```python
import tensorflow as tf
from tensorflow.keras import layers, models
from sklearn.model_selection import train_test_split

# Load the MNIST dataset
(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

# Preprocess the data
train_images = train_images / 255.0
test_images = test_images / 255.0

# Add a channel dimension to the images
train_images = train_images[..., tf.newaxis]
test_images = test_images[..., tf.newaxis]

# Split the training set into training and validation sets
train_images, val_images, train_labels, val_labels = train_test_split(train_images, train_labels, test_size=0.2, random_state=42)

# Parameters
batch_size = 32
num_classes = 10  # There are 10 classes in the MNIST dataset
num_epochs = 1
image_height, image_width = 28, 28

# Create the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(image_height, image_width, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Create data generators
train_dataset = tf.data.Dataset.from_tensor_slices(
    (train_images, train_labels)).batch(batch_size).shuffle(buffer_size=1024).prefetch(
    buffer_size=tf.data.experimental.AUTOTUNE)
val_dataset = tf.data.Dataset.from_tensor_slices(
    (val_images, val_labels)).batch(batch_size).prefetch(
    buffer_size=tf.data.experimental.AUTOTUNE)
test_dataset = tf.data.Dataset.from_tensor_slices(
    (test_images, test_labels)).batch(batch_size).prefetch(
    buffer_size=tf.data.experimental.AUTOTUNE)

# Train the model
model.fit(
    train_dataset,
    epochs=num_epochs,
    validation_data=val_dataset
)

# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_dataset)
print("Test accuracy:", test_accuracy)

# Predict labels for new images (example: first image from test set)
new_image = test_images[0:1]  # Take the first image from the test set
predictions = model.predict(new_image)
predicted_label = tf.argmax(predictions, axis=1)[0]
print("Predicted label for the first test image:", predicted_label.numpy())
```

```
1500/1500 [==============================] - 54s 35ms/step - loss: 0.2229 - accuracy: 0.9307 - val_loss: 0.0981 - val_accuracy: 0.9697
313/313 [==============================] - 3s 11ms/step - loss: 0.0871 - accuracy: 0.9715
Test accuracy: 0.9714999794960022
1/1 [==============================] - 0s 124ms/step
Predicted label for the first test image: 7
```