CSCI 5922 Fall 2022–HW3

Abulitibu Tuguluke

October 17, 2022

1 Fine-Tuning

(a)

We have a model, say VGG16 network, that had been train on millions of images and has 1000 classification, and we want to use it to classify the image of only 10 different animals (CIFAR), it is better to change the classifier layer from 1000 to 10 while retrain the data, that is the idea of transfer learning.
Fine-tuning wants to achieve the same goal: Why not use a 'invented wheel' model, by 'modified' the architecture to solve 'tailored' problem, to speed up the process and get better result.
In R-CNN's case, fine-tuning is implemented by replacing the final layer with a 'user-defined' neural net classifier(choice of selected number of classes), and re-train to update the weights.

(b)

For fully Convolutional semantic segmentation, the motivation is similar, to transfer the 'learned representation'[1] by fine-tuning, so that to have better classification predictions.
Before adding skip connections, ahead of the pixel prediction/segmentation g.t., fine-tuning is implemented by training to update the weight using back-propagation.

2 Recurrent Neural Networks; i.e., RNNs

(a)

- One-to-many: In hand-written mathematical formulas' recognition, one section of the image input to math symbol/oprators/structures etc, more if it is a word problem.
- Many-to-many: RNN time-series regression application using SMAP[2] data to predict weather, all soil moisture/wind speed/rain/carbons etc, 'mapping(classify)' to future soil moisture/wind speed/rain/-carbons point.
- Many-to-one: Same as last application,but instead of 'mapping' soil moisture/wind speed/rain/carbons, only to price of sugar.

(b)

We will use the applications from the previous section.
The inputs are encoded/embedding vectors for info representation, in One-to-many case, it could be a tokenized image for particular math handwritten symbol. For Many-to-many and Many-to-one, they could be encoded vectors for numerical inputs of soil moisture, wind speed, temperatures in different time step.
The Outputs are decoded representation of predicted values. in One-to-many case, they are the math symbol in that image, along with the symbol or operations followed by such, they could also include the multiple

[1]Fully Convolutional Networks for Semantic Segmentation
[2]`https://smap.jpl.nasa.gov/data/`

guesses for n-gram prediction. For Many-to-many, they are the numerical values for the predicted(future) soil moisture, wind speed, temperatures etc.. In Many-to-many's case, it is the value for sugar commodity price (or yield) in trading season, or even binary: Will the stock price be up or down?
The architectures follows the RNN principle: Many-to-many and Many-to-one's hidden layer h_{i+1} is fed by h_i's info for weight learning, we can pick software for many-out, Sigmoid for one.

(c)

If we assume input and output are the same size while ignoring the biases, then the number of model parameters will quadruples as well, since W, U, V are shared.

3 Attention

$$\underset{\text{Input A}}{\begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline\end{array}} \qquad \underset{\text{Input B}}{\begin{array}{|c|c|c|}\hline 2 & 0 & 1 \\ \hline\end{array}}$$

$$\underset{\text{Query weights}}{\begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & 1 \\ \hline 1 & 2 \\ \hline\end{array}} \qquad \underset{\text{Key weights}}{\begin{array}{|c|c|}\hline 2 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline\end{array}} \qquad \underset{\text{Value weights}}{\begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & 1 \\ \hline 1 & 2 \\ \hline\end{array}}$$

Figure 1: Inputs and weight matrices for computing self-attention.

$$\text{Input A:}\begin{cases} W_{Q_A}: \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & 1 \\ \hline 1 & 2 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline\end{array} \\ W_{K_A}: \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 2 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline\end{array} \\ W_{V_A}: \begin{array}{|c|c|c|}\hline 0 & 1 & 0 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & 1 \\ \hline 1 & 2 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 2 & 1 \\ \hline\end{array} \end{cases}, \text{Input B:}\begin{cases} W_{Q_B}: \begin{array}{|c|c|c|}\hline 2 & 0 & 1 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 1 & 2 \\ \hline 0 & 1 \\ \hline 1 & 2 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 3 & 6 \\ \hline\end{array} \\ W_{K_B}: \begin{array}{|c|c|c|}\hline 2 & 0 & 1 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 2 & 0 \\ \hline 1 & 0 \\ \hline 2 & 1 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 6 & 1 \\ \hline\end{array} \\ W_{V_B}: \begin{array}{|c|c|c|}\hline 2 & 0 & 1 \\ \hline\end{array} \times \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 2 & 1 \\ \hline 1 & 2 \\ \hline\end{array} = \begin{array}{|c|c|}\hline 3 & 2 \\ \hline\end{array} \end{cases}$$

Attention score(query-key similarity):

$$\begin{cases} W_{Q_A} \times W_{K_A} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline\end{array} \times \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline\end{array} = 0 \\ W_{Q_A} \times W_{K_B} = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline\end{array} \times \begin{array}{|c|}\hline 6 \\ \hline 1 \\ \hline\end{array} = 1 \end{cases}, \begin{cases} W_{Q_B} \times W_{K_B} = \begin{array}{|c|c|}\hline 3 & 6 \\ \hline\end{array} \times \begin{array}{|c|}\hline 6 \\ \hline 1 \\ \hline\end{array} = 24 \\ W_{Q_B} \times W_{K_A} = \begin{array}{|c|c|}\hline 3 & 6 \\ \hline\end{array} \times \begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline\end{array} = 3 \end{cases}$$

Attention weights without scale:

$$\begin{cases} \text{Input A: } \begin{array}{|c|c|}\hline 0 & 1 \\ \hline\end{array} \\ \text{Input B: } \begin{array}{|c|c|}\hline 3 & 24 \\ \hline\end{array} \end{cases}$$

Resulting representations of the input tokens from self-attention:

$$\begin{cases} \text{Input A: } 0 \times W_{V_A} + 1 \times W_{V_B} = 0 \times \boxed{2 \mid 1} + 1 \times \boxed{3 \mid 2} = \boxed{3 \mid 2} \\ \text{Input B: } 3 \times W_{V_A} + 24 \times W_{V_B} = 3 \times \boxed{2 \mid 1} + 24 \times \boxed{3 \mid 2} = \boxed{78 \mid 51} \end{cases}$$

4 Transformers

RNN training normally 'unrolled' in time, that is, hiiden layer h_{i+1} had to wait for h_i, while Transformer can process all input at the same time, that is, each step's computation is independent of other steps, which make it adaptable for parallelization.
Transformer also has new mechanism such as self-attention and positional encoding, something like a 'time representation', which we don't see in RNN.