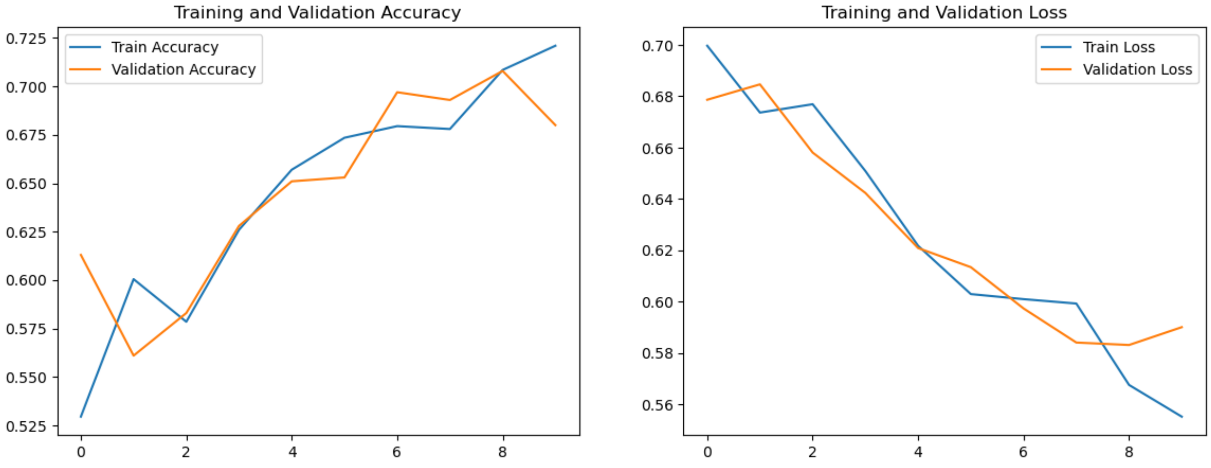
In [2]:

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

# Settings
IMG_SIZE = 160
BATCH_SIZE = 32
BASE_DIR = "C:/Users/shuj/Downloads/kagglecatsanddogs_5340/cats_and_dogs"

# Data generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Flow from directories
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Build CNN model from scratch
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3
    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(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

# Train
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate on test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy: {test_accuracy:.4f}")

# Plot curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, label='Train Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, label='Train Loss')
plt.plot(epochs, val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
63/63 [==============================] - 21s 332ms/step - loss: 0.6997 - accuracy: 0
.5295 - val_loss: 0.6787 - val_accuracy: 0.6130
Epoch 2/10
63/63 [==============================] - 15s 234ms/step - loss: 0.6737 - accuracy: 0
.6005 - val_loss: 0.6847 - val_accuracy: 0.5610
Epoch 3/10
63/63 [==============================] - 15s 238ms/step - loss: 0.6769 - accuracy: 0
.5785 - val_loss: 0.6581 - val_accuracy: 0.5830
Epoch 4/10
63/63 [==============================] - 16s 257ms/step - loss: 0.6509 - accuracy: 0
.6260 - val_loss: 0.6424 - val_accuracy: 0.6280
Epoch 5/10
63/63 [==============================] - 15s 240ms/step - loss: 0.6218 - accuracy: 0
.6570 - val_loss: 0.6209 - val_accuracy: 0.6510
Epoch 6/10
63/63 [==============================] - 16s 261ms/step - loss: 0.6030 - accuracy: 0
.6735 - val_loss: 0.6135 - val_accuracy: 0.6530
Epoch 7/10
63/63 [==============================] - 13s 193ms/step - loss: 0.6010 - accuracy: 0
.6795 - val_loss: 0.5974 - val_accuracy: 0.6970
Epoch 8/10
63/63 [==============================] - 12s 185ms/step - loss: 0.5993 - accuracy: 0
.6780 - val_loss: 0.5841 - val_accuracy: 0.6930
Epoch 9/10
63/63 [==============================] - 16s 258ms/step - loss: 0.5676 - accuracy: 0
.7085 - val_loss: 0.5832 - val_accuracy: 0.7080
Epoch 10/10
63/63 [==============================] - 15s 240ms/step - loss: 0.5553 - accuracy: 0
.7210 - val_loss: 0.5901 - val_accuracy: 0.6800
32/32 [==============================] - 4s 121ms/step - loss: 0.5948 - accuracy: 0.
6950

