ABSTRACT
Optimizing compilers for efficient machine learning are more important than ever due to the rising ubiquity of machine learning. Predictive models to guide compiler optimization are sometimes used to derive a sequence of loop transformations to optimize memory access performance via deploying learned models. However, training models for loop optimization often requires prohibitively expensive training data generation when predicting the combined effects of a transformation sequence. In this paper, we present a learning strategy called Composed Singular Prediction that significantly reduces the training data generation cost in the context of learned loop transformation models. The learned models are then deployed to predict data locality optimization schedules for Conv2d kernels to achieve performance improvements up to 4.0x against Intel oneDNN while scaling 500x in training data collection time.

INTRODUCTION
- Learned models for compiler optimization are popular
- Datasets are not readily available
- Training data are generated on a case-by-case basis
- Requires some form of Design Space Exploration (DSE)
- Often requires sampling and/or pruning to handle the search complexity

Fig 1. Performance distribution of loop permutation for Conv2d

Fig 2. Performance distribution of loop tile and loop permutation for Conv2d

Conv2d in MLIR
Inputs are written in affine and std dialects. A custom LLVM pass consumes the MLIR input, applies loop transformations, and generates a code for the selected architecture.

CASE STUDY
- Data locality optimization for Conv2d kernel
- Loop tiling and Loop permutation with 2 Singular models
- MLIR/LLVM-based compiler
- Static loop unrolling and vectorization
- Generates code for Intel Xeon AVX-512 systems

The permutation model computes,
\[ p^* = \text{argmax}_{p} g_p(c, s) \]

The tile model computes,
\[ s^* = \text{argmax}_{s} g_s(c, s) \]

Performance of Composed Singular Prediction for the 2 transformations,
\[ \psi(f(c, s^*, p^*)) \]

METHODOLOGY
Search space for training data generation consists of different transformation schedule instances.
\[ s = (s_1, s_2, ..., s_n), s_1 \in S_1, ..., s_n \in S_n \]

Classical search space, Multiplicative Domain Formulation (MDF)
\[ \text{MDF}: (S_1 \times S_2 \times \cdots \times S_n) \rightarrow \mathcal{O}(|S_1| \times \cdots \times |S_n|) \]

With MDF, the learning task is to find a function \( f \) such that,
\[ s^* = \text{argmax}_{s} \psi(f(c, s)) \]

An alternative search space Additive Domain Formulation (ADF) can be defined as,
\[ \text{ADF}: (S_1) + (S_2) + \cdots + (S_n) \rightarrow \mathcal{O}(|S_1| + \cdots + |S_n|) \]

With ADF, the task is to learn a set of Singular Functions \( g_s \), such that,
\[ \delta = \{ \delta_1: \delta_1 = \text{argmax}_{s} g_s(c, s) \} \]

\[ \psi_{\delta_1} = \psi(f(c, \delta_1), \delta_1) \]

Feature Representation
Training data generation for the tile model queries the loop permutation model,
\[ \psi_{\delta_1} \times \psi_f(c, \delta_1) \]

RESULTS
Loop Permutation Singular Model Accuracy
The performance distributions of permutation model’s predictions and its corresponding labels for all permutations of a select set of Conv2d layers.

Performance of predicted permutation schedules
Performance of permutation schedules are compared against Intel oneDNN’s matching data layout implementation and the library’s best performing data layout.

Performance speed-ups of the tile model’s predictions against untiled Conv2d and Intel oneDNN library implementation.

Further evidence for domain correlation is observed by evaluating a small MDF (D2) test set against the model trained with ADF (D1) training data.

ACKNOWLEDGEMENTS
This work began in Summer 2020 at Intel Labs Machine Programming Research (MPR) Group. Anand Verkant, Justin Gottschlich (Intel Labs), Abhishek Kulkarni, Pushkar Ratnakar (Intel), Uylek Srikumar (University of Utah).

Intel Academic Compute Environment (ACE)

Funding
DOD SciDAC, NSF