The Ultimate Compression Pipeline ~ Ajay Maheshwari

In [191]:
```
! pip install -q tensorflow-model-optimization
```

In [192]:
```
import tensorflow as tf
import tf_keras as keras

import numpy as np
import tempfile
import zipfile
import os
```

In [193]:
```
def get_gzipped_model_size(model):
  with tempfile.NamedTemporaryFile(suffix=".h5") as temp_file:
    model.save(temp_file.name)

    _, zipped_file = tempfile.mkstemp('.zip')
    with zipfile.ZipFile(zipped_file, 'w', compression=zipfile.ZIP_DEFLATED) as f:
      f.write(temp_file.name)

    # print(f"Zipped model is saved at: {zipped_file}")

    x = os.path.getsize(zipped_file)

    os.remove(zipped_file)
    # print(f"Temporary zip file removed: {zipped_file}")

    return x / 1000


def print_model_weights_sparsity(model):
    for layer in model.layers:
        if isinstance(layer, keras.layers.Wrapper):
            weights = layer.trainable_weights
        else:
            weights = layer.weights
        for weight in weights:
            if "kernel" not in weight.name or "centroid" in weight.name:
                continue
            weight_size = weight.numpy().size
            zero_num = np.count_nonzero(weight == 0)
            print(
                f"{weight.name}: {zero_num/weight_size:.2%} sparsity ",
                f"({zero_num}/{weight_size})",
            )


def get_gzipped_model_size2(file):

  _, zipped_file = tempfile.mkstemp('.zip')
  with zipfile.ZipFile(zipped_file, 'w', compression=zipfile.ZIP_DEFLATED) as f:
    f.write(file)

  return os.path.getsize(zipped_file)/1000


def eval_model(interpreter):
  input_index = interpreter.get_input_details()[0]["index"]
  output_index = interpreter.get_output_details()[0]["index"]

  prediction_digits = []
  for i, test_image in enumerate(test_images):
    test_image = np.expand_dims(test_image, axis=0).astype(np.float32)
    interpreter.set_tensor(input_index, test_image)

    interpreter.invoke()

    output = interpreter.tensor(output_index)
    digit = np.argmax(output()[0])
    prediction_digits.append(digit)

  prediction_digits = np.array(prediction_digits)
  accuracy = (prediction_digits == test_labels).mean()
  return accuracy

model_acc = []
model_sz = []
```

