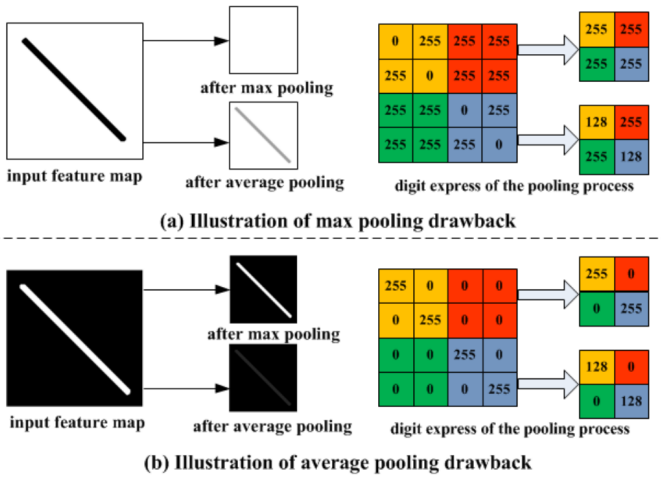

Figure 6. Example illustrating the drawbacks of max pooling and average pooling (Figure from (Yu et al., 2014))

PARAMETER	DEFAULT VALUE
BATCH_SIZE	100
SEED	0
IMAGE_NUM_CHANNELS	1
NUM_LAYERS	4
IMAGE_HEIGHT	28
IMAGE_WIDTH	28
WEIGHT_DECAY_COEF	10^{-5}
NUM_EPOCHS	100
CONVOLUTIONAL KERNEL SIZE	3
NUM_FILTERS	64

Table 1. Default parameters

The framework of this experiment regards convolution layer and dimensional reduction layer (Pooling layer) as one layer. The outputs of the convolution layer are applied ReLU.

4.1. Does the Max-pooling perform better than Average-pooling?

For a data set of black and white images, Different results may be generated for max pooling and avg pooling when the same content is represented by different data, because both approaches have its drawback(Yu et al., 2014) (see figure 6). Max pooling is able to select superior invariant features, which consequently leads to faster convergence and improves generalisation performance (Nagi et al., 2011). Hence, this report is going to explore the differences in performance between two approaches in terms of accuracy of predictions, generalisation, etc.

Since this part is going to compare the difference between max-pooling and average-pooling, so this part only change dimensional reduction type, and other parameters are set as default.

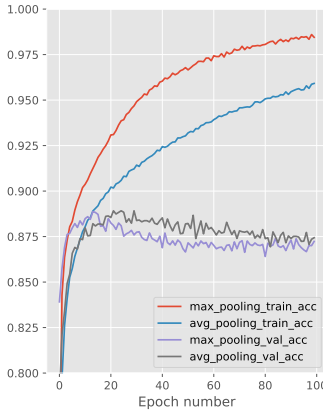

Figure 7. Accuracy of different models

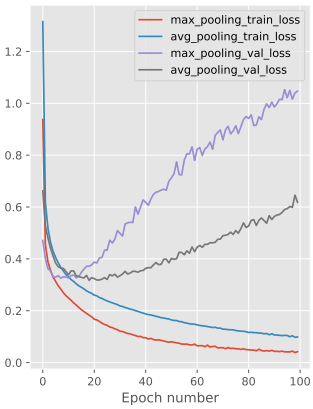

Figure 8. Error of different models

The figure 7 shows there are the same increasing tendency on the accuracy of training set between max pooling and average pooling, and the accuracy of max pooling on training data is higher than the average pooling (approximately 0.025). But if you look at the results on the validation set, it seems that the performance of max pooling is even worse. This. Combined with the results in figure 8, it can be seen that max pooling is easier to overfit.

Model	Accuracy	Loss	Time	Memory
Max pooling	0.882468354	0.35714045	212.16 it/s	845MiB
Average pooling	0.877151899	0.35369727	198.42 it/s	817MiB

Table 2. Performance results on test set

As shown in the above table 2, the accuracy of maxpooling in the test set is slightly higher than the average pooling. There is almost no difference on loss, which means the generalizations of the two models may be similar.

4.2. How does the number of layers affect the model?

Article, Network in Network, points out that the deeper the layer number is, the better the abstraction ability is (Lin et al., 2013). While maybe approximately 5 layers is enough to learn a decent expression of images. Because a research result shows that direct use of features extracted from the OverFeat network can produce a good result (Sharif Razavian et al., 2014).

