Problem Set 3
CSCI 5922: Neural Networks and Deep Learning

Rajeev R Menon (110581437)

November 3, 2022

Problem 1

Part a

Feature identification in deep learning is usually done by the first few layers. For similar tasks, the last few layers can be kept the same while keeping the first few layers different [1]. Fine-tuning involves the use of knowledge gained during the training of one particular model in another model. Specifically, it involves the use of weights from pre-trained models in a new model. Fine-tuning is usually used to accelerate machine learning by reducing training times. In the case of R-CNNs for object detection, there is not enough labeled data for training a large CNN. This lack of labeled data in the case of R-CNNs was tackled by supervised pre-training on the ImageNet Large Scale Visual Recognition Challenge dataset and domain-specific fine-tuning on the PASCAL Visual Object Classes dataset [2].

Part b

Training all the weights from scratch takes a lot of time to form the base classification network. Fine-tuning the output classifier alone gives only a fraction of the full fine-tuning performance. Therefore, in fully convolutional semantic segmentation models, fine-tuning is done on all layers by back-propagation through the whole neural network [3].

Problem 2

Part a

Image searches in popular search engines like Google can be quoted as an example of a one-to-many problem. Here a single image is usually recognized and described using tokens that are used to query relevant results.

Forecasting the value of a particular stock can be used as an example of a many-to-one problem. Here a single value, the stock price, is predicted from multiple trends and factors.

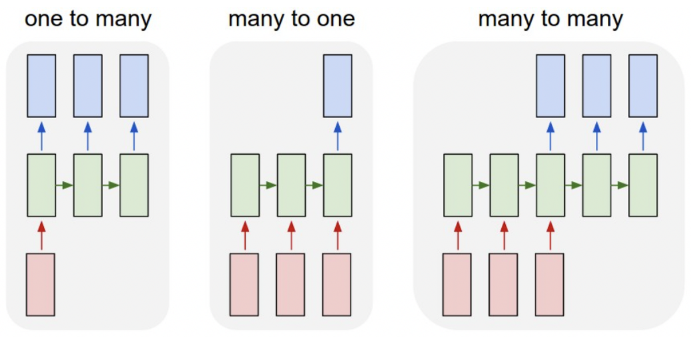

Figure 1: Types of RNNs [4]

Document generation in the field of banking involves the many-to-many problem. Here forms and documents containing multiple words are generated based on client information specific to a person.

Part b

Recurrent neural networks for the one-to-many problem take in single input and predicts a sequence. RNNs for many-to-one problems take a sequence as input and output a single value. In neural networks for the many-to-many problem, both the input and output are sequences. All three problems are represented in Figure 1. The rectangles represent various vectors and the arrows represent functions. The input is colored red while the output is colored blue [3].

Part c

The recurrent neural network in question consists of single input, hidden, and output layers. In the case of RNNs, the same weights are run across each input time step. Quadrupling the input sequence in this case adds to the number of input time steps. Since each input time step is run sequentially through the same weights, the number of model parameters would be unaffected.

Problem 3

Part a

$$\begin{aligned} Key\,matrix\,for\,Input\,A &= \begin{bmatrix}0 & 1 & 0\end{bmatrix} . \begin{bmatrix}2 & 0\\1 & 0\\2 & 1\end{bmatrix} \\ &= \begin{bmatrix}(0\times 2+1\times 1+0\times 2) & (0\times 0+1\times 0+0\times 1)\end{bmatrix} \\ &= \begin{bmatrix}1 & 0\end{bmatrix} \end{aligned}$$

$$\begin{aligned} Key\, matrix\, for\, Input\, B &= \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 2 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (2 \times 2 + 0 \times 1 + 1 \times 2) & (2 \times 0 + 0 \times 0 + 1 \times 1) \end{bmatrix} \\ &= \begin{bmatrix} 6 & 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} Query\, matrix\, for\, Input\, A \ &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} (0 \times 1 + 1 \times 0 + 0 \times 1) & (0 \times 2 + 1 \times 1 + 0 \times 2) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} Query\, matrix\, for\, Input\, A \ &= \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} (2 \times 1 + 0 \times 0 + 1 \times 1) & (2 \times 2 + 0 \times 1 + 1 \times 2) \end{bmatrix} \\ &= \begin{bmatrix} 3 & 6 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} Value\, matrix\, for\, Input\, A &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} . \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} (0 \times 1 + 1 \times 2 + 0 \times 1) & (0 \times 0 + 1 \times 1 + 0 \times 2) \end{bmatrix} \\ &= \begin{bmatrix} 2 & 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} Value\, matrix\, for\, Input\, B &= \begin{bmatrix} 2 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \\ &= \begin{bmatrix} (2 \times 1 + 0 \times 2 + 1 \times 1) & (2 \times 0 + 0 \times 1 + 1 \times 2) \end{bmatrix} \\ &= \begin{bmatrix} 3 & 2 \end{bmatrix} \end{aligned}$$