Creating a Base Model - 1.0

In [194]:
```
# Load MNIST dataset
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Normalize the input image so that each pixel value is between 0 to 1.
train_images = train_images / 255.0
test_images  = test_images / 255.0

model = keras.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
```

```
  keras.layers.Dense(300, activation='relu'),
  keras.layers.Dense(100, activation='relu'),
  keras.layers.Dense(10, activation='softmax')
])

opt = keras.optimizers.Adam(learning_rate=1e-3)

# Train the digit classification model
model.compile(optimizer=opt,
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

model.fit(
    train_images,
    train_labels,
    validation_split=0.1,
    epochs=25
)
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use th
e legacy TF-Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
```

```
Epoch 1/25
```

```
/Users/ajaymaheshwari/anaconda3/lib/python3.11/site-packages/tf_keras/src/backend.py:5729: UserWarning: "`sparse_ca
tegorical_crossentropy` received `from_logits=True`, but the `output` argument was produced by a Softmax activation
and thus does not represent logits. Was this intended?
  output, from_logits = _get_logits(
```

```
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2151 - accuracy: 0.9351 - val_loss: 0.0892 - val
_accuracy: 0.9717
Epoch 2/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0880 - accuracy: 0.9727 - val_loss: 0.0752 - val
_accuracy: 0.9778
Epoch 3/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0591 - accuracy: 0.9823 - val_loss: 0.0792 - val
_accuracy: 0.9778
Epoch 4/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0448 - accuracy: 0.9860 - val_loss: 0.0922 - val
_accuracy: 0.9752
Epoch 5/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0342 - accuracy: 0.9889 - val_loss: 0.0763 - val
_accuracy: 0.9795
Epoch 6/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0278 - accuracy: 0.9905 - val_loss: 0.0733 - val
_accuracy: 0.9803
Epoch 7/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0216 - accuracy: 0.9929 - val_loss: 0.0793 - val
_accuracy: 0.9808
Epoch 8/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0211 - accuracy: 0.9926 - val_loss: 0.0888 - val
_accuracy: 0.9795
Epoch 9/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0164 - accuracy: 0.9943 - val_loss: 0.1008 - val
_accuracy: 0.9770
Epoch 10/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0154 - accuracy: 0.9949 - val_loss: 0.1088 - val
_accuracy: 0.9782
Epoch 11/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0160 - accuracy: 0.9947 - val_loss: 0.0976 - val
_accuracy: 0.9808
Epoch 12/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0135 - accuracy: 0.9956 - val_loss: 0.0987 - val
_accuracy: 0.9782
Epoch 13/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0146 - accuracy: 0.9951 - val_loss: 0.0903 - val
_accuracy: 0.9828
Epoch 14/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0111 - accuracy: 0.9965 - val_loss: 0.1019 - val
_accuracy: 0.9782
Epoch 15/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0102 - accuracy: 0.9969 - val_loss: 0.1321 - val
_accuracy: 0.9755
Epoch 16/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0126 - accuracy: 0.9960 - val_loss: 0.1052 - val
_accuracy: 0.9815
Epoch 17/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0099 - accuracy: 0.9970 - val_loss: 0.1174 - val
_accuracy: 0.9803
Epoch 18/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0115 - accuracy: 0.9964 - val_loss: 0.1348 - val
_accuracy: 0.9813
Epoch 19/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0085 - accuracy: 0.9974 - val_loss: 0.1370 - val
_accuracy: 0.9795
Epoch 20/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0094 - accuracy: 0.9970 - val_loss: 0.1158 - val
_accuracy: 0.9800
Epoch 21/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0100 - accuracy: 0.9968 - val_loss: 0.1308 - val
_accuracy: 0.9795
Epoch 22/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0067 - accuracy: 0.9982 - val_loss: 0.1377 - val
_accuracy: 0.9800
Epoch 23/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0098 - accuracy: 0.9971 - val_loss: 0.1264 - val
_accuracy: 0.9820
Epoch 24/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0067 - accuracy: 0.9980 - val_loss: 0.1459 - val
_accuracy: 0.9798
Epoch 25/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0090 - accuracy: 0.9971 - val_loss: 0.1359 - val
_accuracy: 0.9817
```

Out[194]: <tf_keras.src.callbacks.History at 0x2899d5f90>

Evaluate the baseline model and save it for later usage

In [195]:
```
_, baseline_model_accuracy = model.evaluate(
    test_images, test_labels, verbose=0)

print('Baseline test accuracy:', baseline_model_accuracy)

sz = get_gzipped_model_size(model)
print("Base model size: ",  sz , ' KB' )

model_acc.append(baseline_model_accuracy)
model_sz.append(sz)

print(f"Top-1 Error = {(1-baseline_model_accuracy)*100:.2f}%")
```

```
Baseline test accuracy: 0.9801999926567078
Base model size:  2861.216  KB
Top-1 Error = 1.98%
```

```
/Users/ajaymaheshwari/anaconda3/lib/python3.11/site-packages/tf_keras/src/engine/training.py:3098: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format is considered legacy. We recommend using instead the native TF-Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

---------- Checkpoint Point 1 ---------

Pruning and then fine-tuning the model - 2.0

In [196]:
```
import tensorflow_model_optimization as tfmot

pruning_params = {
          'pruning_schedule': tfmot.sparsity.keras.ConstantSparsity(0.7, begin_step=0, frequency=100)
}

callbacks = [
      tfmot.sparsity.keras.UpdatePruningStep()
]


pruned_model = model

for i in range(1):
    prune_low_magnitude = tfmot.sparsity.keras.prune_low_magnitude

    pruned_model = prune_low_magnitude(pruned_model, **pruning_params)

    # learning rate for fine-tuning
    opt = keras.optimizers.Adam(learning_rate=1e-5)


    pruned_model.compile(
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      optimizer=opt,
      metrics=['accuracy'])

