The Ultimate Compression Pipeline ~ Ajay Maheshwari

In [344]:
```
! pip install -q tensorflow-model-optimization
```

In [345]:
```
import tensorflow as tf
import tf_keras as keras

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

In [346]:
```
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}")

    return os.path.getsize(zipped_file) / 1000


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 [347]:
```
# 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.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)
])

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=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`.

Epoch 1/10
1688/1688 [==============================] - 6s 4ms/step - loss: 0.3141 - accuracy: 0.9116 - val_loss: 0.1401 - val
_accuracy: 0.9600
Epoch 2/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1326 - accuracy: 0.9623 - val_loss: 0.0881 - val
_accuracy: 0.9782
Epoch 3/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.0932 - accuracy: 0.9733 - val_loss: 0.0784 - val
_accuracy: 0.9818
Epoch 4/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.0742 - accuracy: 0.9786 - val_loss: 0.0639 - val
_accuracy: 0.9837
Epoch 5/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0613 - accuracy: 0.9821 - val_loss: 0.0617 - val
_accuracy: 0.9843
Epoch 6/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0543 - accuracy: 0.9834 - val_loss: 0.0590 - val
_accuracy: 0.9845
Epoch 7/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.0475 - accuracy: 0.9857 - val_loss: 0.0577 - val
_accuracy: 0.9857
Epoch 8/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0429 - accuracy: 0.9871 - val_loss: 0.0603 - val
_accuracy: 0.9842
Epoch 9/10
1688/1688 [==============================] - 4s 2ms/step - loss: 0.0382 - accuracy: 0.9884 - val_loss: 0.0606 - val
_accuracy: 0.9852
Epoch 10/10
1688/1688 [==============================] - 4s 3ms/step - loss: 0.0349 - accuracy: 0.9893 - val_loss: 0.0628 - val
_accuracy: 0.9833
```

Out[347]: `<tf_keras.src.callbacks.History at 0x289802310>`

Evaluate the baseline model and save it for later usage

In [348]:
```
_, 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)
```

```
Baseline test accuracy: 0.9804999828338623
Zipped model is saved at: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmp4z6cyg7c.zip
Base model size:  235.111  KB
```

```
/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 [349]:
```
import tensorflow_model_optimization as tfmot

pruning_params = {
          'pruning_schedule': tfmot.sparsity.keras.ConstantSparsity(0.5, begin_step=0, frequency=100)
}

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


pruned_model = model

for i in range(4):
    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=10,
      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
```

```
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0367 - accuracy: 0.9906 - val_loss: 0.0573 - val
_accuracy: 0.9847
Epoch 6/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0364 - accuracy: 0.9906 - val_loss: 0.0570 - val
_accuracy: 0.9845
Epoch 7/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0361 - accuracy: 0.9906 - val_loss: 0.0568 - val
_accuracy: 0.9848
Epoch 8/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0358 - accuracy: 0.9906 - val_loss: 0.0566 - val
_accuracy: 0.9848
Epoch 9/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0356 - accuracy: 0.9907 - val_loss: 0.0564 - val
_accuracy: 0.9850
Epoch 10/10
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0354 - accuracy: 0.9908 - val_loss: 0.0563 - val
_accuracy: 0.9850
```

```
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 [350]:
```
pruned_model.compile(
      loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      optimizer=opt,
      metrics=['accuracy'])
```

In [351]:
```
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})",
            )
```

In [352]:
```
_, 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: 0.9807999730110168
Zipped model is saved at: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpravhtlmn.zip
Stripped model size:  191.29  KB
```

