MLP Coursework 2: Exploring Convolutional Networks

Andreas Neokleous

Abstract

Convolutional neural networks have been widely used for image classification since 2012. Their performance depends on the chosen architecture, such as number of convolutional layers, number of filters per layer and the type of dimensionality reduction. The aim of this report is to explain the implementation of convolutional networks, the importance of context and to compare different architectures, by fine-tuning the mentioned hyper-parameters, in the task of image classification. The models were evaluated using the EMNIST Balanced dataset of handwritten images. The highest accuracy was achieved by a convolutional neural network with 32 convolutional filters per layer, 2 convolutional layers and dilated dimensionality reduction.

1. Introduction

The first convolutional neural network (CNN) that achieved strong results on image classification was AlexNet (Krizhevsky et al., 2012). AlexNet was trained on images in the LSVRC-2010 ImageNet training set and achieved a top-5 error rate of 18.9%, which was a significant improvement over the state-of-the-art. The model architecture consisted of five convolutional layers, max-pooling dimensionality reduction layers, two fully connected layers, and at the end a softmax layer to output the 1000-class probability distribution. Since this breakthrough, CNNs have been used widely for many applications such as, image classification, retrieval, detection, segmentation and more.

For the first part of the coursework the ConvolutionalLayer and the MaxPooling2DLayer were implemented. These are described in section 2. I used the provided im2col and col2im functions to implement the forward propagation and backpropagation of both layers (and to calculate the gradients with respect to the convolutional layer parameters). The provided tests were passed.

The experiments in section 4 are form the second part of the coursework and they were influenced by AlexNet's (Krizhevsky et al., 2012) architecture, in particular the type of dimensionality reduction used. The aim of the first two experiments is to explore different network architectures with max-pooling dimensionality reduction, and the third experiment compares max-pooling with other three dimensionality reduction types. The research questions that led to these experiments are:

1. How does the number of convolutional layers affect the performance of CNN with max-pooling?

2. How does the number of convolutional filters per convolutional layer affect the performance of CNN with max-pooling?

3. How does the type of dimensionality reduction affect the performance of CNN?

The baseline for the all three experiments was a CNN with 4 convolutional layers, 64 filters per covolutional layer with max-pooling dimensionality reduction.

To address the **first question**, I trained three different CNN and then compared them to the baseline. The number of layers was different for each CNN, but the rest of the hyper-parameters were the same. I started with a shallow network, with 1 convolutional layer. Then a CNN with 4 layers and finally another one with 5 layers. Each CNN had 64 filters per convolutional layer and max-pooling. All CNNs were run for three times with the same architecture and the mean test accuracy was taken.

The same procedure was taken to address the **second question**, but this time changing the number of filters per layer instead the number of layers. Three CNNs with 32,16 and 8 number of filters were trained and compared to the baseline. Each CNN had 4 convolutional layers and max-pooling. All CNNs were run for three times and the mean test accuracy was taken.

Lastly, four different dimensionality reduction types were explored to address the **third question**. These are: max pooling, average pooling, dilated and strided convolution. Each dimensionality reduction type was tested with four different architectures. The architectures are, 4 layers and 64 filters, 4 layers and 32 filters, 2 layers and 64, filters, 2 layers and 32 filters.

The dataset used to train, validate and test the CNNs is the EMNIST Balanced dataset, which consists of handwritten digit and letter images. Each image is 28x28 and there are 47 classes (10 digits, 52 upper-lower case letters minus 15 hard-to-distinguish letters). The total number of characters in the dataset is 131,600. It is split into training, validation and testing set. The training set consists of 1000 batches and each batch has a 100 characters. The validation and testing set consist of 158 batches each and each batch has 100 characters. The size of the training set is 100,000, the size of validation set is 15,800 and the size of testing set is 15,800.