    # Fine-tune model
    pruned_model.fit(
      train_images,
      train_labels,
      epochs=25,
      validation_split=0.1,
      callbacks=callbacks)

    stripped_pruned_model = tfmot.sparsity.keras.strip_pruning(pruned_model)

    print_model_weights_sparsity(stripped_pruned_model)

    pruned_model = stripped_pruned_model
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use the legacy TF-Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
```

```
Epoch 1/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0694 - accuracy: 0.9773 - val_loss: 0.1067 - val_accuracy: 0.9692
Epoch 2/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0504 - accuracy: 0.9842 - val_loss: 0.0923 - val_accuracy: 0.9743
Epoch 3/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0385 - accuracy: 0.9887 - val_loss: 0.0846 - val_accuracy: 0.9765
Epoch 4/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0313 - accuracy: 0.9912 - val_loss: 0.0798 - val_accuracy: 0.9778
Epoch 5/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0265 - accuracy: 0.9928 - val_loss: 0.0767 - val_accuracy: 0.9790
Epoch 6/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0229 - accuracy: 0.9940 - val_loss: 0.0744 - val_accuracy: 0.9798
Epoch 7/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0202 - accuracy: 0.9948 - val_loss: 0.0728 - val_accuracy: 0.9800
Epoch 8/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0180 - accuracy: 0.9955 - val_loss: 0.0715 - val_accuracy: 0.9807
```

```
_accuracy: 0.9807
Epoch 9/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0162 - accuracy: 0.9960 - val_loss: 0.0704 - val
_accuracy: 0.9822
Epoch 10/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0147 - accuracy: 0.9964 - val_loss: 0.0695 - val
_accuracy: 0.9823
Epoch 11/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0134 - accuracy: 0.9969 - val_loss: 0.0688 - val
_accuracy: 0.9820
Epoch 12/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0123 - accuracy: 0.9971 - val_loss: 0.0682 - val
_accuracy: 0.9818
Epoch 13/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0113 - accuracy: 0.9975 - val_loss: 0.0678 - val
_accuracy: 0.9822
Epoch 14/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0104 - accuracy: 0.9978 - val_loss: 0.0674 - val
_accuracy: 0.9825
Epoch 15/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0096 - accuracy: 0.9979 - val_loss: 0.0671 - val
_accuracy: 0.9827
Epoch 16/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0089 - accuracy: 0.9981 - val_loss: 0.0669 - val
_accuracy: 0.9832
Epoch 17/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0083 - accuracy: 0.9983 - val_loss: 0.0668 - val
_accuracy: 0.9835
Epoch 18/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0077 - accuracy: 0.9985 - val_loss: 0.0666 - val
_accuracy: 0.9835
Epoch 19/25
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0072 - accuracy: 0.9988 - val_loss: 0.0665 - val
_accuracy: 0.9838
Epoch 20/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0068 - accuracy: 0.9990 - val_loss: 0.0665 - val
_accuracy: 0.9838
Epoch 21/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0063 - accuracy: 0.9991 - val_loss: 0.0665 - val
_accuracy: 0.9838
Epoch 22/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0059 - accuracy: 0.9992 - val_loss: 0.0666 - val
_accuracy: 0.9838
Epoch 23/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0056 - accuracy: 0.9992 - val_loss: 0.0667 - val
_accuracy: 0.9840
Epoch 24/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0052 - accuracy: 0.9993 - val_loss: 0.0667 - val
_accuracy: 0.9842
Epoch 25/25
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0049 - accuracy: 0.9994 - val_loss: 0.0668 - val
_accuracy: 0.9842
dense_59/kernel:0: 70.00% sparsity  (164640/235200)
dense_60/kernel:0: 70.00% sparsity  (21000/30000)
dense_61/kernel:0: 70.00% sparsity  (700/1000)
```

In [197]:
```
pruned_model.compile(
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      optimizer=opt,
      metrics=['accuracy'])
```

In [198]:
```
_, pruned_model_accuracy = pruned_model.evaluate(
    test_images, test_labels, verbose=0)

print('Pruned Model test accuracy:', pruned_model_accuracy)

sz = get_gzipped_model_size(pruned_model)
print("Stripped model size: ",  sz , ' KB' )

model_acc.append(pruned_model_accuracy)
model_sz.append(sz)
```

```
Pruned Model test accuracy: 0.9828000068664551
Stripped model size:  2155.777  KB
```

---------- Checkpoint Point 2 ---------

Knowledge Distillation - 3.0

In [199]:
```
from keras import layers
from keras import ops
import numpy as np

class Distiller(keras.Model):
    def __init__(self, student, teacher):
        super().__init__()
        self.teacher = teacher
        self.student = student

    def compile(
        self,
        optimizer,
        metrics,
        student_loss_fn,
        distillation_loss_fn,
        alpha=0.1,
        temperature=3,
    ):
```

