In [6]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.applications import VGG19
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
from tensorflow.image import grayscale_to_rgb, resize
import numpy as np
```

In [7]:
```
# Load and preprocess the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize pixel values to [0, 1] and reshape for VGG input shape
x_train = (x_train.astype('float32') / 255.0).reshape(-1, 28, 28, 1)
x_test = (x_test.astype('float32') / 255.0).reshape(-1, 28, 28, 1)

# Convert class labels to one-hot encoding
y_train, y_test = to_categorical(y_train), to_categorical(y_test)
```

In [8]:
```
# Load pre-trained VGG19 model with customized input shape
base_model = VGG19(weights='imagenet', include_top=False, input_shape=(48, 48, 3))

# Freeze the layers of the pre-trained VGG19 model
for layer in base_model.layers:
    layer.trainable = False
```

```
WARNING:tensorflow:From D:\JUPYTER FOLDER\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing
_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instea
d.

WARNING:tensorflow:From D:\JUPYTER FOLDER\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: T
he name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

In [9]:
```
# Extend VGG19 with new classification layers
model = Model(inputs=base_model.input, outputs=Dense(10, activation='softmax')(Dense(1024, activation='relu')

#outputs=Dense(10, activation='softmax')(Dense(1024, activation='relu')(Flatten()(base_model.output))))

# Prepare data for VGG input dimensions
x_train_vgg = grayscale_to_rgb(resize(x_train, (48, 48), method='bicubic'))
x_test_vgg = grayscale_to_rgb(resize(x_test, (48, 48), method='bicubic'))

# Compile, train, and evaluate the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
WARNING:tensorflow:From D:\JUPYTER FOLDER\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name t
f.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.
```

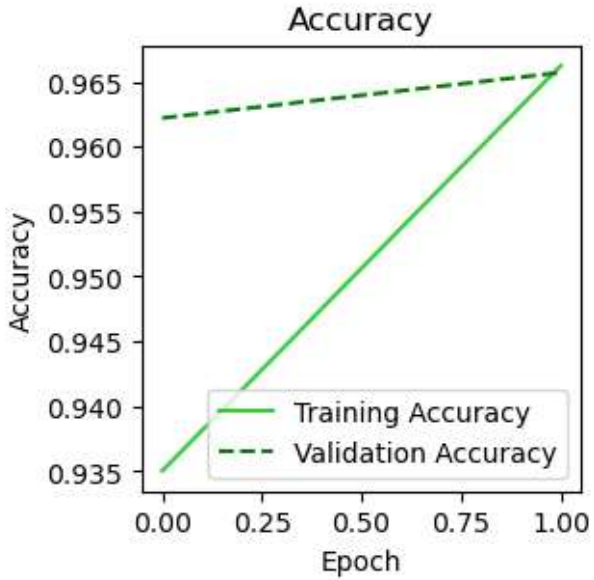
In [10]:
```
history = model.fit(x_train_vgg, y_train, epochs=2, batch_size=50, validation_data=(x_test_vgg, y_test))
```

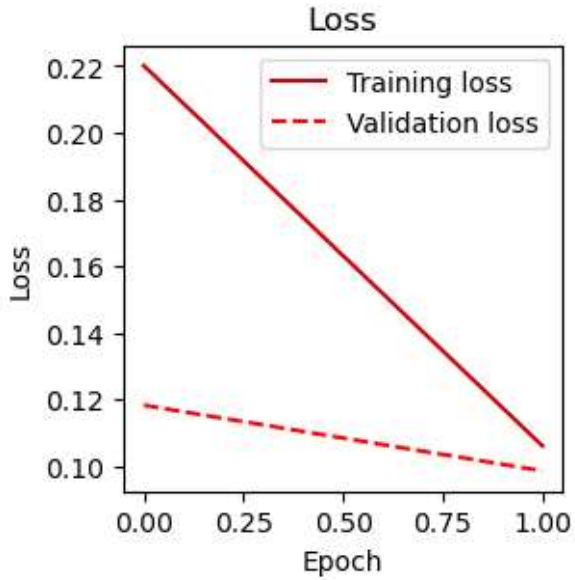
```
Epoch 1/2
WARNING:tensorflow:From D:\JUPYTER FOLDER\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.rag
ged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From D:\JUPYTER FOLDER\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The na
me tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_f
unctions instead.

1200/1200 [==============================] - 655s 543ms/step - loss: 0.2201 - accuracy: 0.9350 - val_loss:
0.1184 - val_accuracy: 0.9622
Epoch 2/2
1200/1200 [==============================] - 638s 532ms/step - loss: 0.1062 - accuracy: 0.9662 - val_loss:
0.0987 - val_accuracy: 0.9657
```

In [11]:
```
# Evaluate and print test accuracy
test_loss, test_accuracy = model.evaluate(x_test_vgg, y_test)
print("Test accuracy:", test_accuracy)
```

```
313/313 [==============================] - 92s 292ms/step - loss: 0.0987 - accuracy: 0.9657
Test accuracy: 0.9656999707221985
```

In [12]:
```
plt.figure(figsize=(10,3))
plt.subplot(1,3,1)
plt.title('Accuracy')
plt.plot(history.history['accuracy'],label='Training Accuracy',color='limegreen')
plt.plot(history.history['val_accuracy'],label='Validation Accuracy',color='green', linestyle='dashed')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,3,3)
plt.title('Loss')
plt.plot(history.history['loss'], label='Training loss', color='#cc0000',)
plt.plot(history.history['val_loss'], label='Validation loss',color = "#ff0000", linestyle = 'dashed')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```





In [13]:
```
# Load some test images
test_images = x_test_vgg[:10]

# Make predictions on the test images
predictions = model.predict(test_images)

#Plot the test images and their predictions
for i in range(len(test_images)):
    plt.subplot(2, 5, i + 1)
    plt.imshow(test_images[i], cmap='gray')
    plt.title(f"Predict: {np.argmax(predictions[i])}")
    plt.axis('off')
plt.show()
```

```
1/1 [==============================] - 0s 468ms/step

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integer
s).
```


Predict: 7
Predict: 2
Predict: 1
Predict: 0
Predict: 4


Predict: 1
Predict: 4
Predict: 9
Predict: 5
Predict: 9

In [ ]: