Continual Stereo Matching of Continuous Driving Scenes
with Growing Architecture – Supplementary Material

Chenghao Zhang\textsuperscript{1,2}, Kun Tian\textsuperscript{1,2}, Bin Fan\textsuperscript{3}, Gaofeng Meng\textsuperscript{1,2,4,}\textsuperscript{*}, Zhaoxiang Zhang\textsuperscript{1,4}, Chunhong Pan\textsuperscript{1}
\textsuperscript{1} National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
\textsuperscript{2} School of Artificial Intelligence, University of Chinese Academy of Sciences
\textsuperscript{3} School of Automation and Electrical Engineering, University of Science and Technology Beijing
\textsuperscript{4} CAS Centre for Artificial Intelligence and Robotics, HK Institute of Science and Innovation
{chenghao.zhang, kun.tian, gfmeng, zzhang, chpan}@nlpr.ia.ac.cn, {bin.fan}@iee.org

1. Datasets and Evaluation Metrics

datasets. We evaluate the proposed method on DrivingStereo [12], KITTI raw [5], and Virtual KITTI [2, 4] datasets.

DrivingStereo is a large-scale outdoor stereo dataset in driving scenarios containing \(\sim 17K\) sequential training pairs and \(\sim 8K\) testing paris. From the entire dataset, we use the specially selected 2000 frames with four different kinds of weather (cloudy, foggy, rainy, sunny) for continual stereo. Each scene includes 500 stereo pairs with 400 pairs for training and 100 pairs for testing. The resolution of the image is 881 \(\times\) 400.

KITTI raw collects real-world outdoor stereo video sequences covering heterogeneous environments, namely residential, city, road and campus. It contains about 43k video frames with sparse depth labels [11] converted into disparities by knowing the camera parameters. Since the labels of the real-world data are hard to obtain, we extract two videos from each scene as training and testing to mimic the real environments. Table 1 shows the specific division with \(\sim 1000\) pairs for training and \(\sim 100\) pairs for testing on each scene. The resolution of the image is about 1248 \(\times\) 384.

Virtual KITTI is a synthetic clone of the real KITTI dataset containing five sequences named Scene 01, 02, 06, 18, and 20. Each scene has nine variants with five different kinds of weather and four modified camera configurations. Since the weather condition has been considered in DrivingStereo, we select the five scenes with four camera configurations and divide them into 80\% training sets and 20\% testing sets for continual stereo.

Metrics. We use the following metrics for evaluation.

The end-point-error (EPE) metric measures the average pixel error between the predicted disparity and the ground truth disparity. The bad pixel error for all pixels (D1-all)

\[
\text{EPE} = \frac{1}{N} \sum_{i=1}^{N} R_{N,i,i},
\]

\[
\text{BWT} = \frac{1}{N-1} \sum_{i=1}^{N-1} R_{N,i,i} - R_{i,i}.
\]

To evaluate the reusability of the learned cells, the average reuse rate (ARR) is introduced to calculate the average proportion of the parameters of the old cells in the current architecture. Define the number of parameters of the current model \(h^t\) as \(\phi_t\), then we have ARR formulated as:

\[
\text{ARR} = \frac{1}{N-1} \sum_{i=2}^{N} \frac{\sum_{j=1}^{t-1} \phi_j \cap \phi_t}{\phi_t},
\]

where \(\phi_j \cap \phi_t\) refers to the number of parameters of cells in \(h^3\) that are resued in \(h^t\).

\begin{table}[ht]
\centering
\begin{tabular}{|c|c|c|}
\hline
Scene & Training Sequence & Testing Sequence \\
\hline
Residential & 2011_09_30_drive_0034 & 2011_09_26_drive_0079 \\
City & 2011_09_29_drive_0071 & 2011_09_26_drive_0113 \\
Road & 2011_10_03_drive_0042 & 2011_09_26_drive_0027 \\
Campus & All sequences except testing & 2011_09_28_drive_0037 \\
\hline
\end{tabular}
\caption{Division of the KITTI raw dataset.}
\end{table}
2. Training Details