```
    ):
        super().compile(optimizer=optimizer, metrics=metrics)
        self.student_loss_fn = student_loss_fn
        self.distillation_loss_fn = distillation_loss_fn
        self.alpha = alpha
        self.temperature = temperature

    def compute_loss(
        self, x=None, y=None, y_pred=None, sample_weight=None, allow_empty=False
    ):
        teacher_pred = self.teacher(x, training=False)
        student_loss = self.student_loss_fn(y, y_pred)

        distillation_loss = self.distillation_loss_fn(
            ops.softmax(teacher_pred / self.temperature, axis=1),
            ops.softmax(y_pred / self.temperature, axis=1),
        ) * (self.temperature**2)

        loss = self.alpha * student_loss + (1 - self.alpha) * distillation_loss
        return loss

    def call(self, x):
        return self.student(x)
```

In [200]:

```
teacher_model = keras.Sequential(
    [
        keras.layers.Flatten(input_shape=(28, 28)),
        keras.layers.Dense(500, activation='relu'),
        keras.layers.Dense(400, activation='relu'),
        keras.layers.Dense(300, activation='relu'),
        keras.layers.Dense(200, activation='relu'),
        keras.layers.Dense(100, activation='relu'),
        keras.layers.Dense(10, activation='softmax')
    ],
    name="teacher",
)

# teacher = keras.Sequential(
#     [
#           keras.layers.InputLayer(input_shape=(28, 28)),
#           keras.layers.Reshape(target_shape=(28, 28, 1)),
#           keras.layers.Conv2D(filters=12, kernel_size=(3, 3),
#                               activation=tf.nn.relu),
#           keras.layers.MaxPooling2D(pool_size=(2, 2)),
#           keras.layers.Flatten(),
#           keras.layers.Dense(10)
#     ],
#     name="teacher",
# )


teacher.compile(
    optimizer=keras.optimizers.Adam(),
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
)
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use th
e legacy TF-Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
```

In [201]:

```
teacher.fit(train_images, train_labels, epochs=20)
teacher.evaluate(test_images, test_labels)
```

```
Epoch 1/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0407 - sparse_categorical_accuracy: 0.9936
Epoch 2/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0105 - sparse_categorical_accuracy: 0.9969
Epoch 3/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0073 - sparse_categorical_accuracy: 0.9977
Epoch 4/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0041 - sparse_categorical_accuracy: 0.9987
Epoch 5/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0038 - sparse_categorical_accuracy: 0.9987
Epoch 6/20
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0014 - sparse_categorical_accuracy: 0.9995
Epoch 7/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0024 - sparse_categorical_accuracy: 0.9993
Epoch 8/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0038 - sparse_categorical_accuracy: 0.9988
Epoch 9/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0030 - sparse_categorical_accuracy: 0.9992
Epoch 10/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0023 - sparse_categorical_accuracy: 0.9994
Epoch 11/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0024 - sparse_categorical_accuracy: 0.9995
Epoch 12/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0035 - sparse_categorical_accuracy: 0.9991
Epoch 13/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0026 - sparse_categorical_accuracy: 0.9993
Epoch 14/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0014 - sparse_categorical_accuracy: 0.9996
Epoch 15/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0024 - sparse_categorical_accuracy: 0.9994
Epoch 16/20
1875/1875 [==============================] - 7s 4ms/step - loss: 4.9741e-04 - sparse_categorical_accuracy: 0.9998
Epoch 17/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0029 - sparse_categorical_accuracy: 0.9992
Epoch 18/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0027 - sparse_categorical_accuracy: 0.9995
Epoch 19/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0021 - sparse_categorical_accuracy: 0.9993
Epoch 20/20
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0016 - sparse_categorical_accuracy: 0.9996
313/313 [==============================] - 1s 1ms/step - loss: 0.0651 - sparse_categorical_accuracy: 0.9919
```

Out[201]:
```
[0.06505972892045975, 0.9919000267982483]
```

In [202]:
```
distiller = Distiller(student=pruned_model, teacher=teacher)
distiller.compile(
    optimizer=keras.optimizers.Adam(),
    metrics=[keras.metrics.SparseCategoricalAccuracy()],
    student_loss_fn=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    distillation_loss_fn=keras.losses.KLDivergence(),
    alpha=0.1,
    temperature=10,
)
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.Adam` runs slowly on M1/M2 Macs, please use th
e legacy TF-Keras optimizer instead, located at `tf.keras.optimizers.legacy.Adam`.
```

In [203]:
```
# Distill teacher se student
distiller.fit(train_images, train_labels, epochs=15)

