Convolutional Networks 1

CS 287
Review: NGram Issues

In training we might see,

the arizona corporations commission authorized

But at test we see,

the colorado businesses organization ___

▶ Does this training example help here?
  ▶ Not really. No count overlap.

▶ Does backoff help here?
  ▶ Maybe, if we have seen organization.
  ▶ Mostly get nothing from the earlier words.
Review: NGram Issues

In training we might see,

the arizona corporations commission authorized

But at test we see,

the colorado businesses organization ___

▶ Does this training example help here?
  ▶ Not really. No count overlap.

▶ Does backoff help here?
  ▶ Maybe, if we have seen organization.
  ▶ Mostly get nothing from the earlier words.
Review: NGram Issues

In training we might see,

the arizona corporations commission authorized

But at test we see,

the colorado businesses organization ___

- Does this training example help here?
  - Not really. No count overlap.

- Does backoff help here?
  - Maybe, if we have seen organization.
  - Mostly get nothing from the earlier words.
Optional, direct connection layers,

\[ NN_{DMLP_1}(x) = [\tanh(xW^1 + b^1), x]W^2 + b^2 \]

- \( W^1 \in \mathbb{R}^{d_{in} \times d_{hid}}, b^1 \in \mathbb{R}^{1 \times d_{hid}} \); first affine transformation
- \( W^2 \in \mathbb{R}^{(d_{hid} + d_{in}) \times d_{out}}, b^2 \in \mathbb{R}^{1 \times d_{out}} \); second affine transformation
Review: A Neural Probabilistic Language Model (Bengio, 2003)

Dashed-lines show the optional direct connections, $C = ν$. 
Both count-based models and feed-forward NNLMs are Markovian language models, 

Comparison:

- Training Speed: ngrams are much faster (more coming)
- Usage Speed: ngrams very fast, NN can be fast with some tricks.
- Memory: NN models can be much smaller (but there are big ones)
- Accuracy: Comparable for small data, NN does better with more.

Advantages of NN model

- Can be trained end-to-end.
- Does not require smoothing methods.
Neural language models can be poor at assigning very high probability to high confidence decisions, for instance major league baseball or united states of america.

▶ Give a high-level explanation of why this might occur compared to an n-gram model.

▶ Describe a variant of the Bengio model that is able to incorporate extra parameters to allow for rare cases that should have high probability.
Contents

Text Classification Review

Convolutions

Applications

Vision
**Sentiment**

**Good Sentences**
- A thoughtful, provocative, insistently humanizing film.
- Occasionally melodramatic, it’s also extremely effective.
- Guaranteed to move anyone who ever shook, rattled, or rolled.

**Bad Sentences**
- A sentimental mess that never rings true.
- This 100-minute movie only has about 25 minutes of decent material.
- Here, common sense flies out the window, along with the hail of bullets, none of which ever seem to hit Sascha.
Review Linear Models for Classification

Linear model,

\[ \hat{y} = f(xW + b) \]

- \( W \in \mathbb{R}^{d_{in} \times d_{out}} \), \( b \in \mathbb{R}^{1 \times d_{out}} \); model parameters
- \( f : \mathbb{R}^{d_{out}} \rightarrow \mathbb{R}^{d_{out}} \); activation function
- Sometimes \( z = xW + b \) informally “score” vector.
- Note \( z \) and \( \hat{y} \) are not one-hot.

