Continual and Multi-Task Architecture Search

Ramkanth Pasunuru
www.rama-kanth.com

Mohit Bansal
www.cs.unc.edu/~mbansal/
Introduction

In this work, we extend architecture search approach to two important paradigms of transfer learning.
In this work, we extend architecture search approach to two important paradigms of transfer learning.

**Continual Learning**

Model parameters evolve and adapt when trained sequentially on a new task.
In this work, we extend architecture search approach to two important paradigms of transfer learning.

**Continual Learning**
Model parameters evolve and adapt when trained sequentially on a new task.

**Multi-Task Learning**
Given multiple tasks in parallel, learns a generalizable cell structure.
Definition: Continual learning (CL) is the ability to learn continually from a stream of data, building on what was learnt previously, while being able to reapply, adapt and generalize it to new situations.
Definition: Continual learning (CL) is the ability to learn continually from a stream of data, building on what was learnt previously, while being able to reapply, adapt and generalize it to new situations.

Key Challenges:

Transfer and Adapt
Continual Learning

**Definition:** Continual learning (CL) is the ability to learn continually from a stream of data, building on what was learnt previously, while being able to \textit{reapply}, \textit{adapt} and \textit{generalize} it to new situations.

**Key Challenges:**

- Transfer and Adapt
- Catastrophic Forgetting

[Donahue et al., 2014; Zeneke et al., 2017; Yoon et al., 2018]
Continual Learning

**Definition:** Continual learning (CL) is the ability to learn continually from a stream of data, building on what was learnt previously, while being able to *reapply, adapt* and *generalize* it to new situations.

**Key Challenges:**

- **Transfer and Adapt**
- **Catastrophic Forgetting**
- **Bounded System Size**

[Donahue et al., 2014; Zeneke et al., 2017; Yoon et al., 2018]
Previous Work

- Regularization to penalize functional or shared parameters’ change
  [Razavian et al., 2014; Li and Hoiem, 2017; Hinton et al., 2015; Jung et al., 2016; Kirkpatrick et al., 2017; Donahue et al., 2014; Yosinski et al., 2014]

- Copying the previous task and augmenting with new task’s features
  [Rusu et al., 2016]

- Intelligent synapses to accumulate task-related information
  [Zeneke et al., 2017]

- Dynamically expandable network based on incoming new data
  [Yoon et al., 2018]
Leverage Neural Architecture Search!

Continual Architecture Search (CAS): continually evolve the model parameters during the sequential training of several tasks by leveraging neural architecture search.

Idea!

[Donahue et al., 2014; Zeneke et al., 2017; Yoon et al., 2018]
**NAS**

**Neural Architecture Search (NAS):** has been recently introduced for automatic learning of the model structure for the given dataset/task.

- Shown good improvements on image classification and language modeling
- Computationally feasible NAS approaches:
  - Tree-structured search space
  - $\epsilon$-greedy exploration

[Zoph & Le, 2017; Baker et al., 2017; Negrinho & Gordon, 2017]
Neural Architecture Search (NAS): has been recently introduced for automatic learning of the model structure for the given dataset/task.

- Shown good improvements on image classification and language modeling
- Computationally feasible NAS approaches:
  - Tree-structured search space
  - $\epsilon$-greedy exploration

Efficient Neural Architecture Search (ENAS): A weight-sharing strategy among search parameters [Pham et al., 2018]

[Zoph & Le, 2017; Baker et al., 2017; Negrinho & Gordon, 2017]
Figure: An example of a recurrent cell in our search space with 4 computational nodes. Left: The computational DAG that corresponds to the recurrent cell. The red edges represent the flow of information in the graph. Middle: The recurrent cell. Right: The outputs of the controller RNN that result in the cell in the middle and the DAG on the left. Note that nodes 3 and 4 are never sampled by the RNN, so their results are averaged and are treated as the cell’s output.
**Figure:** The graph represents the entire search space while the red arrows define a model in the search space, which is decided by a controller. Here, node 1 is the input to the model whereas nodes 3 and 6 are the model’s outputs.
Stage 1:
- Controller samples a cell structure and use the task’s performance as feedback
- Controller learns optimal cell structure
ENAS for Text Classification

Stage 2:
- Retrain the model using the learned optimal cell structure in stage-1

CONTROLLER
Label Distribution

Max-pooling
Concatenation
Max-pooling
Label Distribution
ENAS for Sequence Generation

CONTROLLER

Video Encoder

Caption Decoder
Continual and Multi-Task Architecture Search

Continual Architecture Search (CAS)

Task1 (dataset $d_1$)