distiller.evaluate(test_images, test_labels)
```

```
Epoch 1/15
1875/1875 [==============================] - 9s 4ms/step - sparse_categorical_accuracy: 0.9954
Epoch 2/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9967
Epoch 3/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9963
Epoch 4/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9963
Epoch 5/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9965
Epoch 6/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9979
Epoch 7/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9975
Epoch 8/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9972
Epoch 9/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9977
Epoch 10/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9973
Epoch 11/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9981
Epoch 12/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9972
Epoch 13/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9984
Epoch 14/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9980
Epoch 15/15
1875/1875 [==============================] - 8s 4ms/step - sparse_categorical_accuracy: 0.9974
313/313 [==============================] - 0s 508us/step - sparse_categorical_accuracy: 0.9831
```

Out[203]:
```
0.9830999970436096
```

In [204]:
```
_, acc = distiller.student.evaluate(
    test_images, test_labels, verbose=0)

print('Distilled Model test accuracy:', acc)

sz = get_gzipped_model_size(distiller.student)
print("Distilled model size: ",  sz , ' KB' )

model_acc.append(acc)
model_sz.append(sz)
```

```
Distilled Model test accuracy: 0.9830999970436096
Distilled model size:  2646.814  KB
```

---------- Checkpoint Point 3 ---------

Weight Clustering - 4.0

In [205]:
```
def print_model_weight_clusters(model):
    for layer in model.layers:
        if isinstance(layer, keras.layers.Wrapper):
            weights = layer.trainable_weights
        else:
            weights = layer.weights
        for weight in weights:
            # ignore auxiliary quantization weights
            if "quantize_layer" in weight.name:
                continue
            if "kernel" in weight.name:
                unique_count = len(np.unique(weight))
                print(
                    f"{layer.name}/{weight.name}: {unique_count} clusters "
                )
```

In [206]:
```
import tensorflow_model_optimization as tfmot
from tensorflow_model_optimization.python.core.clustering.keras.experimental import (
    cluster,
)

cluster_weights = tfmot.clustering.keras.cluster_weights
CentroidInitialization = tfmot.clustering.keras.CentroidInitialization

cluster_weights = cluster.cluster_weights
```

```
clustering_params = {
  'number_of_clusters': 8,
  'cluster_centroids_init': CentroidInitialization.KMEANS_PLUS_PLUS,
  'preserve_sparsity': True
}

sparsity_clustered_model = cluster_weights(distiller.student, **clustering_params)

sparsity_clustered_model.compile(optimizer='adam',
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

print('Train sparsity preserving clustering model:')
sparsity_clustered_model.fit(train_images, train_labels,epochs=10, validation_split=0.1)
```