Class prediction,

\[ \hat{c} = \arg \max_{i \in C} \hat{y}_i = \arg \max_{i \in C} (xW + b)_i \]
Features 1: Sparse Bag-of-Words Features

Representation is counts of input words,

- $\mathcal{F}$; the vocabulary of the language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

A sentimental mess

$$\mathbf{x} = \delta(\text{word:A}) + \delta(\text{word:sentimental}) + \delta(\text{word:mess})$$

$$\mathbf{x}^\top = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$\text{word:A}$

$\text{word:mess}$

$\text{word:sentimental}$
Features 2: Sparse Bag-of-Bigrams Features

Representation is counts of input bigrams,

- $\mathcal{F}$; the vocabulary of the bigram language.
- $\mathbf{x} = \sum_i \delta(f_i)$

Example: Movie review input,

A sentimental mess

$\mathbf{x} = \delta(\text{word:A}) + \delta(\text{bigram:A:sentimental})$
$+ \delta(\text{word:sentimental}) + \delta(\text{bigram:sentimental:mess})$
$+ \delta(\text{word:mess})$
Features 3: Continuous Bag-of-Words Features

\[ \mathbf{x} = \sum_{i=1}^{k} \nu(f_i; \theta) = \sum_{i=1}^{k} \delta(f_i) \mathbf{W^0} \]

- \( \mathcal{F} \); the vocabulary of the language.
- \( \mathbf{x} = \sum_i \delta(f_i) \)

Example: Movie review input,

\[ \mathbf{x} = \nu(\text{word:A}) + \nu(\text{word:sentimental}) + \nu(\text{word:mess}) \]

\[ \mathbf{x}^\top = \begin{bmatrix} 0.2 \\ \vdots \\ 1.2 \\ -0.5 \end{bmatrix} + \begin{bmatrix} 0.8 \\ \vdots \\ 1.0 \\ -1.0 \end{bmatrix} + \begin{bmatrix} 0.1 \\ \vdots \\ 9.2 \\ -2.0 \end{bmatrix} = \begin{bmatrix} 1.1 \\ \vdots \\ 11.4 \\ -3.5 \end{bmatrix} \]
Features 4: Continuous Bag-of-Bigrams Features?

Representation is counts of input bigrams,

- $\mathcal{F}$; the vocabulary of the bigram language.
- $x = \sum_i \delta(f_i)$

Example: Movie review input,

A sentimental mess

\[
x = \nu(\text{word: A}) + \nu_2(\text{bigram: A: sentimental}) \\
+ \nu(\text{word: sentimental}) + \nu_2(\text{bigram: sentimental: mess}) \\
+ \nu(\text{word: mess})
\]
Neural Network

One-layer multi-layer perceptron architecture,

$$NN_{MLP_1}(x) = g(xW^1 + b^1)W^2 + b^2$$

- $xW + b$; perceptron
- $x$ is the dense representation in $\mathbb{R}^{1 \times d_{in}}$
- $W^1 \in \mathbb{R}^{d_{in} \times d_{hid}}, b^1 \in \mathbb{R}^{1 \times d_{hid}}$; first affine transformation
- $W^2 \in \mathbb{R}^{d_{hid} \times d_{out}}, b^2 \in \mathbb{R}^{1 \times d_{out}}$; second affine transformation
- $g: \mathbb{R}^{d_{hid} \times d_{hid}}$ is an activation non-linearity (often pointwise)
- $g(xW^1 + b^1)$ is the hidden layer
Windowed Classification

Alternative method, windows into MLP.

**Goal:** predict $t_5$.

- Windowed word model.
  - $w_1$, $w_2$, $[w_3, w_4, w_5, w_6, w_7]$, $w_8$
  - $w_3$, $w_4$; left context
  - $w_5$; Word of interest
  - $w_6$, $w_7$; right context
  - $d_{\text{win}}$; size of window ($d_{\text{win}} = 5$)
Idea: Use window at each location.

\[
\begin{bmatrix}
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \\
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \\
w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 \\
\vdots
\end{bmatrix}
\]

Each maps from window of embeddings to $d_{\text{hid}}$
Convolution Formally

Let our input be the embeddings of the full sentence, $X \in \mathbb{R}^{n \times d^0}$

$$X = [v(w_1), v(w_2), v(w_3), \ldots, v(w_n)]$$

Define a window model as $\mathcal{NN}_{window} : \mathbb{R}^{1 \times (d_{win}d^0)} \rightarrow \mathbb{R}^{1 \times d_{hid}}$, $\mathcal{NN}_{window}(x_{win}) = x_{win}W^1 + b^1$