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

Knowledge Distillation - 3.0

In [353]:
```
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(
```

```
        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 [354]:
```
teacher = keras.Sequential(
    [
        keras.layers.InputLayer(input_shape=(28, 28)),
        keras.layers.Reshape(target_shape=(28, 28, 1)),
        keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation=tf.nn.relu),  # Increase filters
        keras.layers.MaxPooling2D(pool_size=(2, 2)),
        keras.layers.Conv2D(filters=24, kernel_size=(3, 3), activation=tf.nn.relu),  # Add another layer
        keras.layers.MaxPooling2D(pool_size=(2, 2)),
        keras.layers.Flatten(),
        keras.layers.Dense(units=128, activation=tf.nn.relu),  # Add a hidden layer
        keras.layers.Dense(10)
    ],
    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 [355]:
```
teacher.fit(train_images, train_labels, epochs=12)
teacher.evaluate(test_images, test_labels)
```

```
Epoch 1/12
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1660 - sparse_categorical_accuracy: 0.9498
Epoch 2/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0514 - sparse_categorical_accuracy: 0.9848
Epoch 3/12
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0356 - sparse_categorical_accuracy: 0.9887
Epoch 4/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0267 - sparse_categorical_accuracy: 0.9915
Epoch 5/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0207 - sparse_categorical_accuracy: 0.9937
Epoch 6/12
1875/1875 [==============================] - 10s 6ms/step - loss: 0.0156 - sparse_categorical_accuracy: 0.9954
Epoch 7/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0131 - sparse_categorical_accuracy: 0.9956
Epoch 8/12
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0109 - sparse_categorical_accuracy: 0.9963
Epoch 9/12
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0086 - sparse_categorical_accuracy: 0.9970
Epoch 10/12
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0078 - sparse_categorical_accuracy: 0.9976
Epoch 11/12
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0069 - sparse_categorical_accuracy: 0.9977
Epoch 12/12
1875/1875 [==============================] - 10s 6ms/step - loss: 0.0060 - sparse_categorical_accuracy: 0.9981
313/313 [==============================] - 1s 3ms/step - loss: 0.0411 - sparse_categorical_accuracy: 0.9896
```

Out[355]:
```
[0.04109995812177658, 0.9896000027656555]
```

In [356]:
```
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 [357]:
```
# Distill teacher se student
distiller.fit(train_images, train_labels, epochs=15)

distiller.evaluate(test_images, test_labels)
```

```
Epoch 1/15
1875/1875 [==============================] - 17s 9ms/step - sparse_categorical_accuracy: 0.9789
Epoch 2/15
1875/1875 [==============================] - 13s 7ms/step - sparse_categorical_accuracy: 0.9702
```

```
Epoch 3/15
1875/1875 [==============================] - 10s 6ms/step - sparse_categorical_accuracy: 0.9651
Epoch 4/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9628
Epoch 5/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9639
Epoch 6/15
1875/1875 [==============================] - 11s 6ms/step - sparse_categorical_accuracy: 0.9656
Epoch 7/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9677
Epoch 8/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9701
Epoch 9/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9711
Epoch 10/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9723
Epoch 11/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9735
Epoch 12/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9740
Epoch 13/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9746
Epoch 14/15
1875/1875 [==============================] - 10s 5ms/step - sparse_categorical_accuracy: 0.9750
Epoch 15/15
1875/1875 [==============================] - 10s 6ms/step - sparse_categorical_accuracy: 0.9754
313/313 [==============================] - 1s 845us/step - sparse_categorical_accuracy: 0.9783
```

Out[357]:
```
0.9782999753952026
```

In [358]:
```
_, 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: 0.9782999753952026
Zipped model is saved at: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmp_n211f84.zip
Distilled model size:  217.632  KB
```

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

Weight Clustering - 4.0

In [359]:
```
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 [360]:
```
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=6, validation_split=0.1)
```

```
Train sparsity preserving clustering model:
Epoch 1/6
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0843 - accuracy: 0.9749 - val_loss: 0.0609 - val
_accuracy: 0.9823
Epoch 2/6
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0667 - accuracy: 0.9801 - val_loss: 0.0634 - val
_accuracy: 0.9818
Epoch 3/6
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0668 - accuracy: 0.9794 - val_loss: 0.0728 - val
```

```
_accuracy: 0.9785
Epoch 4/6
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0621 - accuracy: 0.9806 - val_loss: 0.0718 - val
_accuracy: 0.9810
Epoch 5/6
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0605 - accuracy: 0.9805 - val_loss: 0.0666 - val
_accuracy: 0.9817
Epoch 6/6
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0539 - accuracy: 0.9828 - val_loss: 0.0569 - val
_accuracy: 0.9853
```