```
Train sparsity preserving clustering model:
Epoch 1/10
1688/1688 [==============================] - 11s 6ms/step - loss: 0.0060 - accuracy: 0.9980 - val_loss: 0.0077 - va
l_accuracy: 0.9977
Epoch 2/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0037 - accuracy: 0.9990 - val_loss: 0.0117 - va
l_accuracy: 0.9958
Epoch 3/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0027 - accuracy: 0.9991 - val_loss: 0.0130 - va
l_accuracy: 0.9973
Epoch 4/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0047 - accuracy: 0.9986 - val_loss: 0.0189 - va
l_accuracy: 0.9935
Epoch 5/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0072 - accuracy: 0.9976 - val_loss: 0.0290 - va
l_accuracy: 0.9923
Epoch 6/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0046 - accuracy: 0.9987 - val_loss: 0.0289 - va
l_accuracy: 0.9920
Epoch 7/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0062 - accuracy: 0.9982 - val_loss: 0.0203 - va
l_accuracy: 0.9938
Epoch 8/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.0227 - va
l_accuracy: 0.9940
Epoch 9/10
1688/1688 [==============================] - 10s 6ms/step - loss: 0.0046 - accuracy: 0.9984 - val_loss: 0.0264 - va
l_accuracy: 0.9943
Epoch 10/10
1688/1688 [==============================] - 12s 7ms/step - loss: 0.0048 - accuracy: 0.9985 - val_loss: 0.0296 - va
l_accuracy: 0.9925
```

Out[206]:
```
<tf_keras.src.callbacks.History at 0x285acbed0>
```

In [207]:
```
stripped_clustered_model = tfmot.clustering.keras.strip_clustering(sparsity_clustered_model)

print("Model sparsity:\n")
print_model_weights_sparsity(stripped_clustered_model)

print("\nModel clusters:\n")
print_model_weight_clusters(stripped_clustered_model)
```

```
Model sparsity:

kernel:0: 44.18% sparsity  (103913/235200)
kernel:0: 17.72% sparsity  (5315/30000)
kernel:0: 32.50% sparsity  (325/1000)

Model clusters:

dense_59/kernel:0: 8 clusters
dense_60/kernel:0: 8 clusters
dense_61/kernel:0: 8 clusters
```

In [208]:
```
stripped_clustered_model.compile(optimizer=opt,
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [209]:
```
_, stripped_clustered_model_accuracy = stripped_clustered_model.evaluate(
    test_images, test_labels, verbose=0)

print('Clustered Model test accuracy:', stripped_clustered_model_accuracy)

sz = get_gzipped_model_size(stripped_clustered_model)
print("Clustered model size: ",  sz , ' KB' )

model_acc.append(stripped_clustered_model_accuracy)
model_sz.append(sz)
```

```
Clustered Model test accuracy: 0.9824000000953674
Clustered model size:  1859.576  KB
```

---------- Checkpoint Point 4 ---------

Quantization - 5.0

In [210]:
```
quant_aware_annotate_model = tfmot.quantization.keras.quantize_annotate_model(
              stripped_clustered_model)
quant_model = tfmot.quantization.keras.quantize_apply(
              quant_aware_annotate_model,
              tfmot.experimental.combine.Default8BitClusterPreserveQuantizeScheme(preserve_sparsity=True))

quant_model.compile(optimizer='adam',
```

```
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
print('Training after quantization model:')
quant_model.fit(train_images, train_labels, batch_size=128, epochs=3, validation_split=0.1)
```

```
Training after quantization model:
Epoch 1/3
WARNING:tensorflow:Gradients do not exist for variables ['dense_59/kernel:0', 'dense_60/kernel:0', 'dense_61/kerne
l:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss` argument?

WARNING:tensorflow:Gradients do not exist for variables ['dense_59/kernel:0', 'dense_60/kernel:0', 'dense_61/kerne
l:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss` argument?

WARNING:tensorflow:Gradients do not exist for variables ['dense_59/kernel:0', 'dense_60/kernel:0', 'dense_61/kerne
l:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss` argument?

WARNING:tensorflow:Gradients do not exist for variables ['dense_59/kernel:0', 'dense_60/kernel:0', 'dense_61/kerne
l:0'] when minimizing the loss. If you're using `model.compile()`, did you forget to provide a `loss` argument?

422/422 [==============================] - 4s 8ms/step - loss: 0.0015 - accuracy: 0.9997 - val_loss: 0.0241 - val_a
ccuracy: 0.9953
Epoch 2/3
422/422 [==============================] - 3s 8ms/step - loss: 6.8470e-05 - accuracy: 1.0000 - val_loss: 0.0281 - v
al_accuracy: 0.9948
Epoch 3/3
422/422 [==============================] - 3s 8ms/step - loss: 2.1337e-05 - accuracy: 1.0000 - val_loss: 0.0289 - v
al_accuracy: 0.9952
```

