VIP Cheatsheet: Convolutional Neural Networks

Afshine AMIDI and Shervine AMIDI

November 26, 2018

Overview

Architecture of a traditional CNN – Convolutional neural networks, also known as CNNs, are a specific type of neural networks that are generally composed of the following layers:

- Convolutional layer (CONV)
- Pooling (POOL)
- Fully Connected (FC)

The convolution layer and the pooling layer can be fine-tuned with respect to hyperparameters that are described in the next sections.

Types of layer

Convolutional layer (CONV) – The convolution layer (CONV) uses filters that perform convolution operations as it is scanning the input with respect to its dimensions. Its hyperparameters include the filter size \( F \) and stride \( S \). The resulting output \( O \) is called feature map or activation map.

Pooling (POOL) – The pooling layer (POOL) is a downsampling operation, typically applied after a convolution layer, which does some spatial invariance. In particular, max and average pooling are special kinds of pooling where the maximum and average value is taken, respectively.

Filter hyperparameters

The convolution layer contains filters for which it is important to know the meaning behind its hyperparameters.

- Dimensions of a filter – A filter of size \( F \times F \) applied to an input containing \( C \) channels is a \( F \times F \times C \) volume that performs convolutions on an input of size \( I \times I \times C \) and produces an output feature map (also called activation map) of size \( O \times O \times 1 \).

- Stride – For a convolutional or a pooling operation, the stride \( S \) denotes the number of pixels by which the window moves after each operation.
Zero-padding – Zero-padding denotes the process of adding \( P \) zeroes to each side of the boundaries of the input. This value can either be manually specified or automatically set through one of the three modes detailed below:

<table>
<thead>
<tr>
<th>Value</th>
<th>Valid</th>
<th>P_{start} = \left\lfloor S \left( \frac{1}{2} \right) - 1 + F - S \right\rfloor \frac{1}{2}</th>
<th>P_{start} \in [0, F - 1]</th>
<th>P_{end} = \left\lceil S \left( \frac{1}{2} \right) - 1 + F - S \right\rceil \frac{1}{2}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration</td>
<td>- No padding</td>
<td>- Padding such that feature map size has size ( \left\lfloor \frac{I}{2} \right\rfloor )</td>
<td>- Maximum padding such that end convolutions are applied on the limits of the input</td>
<td>- Also called 'half' padding</td>
</tr>
<tr>
<td>Purpose</td>
<td>- Drops last convolution if dimensions do not match</td>
<td>- Output size is mathematically convenient</td>
<td>- Filter 'sees' the input end-to-end</td>
<td></td>
</tr>
</tbody>
</table>

### Tuning hyperparameters

Parameter compatibility in convolution layer – By noting \( I \) the length of the input volume size, \( F \) the length of the filter, \( P \) the amount of zero padding, \( S \) the stride, then the output size \( O \) of the feature map along that dimension is given by:

\[
O = \frac{I - F + P_{start} + P_{end}}{S} + 1
\]

Remark: often times, \( P_{start} = P_{end} \equiv P \), in which case we can replace \( P_{start} + P_{end} \) by \( 2P \) in the formula above.

Receptive field – The receptive field at layer \( k \) is the area denoted \( R_k \times R_k \) of the input that each pixel of the \( k \)-th activation map can 'see'. By calling \( F_j \) the filter size of layer \( j \) and \( S_i \) the stride value of layer \( i \) and with the convention \( S_0 = 1 \), the receptive field at layer \( k \) can be computed with the formula:

\[
R_k = 1 + \sum_{j=1}^{k} \left( F_j - 1 \right) \prod_{i=0}^{j-1} S_i
\]

In the example below, we have \( F_1 = F_2 = 3 \) and \( S_1 = S_2 = 1 \), which gives \( R_2 = 1 + 2 \cdot 2 = 1 + 2 + 2 = 5 \).

