Amr Mohamed Abd Albadee

Dr/ Mohamed El Shenawy

Neural Networks and Deep Learning

Final Assessment Report

Question 1

1) FasterR-CNN

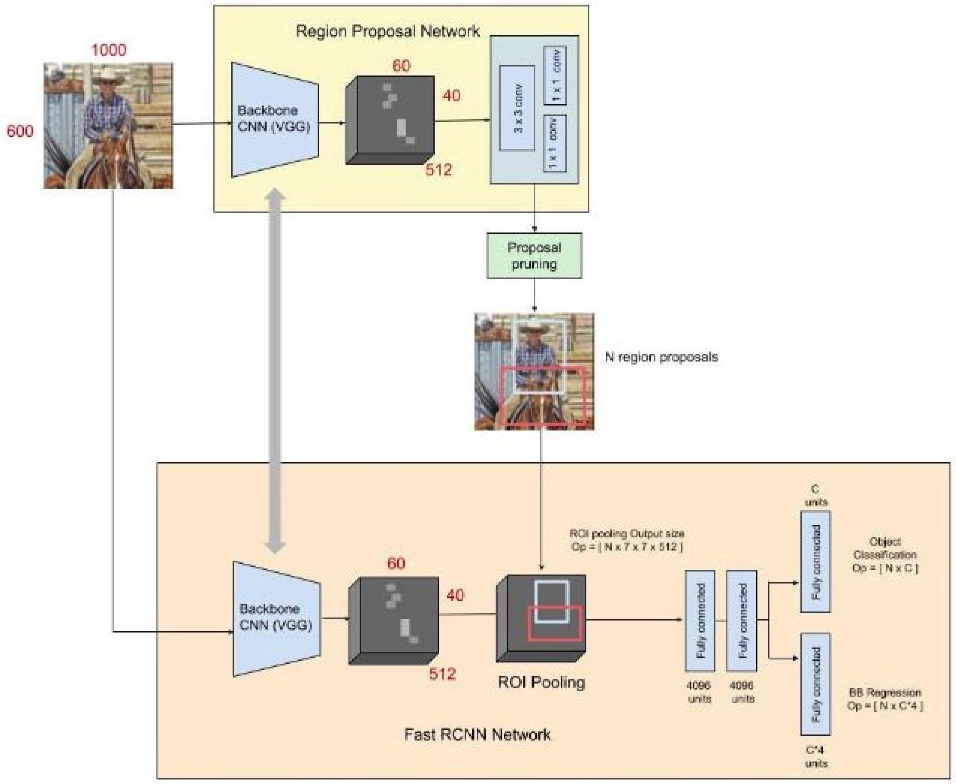
- **Basic Operation:** Bounding boxes are generated first to detect the image's objects , using a regional algorithm. Then, a set of features are extracted from those objects using CNN. Then, classify those objects to suitable classes. Finally, making object boundaries more precise using regression technique.
- **Architecture:** it consists of 2 parts, first the detection pipeline using RPN, then a detector network of Fast R-CNN . **Here is what happens exactly in the RPN network:**
    1. The input image is sized to 600px * 1000px
    2. Image is pipelined to a backbone network that has stride of 16, so every two adjacent pixels in feature output resembles 2 points who are 16px far away in the input.
    3. For each location on the feature map output , a set of anchors is created in the original input image. Anchors who have different sizes and aspect ratios, are used to estimate probable objects, also their location and size. A set of 9 anchors is being used in this paper.

4. Checking spanning anchors and refining them to give object proposals.
5. 512 convolutional filters of size 3*3 which gives a feature map of 512 dimension for every backbone network output.
6. 18 convolutional filters of size 1*1 which gives containing-object probabilities for every point in the backbone network output.
7. Regression layer of 36 neurons which gives 4 regression coefficients for every anchor, and helps anchor refining their coordinations.