Out[210]: `<tf_keras.src.callbacks.History at 0x2910987d0>`

In [211]:

```
print("Final Model clusters:")
print_model_weight_clusters(quant_model)
print("\nFinal Model sparsity:")
print_model_weights_sparsity(quant_model)
```

```
Final Model clusters:
quant_dense_59/dense_59/kernel:0: 8 clusters
quant_dense_60/dense_60/kernel:0: 8 clusters
quant_dense_61/dense_61/kernel:0: 8 clusters

Final Model sparsity:
dense_59/kernel:0: 50.71% sparsity  (119281/235200)
dense_60/kernel:0: 21.72% sparsity  (6515/30000)
dense_61/kernel:0: 33.00% sparsity  (330/1000)
```

In [212]:

```
converter = tf.lite.TFLiteConverter.from_keras_model(quant_model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
final_tflite_model = converter.convert()
final_model_file = 'final_model.tflite'
# Save the model.
with open(final_model_file, 'wb') as f:
    f.write(final_tflite_model)


sz = get_gzipped_model_size2(final_model_file)
print("Final model size: ", sz, ' KB')

model_sz.append(sz)
```

```
INFO:tensorflow:Assets written to: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpkwpscfqz/assets

INFO:tensorflow:Assets written to: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpkwpscfqz/assets
/Users/ajaymaheshwari/anaconda3/lib/python3.11/site-packages/tensorflow/lite/python/convert.py:964: UserWarning: St
atistics for quantized inputs were expected, but not specified; continuing anyway.
  warnings.warn(

Final model size:  76.195  KB

W0000 00:00:1717008753.144153  103872 tf_tfl_flatbuffer_helpers.cc:390] Ignored output_format.
W0000 00:00:1717008753.145295  103872 tf_tfl_flatbuffer_helpers.cc:393] Ignored drop_control_dependency.
2024-05-30 00:22:33.147532: I tensorflow/cc/saved_model/reader.cc:83] Reading SavedModel from: /var/folders/px/z8lb
6znd6q95tq6vlznyb0s40000gn/T/tmpkwpscfqz
2024-05-30 00:22:33.157069: I tensorflow/cc/saved_model/reader.cc:51] Reading meta graph with tags { serve }
2024-05-30 00:22:33.157085: I tensorflow/cc/saved_model/reader.cc:146] Reading SavedModel debug info (if present) f
rom: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpkwpscfqz
2024-05-30 00:22:33.178042: I tensorflow/cc/saved_model/loader.cc:234] Restoring SavedModel bundle.
2024-05-30 00:22:33.232863: I tensorflow/cc/saved_model/loader.cc:218] Running initialization op on SavedModel bund
le at path: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpkwpscfqz
2024-05-30 00:22:33.244557: I tensorflow/cc/saved_model/loader.cc:317] SavedModel load for tags { serve }; Status:
success: OK. Took 97035 microseconds.
```

In [213]:

```
interpreter = tf.lite.Interpreter(final_model_file)
interpreter.allocate_tensors()

final_test_accuracy = eval_model(interpreter)

print('Final test accuracy:', final_test_accuracy)

model_acc.append(final_test_accuracy)
```

```
Final test accuracy: 0.9843
```

---------- Checkpoint Point 5 ---------

In [214]:

```
for i in range(len(model_acc)):
    print(f"Accuracy = { round(model_acc[i]*100,2)}  |  size = {model_sz[i]} KB  |  Top-1 Error = {(1-model_acc[i])*
```

```
Accuracy = 98.02  |  size = 2861.216 KB  |  Top-1 Error = 1.98%
Accuracy = 98.28  |  size = 2155.777 KB  |  Top-1 Error = 1.72%
Accuracy = 98.31  |  size = 2646.814 KB  |  Top-1 Error = 1.69%
Accuracy = 98.24  |  size = 1859.576 KB  |  Top-1 Error = 1.76%
Accuracy = 98.43  |  size = 76.195 KB  |  Top-1 Error = 1.57%
```

```
-------- Final Comparison Summary --------
```