$\theta_{1,k} \in \theta_1$

Task2 (dataset $d_2$)

$\theta_{2,k} \in \theta_2$
Continual Architecture Search (CAS)

Task1 (dataset $d_1$)

$\theta_{1,k} \in \theta_1$

Task2 (dataset $d_2$)

$\theta_{2,k} \in \theta_2$

$\psi_{2,k} \in \psi_2$

$\theta_{2,k} = \theta_{1,k} + \psi_{2,k}$
Continual Architecture Search (CAS)

Task1 (dataset $d_1$)

$\theta_{1,k} \in \theta_1$

Task2 (dataset $d_2$)

$\theta_{2,k} = \theta_{1,k} + \psi_{2,k}$

$\psi_{2,k} \in \psi_2$

$\theta_1$ are block-sparse in nature

[Scardapane et al., 2017]
Continual Architecture Search (CAS)

\[ \theta_1, k \in \theta_1 \]

\[ \psi_{2,k} \in \psi_2 \]

\[ \theta_{2,k} = \theta_{1,k} + \psi_{2,k} \]

\[ \theta_1 \text{ are block-sparse in nature} \]

\[ \psi_2 \text{ is orthogonal to } \theta_1 \]

[ Bousmalis et al., 2016; Scardapane et al., 2017 ]
Continual Architecture Search (CAS)

**Task1** (dataset $d_1$)  

**Task2** (dataset $d_2$)

Condition 1: When training the model on dataset $d_1$, we constrain the model parameters $\theta_{1,k} \in \Theta_1$ to be sparse, specifically, to be block sparse, i.e., minimize

$$
\sum_{i=1}^{m} \left\| \theta_{1,k}[i, :] \right\|_2\right\|_1
$$

Continual and Multi-Task Architecture Search  

R. Pasunuru & M. Bansal

[Scardapane et al., 2017]
Continual and Multi-Task Architecture Search

**Continual Architecture Search (CAS)**

**Task1 (dataset \( d_1 \))**

\[ \theta_{1,k} \in \theta_1 \]

**Task2 (dataset \( d_2 \))**

\[ \psi_{2,k} \in \psi_2 \]

\[ \theta_{2,k} = \theta_{1,k} + \psi_{2,k} \]

\[ \theta_{2,k} \in \theta_2 \]

\[ \psi_{2,k} \in \psi_2 \]

**Condition 2:** When training the model on dataset \( d_2 \), we start from \( \theta_{1,k} \), keep it constant, and update \( \psi_{2,k} \) such that:

1. \( \psi_{2,k} \) is block sparse, i.e.,
   \[
   \sum_{i=1}^{m} \| \| \psi_{2,k}[i, :] \|_2 \|_1
   \]

2. \( \theta_{1,k} \) and \( \psi_{2,k} \) are orthogonal.
CAS Step-1

Dataset $d_1$

Step-1

$\text{dag}_1$

$\theta_{11}$ $\theta_{12}$ $\theta_{13}$ $\theta_{14}$

$\text{Avg}$

Train on Dataset $d_1$

Apply Condition 1

Learn parameters $\theta_1$ and cell structure $\text{dag}_1$
Continual and Multi-Task Architecture Search

R. Pasunuru & M. Bansal

CAS Step-2

Step-2 Dataset \( d_2 \)

\[ \text{Train on Dataset } d_2 \text{ and initialize the parameters with } \theta_1 \]

Apply Condition 2

\[ \text{Learn parameters } \theta_2 \text{ and cell structure } \text{dag}_2 \]
CAS Evaluation

Figure: Continual architecture search (CAS) approach: green, solid edges (weight parameters) are shared, newly-learned edges are represented with red, dashed edges.
Figure: Continual architecture search (CAS) approach: green, solid edges (weight parameters) are shared, newly-learned edges are represented with red, dashed edges.
CAS Evaluation

Figure: Continual architecture search (CAS) approach: green, solid edges (weight parameters) are shared, newly-learned edges are represented with red, dashed edges.
**Figure:** Continual architecture search (CAS) approach: green, solid edges (weight parameters) are shared, newly-learned edges are represented with red, dashed edges.
### CAS Results on Text Classification

<table>
<thead>
<tr>
<th>Models</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PREVIOUS WORK</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiLSTM+ELMo (2018)</td>
<td>69.4</td>
<td>50.1</td>
<td>65.1</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn (2018)</td>
<td>61.1</td>
<td>50.3</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>BASELINES</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline (with ELMo)</td>
<td>73.2</td>
<td>52.3</td>
<td>65.1</td>
</tr>
<tr>
<td>ENAS (Architecture Search)</td>
<td>74.5</td>
<td>52.9</td>
<td>65.1</td>
</tr>
</tbody>
</table>