Then the detector or FastR-CNN architecture are below :

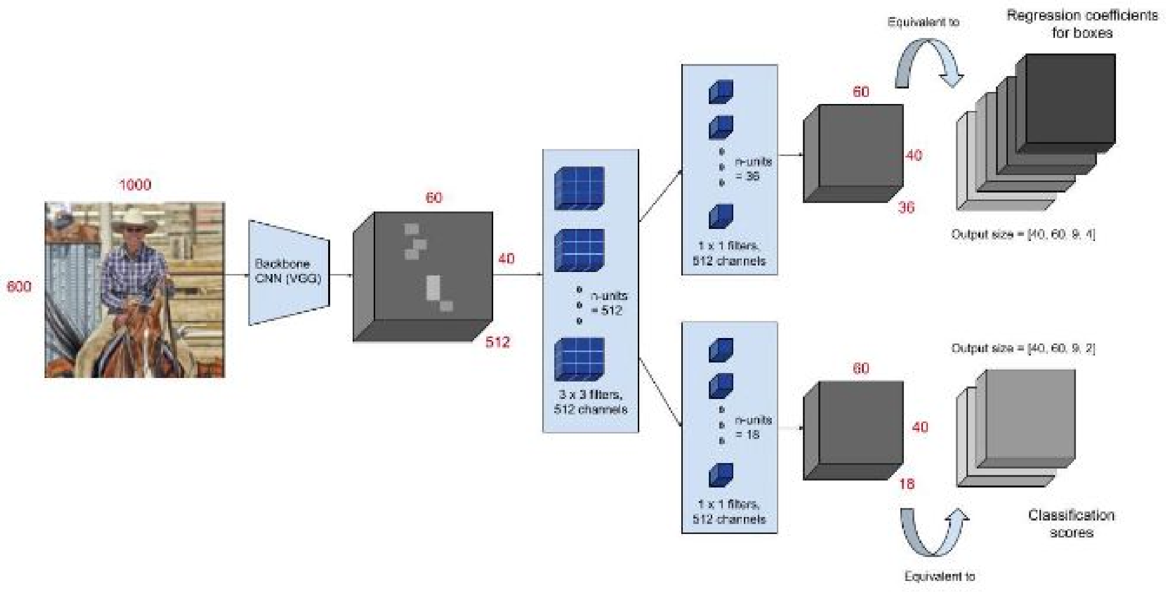
1. Input image is pipelined to the backbone CNN, getting a feature map of size (60,40,150).
2. The ROI pooling layer takes proposals from RPN and their corresponding regions, divides those regions to subwindows and performs a max-pooling operation on them.
3. Output of the ROI layer is passed through 2 fully connected layers of 4096 neurons each.
4. Classification layer of C units, each unit corresponding to seperate class in the task, its neurons has softmax activation function to get scores.
5. Specific Regressor for each class, which have 4 parameters , help to refine and improve boxes boundaries.

   Here is the shape of the architecture of FasterR-CNN,

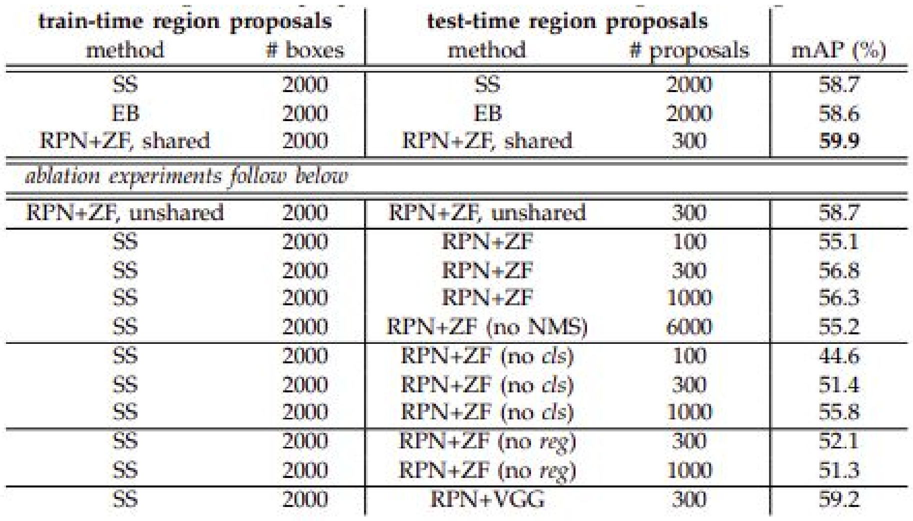


- Training procedure: RPN is trained independently, initialized weights of backbone CNN are from imagenet network, and fine tuning the backbone to do regional proposals. Fast R-CNN is trained also independently, initialized weights of backbone CNN are from imagenet network, and fine tuning the backbone to do object detection. Through sharing weights, RPN weights are initialized from fasterR-CNN weights, but the common layer weights between the two networks don't change. Finally, the detector is tuned through RPN weights.
- Design Consideration: using RPN "regional proposal network" handles computational cost of selective search algorithm in R-CNN, as it takes around 2

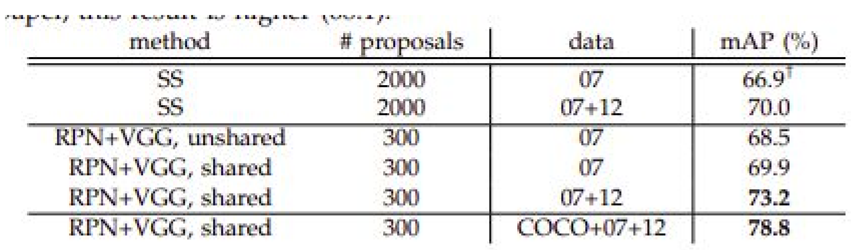
seconds per image. The cost becomes only 10 ms for image, and layers become shareable between regional stage and following stages, which helps more in the feature extraction process. Here is the architecture of RPN.


Equivalent to
Regression coefficients for boxes
1000
600
Backbone CNN (VGG)
60
40
512
n-units = 512
3 x 3 filters, 512 channels
n-units = 36
1 x 1 filters, 512 channels
60
40
36
Output size = [40, 60, 9, 4]
Output size = [40, 60, 9, 2]
n-units = 18
1 x 1 filters, 512 channels
60
40
18
Classification scores
Equivalent to

- Performance on benchmark datasets: the model was trained on PASCAL dataset. RPN+VGG backbone as a network itself without sharing weights with FastR-CNN , over performed selective search algorithm (SS). With sharing weights, using VGG or ZF along with RPN over performed SS. classification Branch was an important factor, as it maintained high recall vs IoU overlap ratio. Using 3 scales with a single aspect ratio, performs well as changing the aspect ratio for every scale, and it is better to use many anchors “3 in this case” than using only one anchor. [1]



train-time region proposals method	# boxes	test-time region proposals method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2



method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Question 2

Attention model: was designed for machine Translation, but spreaded to many other applications like text summarization and image captioning. The idea comes from that humans only focus on some words to do translation, as minds cannot memorize long sequences. So, the human takes group by group to do the translation as each group is related and expresses certain meaning. The model computes a set of attention weights, as every word of the sequence has its weight. It shows the context with more importance as important context has big weights and non-relative or unimportant one has small weights as it does not affect much on the process.

Often bidirectional LSTM is used to compute instance feature vectors. each vector contains features computed from for recurrence step activation and backward recurrence step activation, those features are for every instance. After that, a unidirectional LSTM is used, in which every block takes context and the last block output as its input, and its output is one word in the sequence. The context is formalized from summation of weighted features vectors for many time steps by their attention weights. The all weights are summed up to one. A neural network is used to learn the function used to generate those weights. The disadvantage of the algorithm is that it runs at quadratic cost. [2]

