#include "tensorflow/core/framework/op.h"

#include "tensorflow/core/framework/shape\_inference.h"

#include "tensorflow/core/framework/op\_kernel.h"

#include <twml.h>

#include "tensorflow\_utils.h"

#include <map>

#include <vector>

#include <set>

REGISTER\_OP("FeatureMask")

.Attr("T: {int64, int8}")

.Input("keep: T")

.Attr("list\_keep: list(int)")

.Output("mask: bool")

.SetShapeFn([](::tensorflow::shape\_inference::InferenceContext\* c) {

return Status::OK();

}).Doc(R"doc(

A tensorflow OP that creates a mask of the indices that should be kept.

Attribute

list\_keep: list of values which should be kept(list(int))

Input

keep: Tensor for which we will apply the mask (int64, int8)

Outputs

mask: boolean Tensor. (bool)

)doc");

template <typename T>

class FeatureMask : public OpKernel {

private:

std::set<int64> feature\_set\_keep;

public:

explicit FeatureMask(OpKernelConstruction\* context)

: OpKernel(context) {

std::vector<int64> feature\_list\_keep;

OP\_REQUIRES\_OK(context, context->GetAttr("list\_keep", &feature\_list\_keep));

// create set that contains the content of the feature\_list\_keep, since tensorflow does not allow

// me to directly ouput the contents of list\_keep to a set

feature\_set\_keep = std::set<int64>(feature\_list\_keep.begin(), feature\_list\_keep.end());

}

void Compute(OpKernelContext\* context) override {

// Get size of the input\_vector and create TensorShape shape

const Tensor& input = context->input(0);

auto keep = input.flat<T>();

// Create an output tensor

Tensor\* output\_mask = nullptr;

// Output shape is determined and now we can copy the contents of the vector to the output Tensor.

const int total\_size\_out = static\_cast<int>(keep.size());

TensorShape shape\_out = {total\_size\_out};

OP\_REQUIRES\_OK(context, context->allocate\_output(0, shape\_out, &output\_mask));

auto output\_mask\_ = output\_mask->flat<bool>();

// Check if value is in set, output is boolean

for (int j = 0; j < keep.size(); j++){

output\_mask\_(j) = (feature\_set\_keep.count(keep(j)));

}

}

};

#define REGISTER(Type) \

\

REGISTER\_KERNEL\_BUILDER( \

Name("FeatureMask") \

.Device(DEVICE\_CPU) \

.TypeConstraint<Type>("T"), \

FeatureMask<Type>); \

REGISTER(int64);

REGISTER(int8);