2. Implementing convolutional networks

2.1. Convolution Layer

CNNs are similar to regular neural networks in the sense that both consist of weights, biases, a loss function on the last layer and neurons which perform a dot product with some inputs (CS231n). The difference is that CNNs keep the spatial structure of the input, rather than stretching it. CNNs make the assumption that the input is always an image of depth (dimension) x height x width and its purpose is to learn the features of the image. In CNNs the neurons of a layer are connected only to a small region to the layer before it, in contrast with the fully connected layers. To achieve this, a convolution filter (or kernel) is used, which has the same depth (dimension) as the input image or previous layer's output, but usually different height and width. In forward propagation, the kernel 'slides' (convolves) to the whole input, performing element-wise dot products between the input region and the kernel, starting from the top left corner. The sum of the dot products for a region corresponds to one element of the output as shown in Figure 1.

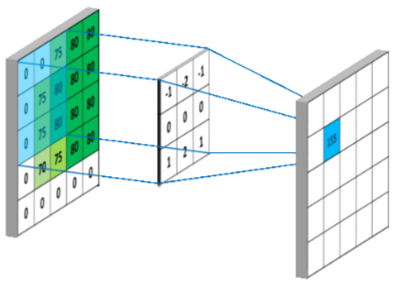

Figure 1. Kenrel Operation (mlnotebook)

More than one filter can be used, each resulting to its own output (feature map). The number of feature maps determine the depth of the total output volume of the layer. Therefore, one layer has multiple feature maps. The backpropagation is also a convolution with flipped kernel.

2.2. Pooling Layer

Pooling layer is a form of down-sampling. The purpose of pooling layers is to reduce the height and width dimensions of the representations. The depth remains the same. This increase the context, reduces the number of trainable parameter, reduces computation for each layer, thus reducing the training time. There are different types of pooling layers, such as min,average and max pooling. The most common is max pooling. The pooling layer has a filter and its size determines the region that the pooling operation is applied on. Usually the stride is adjusted based on the pooling filter size so there is no overlapping. In forward propagation for max pooling, the filter 'slides' over the input values, performing a max operation, as shown in Figure 2. Pooling is applied to each feature map separately. In backpropagation the gradients are passed back only to the highest neurons of each region, and the others get zero gradient.

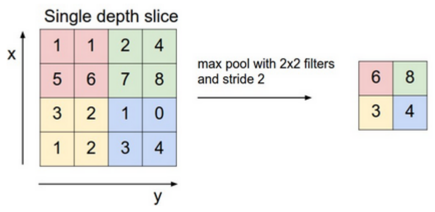

Figure 2. Max Pooling Operation (CS231n)

2.3. Implementation of ConvolutionalLayer and MaxPooling2D

For part 1 of the coursework the convolutional and max pooling layers were implemented using the serialization approach. The provided tests, for both convolutional and max pooling layers, were passed. The function im2col stretches the input regions into columns, and a single matrix multiplication is performed. For instance in the forward propagation of the convolutional layer, the input (2, 3, 4, 4) is transformed into (12, 18) using im2col. The kernels are transformed from (2, 3, 2, 2) to (2,12). A single dot product is then performed plus the biases which results in (18,2) shape. The result is transformed to the output dimensions (2, 2, 3, 3). In backpropagation, the dot product of the reshaped gradients with respect to the outputs and the reshaped kernels is calculated. This is then fed into the col2im which performs the inverse operation of im2col. The same idea was used for the max pooling layer.

The advantage of this approach is that a single operation is more efficient than many small operations. On the other hand, it used a lot of memory because the of the large size of the stretched matrix.

3. Context in convolutional networks

The purpose of increasing the receptive field of a neuron is to allow it to increase the context that it "sees".

The receptive field is another terminology for the filter. When a filter is applied, the output neuron is only connected to a local region, and its field of view is the size (Height x Width) of the filter projected on the input volume. In fact, the height and width of the receptive field of a neuron are always local and equal to the size of filter. However, the depth of the receptive field of a neuron is global. As more layers are added, the receptive field of a neuron is increasing in depth. The depth is increased by the number of feature maps per layer. The neurons that look at the same spatial region on the input volume and are in different depths, they capture different features. Figure 3 shows 5 neurons in different depths that look at the same spatial input region.

To be able to compare features that are spatially apart while keeping small convolution filters, we can use pooling layers, increase striding or use dilated convolutions (Yu & Koltun, 2015).

