In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Input
from tensorflow.keras.utils import to_categorical

train_data = pd.read_csv('./Train.csv')
print("Shape of train_data:", train_data.shape)
```

```
Shape of train_data: (42000, 785)
```

In [2]:
```
X = train_data.iloc[:, 1:]
y = train_data.iloc[:, 0]

print("Shape of X after separating features:", X.shape)
```

```
Shape of X after separating features: (42000, 784)
```

In [3]:
```
if not isinstance(X, pd.DataFrame):
    X = pd.DataFrame(X)
X = X.apply(pd.to_numeric, errors='coerce')
X = X.fillna(0)
X = X.values / 255.0
X = X.reshape(-1, 28, 28, 1)
print("Shape of X after reshaping:", X.shape)
```

```
Shape of X after reshaping: (42000, 28, 28, 1)
```

In [4]:
```
y = to_categorical(y, num_classes=10)
print("Shape of y after one-hot encoding:", y.shape)
```

```
Shape of y after one-hot encoding: (42000, 10)
```

In [5]:
```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
print("X_train shape:", X_train.shape)
```

```
X_train shape: (33600, 28, 28, 1)
```

In [6]:
```
model = Sequential([
    Input(shape=(28, 28, 1)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 64)                8256

 dense_2 (Dense)             (None, 10)                650

=================================================================
Total params: 109,386
Trainable params: 109,386
Non-trainable params: 0
_________________________________________________________________
```

In [13]:
```
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_val, y_val))
```

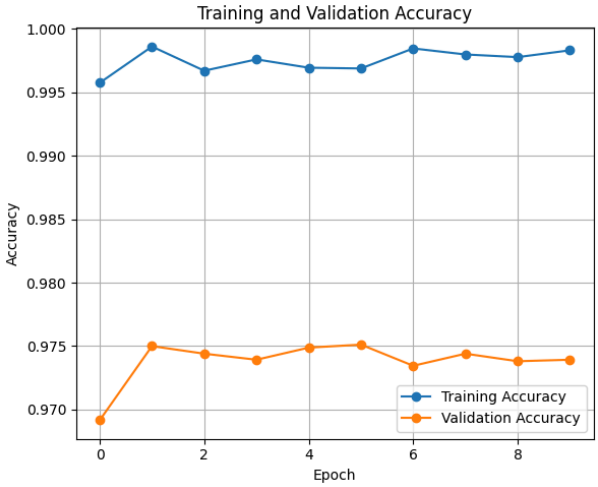
```
Epoch 1/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0126 - accuracy: 0.9957 - val_
loss: 0.1851 - val_accuracy: 0.9692
Epoch 2/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0050 - accuracy: 0.9986 - val_
loss: 0.1536 - val_accuracy: 0.9750
Epoch 3/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0100 - accuracy: 0.9967 - val_
loss: 0.1546 - val_accuracy: 0.9744
Epoch 4/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0068 - accuracy: 0.9976 - val_
loss: 0.1589 - val_accuracy: 0.9739
Epoch 5/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0103 - accuracy: 0.9969 - val_
loss: 0.1661 - val_accuracy: 0.9749
Epoch 6/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0099 - accuracy: 0.9969 - val_
loss: 0.1526 - val_accuracy: 0.9751
Epoch 7/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0049 - accuracy: 0.9985 - val_
loss: 0.1608 - val_accuracy: 0.9735
Epoch 8/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0071 - accuracy: 0.9980 - val_
loss: 0.1714 - val_accuracy: 0.9744
Epoch 9/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0071 - accuracy: 0.9978 - val_
loss: 0.1770 - val_accuracy: 0.9738
Epoch 10/10
1050/1050 [==============================] - 2s 2ms/step - loss: 0.0052 - accuracy: 0.9983 - val_
loss: 0.1855 - val_accuracy: 0.9739
```

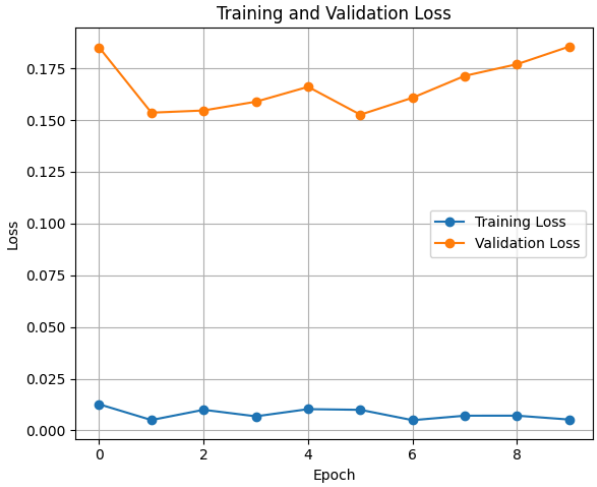
In [14]:

```python
val_loss, val_accuracy = model.evaluate(X_val, y_val)
print(f"Validation Accuracy: {val_accuracy * 100:.2f}%")
print(f"Validation loss: {val_loss * 100:.2f}%")

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.grid(True)
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss', marker='o')
plt.plot(history.history['val_loss'], label='Validation Loss', marker='o')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
263/263 [==============================] - 0s 1ms/step - loss: 0.1855 - accuracy: 0.9739
Validation Accuracy: 97.39%
Validation loss: 18.55%
```

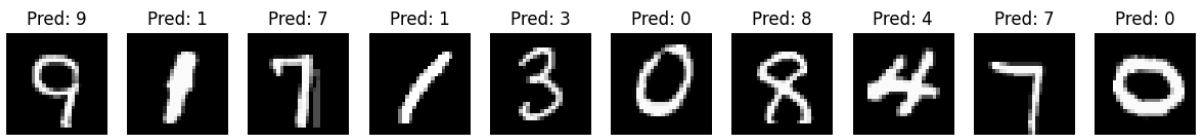




In [17]:
```
test_data = pd.read_csv('./test.csv')
test_images = test_data.values / 255.0
test_images = test_images.reshape(-1, 28, 28, 1)


predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)
num_samples = 10
plt.figure(figsize=(15, 4))
for i in range(num_samples):
    plt.subplot(1, num_samples, i + 1)
    plt.imshow(test_images[i].reshape(28, 28), cmap='gray')
    plt.title(f"Pred: {predicted_labels[i]}")
    plt.axis('off')
plt.suptitle("Sample Predictions on Test Images", fontsize=16)
plt.show()
```

```
175/175 [==============================] - 0s 913us/step
```



In [ ]: