دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: تکلیف چهارم درس یادگیری عمیق

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم‌سال تحصیلی: پاییز ۱۴۰۲

مدرّس: دکتر سمانه حسینی سمنانی

دستیاران آموزشی: مریم محمدی-علی بزرگ زادارباب

۱

Two common approaches are as follows:

1.1 Develop Model Approach

1. Select Source Task. You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.

2. Develop Source Model. Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.

3. Reuse Model. The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

4. Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

1.2 Pre-trained Model Approach

1. Select Source Model. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.

2. Reuse Model. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.

3. Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

This second type of transfer learning is common in the field of deep learning.

The paper is employing the ”Pre-trained Model Approach” for transfer learning. Here are the key points supporting this conclusion:

1. Selection of Source Model: The abstract mentions the use of a ”high-capacity convolutional neural network (CNN)” for the proposed detection algorithm. The paper references the success of Krizhevsky et al. [25] in 2012, which rekindled interest in CNNs and achieved higher image classification accuracy on the ImageNet challenge.

2. Reuse of Pre-trained Model: The abstract explicitly mentions the approach of ”supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning,” indicating the utilization of a pre-trained model on a

related task. The paper mentions that their method combines ”region proposals with CNNs,” and they name their approach ”R-CNN: Regions with CNN features.” This suggests the adoption of a pre-trained CNN for feature extraction.

3. Fine-tuning: The abstract mentions the second key insight that ”when labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost.”

4. Comparison with Another Model: The paper compares their R-CNN approach with OverFeat, a sliding-window detector based on a similar CNN architecture, and reports that R-CNN outperforms OverFeat.

In summary, the paper is using a pre-trained convolutional neural network for object detection, and it emphasizes the effectiveness of supervised pre-training on an auxiliary task followed by domain-specific fine-tuning to boost performance.

۲

Here are the steps you can follow:

1. Annotate Bounding Boxes: For each image in your dataset, you need to annotate the bounding boxes around the objects (cats and dogs in this case). The bounding box annotations should include the coordinates of the box, typically represented as (xmin, ymin, xmax, ymax).

2. Update Data Pipeline: Modify your data loading and preprocessing pipeline to handle the new bounding box annotations. Your input data should now include both the image and the corresponding bounding box information.

۳

In the context of object detection, several challenges need to be addressed, and modifications to the loss function are necessary to handle these challenges. Let’s discuss the problems and corresponding modifications based on the Fast R-CNN paper:

1. Localization Accuracy: Problem: Accurate localization of objects within an image is crucial. Object detection involves predicting bounding boxes that tightly enclose the objects of interest. Modification: The paper introduces a bounding box regression loss to improve localization accuracy. This loss helps the model learn to refine the predicted bounding box coordinates.

2. Multi-Object Detection: Problem: An image may contain multiple objects of different classes, and the model needs to detect and classify each of them accurately. Modification: The Fast R-CNN architecture includes a multi-task loss function that jointly optimizes for both classification and bounding-box regression. This modification allows the model to handle multiple object instances in a single forward pass.

3. Proposal Generation and Processing: Problem: Generating and processing a large number of region proposals for object detection can be computationally expensive. Modification: The RoI (Region of Interest) pooling layer is introduced to efficiently process region proposals. This layer extracts fixed-length feature vectors from the feature map, making the computation more tractable.

4. Training Speed and Efficiency: Problem: Training deep networks for object detection can be time-consuming, particularly when using multi-stage pipelines like R-CNN. Modification: The Fast R-CNN approach streamlines the training process into a single-stage algorithm, allowing for faster training. The hierarchical sampling strategy and shared computation during training further enhance efficiency.

5. Scale Invariance: Problem: Objects in images can vary in size, and the model needs to be invariant to these scale differences. Modification: The paper explores two approaches for achieving scale invariance. One is through ”brute force” learning, where the network directly learns scale invariance during training. The other approach uses image pyramids during both training and testing to provide approximate scale invariance.

6. Loss Function Modifications: Problem: Traditional classification loss may not be sufficient for object detection, where precise localization is crucial. Modification: The Fast R-CNN introduces a multi-task loss function that includes both a classification loss (log loss) and a bounding-box regression loss. The classification loss penalizes incorrect class predictions, and the regression loss penalizes errors in bounding box predictions.

In summary, Fast R-CNN addresses multiple challenges in object detection, including localization accuracy, handling multiple objects, efficient proposal processing, training speed, and scale invariance. The modifications in the loss function involve a multi-task approach, combining classification and bounding-box regression losses to train the model effectively for both tasks.

۴

Accuracy in Object Detection

In object detection, accuracy is typically measured using metrics that consider both the classification and localization aspects of the detected objects. Common metrics include:

Intersection over Union (IoU)

IoU measures the overlap between the predicted bounding box and the ground truth bounding box. It is calculated as the area of intersection divided by the area of the union. The IoU is calculated as: $IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$ A common IoU threshold for considering a detection as correct is 0.5, but this threshold can vary depending on the application.

Precision, Recall, and F1 Score

Precision is the ratio of true positive detections to the total number of predicted positive detections. Recall is the ratio of true positive detections to the total number of ground truth positive instances. F1 Score is the harmonic mean of precision and recall.

Precision, Recall, and F1 Score are defined as follows: Precision $= \frac{\text{True Positives}}{\text{True Positives + False Positives}}$
Recall $= \frac{\text{True Positives}}{\text{True Positives + False Negatives}}$
F1 Score $= 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}}$

Mean Average Precision (mAP)

mAP is a widely used metric for evaluating the performance of object detection models. It considers precision and recall across different levels of confidence thresholds for object detections. Here are the steps involved in calculating mAP:

Precision-Recall Curve

For each class, calculate precision and recall at different confidence thresholds. Precision is computed as the ratio of true positive detections to the total number of predicted positive detections at a given threshold. Recall is computed as the ratio of true positive detections to the total number of ground truth positive instances at a given threshold. Plot precision-recall pairs to form a curve.

Interpolation of Precision-Recall Curve

Interpolate the precision values at a set of equally spaced recall levels (e.g., recall levels from 0 to 1 with small steps). This interpolation helps smooth the precision-recall curve.

Average Precision (AP)

Compute the area under the interpolated precision-recall curve. AP represents the average precision across different recall levels for a specific class. $AP =$ Area under Precision-Recall Curve

Mean Average Precision (mAP)

The mean of the AP values across all classes is calculated: $mAP = \frac{\sum_{i=1}^{C} AP_i}{C}$ where C is the number of classes.

mAP provides a comprehensive evaluation of object detection performance by considering both precision and recall at various confidence thresholds. It is particularly useful when dealing with datasets with imbalanced class distributions and varying levels of difficulty in detection.

۵

This term penalize the bounding box with inacurate height and width. The square root is present so that erors in small bounding boxes are more penalizing than errors in big bounding boxes. In other words, w_i and h_i are under square-root. This is done to penalize the smaller bounding boxes as we need better prediction on smaller objects than on bigger objects (author's call). Check out the table below and observe how the smaller values are punished more if we follow "square-root" method (look at the inflection point when we have 0.3 and 0.2 as the input values) (PS: I have kept the ratio of var1 and var2 same just for explanation):

var1	**var2**	**(var1 - var2)^2**	**(sqrtvar1 - sqrtvar2)^2**
0.0300	0.020	9.99e-05	0.001
0.0330	0.022	0.00012	0.0011
0.0693	0.046	0.000533	0.00233
0.2148	0.143	0.00512	0.00723
0.3030	0.202	0.01	0.01
0.8808	0.587	0.0862	0.0296
4.4920	2.994	2.2421	0.1512

۶

Vanishing and exploding gradient problems are common challenges associated with training Recurrent Neural Networks (RNNs).

6.1 Vanishing Gradient:

1. Issue: During backpropagation through time (BPTT), gradients may become extremely small as they are propagated through many time steps. This makes it difficult for the network to learn long-term dependencies.
2. Cause: The repeated multiplication of small gradient values (typically less than 1.0) during BPTT leads to vanishing gradients.
3. Consequence: The model fails to capture information from earlier time steps, limiting its ability to learn sequential patterns over long distances.

6.2 Exploding Gradient:

1. Issue: Gradients can also become extremely large during BPTT, leading to numeric instability and difficulty in training the network.

2. Cause: The repeated multiplication of large gradient values (typically greater than 1.0) during BPTT results in exploding gradients.

3. Consequence: Large gradients may cause the model parameters to be updated excessively, leading to poor convergence and potential divergence during training.

6.3 Solving Exploding Gradient Problem:

Solving the exploding gradient problem is relatively straightforward, and a common approach is to use gradient clipping. Gradient clipping involves scaling down the gradients if they exceed a predefined threshold.

۷

1. What is C_{t-1}?

C_{t-1} represents the cell state at the previous time step ($t-1$) in the LSTM. The cell state is a memory unit that can store information over long sequences, allowing the LSTM to capture long-term dependencies in data. C_{t-1} carries information from the past and is updated and modified during the current time step based on input data and gating mechanisms.

2. Why do we use the sigmoid function for making f_t?

In the LSTM, f_t is the forget gate activation vector at time step t. The forget gate's purpose is to decide what information from the previous cell state (C_{t-1}) should be discarded or retained for the current time step. The sigmoid function is used to create the forget gate activation vector because it squashes values between 0 and 1, allowing the network to control the flow of information. If a component of f_t is close to 0, it means the LSTM should forget the corresponding information, and if it's close to 1, the information should be retained.

3. What is the concept of $C_{t-1} * f_t$?

The term $C_{t-1} * f_t$ represents an element-wise multiplication between the previous cell state (C_{t-1}) and the forget gate activation vector (f_t). This operation determines which information from the previous cell state should be preserved and which should be discarded. Element-wise multiplication scales each component of the cell state (C_{t-1}) by the corresponding component of the forget gate (f_t). If a component of f_t is close to 0, the information in the corresponding component of C_{t-1} will be scaled down or "forgotten." If f_t is close to 1, the information will be retained.

منابع

[1] https://machinelearningmastery.com/transfer-learning-for-deep-learning/

[2] https://stats.stackexchange.com/questions/287486/yolo-loss-function-explanation