Both pooling and striding have a similar down-sampling

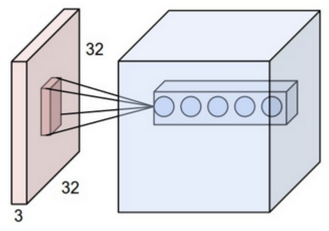

Figure 3. Each neuron (blue circles) is connected only locally to the spatial region of the input, but to the full depth. (CS231n)

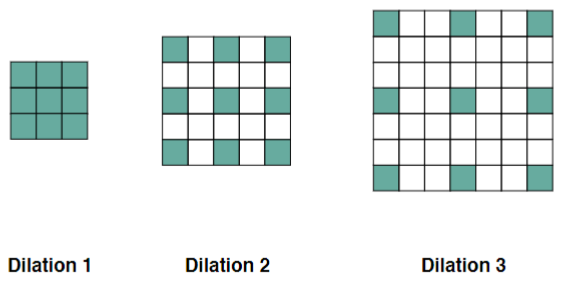

Figure 4. Increasing dilation of 3x3 filter (et al., 2018)

effect. Pooling was discussed in section 2 and striding increases the step size of the the kernel. Both result in a smaller feature map.

Dilation (Yu & Koltun, 2015) incrementally introduces spaces between the cells of the filter. The size of the receptive field is increased while the number of parameters remain constant. This is achieved by increase the distance between filter's cells as Figure 4 shows. This allows the effective receptive field to grow larger with fewer layers (CS231n).

4. Experiments

In this section the experiments that were carried out to address the following research questions are described:

1. How does the number of convolutional layers affect the performance of CNN with max-pooling?

2. How does the number of convolutional filters per convolutional layer affect the performance of CNN with max-pooling?

3. How does the type of dimensionality reduction affect the performance of CNN?

The baseline for all three experiment is a CNN with 4 layers, 64 filters with max pooling. The number of epochs for every experiment was 50.

4.1. Experiment 1: Number of Layers on CNN with Max Pooling

The purpose of this experiment was to discover the effect that the number of layers has on the performance of CNN with Max Pooling.

Each CNN architecture was run 3 times and the mean accuracy on the test set was used to compare the architectures.

Layers: 1	3	4	5
Acc(test): 0.73746	0.87632	0.88004	0.87645

Table 1. Test Set Accuracy of CNNs with Different Layers with 64 Filters and Max Pooling

Table 1 shows that the baseline, which has 4 layers achieved the highest accuracy. The worst was the shallow layer with 1 layer, as expected. It is worth noting that more layers do not always improve the performance of the model. This however, depends on the on the other hyper-parameters, as well as the dataset type.

4.2. Experiment 2: Number of Filters on CNN with Max Pooling

The second experiment is meant to find how the number of filters affect the performance of CNN with Max Pooling. Each CNN architecture was run 3 times and the mean accuracy on the test set was used to compare the architectures.

Filters: 8	16	32	64
Acc(test): 0.82951	0.86814	0.87413	0.88004

Table 2. Test Set Accuracy of CNNs with Different Number of Filters with Layers and Max Pooling

Table 2 show that the highest accuracy was achieved by the baseline with 64 filter per layer. More filters per layers means that the model learns more features per layer. Thus, it achieves a higher classification performance. The less filters per layer, the lower the test accuracy of the CNN with max pooling.

4.3. Experiment 3: Max Pooling vs Average Pooling vs Strided vs Dilated

The purpose of this experiment was to compare Max Pooling, Average Pooling, Convolutions with Stride and Dilated Convolutions. For each dimensionality type, four different CNN architectures were tested:

- 2 layers and 32 filters per layer
- 2 layers and 64 filters per layer
- 4 layers and 32 filters per layer
- 4 layers and 64 filters per layer

	Max	Average	Dilated	Strided
2L32F:	0.85677	0.85209	0.88342	0.87070
2L64F:	0.87171	0.86949	0.88247	0.87639
4L32F:	0.87414	0.87582	0.87943	0.87203
4L64F:	0.88004	0.88089	0.88133	0.87861

Table 3. Test Set Accuracy Table of CNNs with Different Dimensioanlity Reduction Types. L=Layer, F=Number of Filters