In neural networks, the depth of the model has an effect on the results.Therefore, this part hopes to explore whether the deeper the model is, the better the result of the model will be. This part compares the influence of the number of model layers, so we set the number of layers in the model to 2, 3, 4, 5, 6, and 8. The other parameters are set to default. In addition, the default hyper-parameters of strding are kernel size=3, stride=2, pading=1 and the default hyper-parameters of dilation are kernel size=3, padding=1, stride=1, dilation=i+2 (i: 0,1,2,...). Because convolution kernel of dilation size increases, this method does not have the result of higher layers Numbers

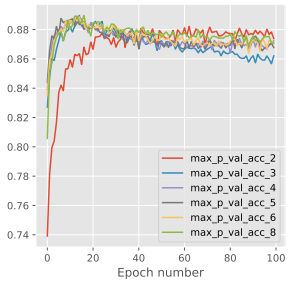

Figure 9. Results of different layers(maxpooling)

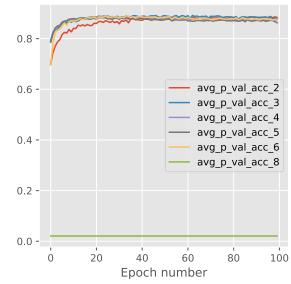

Figure 10. Results of different layers(avgpooling)

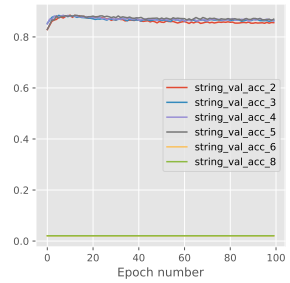

Figure 11. Results of different layers(striding)

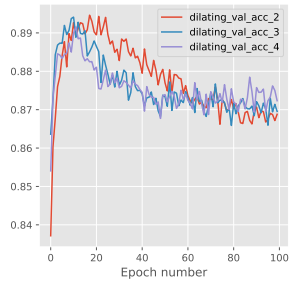

Figure 12. Results of different layers(dilation)

For the first model (figure 9), the result of 2 layers is slightly worse than the others. For the rest of the models, the results seem similar, and on this framework, adding layers doesn't seem to improve performance much. It is worth noting that figures 10 and 11 have some models that are not being learned when the number of layers is large.

Table 5 (See appendix B) shows that the running time of the model increases with the number of layers (depth) of the model and the dilation takes more time than the other methods. From the perspective of test set loss, increasing the number of layers does not seem to significantly improve the generalization, and sometimes even reduce the ability of generaliztion.

4.3. Whether the overlapping pooling will improve the performances?

Overlapping pooling is a special application pooling approach. This method makes the stride of the pooling layer smaller than the pooling size, which causes overlapping parts in the pooling process. The results show that this operation can improve the performance of the model to some extent (Krizhevsky et al., 2012).

Therefore, This report would like to explore whether such improvement really exist. For the comparison, the results on experiment 1 will be treated as the baseline in order to observe how many promotions there are. In this experiment, almost all the parameters settings are the same as settings in experiment 1. We also adopt 4 layers and 64 kernels. In order to implement the overlapping pooling, the stride of pooling is set to 1 on account of the pooling size of 2.

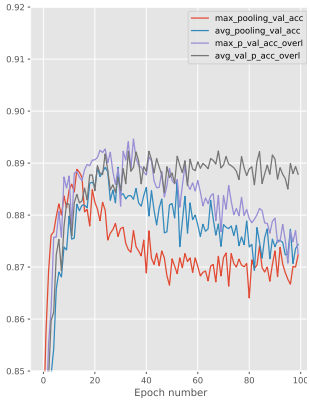

Figure 13. Accuracy of different models

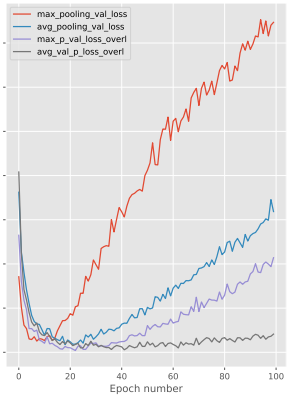

Figure 14. Error of different models

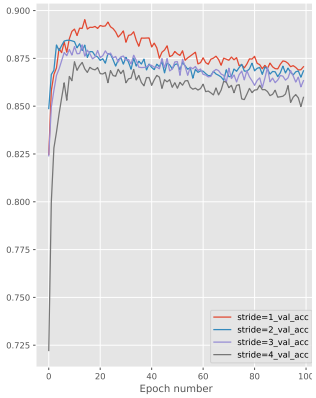

Figure 15. Accuracy of different stride(striding)

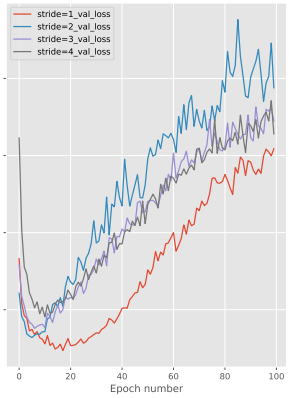

Figure 16. Error of different stride(striding)

In this case, compare with the baseline, both max pooling and average pooling applying overlapping method get higher score on accuracy rate(Black line and purple line), and reduce the overfitting, especially average pooling (Black line).