✅ Test Accuracy: 0.6950
```



In [3]:

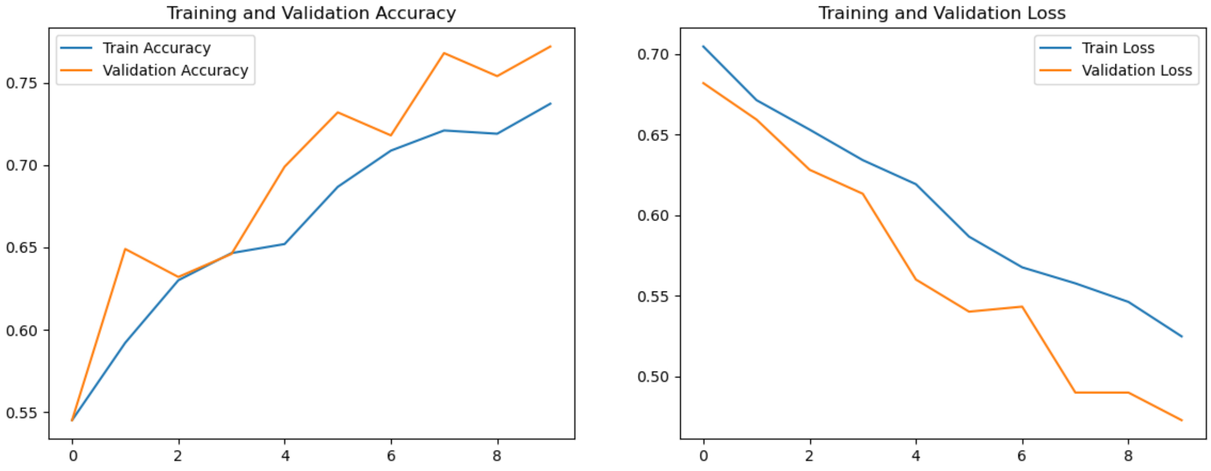
```
# Data generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Flow from directories
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Build CNN model from scratch
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3
    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(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

# Train
```

```python
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate on test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy: {test_accuracy:.4f}")

# Plot curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, label='Train Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, label='Train Loss')
plt.plot(epochs, val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

```
Found 4000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
125/125 [==============================] - 36s 283ms/step - loss: 0.7047 - accuracy:
0.5450 - val_loss: 0.6820 - val_accuracy: 0.5450
Epoch 2/10
125/125 [==============================] - 28s 222ms/step - loss: 0.6714 - accuracy:
0.5920 - val_loss: 0.6594 - val_accuracy: 0.6490
Epoch 3/10
125/125 [==============================] - 27s 216ms/step - loss: 0.6531 - accuracy:
0.6300 - val_loss: 0.6282 - val_accuracy: 0.6320
Epoch 4/10
125/125 [==============================] - 28s 221ms/step - loss: 0.6342 - accuracy:
0.6465 - val_loss: 0.6133 - val_accuracy: 0.6460
Epoch 5/10
125/125 [==============================] - 28s 221ms/step - loss: 0.6193 - accuracy:
0.6520 - val_loss: 0.5601 - val_accuracy: 0.6990
Epoch 6/10
125/125 [==============================] - 30s 239ms/step - loss: 0.5867 - accuracy:
0.6867 - val_loss: 0.5401 - val_accuracy: 0.7320
Epoch 7/10
125/125 [==============================] - 26s 203ms/step - loss: 0.5677 - accuracy:
0.7088 - val_loss: 0.5433 - val_accuracy: 0.7180
Epoch 8/10
125/125 [==============================] - 28s 225ms/step - loss: 0.5577 - accuracy:
0.7210 - val_loss: 0.4899 - val_accuracy: 0.7680
Epoch 9/10
125/125 [==============================] - 26s 206ms/step - loss: 0.5462 - accuracy:
0.7190 - val_loss: 0.4899 - val_accuracy: 0.7540
Epoch 10/10
125/125 [==============================] - 28s 221ms/step - loss: 0.5248 - accuracy:
0.7372 - val_loss: 0.4728 - val_accuracy: 0.7720
32/32 [==============================] - 4s 117ms/step - loss: 0.5061 - accuracy:
0.7600
```

```
C:\Users\shuj\anaconda3\envs\convnet_hw3\lib\site-packages\PIL\TiffImagePlugin.py:94
9: UserWarning: Truncated File Read
  warnings.warn(str(msg))
```