The convolution is defined as $\mathcal{NN}_{conv} : \mathbb{R}^{n \times d^0} \rightarrow \mathbb{R}^{(n-d_{win}+1) \times d_{hid}}$, $\mathcal{NN}_{conv}(X) = \tanh \left[ \begin{array}{c} \mathcal{NN}_{window}(X_{1:d_{win}}) \\ \mathcal{NN}_{window}(X_{2:d_{win}+1}) \\ \vdots \\ \mathcal{NN}_{window}(X_{n-d_{win}:n}) \end{array} \right]$
Pooling

- Unfortunately $\mathbf{NN}_{\text{conv}} : \mathbb{R}^{n \times d^0} \mapsto \mathbb{R}^{(n - d_{\text{win}} + 1) \times d_{\text{hid}}}$.

- Need to map down to $d_{\text{out}}$ for different $n$

- Recall pooling operations.

- Pooling “over-time” operations $f : \mathbb{R}^{n \times m} \mapsto \mathbb{R}^{1 \times m}$

  1. $f_{\text{max}}(\mathbf{X})_{1,j} = \max_i X_{i,j}$
  2. $f_{\text{min}}(\mathbf{X})_{1,j} = \min_i X_{i,j}$
  3. $f_{\text{mean}}(\mathbf{X})_{1,j} = \sum_i X_{i,j} / n$

$$f(\mathbf{X}) = \begin{bmatrix} \downarrow & \downarrow & \cdots \\ \downarrow & \downarrow & \cdots \\ \cdots \\ \downarrow & \downarrow & \cdots \end{bmatrix} = \begin{bmatrix} \cdots \end{bmatrix}$$
Putting it together

\[ \hat{y} = \text{softmax}(f_{\text{max}}(\text{NN}_{\text{conv}}(X))W^2 + b^2) \]

- \( W^2 \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{out}}} \), \( b^2 \in \mathbb{R}^{1 \times d_{\text{out}}} \)
- Final linear layer \( W^2 \) uses learned window features
Multiple Convolutions

\[ \hat{y} = \text{softmax}( [ f(\mathcal{N}N_{conv}^1(\mathbf{X})), f(\mathcal{N}N_{conv}^2(\mathbf{X})), \ldots, f(\mathcal{N}N_{conv}^f(\mathbf{X})) ] W^2 + b^2 ) \]

- Concat several convolutions together.
- Each $\mathcal{N}N^1$, $\mathcal{N}N^2$, etc uses a different $d_{\text{win}}$
- Allows for different window-sizes (similar to multiple n-grams)
Convolution Diagram (Kim, 2014)

- $n = 9$, $d_{\text{hid}} = 4$, $d_{\text{out}} = 2$

- red- $d_{\text{win}} = 2$, blue- $d_{\text{win}} = 3$, (ignore back channel)
## Classification Results