Out[360]: <tf_keras.src.callbacks.History at 0x30c393d90>

In [361]:
```
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: 33.33% sparsity  (36/108)
kernel:0: 70.63% sparsity  (14324/20280)

Model clusters:

conv2d_29/kernel:0: 8 clusters
dense_29/kernel:0: 8 clusters
```

In [362]:
```
stripped_clustered_model.compile(optimizer=opt,
              loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [363]:
```
_, 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: 0.9821000099182129
Zipped model is saved at: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmpgqmo6le8.zip
Clustered model size:  152.239  KB
```

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

Quantization - 5.0

In [364]:
```
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 ['conv2d_29/kernel:0', 'dense_29/kernel: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 ['conv2d_29/kernel:0', 'dense_29/kernel: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 ['conv2d_29/kernel:0', 'dense_29/kernel: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 ['conv2d_29/kernel:0', 'dense_29/kernel:0'] when minimizing
the loss. If you're using `model.compile()`, did you forget to provide a `loss` argument?

422/422 [==============================] - 4s 7ms/step - loss: 0.0492 - accuracy: 0.9864 - val_loss: 0.0560 - val_a
ccuracy: 0.9853
Epoch 2/3
422/422 [==============================] - 3s 7ms/step - loss: 0.0393 - accuracy: 0.9883 - val_loss: 0.0557 - val_a
ccuracy: 0.9860
Epoch 3/3
422/422 [==============================] - 3s 7ms/step - loss: 0.0373 - accuracy: 0.9884 - val_loss: 0.0554 - val_a
ccuracy: 0.9867
```

Out[364]: <tf_keras.src.callbacks.History at 0x30c6c9890>

In [365]:
```
print("Final Model clusters:")
print_model_weight_clusters(quant_model)
print("\nFinal Model sparsity:")
print_model_weights_sparsity(quant_model)
```

```
Final Model clusters:
quant_conv2d_29/conv2d_29/kernel:0: 8 clusters
quant_dense_29/dense_29/kernel:0: 8 clusters
```

```
Final Model sparsity:
conv2d_29/kernel:0: 33.33% sparsity  (36/108)
dense_29/kernel:0: 70.77% sparsity  (14352/20280)
```

In [366]:
```
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/tmph56pi8mm/assets
```

```
INFO:tensorflow:Assets written to: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmph56pi8mm/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(
W0000 00:00:1716489469.323926  555595 tf_tfl_flatbuffer_helpers.cc:390] Ignored output_format.
W0000 00:00:1716489469.324425  555595 tf_tfl_flatbuffer_helpers.cc:393] Ignored drop_control_dependency.
2024-05-24 00:07:49.325251: I tensorflow/cc/saved_model/reader.cc:83] Reading SavedModel from: /var/folders/px/z8lb
6znd6q95tq6vlznyb0s40000gn/T/tmph56pi8mm
2024-05-24 00:07:49.326528: I tensorflow/cc/saved_model/reader.cc:51] Reading meta graph with tags { serve }
2024-05-24 00:07:49.326533: I tensorflow/cc/saved_model/reader.cc:146] Reading SavedModel debug info (if present) f
rom: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmph56pi8mm
2024-05-24 00:07:49.342435: I tensorflow/cc/saved_model/loader.cc:234] Restoring SavedModel bundle.
2024-05-24 00:07:49.388163: I tensorflow/cc/saved_model/loader.cc:218] Running initialization op on SavedModel bund
le at path: /var/folders/px/z8lb6znd6q95tq6vlznyb0s40000gn/T/tmph56pi8mm
2024-05-24 00:07:49.397341: I tensorflow/cc/saved_model/loader.cc:317] SavedModel load for tags { serve }; Status:
success: OK. Took 72092 microseconds.
```

```
Final model size:  6.38  KB
```

In [367]:
```
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: 0.9819
```

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

In [368]:
```
for i in range(len(model_acc)):
    print(f"Accuracy = { round(model_acc[i]*100,2)} with size = {model_sz[i]} KB ")
```

```
Accuracy = 98.05 with size = 235.111 KB
Accuracy = 98.08 with size = 191.29 KB
Accuracy = 97.83 with size = 217.632 KB
Accuracy = 98.21 with size = 152.239 KB
Accuracy = 98.19 with size = 6.38 KB
```

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