**Table:** Test results on GLUE tasks for various models: Baseline, ENAS, and CAS (continual architecture search). The CAS results maintain statistical equality across each step.
### CAS Results on Text Classification

<table>
<thead>
<tr>
<th>Models</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PREVIOUS WORK</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiLSTM+ELMo (2018)</td>
<td>69.4</td>
<td>50.1</td>
<td>65.1</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn (2018)</td>
<td>61.1</td>
<td>50.3</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>BASELINES</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline (with ELMo)</td>
<td>73.2</td>
<td>52.3</td>
<td>65.1</td>
</tr>
<tr>
<td>ENAS (Architecture Search)</td>
<td>74.5</td>
<td>52.9</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>CAS RESULTS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAS Step-1 (QNLI training)</td>
<td>73.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CAS Step-2 (RTE training)</td>
<td>73.6</td>
<td>54.1</td>
<td>N/A</td>
</tr>
<tr>
<td>CAS Step-3 (WNLI training)</td>
<td>73.3</td>
<td>54.0</td>
<td>64.4</td>
</tr>
</tbody>
</table>

**Table:** Test results on GLUE tasks for various models: Baseline, ENAS, and CAS (continual architecture search). The CAS results maintain statistical equality across each step.
### CAS Results on Text Classification

<table>
<thead>
<tr>
<th>Models</th>
<th>QNLI</th>
<th>RTE</th>
<th>WNLI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Previous Work</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BiLSTM+ELMo (2018)</td>
<td>69.4</td>
<td>50.1</td>
<td>65.1</td>
</tr>
<tr>
<td>BiLSTM+ELMo+Attn (2018)</td>
<td>61.1</td>
<td>50.3</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Baseline (with ELMo)</td>
<td>73.2</td>
<td>52.3</td>
<td>65.1</td>
</tr>
<tr>
<td>ENAS (Architecture Search)</td>
<td>74.5</td>
<td>52.9</td>
<td>65.1</td>
</tr>
<tr>
<td><strong>CAS Results</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAS Step-1 (QNLI training)</td>
<td>73.8</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CAS Step-2 (RTE training)</td>
<td>73.6</td>
<td>54.1</td>
<td>N/A</td>
</tr>
<tr>
<td>CAS Step-3 (WNLI training)</td>
<td>73.3</td>
<td>54.0</td>
<td>64.4</td>
</tr>
</tbody>
</table>

**Table:** Test results on GLUE tasks for various models: Baseline, ENAS, and CAS (continual architecture search). The CAS results maintain statistical equality across each step.

Note that we use same settings for both normal and architecture search models, unless otherwise specified. More details in appendix.

Further, if we use the same cell structure, the model performance is same as our baseline model.

Note that ENAS random search baseline vs. optimal architecture search based models.

ENAS Models: ENAS (Random search), Baseline (with ELMo)

Comparison to previous work (2018) vs. ENAS (Architecture Search)

On validation set, QNLI step-3 vs. step-2 performance is 61.0 vs. 60.6 on validation set, 73.9 vs. 74.1, which is statistically equal. Similarly, on RTE, step-2 (74.1 vs. 74.2 on validation set which is statistically equal on validation set).

We observe that even though we learn the architecture search (CAS) approach on QNLI, RTE, and WNLI (in the order of primary task is then trained only on its own data). CAS Step-1 vs. ENAS (74.5 vs. 73.8); however, this is not statistically significant on it.

This is because the learned optimal cell structure is in fact better than the random cell structure.

We also performed important ablation experiments to understand the method can maintain the performance of a task in method can maintain the performance on QNLI (as well as maintain the performance on QNLI (as well as maintaining statistical equality across each step).

CAS Condition Ablation:

We only report single-task (and not 9-task multi-task) results from the GLUE benchmark for fair comparison to our benchmark baselines (see next paragraph on ‘CAS Condition Ablation’).

Further, we use bidirectional LSTM+ELMo+Attn, BiLSTM+ELMo for video captioning tasks. We calculate the statistical significance on it.

Since the test set is hidden, we are not able to calculate the statistical significance on it.