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>SST-1</th>
<th>SST-2</th>
<th>Subj</th>
<th>TREC</th>
<th>CR</th>
<th>MPQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-rand</td>
<td>76.1</td>
<td>45.0</td>
<td>82.7</td>
<td>89.6</td>
<td>91.2</td>
<td>79.8</td>
<td>83.4</td>
</tr>
<tr>
<td>CNN-static</td>
<td>81.0</td>
<td>45.5</td>
<td>86.8</td>
<td>93.0</td>
<td>92.8</td>
<td>84.7</td>
<td><strong>89.6</strong></td>
</tr>
<tr>
<td>CNN-non-static</td>
<td><strong>81.5</strong></td>
<td>48.0</td>
<td>87.2</td>
<td>93.4</td>
<td>93.6</td>
<td>84.3</td>
<td>89.5</td>
</tr>
<tr>
<td>CNN-multichannel</td>
<td>81.1</td>
<td>47.4</td>
<td><strong>88.1</strong></td>
<td>93.2</td>
<td>92.2</td>
<td><strong>85.0</strong></td>
<td>89.4</td>
</tr>
<tr>
<td>RAE (Socher et al., 2011)</td>
<td>77.7</td>
<td>43.2</td>
<td>82.4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>86.4</td>
</tr>
<tr>
<td>MV-RNN (Socher et al., 2012)</td>
<td>79.0</td>
<td>44.4</td>
<td>82.9</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>RNTN (Socher et al., 2013)</td>
<td>—</td>
<td>45.7</td>
<td>85.4</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>DCNN (Kalchbrenner et al., 2014)</td>
<td>—</td>
<td>48.5</td>
<td>86.8</td>
<td>—</td>
<td>93.0</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Paragraph-Vec (Le and Mikolov, 2014)</td>
<td>—</td>
<td><strong>48.7</strong></td>
<td>87.8</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Convolutional Vocabulary

- **kernel size** or **filter width**; window size $d_{\text{win}}$
- **filter**; column of matrix $W^1$ in $\mathbb{R}^{(d^0 \times d_{\text{win}}) \times 1}$
- **feature map**; column of $NN_{\text{conv}}$, $d_{\text{hid}}$ of these
- **fully-connected layer**; affine or linear + activation
- **random, static, non-static**; embedding layer setup
- **temporal convolution, time-delay convolution**; names for one-dimensional convolutions
Why is it called a convolution?

- Let $x$ and $y$ be in $\mathbb{R}^n$ and $\mathbb{R}^m$

$$[x \ast y]_i = \sum_{j=1}^{m} x_{i-j}y_j$$

- Circular, $i - k$ wraps around.

- For NN, include padding
Contents

Text Classification Review

Convolutions

Applications

Vision
He would n’t accept anything of value from those he was writing about

[A0 He ] [AM-MOD would ] [AM-NEG n’t ] [V accept ] [A1 anything of value ] from [A2 those he was writing about ]

▶ V: verb
▶ A0: acceptor
▶ A1: thing accepted
▶ A2: accepted-from
▶ A3: attribute
▶ AM-MOD: modal
▶ AM-NEG: negation
Other Language Applications (Collobert et al. 2011)
First given a verb $w_i$ e.g. accept.

Then consider a word $w_j$ e.g. n’t

For a word $w_k$ features are

$$v(w_k), v_2(cap(w_k)), v_3(i - k), v_4(j - k)$$

Convolution over sentence is used to predict role.

$O(n \times |\text{verbs}|)$ convolutions per sentence
Feed-forward NLM (Bengio, Ducharme, and Vincent 2003)

\[ p(w_t | w_1, \ldots, w_{t-1}) = \text{softmax}(P h_{t-1} + q) \]

\[ h_{t-1} = W \begin{bmatrix} x_{t-1} \\ x_{t-2} \\ x_{t-3} \end{bmatrix} \]
**Issue**: The fundamental unit of information is still the **word**

Separate embeddings for “trading”, “trade”, “trades”, etc.

![Log-Freq vs Word (ranked by freq)](image)
Character-level CNN (CharCNN)
Character-level CNN (CharCNN)

\[ C \in \mathbb{R}^{d \times l} : \text{Representation of absurdity} \]

