

LOOP AND LIBRARY FUSION

BACKGROUND

[0001] This specification relates to constructing and compiling computational graphs.

[0002] A computational graph defines sequences of operations by the types of operations, the data that is input into and output from each operation, and computational dependencies. A compiler translates a computational graph of operations to produce compiled code.

SUMMARY

[0003] This specification describes technologies relating to computational graph systems in general, and specifically to systems and methods for representing computations as graph operations that can be translated into efficient compiled code.

[0004] The computational graph includes nodes, connector directed edges, and parameter directed edges. Each node represents a respective operation, each connector directed edge connects a respective first node to a respective second node that represents an operation that receives, as input, an output of an operation represented by the respective first node, and each parameter directed edge connects into a respective node and represents a flow of one or more parameters of a neural network as input to the operation represented by the respective node.

[0005] In general, one innovative aspect of the subject matter described in this specification can be embodied in a system comprising one or more computers and one or more storage devices storing instructions that are operable, when executed by the one or more computers to cause the one or more computers to perform operations that implement an example method. An example method comprises: obtaining an un-optimized computational graph comprising a plurality of nodes representing operations and a plurality of directed edges representing data dependencies between the operations; analyzing the un-optimized computational graph using pattern matching to determine fusible operations that can be fused together, wherein each fusible operation of the fusible operations is decomposed into an index function and a data function; transforming the un-optimized computational graph into an optimized computational graph by replacing the plurality of nodes representing the fusible operations in the un-optimized computational graph with a fusion node representing the single fusion operation; and providing, to a compiler, the optimized computational graph including the single fusion node that the compiler can translate as a call that performs the single fused operation to produce efficient code at a code generation phase of compilation.

[0006] Other embodiments of this aspect include corresponding computer systems, apparatus, and computer programs recorded on one or more computer storage devices, each configured to perform the actions of the methods. For a system of one or more computers to

WHAT IS CLAIMED IS:

1. A method comprising:

obtaining an un-optimized computational graph comprising a plurality of nodes representing operations and a plurality of directed edges representing data dependencies between the operations;

analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together, wherein each fusable operation of the fusable operations is decomposed into an index function and a data function;

generating a single fusion operation using a composition of the index functions and the data functions for the fusable operations;

transforming the un-optimized computational graph into an optimized computational graph, by replacing the plurality of nodes representing the fusable operations in the un-optimized computational graph with a single fusion node representing the single fusion operation; and

providing, to a compiler, the optimized computational graph including the single fusion node that the compiler can translate as a call that performs the single fusion operation to produce efficient code at a code generation phase of compilation.

2. The method of claim 1, further comprising: providing the efficient code to computing devices for execution.

3. The method of claim 2, wherein execution includes: executing fusion operations of the optimized computational graph including the call that performs the single fusion operation.

4. The method of claim 1, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

comparing portions of the un-optimized computational graph with patterns of operations that each correspond to a single fusion operation;

determining that a pattern matches a portion of the un-optimized computational graph; and

determining that the matching portion of the un-optimized computational graph can be replaced in the optimized computational graph with the single fusion operation corresponding to the matching pattern.

5. The method of claim 1, wherein the single fusion operation is a loop operation.

6. The method of claim 1, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

searching the un-optimized computational graph for a node representing a fusable operation that takes as input an output generated by a chain of operations represented by a chain of nodes in the un-optimized computational graph; and

determining that the chain of operations can be replaced in the optimized computational graph with a single fusion operation corresponding to the chain of operations required to produce the input.

7. The method of claim 1, wherein the fusable operations are regular operations.

8. The method of claim 1, wherein the fusable operations are regular operations that are fused into non-regular operations.

9. The method of claim 1, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

finding a sequence of operations in a computational graph using a sequencing algorithm; and

determining that the sequence of operations that can be fused together using composition into a single fusion operation.

10. A system comprising:

one or more computers; and

one or more storage devices storing instructions that are operable, when executed by the one or more computers, to cause the one or more computers to perform operations comprising:

obtaining an un-optimized computational graph comprising a plurality of nodes representing operations and a plurality of directed edges representing data dependencies between the operations;

analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together, wherein each fusable operation of the fusable operations is decomposed into an index function and a data function;

generating a single fusion operation using a composition of the index functions and the data functions for the fusable operations;

transforming the un-optimized computational graph into an optimized computational graph, by replacing the plurality of nodes representing the fusable operations in the un-optimized computational graph with a single fusion node representing the single fusion operation; and
providing, to a compiler, the optimized computational graph including the single fusion node that the compiler can translate as a call that performs the single fusion operation to produce efficient code at a code generation phase of compilation.

11. The system of claim 10, wherein the operations further comprise: providing the efficient code to computing devices for execution.

12. The system of claim 11, wherein execution includes: executing fusion operations of the optimized computational graph including the call that performs the single fusion operation.

13. The system of claim 10, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

comparing portions of the un-optimized computational graph with patterns of operations that each correspond to a single fusion operation;

determining that a pattern matches a portion of the un-optimized computational graph;
and

determining that the matching portion of the un-optimized computational graph can be replaced in the optimized computational graph with the single fusion operation corresponding to the matching pattern.

14. The system of claim 10, wherein the single fusion operation is a loop operation.

15. The system of claim 10, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

searching the un-optimized computational graph for a node representing a fusable operation that takes as input an output generated by a chain of operations represented by a chain of nodes in the un-optimized computational graph; and

determining that the chain of operations can be replaced in the optimized computational graph with a single fusion operation corresponding to the chain of operations required to produce the input.

16. The system of claim 10, wherein the fusable operations are regular operations.

17. The system of claim 10, wherein analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together includes:

finding a sequence of operations in a computational graph using a sequencing algorithm;
and

determining that the sequence of operations that can be fused together using composition into a single fusion operation.

18. One or more non-transitory computer-readable storage media comprising instructions stored thereon that are executable by a processing device and upon such execution cause the processing device to perform operations comprising:

obtaining an un-optimized computational graph comprising a plurality of nodes representing operations and a plurality of directed edges representing data dependencies between the operations;

analyzing the un-optimized computational graph using pattern matching to determine fusable operations that can be fused together, wherein each fusable operation of the fusable operations is decomposed into an index function and a data function;

generating a single fusion operation using a composition of the index functions and the data functions for the fusable operations;

transforming the un-optimized computational graph into an optimized computational graph, by replacing the plurality of nodes representing the fusable operations in the un-optimized computational graph with a single fusion node representing the single fusion operation; and

providing, to a compiler, the optimized computational graph including the single fusion node that the compiler can translate as a call that performs the single fusion operation to produce efficient code at a code generation phase of compilation.