A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification

Ye Zhang and Byron Wallace

Presenter: Ruichuan Zhang
• Introduction
• Background
• Datasets and baseline models
• Sensitivity analysis of hyperparameters
  – Input word vector
  – Filter region size
  – Number of feature maps
  – Activation function
  – Pooling strategy
  – Regularization
• Conclusions
Introduction

• Convolutional Neural Networks (CNNs) achieve good performance in sentence classification

• Problem for practitioners: how to specify the CNN architecture and set the (many) hyperparameters?

• Exploring is expensive
  – Slow training
  – Vast space of model architecture and hyperparameter settings

• Need to conduct an empirical evaluation on the effect of varying hyperparameter on performance \( \Rightarrow \) use the results of this paper as a starting point for your own CNN model
Background: CNNs

- Input layer
- Hidden layer
- Output layer
Background: CNNs

Sentence matrix
7 X 5

3 region sizes: 2, 3, 4
2 filters for each size
Totally 6 filters

Convolution

Activation function

2 feature maps for each size
Background: CNNs

1-max pooling

2 feature maps for each size

6 vectors concatenated → single feature vector

Regularization & Softmax

2 classes
Datasets and Baseline Model

• Nine sentence classification datasets [short to medium average sentence length (3-23)]
  – Examples
    • SST: Stanford Sentiment Treebank (average length: 18)
    • CR: customer review dataset (average length: 19)

• Baseline CNN configuration (Kim, 2014):
  – **Input word vector**: Google word2vec
  – **Filter region size**: 3, 4, and 5
  – **Number of feature maps**: 100
  – **Activation function**: ReLU
  – **Pooling**: 1-max pooling
  – **Regularization**: dropout rate 0.5, l2 norm constraint 3
Datasets and Baseline Model

- Baseline CNNs configuration:
  - 100 times 10-fold CV
  - Record mean and range of accuracy
- Each sensitivity analysis:
  - Hold all other settings constant, vary the factor of interest
- Each configuration
  - Replicate the experiment 10 times, each replication a 10-fold CV
  - Record **average CV means** and **ranges** of accuracy
Effect of Input Word Vectors

• Three types of word vector
  – **Word2vec**: 100 billion words from Google News, 300-dimensional
  – **GloVe**: 840 billions of tokens from web data, 300-dimensional
  – Concatenated word2vec and GloVe: 600-dimensional

• Performance depends on dataset
• **Not helpful** to concatenate
• **One-hot vector**: poorly [when training dataset is small to moderate]
Effect of Filter Region Size

- **Filter**
  - Word embedding matrix $A: s \times d$
  - Filter matrix $W$ with region size $h: h \times d$
  - Output sequence of length $s-h+1$: $o$, $o_i = W \cdot A[i:i+h-1]$

E.g., filter with region size 3
Effect of Filter Region Size

- **One** region size
  - Each dataset has its own optimal filter size
  - A coarse search over 1 to 10
  - Longer sentence (e.g., CR): larger filter size
Effect of Filter Region Size

- **Multiple** region sizes
  - Combining close-to-optimal sizes: improve performance
  - Adding far-from-optimal sizes: decrease performance

<table>
<thead>
<tr>
<th>Multiple region size</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3)</td>
<td>91.21 (90.88, 91.52)</td>
</tr>
<tr>
<td>(5)</td>
<td>91.20 (90.96, 91.43)</td>
</tr>
<tr>
<td>(2,3,4)</td>
<td>91.48 (90.96, 91.70)</td>
</tr>
<tr>
<td>(3,4,5)</td>
<td>91.56 (91.24, 91.81)</td>
</tr>
<tr>
<td>(4,5,6)</td>
<td>91.48 (91.17, 91.68)</td>
</tr>
<tr>
<td>(7,8,9)</td>
<td>90.79 (90.57, 91.26)</td>
</tr>
<tr>
<td>(14,15,16)</td>
<td>90.23 (89.81, 90.51)</td>
</tr>
<tr>
<td><strong>(2,3,4,5)</strong></td>
<td><strong>91.57 (91.25, 91.94)</strong></td>
</tr>
<tr>
<td>(3,3,3)</td>
<td>91.42 (91.11, 91.65)</td>
</tr>
<tr>
<td>(3,3,3,3)</td>
<td>91.32 (90.53, 91.55)</td>
</tr>
</tbody>
</table>
Effect of Number of Feature Maps

- Number of feature maps (for each filter region size)
  - 10, 50, 100, 200, 400, 600, 1000, 2000
- Optimums depend on dataset; fall in [100, 600]
- Over 600: no much improvement and longer training time
Effect of Activation Function

- Activation functions $f$: $c_i = f(o_i + b)$
- Examples:

<table>
<thead>
<tr>
<th>Function</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softplus</td>
<td>$f(x) = \ln(1 + e^x)$</td>
</tr>
<tr>
<td>ReLu</td>
<td>$f(x) = \begin{cases} 0 &amp; \text{for } x &lt; 0 \ x &amp; \text{for } x \geq 0 \end{cases}$</td>
</tr>
<tr>
<td>Tanh</td>
<td>$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$</td>
</tr>
<tr>
<td>Sigmoid</td>
<td>$f(x) = \frac{1}{1 + e^{-x}}$</td>
</tr>
<tr>
<td>Identity</td>
<td>$f(x) = x$</td>
</tr>
</tbody>
</table>

- **Tanh, Iden, ReLU** perform better
- No significant difference among the good ones
Effect of Pooling Strategy

- **Baseline strategy**: 1-max pooling
  
  Feature sequence: $c$  
  
  max pooling $\rightarrow$ Maximum $\hat{c}$

- **Strategy 1**: Max pooling over local region (size=3, 10, 20, 30): worse
  
  Feature sequence: $c$
  
  $\{\text{max pooling} \rightarrow \text{Local maximum}\}$  
  
  Concat

- **Strategy 2**: K-max pooling (k=5, 10, 15, 20): worse

- **Strategy 3**: Average pooling over local region (size=3, 10, 20, 30): (much) worse
Effect of Regularization

- Dropout (before the output layer)
  - \( y = w \cdot z + b \), with a probability \( p \) that \( z_i \) is dropped out
  - \( z \) is concatenated maximum values \( \hat{c} \)
  - Dropout rate from 0.1 to 0.5: helps a little
  - Dropout before convolution: similar range and effect
Effect of Regularization

- L2-norm constraint
  - Force $\| \mathbf{w} \|_2 = s$, whenever $\| \mathbf{w} \|_2 > s$
  - L2-norm constraint does not improve performance much
  - Does not harm too, so use one
Conclusions (and Practitioners’ Guide)

• Use word2vec or GloVe rather than one-hot vector
• Line-search over single filter size from 1-10, and then combine multiple ‘good’ region sizes
• Adjust the number of feature maps for each filter size from 100 to 600
• Use 1-max pooling
• Test different activation functions (at least) ReLU and tanh
• Use small dropout rate (0.0-0.5) and a (large) max norm constraint and try larger values when optimal number of feature maps is large (over 600)
• Repeat CV to assess the performance of a model