$$\begin{aligned}
Attention\,weights\,for\,Input\,A &= softmax(Query\,matrix\,for\,Input\,A\,.\,Key\,matrix\,for\,Input\,A, \\
&\quad Query\,matrix\,for\,Input\,A\,.\,Key\,matrix\,for\,Input\,B) \\
&= softmax(\begin{bmatrix}0 & 1\end{bmatrix} \times \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0 & 1\end{bmatrix} \times \begin{bmatrix}6\\1\end{bmatrix}) \\
&= softmax(\begin{bmatrix}0 \times 1 + 1 \times 0\end{bmatrix}, \begin{bmatrix}0 \times 6 + 1 \times 1\end{bmatrix}) \\
&= softmax(0, 1) \\
&= \begin{bmatrix}\frac{e^0}{e^0+e^1} & \frac{e^1}{e^0+e^1}\end{bmatrix} \\
&= \begin{bmatrix}0.2689 & 0.7310\end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
Attention\,weights\,for\,Input\,B &= softmax(Query\,matrix\,for\,Input\,B\,.\,Key\,matrix\,for\,Input\,A, \\
&\quad Query\,matrix\,for\,Input\,B\,.\,Key\,matrix\,for\,Input\,B) \\
&= softmax(\begin{bmatrix}3 & 6\end{bmatrix} . \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}3 & 6\end{bmatrix} . \begin{bmatrix}6\\1\end{bmatrix}) \\
&= softmax(\begin{bmatrix}3 \times 1 + 6 \times 0\end{bmatrix}, \begin{bmatrix}3 \times 6 + 6 \times 1\end{bmatrix}) \\
&= softmax(3, 24) \\
&= \begin{bmatrix}\frac{e^3}{e^3+e^(24)} & \frac{e^(24)}{e^3+e^(24)}\end{bmatrix} \\
&= \begin{bmatrix}7.58E - 10 & 0.9999\end{bmatrix}
\end{aligned}$$

Part b

$$\begin{aligned}
Representation\,for\,input\,A &= Attention\,Weight\,0\,for\,Input\,A.Value\,Matrix\,For\,Input\,A \\
&+ Attention\,Weight\,1\,for\,Input\,A.Value\,Matrix\,For\,Input\,A \\
&= 0.2689\,.\begin{bmatrix}2 & 1\end{bmatrix} + 0.7311\,.\begin{bmatrix}3 & 2\end{bmatrix} \\
&= \begin{bmatrix}0.5378 & 0.2689\end{bmatrix} + \begin{bmatrix}2.1933 & 1.4622\end{bmatrix} \\
&= \begin{bmatrix}2.7311 & 1.7311\end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
Representation\,for\,input\,B &= Attention\,Weight\,0\,for\,Input\,B.Value\,Matrix\,For\,Input\,B \\
&+ Attention\,Weight\,1\,for\,Input\,B.Value\,Matrix\,For\,Input\,B \\
&= 7.5E{-}10.\ \begin{bmatrix}2 & 1\end{bmatrix} + 1 \times \begin{bmatrix}3 & 2\end{bmatrix} \\
&= \begin{bmatrix}7.5E{-}10 \times 2 & 7.5E{-}10 \times 1\end{bmatrix} + \begin{bmatrix}1 \times 3 & 1 \times 2\end{bmatrix} \\
&= \begin{bmatrix}3 & 2\end{bmatrix}
\end{aligned}$$

Problem 4

In transformer neural networks, sentences are processed as a single unit instead of one word or token at a time. Therefore transformers do not rely on past hidden states to capture dependencies with previous words. Recurrent neural networks require sequential processing of words or tokens. In RNNs past information is retained through past hidden states, each state dependent on previously seen states. Therefore transformer networks can be trained in parallel and are faster than RNNs that have to be trained sequentially.

Since RNNs use past hidden states to retain information, the encoding of specific words is retained only for a single step. This means that encoding of a word directly affects only the representation of the next word, and its effect is diminished after a few steps. This problem of recurrence is solved in transformers using positional embeddings. Transformer networks use fixed pre-learned weights that encode information related to the position of a word in a sentence.

References

[1] Too, Edna Chebet, et al. "A comparative study of fine-tuning deep learning models for plant disease identification." Computers and Electronics in Agriculture 161 (2019): 272-279.

[2] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[3] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[4] Karpathy, Andrej. "The Unreasonable Effectiveness of Recurrent Neural Networks" http://karpathy.github.io/2015/05/21/rnn-effectiveness. May 21, 2015.