Commonly used activation functions

Rectified Linear Unit – The rectified linear unit layer (ReLU) is an activation function \( g \) that is used on all elements of the volume. It aims at introducing non-linearities to the network. Its variants are summarized in the table below:
ReLU

\[ g(z) = \max(0, z) \]

Leaky ReLU

\[ g(z) = \max(\epsilon z, z) \]

with \( \epsilon \ll 1 \)

ELU

\[ g(z) = \max(\alpha(e^z - 1), z) \]

with \( \alpha \ll 1 \)

<table>
<thead>
<tr>
<th>Non-linearity complexities</th>
<th>biologically interpretable</th>
<th>Addresses dying ReLU issue for negative values</th>
<th>Differentiable everywhere</th>
</tr>
</thead>
</table>

**Softmax** – The softmax step can be seen as a generalized logistic function that takes as input a vector of scores \( z \in \mathbb{R}^n \) and outputs a vector of output probability \( p \in \mathbb{R}^n \) through a softmax function at the end of the architecture. It is defined as follows:

\[
p = \left( \frac{e^{x_1}}{\sum_{j=1}^{n} e^{x_j}}, \ldots, \frac{e^{x_n}}{\sum_{j=1}^{n} e^{x_j}} \right)
\]

where

\[
p_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}
\]

**Object detection**

**Types of models** – There are 3 main types of object recognition algorithms, for which the nature of what is predicted is different. They are described in the table below:

<table>
<thead>
<tr>
<th>Image classification</th>
<th>Classification w. localization</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional CNN</td>
<td>Simplified YOLO, R-CNN</td>
<td>YOLO, R-CNN</td>
</tr>
</tbody>
</table>

- Classifies a picture
- Predicts probability of object

- Detects object in a picture
- Predicts probability of object and where it is located

- Detects up to several objects in a picture
- Predicts probabilities of objects and where they are located

**Bounding box detection**

Detects the part of the image where the object is located

**Landmark detection**

- Detects a shape or characteristics of an object (e.g. eyes)
- More granular

Box of center \((b_x, b_y)\), height \(b_h\) and width \(b_w\)

Reference points \((l_{1x}, l_{1y}), \ldots, (l_{nx}, l_{ny})\)

**Intersection over Union** – Intersection over Union, also known as IoU, is a function that quantifies how correctly positioned a predicted bounding box \(B_p\) is over the actual bounding box \(B_a\). It is defined as:

\[
\text{IoU}(B_p, B_a) = \frac{B_p \cap B_a}{B_p \cup B_a}
\]

Remark: we always have \(\text{IoU} \in [0, 1]\). By convention, a predicted bounding box \(B_p\) is considered as being reasonably good if \(\text{IoU}(B_p, B_a) \geq 0.5\).

**Anchor boxes** – Anchor boxing is a technique used to predict overlapping bounding boxes. In practice, the network is allowed to predict more than one box simultaneously, where each box prediction is constrained to have a given set of geometrical properties. For instance, the first prediction can potentially be a rectangular box of a given form, while the second will be another rectangular box of a different geometrical form.

**Non-max suppression** – The non-max suppression technique aims at removing duplicate overlapping bounding boxes of a same object by selecting the most representative ones. After having removed all boxes having a probability prediction lower than 0.6, the following steps are repeated while there are boxes remaining:

1. Step 1: Pick the box with the largest prediction probability.
2. Step 2: Discard any box having an IoU \(\geq 0.5\) with the previous box.
YOLO – You Only Look Once (YOLO) is an object detection algorithm that performs the following steps:

- Step 1: Divide the input image into a $G \times G$ grid.
- Step 2: For each grid cell, run a CNN that predicts $y$ of the following form:

$$y = \left[ p_c, b_x, b_y, b_w, c_1, c_2, ..., c_p \right]^T \in \mathbb{R}^{G \times G \times k \times (5+p)}$$

where $p_c$ is the probability of detecting an object, $b_x, b_y, b_w$ are the properties of the detected bounding box, $c_1, c_2, ..., c_p$ is a one-hot representation of which of the $p$ classes were detected, and $k$ is the number of anchor boxes.

- Step 3: Run the non-max suppression algorithm to remove any potential duplicate overlapping bounding boxes.

Remark: when $p_c = 0$, then the network does not detect any object. In that case, the corresponding predictions $b_x, ..., c_p$ have to be ignored.