The training protocol of our RAG framework consists of three stages. In the cell level search stage, the searching batch size is set to 12, and the total sampling times are set to 100. In the network level growth stage, the searching batch size is 8, and the total sampling times are also set to 100. In the task-specific model training stage, the training batch size is 8. We train our model for 400 epochs on the DrivingStereo dataset and 300 epochs on the KITTI raw and Virtual KITTI datasets. The total training requires 1.2 GPU days for each scene of DrivingStereo on a single TITAN RTX GPU.

For the Scene Router module, the left images are randomly cropped to the size of $380 \times 380$. Then they are resized to $224 \times 224$ as the inputs of the feature extractor used in [1] to yield the feature representation $x_n$ with size of $256 \times 13 \times 13$. The subsequent training protocol follows the description in Section 4.3 of the manuscript. Each task-specific autoencoder is trained for 40 epochs with a batch size of 8. We train our model for 400 epochs on the KITTI raw and RTX GPU.

DrivingStereo dataset and 300 epochs on the KITTI raw and

batch size is 8. We train our model for 400 epochs on the

DynamicStereo dataset and 300 epochs on the KITTI raw and

Virtual KITTI datasets. The total training requires 1.2 GPU

days for each scene of DynamicStereo on a single TITAN RTX GPU.

Table 2. Comparison of the two forms of validation scores on the DrivingStereo dataset.

<table>
<thead>
<tr>
<th>$\delta_j^{m*}$</th>
<th>$\text{EPE}_\downarrow$</th>
<th>$\text{D1}_\downarrow$</th>
<th>$\text{ARR}_\uparrow$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. (4)</td>
<td>0.637</td>
<td>1.21%</td>
<td>50.1%</td>
</tr>
<tr>
<td>Eq. (5)</td>
<td>0.731</td>
<td>1.78%</td>
<td>54.1%</td>
</tr>
</tbody>
</table>

scenes with lower correlation. According to the ablation study, we set it as $\Phi/2$ for trade-offs.

We have also explored another simple implementation form, that is, a weighted linear combination of the model parameters and the error rates, i.e.,

$$\delta_j^{m*} = \mu \cdot \sqrt{1 - \sigma^*} + (1 - \mu) \cdot \log \left( \frac{\phi^{m*}}{\phi} + 1 \right), \quad (5)$$

where $\mu = 0.9$. Table 2 lists the comparison of Eq. (4) and Eq. (5) in terms of the performance and reusability. It can be seen that our proposed method in Eq. (4) achieves better balance between the model performance and parameter efficiency using the form of dot product.

3. Design Analysis of Validation Score

Here we give the design analysis of the validation score in the network level growth. Our model inevitably leads to an increase in the number of parameters during growth. To break this dilemma, we design the Eq. (4) to explicitly incorporate the model parameters into the validation score to yield a compact architecture with high reusability. The validation score can be regarded as a function of the error rate $\sigma^*$ and the number of parameters of selected old cells $\phi^{m*}$. They satisfy $0 < \sigma^* < 1$ and $0 < \phi^{m*} < \Phi$, where $\Phi$ is the number of parameters of a single base model. And $f(\sigma^*, \phi^{m*})$ should obey the following properties:

- $0 < f(\sigma^*, \phi^{m*}) < 1$.
- $f(\sigma^*, \phi^{m*})$ should be negatively correlated with the error rate $\sigma^*$.
- $f(\sigma^*, \phi^{m*})$ should be positively correlated with the number of parameters of old cells $\phi^{m*}$.

To this end, we have designed the following validation score:

$$\delta_j^{m*} = \sqrt{1 - \sigma^*} \cdot \log \left( \frac{\phi^{m*}}{\phi} + 1 \right). \quad (4)$$

The target number of parameters $\phi$ in the above formula controls the compactness of the model. A larger target number can yield a higher reuse rate, which is suitable for continual scenes with higher correlation. Conversely, setting a smaller target number will reduce the reuse rate of the model, but it can achieve better performance for continual tasks.