The CAS results maintain statistical equality across each step.
Continual and Multi-Task Architecture Search

CAS Ablation

- No Condition: 69.1
- Only Condition 2.1 (block-sparse): 71.5
- Only Condition 2.2 (orthogonality): 69.4
- With both Conditions: 74.1

Validation accuracy on QNLI

65 67.5 70 72.5 75
CAS Ablation

- No Condition: 69.1
- Only Condition 2.1 (block-sparse): 71.5
- Only Condition 2.2 (orthogonality): 69.4
- With both Conditions: 74.1

Validation accuracy on QNLI

Difference is Statistically Significant
Continual and Multi-Task Architecture Search

from QNLI and WNLI), versus the RTE-ENAS Multi-Task Cell Structure

stable w.r.t. change in the activation functions.

that those edges are learning weights which are but the activation function is different. This shows e.g., edge between node 0 and node 1 is the same,

step-3 cell uses some common edges w.r.t. the node 1 to node 3). Further, we observe that the cell structure in step-2 (e.g.,

step-1 and step-2 share some common edges and with some new edge connections in step-2 (e.g.,

the cell structures in CAS preserve the properties of certain edges while creating new edges for new
capabilities. We notice that the cell structure in RTE, and WNLI tasks. Overall, we observe that CAS approach, where we sequentially train QNLI,

presents the cell structure in each step for the Evolved Cell Structure with CAS
generalizable than the standard LSTM cell.

which suggests that the multi-task cell is more 2.94 for multi-task cell vs. 2.81 for LSTM cell,

man study as Sec. Human Evaluation:

that a cell learned on multiple tasks is more gener-

izable to other tasks.


Table 3: Video captioning results with Baseline, ENAS, and CAS models. Baseline is reproduced numbers from Table 5: Comparison of MAS cell on DiDeMO task.

Table 4: Comparison of MAS cell on RTE task.

CAS Step-2 (MSVD training)

CAS Step-1 (MSR-VTT training)

ENAS

Baseline (Pasunuru and Bansal, 2017b)

We performed a similar hu-

Table: Video captioning results with Baseline, ENAS, and CAS.

<table>
<thead>
<tr>
<th>Models</th>
<th>MSR-VTT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>MSVD</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>B</td>
<td>R</td>
<td>M</td>
<td>AVG</td>
<td>C</td>
<td>B</td>
<td>R</td>
<td>M</td>
</tr>
<tr>
<td>Baseline (Pasunuru and Bansal, 2017b)</td>
<td>48.2</td>
<td>40.8</td>
<td>60.7</td>
<td>28.1</td>
<td>44.5</td>
<td>85.8</td>
<td>52.5</td>
<td>71.2</td>
<td>35.0</td>
</tr>
<tr>
<td>ENAS</td>
<td>48.9</td>
<td>41.3</td>
<td>61.2</td>
<td>28.1</td>
<td>44.9</td>
<td>87.2</td>
<td>52.9</td>
<td>71.7</td>
<td>35.2</td>
</tr>
<tr>
<td>CAS Step-1 (MSR-VTT training)</td>
<td>48.9</td>
<td>41.1</td>
<td>60.5</td>
<td>27.5</td>
<td>44.5</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CAS Step-2 (MSVD training)</td>
<td>48.4</td>
<td>40.1</td>
<td>59.9</td>
<td>27.1</td>
<td>43.9</td>
<td>88.1</td>
<td>52.4</td>
<td>71.3</td>
<td>35.1</td>
</tr>
</tbody>
</table>

CAS Results on Video Captioning

[Chen & Dolan, 2011; Xu et al., 2016]
Continual and Multi-Task Architecture Search

CAS Results on Video Captioning

<table>
<thead>
<tr>
<th>Models</th>
<th>MSR-VTT</th>
<th></th>
<th>MSVD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>B</td>
<td>R</td>
</tr>
<tr>
<td>Baseline (Pasunuru and Bansal, 2017b)</td>
<td>48.2</td>
<td>40.8</td>
<td>60.7</td>
</tr>
<tr>
<td>ENAS</td>
<td>48.9</td>
<td>41.3</td>
<td>61.2</td>
</tr>
<tr>
<td>CAS Step-1 (MSR-VTT training)</td>
<td>48.9</td>
<td>41.1</td>
<td>60.5</td>
</tr>
<tr>
<td>CAS Step-2 (MSVD training)</td>
<td>48.4</td>
<td>40.1</td>
<td>59.9</td>
</tr>
</tbody>
</table>

Table: Video captioning results with Baseline, ENAS, and CAS.

Difference is Statistically insignificant (also based on Human evaluation)
Hence CAS is maintaining performance sequentially

[Chen & Dolan, 2011; Xu et al., 2016]
CAS Learned Cells

(a) Step-1
(b) Step-2
(c) Step-3

Figure: Learned cell structures for step-1, step-2, and step-3 of continual architecture search for GLUE tasks.

Table 3: Video captioning results with Baseline, ENAS, and CAS models. Baseline is reproduced numbers from Pasunuru and Bansal (2017b).