R-CNN – Region with Convolutional Neural Networks (R-CNN) is an object detection algorithm that first segments the image to find potential relevant bounding boxes and then run the detection algorithm to find most probable objects in those bounding boxes.

Remark: although the original algorithm is computationally expensive and slow, newer architectures enabled the algorithm to run faster, such as Fast R-CNN and Faster R-CNN.

### Face verification and recognition

#### Types of models – Two main types of model are summed up in table below:

<table>
<thead>
<tr>
<th>Face verification</th>
<th>Face recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Is this the correct person?</td>
<td>- Is this one of the $K$ persons in the database?</td>
</tr>
<tr>
<td>- One-to-one lookup</td>
<td>- One-to-many lookup</td>
</tr>
</tbody>
</table>

#### One Shot Learning – One Shot Learning is a face verification algorithm that uses a limited training set to learn a similarity function that quantifies how different two given images are. The similarity function applied to two images is often noted $d(image 1, image 2)$.

#### Siamese Network – Siamese Networks aim at learning how to encode images to then quantify how different two images are. For a given input image $x^{(i)}$, the encoded output is often noted as $f(x^{(i)})$.

#### Triplet loss – The triplet loss $\ell$ is a loss function computed on the embedding representation of a triplet of images $A$ (anchor), $P$ (positive) and $N$ (negative). The anchor and the positive example belong to the same class, while the negative example to another one. By calling $\alpha \in \mathbb{R}^+$ the margin parameter, this loss is defined as follows:

$$\ell(A,P,N) = \max(d(A,P) - d(A,N) + \alpha,0)$$

#### Neural style transfer

Motivation – The goal of neural style transfer is to generate an image $G$ based on a given content $C$ and a given style $S$. 

Stanford University

Winter 2019
Activate – In a given layer \(l\), the activation is noted \(a^{[l]}\) and is of dimensions \(n_H \times n_w \times n_c\).

Content cost function – The content cost function \(J_{\text{content}}(C,G)\) is used to determine how the generated image \(G\) differs from the original content image \(C\). It is defined as follows:

\[
J_{\text{content}}(C,G) = \frac{1}{2} \|a^{[l]}(C) - a^{[l]}(G)\|^2
\]

Style matrix – The style matrix \(G^{[l]}\) of a given layer \(l\) is a Gram matrix where each of its elements \(G^{[l]}_{kk'}\) quantifies how correlated the channels \(k\) and \(k'\) are. It is defined with respect to activations \(a^{[l]}\) as follows:

\[
G^{[l]}_{kk'} = \sum_{i=1}^{n_H} \sum_{j=1}^{n_w} a^{[l]}_{ijk} a^{[l]}_{ijk'}
\]

Remark: the style matrix for the style image and the generated image are noted \(G^{[l]}(S)\) and \(G^{[l]}(G)\) respectively.

Style cost function – The style cost function \(J_{\text{style}}(S,G)\) is used to determine how the generated image \(G\) differs from the style \(S\). It is defined as follows:

\[
J_{\text{style}}^{[l]}(S,G) = \frac{1}{(2n_Hn_wn_c)^2} \|(G^{[l]}(S) - G^{[l]}(G))\|_F^2 = \frac{1}{(2n_Hn_wn_c)^2} \sum_{k,k'=1}^{n_c} \left(G^{[l]}_{kk'}(S) - G^{[l]}_{kk'}(G)\right)^2
\]

Overall cost function – The overall cost function is defined as being a combination of the content and style cost functions, weighted by parameters \(\alpha, \beta\), as follows:

\[
J(G) = \alpha J_{\text{content}}(C,G) + \beta J_{\text{style}}(S,G)
\]

Remark: a higher value of \(\alpha\) will make the model care more about the content while a higher value of \(\beta\) will make it care more about the style.

Architectures using computational tricks

Generative Adversarial Network – Generative adversarial networks, also known as GANs, are composed of a generative and a discriminative model, where the generative model aims at generating the most truthful output that will be fed into the discriminative which aims at differentiating the generated and true image.