4. Detailed Structure of Base Model

Our base model comes from the variety of LEAStereo [3] for its good scalability. To better deploy on resource-limited edge devices, we adopt a lightweight version including 4-layer Feature Net and 8-layer Matching Net. Following the training protocol in the original paper, the searched architecture of the base model is shown in Fig 1. For the convenience of description, we use the same schematic diagram as in the original paper. The top two graphs are the searched cell structures for the Feature Net and Matching Net, respectively. The bottom is the searched network-level structure for both networks. The yellow dots refer to the "stem" structure and the blue dots refer to searchable cells. There are three "stem" layers for the Feature Net, which are $3 \times 3$ convolution layer with the stride of three and two layers of $3 \times 3$ convolution with the stride of one. For the Matching Net, there are two "stem" layers of $3 \times 3 \times 3$ convolution with the stride of one. In the continual stereo, the task-specific cell structures are searched for each scene.

5. Implementations of Baselines

We describe the detailed implementations of the continual learning baselines.

EWC [6]: The main idea of the EWC algorithm is to impose constraints over gradient updates so that the gradient updates on the new task do not increase the loss on the old tasks. Only one model with a fixed network structure is adopted and no additional memory space is required. We use the base model to implement it for continual stereo.
iCaRL [9]: The iCaRL algorithm uses a memory bank to store representative samples of the old tasks. They are trained with the data of the new tasks to alleviate forgetting. The memory bank has a fixed size (we set it to 10% of the training set in each task), and old samples will be replaced by samples from new tasks. Similar to the EWC algorithm, we use the base model to implement it.

Expert Gate [1]: Compared with the previous two baselines, the Expert Gate algorithm trains a task-specific model for each scene. It uses the autoencoder gate to judge the relevance of the new task to the old tasks, and further determines whether to use finetuning or Lwf [8] method. We use the output disparity of the previous models on the new scene as the pseudo-label for distillation.

Learn to Grow [7]: It is an architecture growth method close to ours. When a new task arrives, each layer of the network has three candidate operations: "reuse", "adaptation", and "new". The reuse choice reuses the old cells while the adaptation choice adds a small adaptor to the original layer output. The new choice will expand new cells. We re-implement this method using the same search strategy as in our cell-level search.

6. More Visual Comparisons with the Continuous Adaptation

As shown in Fig. 2, we provide more visual comparisons with the continuous adaptation method [10] on the previous scenes and novel scenes, respectively. Our method can overcome forgetting the previously learned challenging scenes to predict good disparity maps, such as rainy and foggy scenes. In addition, when exposed to a novel scene like overcast days at dusk, our method can adaptively select scene-specific architecture paths to obtain better results. In the four samples, the selected architecture are trained on cloudy, foggy, rainy, and rainy, respectively.

References

### Previous Scenes

<table>
<thead>
<tr>
<th>Method</th>
<th>D1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAD-gt-full</td>
<td>31.866%</td>
</tr>
<tr>
<td>Ours</td>
<td>0.004%</td>
</tr>
<tr>
<td>Novel Scenes</td>
<td></td>
</tr>
<tr>
<td>MAD-gt-full</td>
<td>5.003%</td>
</tr>
<tr>
<td>Ours</td>
<td>1.400%</td>
</tr>
</tbody>
</table>

### Novel Scenes

<table>
<thead>
<tr>
<th>Method</th>
<th>D1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAD-gt-full</td>
<td>11.542%</td>
</tr>
<tr>
<td>Ours</td>
<td>8.857%</td>
</tr>
</tbody>
</table>

Figure 2. More visual comparisons of the disparity maps of previously learned scenes (top three rows) and novel scenes (bottom three rows) with the continuous adaptation method [10]. The yellow box marks areas that are significantly improved.