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>-0.8</td>
<td>2.2</td>
<td>0.1</td>
<td>0.5</td>
<td>-0.4</td>
<td>0.4</td>
<td>-0.4</td>
</tr>
<tr>
<td>0.1</td>
<td>1.2</td>
<td>1.5</td>
<td>-0.8</td>
<td>-1.5</td>
<td>0.2</td>
<td>0.1</td>
<td>1.2</td>
</tr>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>-1.2</td>
<td>0.2</td>
<td>-0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>-1.3</td>
</tr>
<tr>
<td>-0.2</td>
<td>-0.5</td>
<td>0.1</td>
<td>0.2</td>
<td>-0.3</td>
<td>0.3</td>
<td>-0.1</td>
<td>1.0</td>
</tr>
</tbody>
</table>

\[ \text{absurdity} \]
Character-level CNN (CharCNN)

\[ H \in \mathbb{R}^{d \times w} : \text{Convolutional filter matrix of width } w = 3 \]
Character-level CNN (CharCNN)

\[ f[1] = \langle C[*, 1 : 3], H \rangle \]
Character-level CNN (CharCNN)

\[ f[1] = \langle C[*, 1 : 3], H \rangle \]
Character-level CNN (CharCNN)

\[ f[2] = \langle C[\ast, 2 : 4], H \rangle \]
Character-level CNN (CharCNN)

\[ f[T - 2] = \langle C[*, T - 2 : T], H \rangle \]
Character-level CNN (CharCNN)

\[ y[1] = \max_i \{ f[i] \} \]
Each filter picks out a character $n$-gram

Character-level CNN (CharCNN)
Character-level CNN (CharCNN)

\[ f'[1] = \langle C[*, 1 : 2], H' \rangle \]
Character-level CNN (CharCNN)

\[ y[2] = \max_{i} \{ f'[i] \} \]
Character-level CNN (CharCNN)

Many filter matrices (25–200) per width (1–7)
Learned Word Representations (In Vocab)

(Based on cosine similarity)

<table>
<thead>
<tr>
<th>Word Embedding</th>
<th>while</th>
<th>his</th>
<th>you</th>
<th>richard</th>
<th>trading</th>
</tr>
</thead>
<tbody>
<tr>
<td>although</td>
<td>your</td>
<td>conservatives</td>
<td>jonathan</td>
<td>advertised</td>
<td></td>
</tr>
<tr>
<td>letting</td>
<td>her</td>
<td>we</td>
<td>rob ert</td>
<td>advertising</td>
<td></td>
</tr>
<tr>
<td>though</td>
<td>my</td>
<td>guys</td>
<td>neil</td>
<td>turnover</td>
<td></td>
</tr>
<tr>
<td>minute</td>
<td>their</td>
<td>i</td>
<td>nancy</td>
<td>turnover</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characters (before highway)</th>
<th>chile</th>
<th>this</th>
<th>your</th>
<th>hard</th>
<th>heading</th>
</tr>
</thead>
<tbody>
<tr>
<td>whole</td>
<td>hhs</td>
<td>young</td>
<td>rich</td>
<td>riche r</td>
<td>training</td>
</tr>
<tr>
<td>meanwhile</td>
<td>is</td>
<td>four</td>
<td>richer</td>
<td>richter</td>
<td>reading</td>
</tr>
<tr>
<td>white</td>
<td>has</td>
<td>youth</td>
<td></td>
<td></td>
<td>leading</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characters (after highway)</th>
<th>meanwhile</th>
<th>hhs</th>
<th>we</th>
<th>eduard</th>
<th>trade</th>
</tr>
</thead>
<tbody>
<tr>
<td>whole</td>
<td>this</td>
<td>your</td>
<td>gerard</td>
<td>training</td>
<td></td>
</tr>
<tr>
<td>though</td>
<td>their</td>
<td>doug</td>
<td>edward</td>
<td>traded</td>
<td></td>
</tr>
<tr>
<td>nevertheless</td>
<td>your</td>
<td>i</td>
<td>carl</td>
<td>trader</td>
<td></td>
</tr>
</tbody>
</table>
Learned Word Representations (In Vocab)

(Based on cosine similarity)