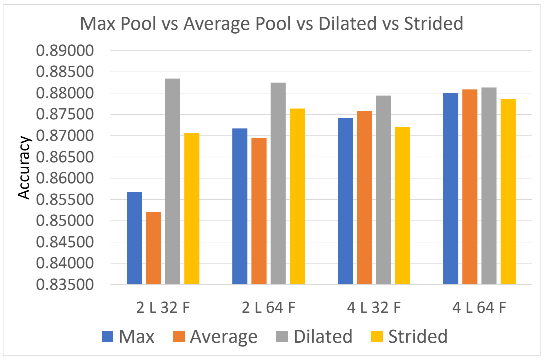

Figure 5. Test Set Accuracy Plot of CNNs with Different Dimensioanlity Reduction Types

Table 3 and Figure 5 show the results of this experiment. The dilated dimensionality reduction achieved the highest accuracy across all tests.

The difference is particularly high between the performance of dilated convolution and the rest, on the architecture with 2 layers and 32 filters. As mentioned on section 3, the dilated filters can capture more information compared to regular filters with fewer layers. The next highest accuracy on this architecture is the strided convolution. This shows that stride is more effective than pooling with few layers. Average pooling performed the worst with max pooling achieving slightly higher accuracy.

On the next architecture with 2 layers and 64 filters, the highest accuracy was achieved again by the dilated convolution. However the cap between the dilated the others has decreased. By just increasing the number of filters the accuracy of max and average pooling has increased considerably. The next highest accuracy after dilated, was achieved by strided convolution, which also saw a small boost in performance from the previous test.

The highest accuracy of the CNN architectures with 4 layers has been achieved by the dilated convolution again. More layers seem to be more favorable to max and average pooling than strided convolution. With 4 layers and 32 filters, the next highest accuracy after dilated is average pooling and slightly behind is max pooling. Strided convolution actually performed worst on this test, than the previous test with the 2 layers.

At the test with 4 layers and 64 filters, the gap between all dimensioanlity reduction types is the lowest. Dilated convolution is again the most accurate, but by a small margin over average pooling. Max pooling is very close to average pooling, while strided convolution is the worst in this case.

The research question was how these four dimeninsality reduction types compare. The result is that Dilated convolutions performed the best for 2 and 4 layers. With 2 layers, strided convolution performed better than max and average pooling. With 4 layers, max and average pooling performed better than strided convoltuion.

5. Discussion

Table 4 shows the 5 CNN architectures that achieved the highest accuracy out of the 22 architectures tested in this report. The initial baseline was a CNN with 4 layers, 64 filters and max pool. There are four setups that achieve higher accuracy than the baseline. The first 3 most accurate setups are all with dilated dimensionality reduction. The highest accuracy was achieved with 2 layers and 32 filters, the second with 2 layers and 64 filters and the third with 4 layers and 64 filters. The fourth most accurate setup was CNN with 4 layers, 64 filter and average pool. Finally, the fifth most accurate is the baseline.

Dim.	Setup	Accuracy
Dilated	2 L 32 F	0.88342
Dilated	2 L 64 F	0.88247
Dilated	4 L 64 F	0.88133
Average	4 L 64 F	0.88089
Max	4 L 64 F	0.88004

Table 4. Top 5 Setups with Highest Test Set Accuracy. L=Layer, F=Number of Filters

6. Conclusions

In this report the implementation of CNN networks and the importance of context were discussed. Different experiments were carried out to address the three research questions that were set. The answers to these are, first, the lower number of filters the worst performance a CNN with max pooling has. Second, more layers do not translate to higher accuracy of a CNN with max pooling. Third, dilated convolutions perform the best out of all dimensionality reduction types, especially with few layers and few number of filters.

References

CS231n. Cs231n convolutional neural networks for visual recognition. http://cs231n.github.io/convolutional-networks/. Accessed: 2018-11-23.

et al., Antreas Antoniou. Dilated densenets for relational reasoning. 2018. URL https://arxiv.org/abs/1811.00410.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E.

Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

mlnotebook. Convolutional neural networks - basics. https://mlnotebook.github.io/post/CNN1/. Accessed: 2018-11-23.

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