Model	Accuracy	Loss	Time	Memory
Max pooling	0.885696203	0.34075135	71.44 it/s	1233MiB
Average pooling	0.884303797	0.34159914	65.19 it/s	1049MiB

Table 3. Performance results on test set(Using Overlapping)

The results of the test set are consistent with the results of the verification set, with higher accuracy and lower loss. Meanwhile, the results show that the max pooling should be slightly better and the average pooling, which is also in line with the empiricism.

4.4. What happens if the stride increases?

In the last question, the overlapping approach is actually the reduction of the stride. Moreover, it can be seen that reducing the stride of the model actually has an effect on the model, so this part hopes to explore the impact of the increase in the stride of models.

In this part, strding is used as the method of dimensional reduction. Only change the stride (from 1 to 4) of this method to observe the transformation of the model performance. In this method, the size of kernels is 3 by 3, and padding is 1.

As it can be seen from figure 15, when the stride decreases, the performance of the model conforms to the results in the previous section, for example, stride=1. While the performance of stride=2 and stride=3 is roughly the same, slightly exceeding 0.875. As the stride changes from 1 to 4, the overall performance of the model decreases and the model of stride=4 has lowest accuracy.

Model	Accuracy	Loss	Time	Memory
Stride=1	0.88556962	0.32997024	54.83 it/s	1093MiB
Stride=2	0.877278481	0.3717412	147.56 it/s	739MiB
Stride=3	0.873607595	0.4062894	198.50 it/s	733MiB
Stride=4	0.862658228	0.41233927	223.02 it/s	699MiB

Table 4. Performance results on test set(Striding)

The test set data shows the same results as on the validation set. As the stride increases, the performance decreases. When stride=1, the model has the best accuracy (0.8856) and generalization ability (0.33).

5. Discussion

In the experiment 1, the validation set shows that the average pooling has better results, however, the testing set shows that the max pooling is slightly better. This may be because the merits of both approaches now depend largely on empiricism. Although both approaches work well for some data sets, it is still difficult to define which approach can do better for a new problem (Yu et al., 2014). The experiment also shows that the convergence speed of Max pooling is slightly faster. This is because max pooling is a way to select important features, so it is able to select superior invariant features which consequently leads to faster convergence (Nagi et al., 2011).

For the question 2 about the number of layers, most of results show that, instead of getting better model with increasing depth, the performances of some models will.

Although models are deeper and are more likely to express information better (Lin et al., 2013), this idea does not work in this case. An analysis of modeling process may explain some of the reasons. One is the neural network for the research of this report is too fixed. Different methods may have their own meanings, so you may need to change the framework when applying them. So it means we can not simply uses a framework like 'convolution + dimensional reduction layer'. Sometime, we need to add fully connected layers, change different activation function, etc so as to better use the idea of models. In addition, The image size might be too small for a high number of layers. After the fourth layer, the feature maps generated by some methods may have become small (2 $\times$ 2). Therefore, it may not be appropriate to continue dimensionality reduction. operations. This warns us not to blindly increase the number of layers of the model to get better results, but to consider case by case.

From the perspective of running time, the running time will increase as the depth of the model increases. This is because the increasing layers lead to more complicated network and more parameters need to be learned. Dilation is the most memory-intensive and time-consuming of these methods because it has more matrix operations and transformations. But if focus on the accuracy, dilation can always generate best prediction (accuracy: 0.8866, layer=3). It is likely that, compared to other parties, this method loses almost no information about the data.

In the experiment of overlapping, both Max pooling and avg pooling, overlapping method are better than non - overlapping method. The cause of this result may be overlapping to reduce the loss of some information. Actually, pooling is an operation for selection or filtering, which means pooling only keep the important information while discard less important information. The less important information, however, is also helpful to make a predictions. Because overlapping pooling takes more information into account, it should be more expensive than the normal pooling. the table 2 and table 3 indicate pooling is almost three times faster than overlapping pooling and pooling and pooling reduces memory requirements by at least 200 MiB.

In order to study the opposite of the third question, the experiment 4 is the exploration of stride. The result of experiment 4 is in accordance with the expectation, since question indicate when stride=1, the performance of models can be improve, and results in figure15 have same pattern. The highest accuracy of striding is 0.8856, when stride is equal to 1.When the stride rises, the model becomes worse and worse, because less and less information is being considered. And, it's going to be faster, and it's going to require less memory.