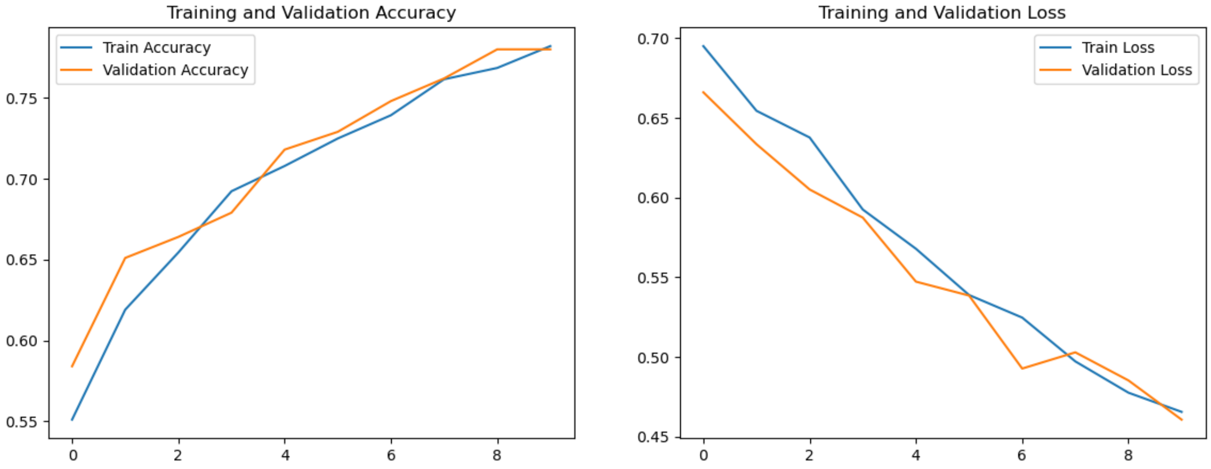
```
✅ Test Accuracy: 0.7600
```



```
In [4]: # Data generators
        train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Flow from directories
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Build CNN model from scratch
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3
    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(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

```python

# Train
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate on test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy: {test_accuracy:.4f}")

# Plot curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, label='Train Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, label='Train Loss')
plt.plot(epochs, val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

```
Found 8000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
250/250 [==============================] - 73s 290ms/step - loss: 0.6950 - accuracy:
0.5510 - val_loss: 0.6660 - val_accuracy: 0.5840
Epoch 2/10
250/250 [==============================] - 58s 230ms/step - loss: 0.6544 - accuracy:
0.6190 - val_loss: 0.6335 - val_accuracy: 0.6510
Epoch 3/10
250/250 [==============================] - 64s 257ms/step - loss: 0.6377 - accuracy:
0.6544 - val_loss: 0.6051 - val_accuracy: 0.6640
Epoch 4/10
250/250 [==============================] - 55s 217ms/step - loss: 0.5925 - accuracy:
0.6923 - val_loss: 0.5875 - val_accuracy: 0.6790
Epoch 5/10
250/250 [==============================] - 54s 215ms/step - loss: 0.5679 - accuracy:
0.7079 - val_loss: 0.5473 - val_accuracy: 0.7180
Epoch 6/10
250/250 [==============================] - 59s 238ms/step - loss: 0.5389 - accuracy:
0.7249 - val_loss: 0.5386 - val_accuracy: 0.7290
Epoch 7/10
250/250 [==============================] - 56s 223ms/step - loss: 0.5247 - accuracy:
0.7393 - val_loss: 0.4928 - val_accuracy: 0.7480
Epoch 8/10
250/250 [==============================] - 54s 217ms/step - loss: 0.4973 - accuracy:
0.7615 - val_loss: 0.5029 - val_accuracy: 0.7620
Epoch 9/10
250/250 [==============================] - 54s 214ms/step - loss: 0.4776 - accuracy:
0.7685 - val_loss: 0.4853 - val_accuracy: 0.7800
Epoch 10/10
250/250 [==============================] - 54s 215ms/step - loss: 0.4656 - accuracy:
0.7820 - val_loss: 0.4607 - val_accuracy: 0.7800
32/32 [==============================] - 4s 121ms/step - loss: 0.4222 - accuracy: 0.
8090

✅ Test Accuracy: 0.8090
```