Transforms: famous architecture of neural networks that has been used recently in sequence to sequence models and wide applications like speech recognition and machine translation. For humans, it is easy to know the meaning of pronouns or what they refer to , humans do that through understanding the context. These dependencies are important in the process of translation or any other task. RNN and CNN are used to deal with that dependency problem. RNN is ineffective in long sequences , as it doesn’t have memory to store relevant information from previous contexts. In LSTM, the information goes through cell states, where it can select relevant information and keep them.but again LSTM can be ineffective when dealing with long sequences, as the probability of keeping words from a long period decreases as sequence increases. Also, a big problem appeared here, as parallelization cannot be done, as each word depends on the last word in the sequence. CNN can work in parallel, as each word can be processed in the same time which previous or next word are processed. Transforms are a combination of CNN and attention models. Encoders consist of two layers which are self-attention and feed forward network FNN. Decoders on the other hand, consists of the same two layers, in addition to a third layer between them called Encoder-Decoder attention. A self-attention layer in Encoders helps them to keep attention to other words while processing a certain word. On the other hand, Encoder-Decoder attention layer in Decoders helps them to focus on certain important relevant parts in the sequence. Every word in the encoders passes through a certain path, dependencies are formed in the attention layer, and those dependencies disappear in the FNN layer. This is how encoders do parallelization while keeping dependencies.

Question 3

Transfer learning is about adapting some knowledge learned over one distribution to a new distribution. It shows the ability of the model to generalize. Using a pre-trained model on a large dataset, the model can learn spatial feature hierarchy which can be then generalized over other tasks over the visual world. By generalization , I mean and that is mostly the case, that the task can be different. For example, we can train on animal classes, but apply the trained model then on humans or tools or any visual classes! That is because many visual entities share common features or basic concepts like edges.

Question 4

Adagrad

The algorithm computes hyperparameters learning rates individually. It is used for large scale problems where it has many hyperparameters, and learning rates cannot be chosen manually. It ensures that each learning rate for each dimension is neither too slow nor too volatile. It uses the data geometry knowledge of past iterations to set lower rates for frequent features “parameters with large derivatives of their loss function” and set higher rates for less frequent ones“parameters with small derivatives of their loss function”. But to consider, accumulation of squared gradients may cause a huge decrease to the learning rate.

<u>RMSProp with Nesterov</u>

<u>Nesterov Momentum</u>

Standard Momentum calculates gradient at current location, then changes to updated accumulated gradient direction. But Nesterov, make a jump in the previous accumulated gradient direction and make a correction then.

<u>RMSProp</u>

AdaGrad converges rapidly in convex function, but when applied to non-convex one, the learning rate will eventually reach a locally convex bowl. According to AdaGrad, the learning rate becomes too small before reaching this point which will cause a huge problem. RMS on the other hand, changes the accumulated gradient to exponentially weighted moving average in which it discards extreme past history, that change leads to rapid convergence after finding the convex bowl. So, RMSpop performs better in non-convex functions. But to consider, RMSProp estimation of second-order moment may have bias in the beginning of training.

RMSProp with Nesterov combines RMSProp algorithm with Nesterov Momentum technique.

<u>Adam</u>

An algorithm that combines the benefits of the last two algorithms (AdaGrad and RMSprop). Adam computes an exponential moving gradient for both the gradient and the square of it “ it has parameters B1 and B2 , they control the rate of decay of those averages”.

Momentum is used in Adam directly in the exponential moving gradient of the first order moment. Adam also has an important feature which is bias correction, it handles the initialization problem of both first and second order moments. So, Adam is considered as fairly robust to hyperparameters choice.

L-BFGS

BFGS algorithm tries to decrease the computation cost of Newton's method. The algorithm is similar to the conjugate gradient algorithm but its update rule is like Newton's method. Instead of massive cost calculation of the inverse Hessian H^{-1} in Newton's method, BFGS approximates it with another matrix M_t , that matrix is refined many times until it is a good approximation of the hessian. After that a direction of p_t is determined, a line search is done in that direction to calculate the step size. The problem here is that the algorithm has to store the M_t matrix which requires complexity of $O(n^2)$ in the memory. LBFGS handles this problem by using the assumption that $M^{(t-1)}$ is an identity matrix. That assumption avoids storing the M matrix at each step. Directions of line search are mutually conjugate. The algorithm can store some of vectors and use them to update M, rather than storing it. So the complexity here is only O(n).

References

[1] Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

[2] Brief Introduction to Attention Models

[3] https://towardsdatascience.com/transformers-141e32e69591

[4] Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning Book.