<table>
<thead>
<tr>
<th>Word Embedding</th>
<th>In Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>while</td>
<td>his</td>
</tr>
<tr>
<td>although</td>
<td>your</td>
</tr>
<tr>
<td>letting</td>
<td>her</td>
</tr>
<tr>
<td>though</td>
<td>my</td>
</tr>
<tr>
<td>minute</td>
<td>their</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Word</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Characters</td>
<td></td>
</tr>
<tr>
<td>(before highway)</td>
<td>chile</td>
</tr>
<tr>
<td></td>
<td>whole</td>
</tr>
<tr>
<td></td>
<td>meanwhile</td>
</tr>
<tr>
<td></td>
<td>white</td>
</tr>
<tr>
<td>Characters</td>
<td></td>
</tr>
<tr>
<td>(after highway)</td>
<td>meanwhile</td>
</tr>
<tr>
<td></td>
<td>whole</td>
</tr>
<tr>
<td></td>
<td>though</td>
</tr>
<tr>
<td></td>
<td>nevertheless</td>
</tr>
</tbody>
</table>
## Learned Word Representations (OOV)

<table>
<thead>
<tr>
<th>Characters (before highway)</th>
<th>Out-of-Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer-aided</td>
<td>computer-guided</td>
</tr>
<tr>
<td>computerized</td>
<td>informed</td>
</tr>
<tr>
<td>disk-drive</td>
<td>performed</td>
</tr>
<tr>
<td>computer</td>
<td>transformed</td>
</tr>
<tr>
<td></td>
<td>inform</td>
</tr>
<tr>
<td></td>
<td>look</td>
</tr>
<tr>
<td></td>
<td>cook</td>
</tr>
<tr>
<td></td>
<td>looks</td>
</tr>
<tr>
<td></td>
<td>shook</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Characters (after highway)</th>
<th>Out-of-Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer-guided</td>
<td>informed</td>
</tr>
<tr>
<td>computer-driven</td>
<td>performed</td>
</tr>
<tr>
<td>computerized</td>
<td>outperformed</td>
</tr>
<tr>
<td>computer</td>
<td>transformed</td>
</tr>
<tr>
<td></td>
<td>looked</td>
</tr>
<tr>
<td></td>
<td>looked</td>
</tr>
<tr>
<td></td>
<td>looking</td>
</tr>
</tbody>
</table>
## Learned Word Representations (OOV)

<table>
<thead>
<tr>
<th>Out-of-Vocabulary</th>
<th>computer-aided</th>
<th>misinformed</th>
<th>loooooook</th>
</tr>
</thead>
<tbody>
<tr>
<td>computer-guided</td>
<td>informed</td>
<td>look</td>
<td></td>
</tr>
<tr>
<td>computerized</td>
<td>performed</td>
<td>cook</td>
<td></td>
</tr>
<tr>
<td>disk-drive</td>
<td>transformed</td>
<td>looks</td>
<td></td>
</tr>
<tr>
<td>computer</td>
<td>inform</td>
<td>shook</td>
<td></td>
</tr>
</tbody>
</table>

### Characters

- **(before highway)**
  - computer-aided
  - computer-guided
  - computerized
  - disk-drive
  - computer
- **(after highway)**
  - computer-driven
  - computerized
  - computer
  - transformed
  - outperformed
  - looked
  - looking
Convolutional Filters

For each filter, visualize 100 substrings with the highest filter response
Convolutional Filters

For each filter, visualize 100 substrings with the highest filter response
Prefixes, Suffixes, Hyphenated, Others

Prefixes: character $n$-grams that start with ‘start-of-word’ character, such as \{un, mis\}. Suffixes defined similarly.
Contents

Text Classification Review

Convolutions

Applications

Vision
Visual Classification
### Speech Convolutions

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Filters</th>
<th>Stride</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softmax</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully connected, 4096</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fully connected, 4096</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max pooling, $2 \times$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 384</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max pooling, $2 \times 2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 192</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max pooling, $2 \times 2$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolution, $3 \times 3$, 96</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input (31x41)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Visualization (Zeiler and Fergus, 2014)
Visualization (Zeiler and Fergus, 2014)