<table>
<thead>
<tr>
<th>Models</th>
<th>MSVD cell</th>
<th>MSR-VTT cell</th>
<th>DiDeMO cell</th>
<th>RTE cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>7.5</td>
<td>13.4</td>
<td>27.5</td>
<td>8.1</td>
</tr>
<tr>
<td>ENAS</td>
<td>7.2</td>
<td>13.1</td>
<td>27.1</td>
<td>7.9</td>
</tr>
<tr>
<td>CAS Step-1</td>
<td>7.1</td>
<td>12.1</td>
<td>25.2</td>
<td>7.9</td>
</tr>
<tr>
<td>CAS Step-2</td>
<td>7.6</td>
<td>12.7</td>
<td>26.7</td>
<td>7.6</td>
</tr>
<tr>
<td>CAS Step-3</td>
<td>7.5</td>
<td>13.1</td>
<td>27.1</td>
<td>7.8</td>
</tr>
</tbody>
</table>

Performance on DiDeMo

<table>
<thead>
<tr>
<th>Performance</th>
<th>MC</th>
<th>B</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>CBRM AVG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>71.3</td>
<td>28.1</td>
<td>44.5</td>
<td></td>
</tr>
<tr>
<td>71.2</td>
<td>28.1</td>
<td>44.5</td>
<td></td>
</tr>
<tr>
<td>71.2</td>
<td>28.1</td>
<td>44.5</td>
<td></td>
</tr>
<tr>
<td>71.2</td>
<td>28.1</td>
<td>44.5</td>
<td></td>
</tr>
</tbody>
</table>

Continual and Multi-Task Architecture Search

Acknowledgments

We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), and faculty awards from Google, Facebook, and Salesforce. The views contained in this article are those of the authors and not of the funding agency.
CAS Learned Cells

Figure: Learned cell structures for step-1, step-2, and step-3 of continual architecture search for GLUE tasks.
CAS Learned Cells

(a) Step-1

(b) Step-2

(c) Step-3

Figure: Learned cell structures for step-1, step-2, and step-3 of continual architecture search for GLUE tasks.
Generalizable Cell on Multiple Tasks

- Architectures found by NAS are dataset dependent
- Human designed cell (e.g., LSTM and GRU) work well across multiple datasets
Generalizable Cell on Multiple Tasks

• Architectures found by NAS are dataset dependent

• Human designed cell (e.g., LSTM and GRU) work well across multiple datasets

Can we learn generalizable NAS cell structures?
Generalizable Cell on Multiple Tasks

- Architectures found by NAS are dataset dependent
- Human designed cell (e.g., LSTM and GRU) work well across multiple datasets

Can we learn generalizable NAS cell structures?

Multi-Task Learning!!
Multi-Task Architecture Search (MAS)

Controller

Shared Model

Figure: Multi-task cell structure learning using joint rewards from n datasets.
Multi-Task Architecture Search (MAS)

Figure: Multi-task cell structure learning using joint rewards from n datasets.
Multi-Task Architecture Search (MAS)

Figure: Multi-task cell structure learning using joint rewards from n datasets.
Multi-Task Architecture Search (MAS)

Figure: Multi-task cell structure learning using joint rewards from n datasets.
MAS Results on Text Classification

Multi-Task cell is learned using QNLI and WNLI dataset

Validation accuracy on RTE

- LSTM cell
- QNLI cell
- WNLI cell
- RTE cell
- Multi-Task Cell
MAS Learned Cells

(a) MAS cell
(b) RTE cell

Figure: Learned Multi-task and RTE cell Structures.
Multi-Task Cell Structure is relatively less complex, i.e., uses several identity functions and very few activation functions in its cell structure.

A MAS cell is more generalizable than the standard LSTM cell.

We performed a similar human study as Sec. 7.2

Table 3: Video captioning results with Baseline, ENAS, and CAS models. Baseline is reproduced numbers from Pasunuru and Bansal (2017b) which uses advanced latest visual features (ResNet-152 and ResNeXt-101).

Figure 4: Learned cell structures for step-1, step-2, and step-3 of continual architecture search for GLUE tasks.

Figure 5: Learned multi-task & RTE cell structures.

**Figure:** Learned Multi-task and RTE cell Structures.
Thanks!

Ramkanth Pasunuru         Mohit Bansal

Code: https://github.com/ramakanth-pasunuru/CAS-MAS

Acknowledgements: We thank the reviewers for their helpful comments. This work was supported by DARPA (YFA17-D17AP00022), and faculty awards from Google, Facebook, and Salesforce.