Furthermore, we could try to consider how other parameters affect the performances of models. Perhaps, we could also try to use a different dataset to test our conclusion from this report.

6. Conclusions

This report aims to get insight into the convolutional neural network and to explore some various approaches of the convolutional neural network as well as their effect. This report has designed 4 research questions to explore the difference between max pooling and average pooling, the effect of the number of layers, whether the overlapping pooling improves performances? and what if the stride changes? To be honest, both max pooling and average pooling can work well on the MNIST dataset, but it is still hard to say which one is better in this case (Yu et al., 2014). An important conclusion to draw from experiment 2 is that although increasing the number of layers makes the model more complex and improves performance under many circumstances, but it is necessary to consider how many layers are appropriate case by case(Lin et al., 2013). Because sometimes, too many layers may lead to the model fail. And complexity means more running time and more space is needed, and It does need a trade-off between performances and cost. There are many hyperparameters of the neural network can be considered, such as strider, which might cause different effects. When the stride < kernel size, it leads to overlapping which is able to improve model but result in the more complex model. On the contrary, lower predictions accuracy but higher speed to train.

References

Goodfellow, Ian, Bengio, Yoshua, Courville, Aaron, and Bengio, Yoshua. *Deep learning*, volume 1. MIT press Cambridge, 2016.

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.

Lin, Min, Chen, Qiang, and Yan, Shuicheng. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Nagi, Jawad, Ducatelle, Frederick, Di Caro, Gianni A, Cireşan, Dan, Meier, Ueli, Giusti, Alessandro, Nagi, Farrukh, Schmidhuber, Jürgen, and Gambardella, Luca Maria. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pp. 342–347. IEEE, 2011.

Ng, Andrew, Ngiam, Jiquan, Foo, Chuan Yu, Mai, Yifan, and Suen, Caroline. Ufldl tutorial, 2010.

Sharif Razavian, Ali, Azizpour, Hossein, Sullivan, Josephine, and Carlsson, Stefan. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

Yu, Dingjun, Wang, Hanli, Chen, Peiqiu, and Wei, Zhihua. Mixed pooling for convolutional neural networks. In *International Conference on Rough Sets and Knowledge Technology*, pp. 364–375. Springer, 2014.

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

Appendix

Appendix 1: Comparison between vectorised operation and for loop

Jupyter Notebook:

```
import numpy as np
size = 100000
a = np.arange(size)
b = np.ones(size)
```

```
%%timeit -n 100 -r 3
c = np.empty(size)
for i in range(size):
    c[i] = a[i] + b[i]
```

186 ms ± 1.16 ms per loop (mean ± std. dev. of 3 runs, 100 loops each)

```
%%timeit -n 100 -r 3
c = a + b
```

180 μs ± 36.7 μs per loop (mean ± std. dev. of 3 runs, 100 loops each)

Appendix 2: Performance results on test set for experiment 2

Model	layers	Accuracy	Loss	time	memory
Max pooling	2	0.871835443	0.37771374	273.44 it/s	819MiB
Max pooling	3	0.882151899	0.35713238	246.47 it/s	831MiB
Max pooling	4	0.882468354	0.35714045	212.16 it/s	845MiB
Max pooling	5	0.877341772	0.3536121	201.93 it/s	835MiB
Max pooling	6	0.876898734	0.38837177	197.64 it/s	835MiB
Max pooling	8	0.882278481	0.37256023	169.27 it/s	837MiB
Avg pooling	2	0.877405063	0.35791504	175.92 it/s	829MiB
Avg pooling	3	0.881139241	0.35427496	190.21 it/s	819MiB
Avg pooling	4	0.877151899	0.35369727	198.42 it/s	817MiB
Avg pooling	5	0.874873418	0.35904276	217.47 it/s	815MiB
Avg pooling	6	0.878987342	0.36150214	250.66 it/s	805MiB
Avg pooling	8	0.021455696	3.850148	167.42 it/s	831MiB
Striding	2	0.876392405	0.3919514	195.41 it/s	733MiB
Striding	3	0.87278481	0.3611034	163.78 it/s	739MiB
Striding	4	0.872405063	0.38118216	146.08 it/s	739MiB
Striding	5	0.876392405	0.3919514	132.72 it/s	739MiB
Striding	6	0.021455696	3.850148	122.11 it/s	741MiB
Striding	8	0.021455696	3.850148	108.97 it/s	743MiB
Dilation	2	0.884936709	0.3261872	71.78 it/s	913MiB
Dilation	3	0.886582278	0.32319847	55.04 it/s	1065MiB
Dilation	4	0.880886076	0.39058015	49.16 it/s	1183MiB

Table 5. Performance results on test set