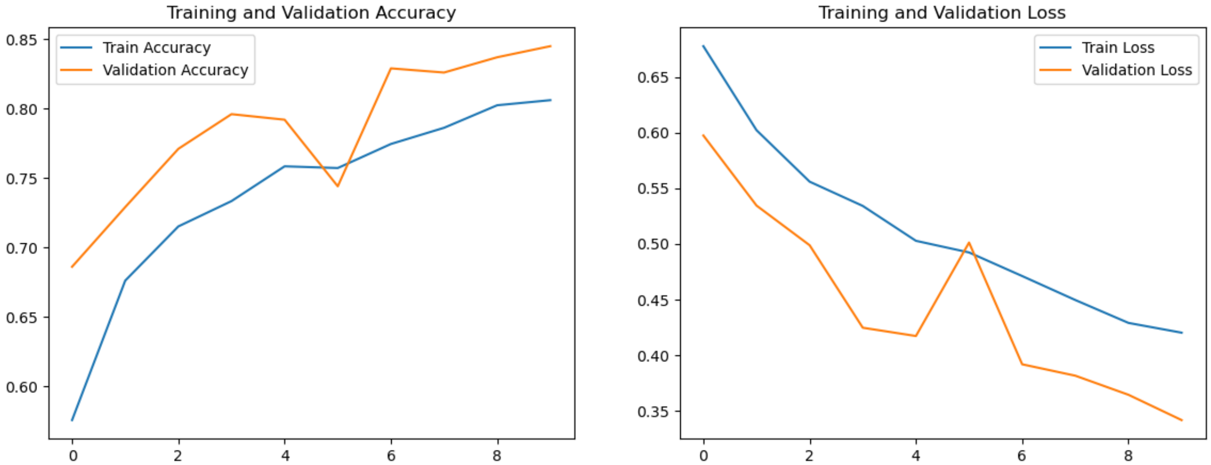


In [5]:

```
# Data generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Flow from directories
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Build CNN model from scratch
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(IMG_SIZE, IMG_SIZE, 3
    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(512, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')
])

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

# Train
```

```
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate on test set
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy: {test_accuracy:.4f}")

# Plot curves
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(acc))

plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(epochs, acc, label='Train Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(epochs, loss, label='Train Loss')
plt.plot(epochs, val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```

```
Found 12000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
375/375 [==============================] - 121s 322ms/step - loss: 0.6776 - accuracy
: 0.5755 - val_loss: 0.5974 - val_accuracy: 0.6860
Epoch 2/10
375/375 [==============================] - 99s 263ms/step - loss: 0.6022 - accuracy:
0.6760 - val_loss: 0.5345 - val_accuracy: 0.7290
Epoch 3/10
375/375 [==============================] - 94s 252ms/step - loss: 0.5559 - accuracy:
0.7151 - val_loss: 0.4989 - val_accuracy: 0.7710
Epoch 4/10
375/375 [==============================] - 110s 293ms/step - loss: 0.5342 - accuracy
: 0.7333 - val_loss: 0.4248 - val_accuracy: 0.7960
Epoch 5/10
375/375 [==============================] - 107s 285ms/step - loss: 0.5028 - accuracy
: 0.7584 - val_loss: 0.4174 - val_accuracy: 0.7920
Epoch 6/10
375/375 [==============================] - 105s 280ms/step - loss: 0.4924 - accuracy
: 0.7572 - val_loss: 0.5013 - val_accuracy: 0.7440
Epoch 7/10
375/375 [==============================] - 93s 247ms/step - loss: 0.4712 - accuracy:
0.7745 - val_loss: 0.3920 - val_accuracy: 0.8290
Epoch 8/10
375/375 [==============================] - 96s 255ms/step - loss: 0.4497 - accuracy:
0.7862 - val_loss: 0.3818 - val_accuracy: 0.8260
Epoch 9/10
375/375 [==============================] - 104s 275ms/step - loss: 0.4291 - accuracy
: 0.8024 - val_loss: 0.3646 - val_accuracy: 0.8370
Epoch 10/10
375/375 [==============================] - 93s 248ms/step - loss: 0.4204 - accuracy:
0.8061 - val_loss: 0.3419 - val_accuracy: 0.8450
32/32 [==============================] - 4s 118ms/step - loss: 0.4260 - accuracy: 0.
8190

✅ Test Accuracy: 0.8190
```



In [6]:

```python
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import matplotlib.pyplot as plt

IMG_SIZE = 160
BATCH_SIZE = 32
BASE_DIR = "C:/Users/shuj/Downloads/kagglecatsanddogs_5340/cats_and_dogs_step1"

# Data generators (use MobileNetV2's preprocess_input)
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

val_test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

# Use the 1000 training image folder again
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Load MobileNetV2 base model (pretrained on ImageNet)
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3),
                         include_top=False,
                         weights='imagenet')
base_model.trainable = False  # Freeze the base

# Build model on top
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

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

# Train
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy (Pretrained + 1000 train images): {test_accuracy:.4f}")
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/m
obilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_160_no_top.h5
9406464/9406464 [==============================] - 0s 0us/step
Epoch 1/10
63/63 [==============================] - 9s 127ms/step - loss: 0.2775 - accuracy: 0.
8840 - val_loss: 0.1277 - val_accuracy: 0.9590
Epoch 2/10
63/63 [==============================] - 7s 117ms/step - loss: 0.1198 - accuracy: 0.
9625 - val_loss: 0.0957 - val_accuracy: 0.9650
Epoch 3/10
63/63 [==============================] - 8s 119ms/step - loss: 0.1108 - accuracy: 0.
9555 - val_loss: 0.0828 - val_accuracy: 0.9710
Epoch 4/10
63/63 [==============================] - 8s 119ms/step - loss: 0.0985 - accuracy: 0.
9665 - val_loss: 0.0762 - val_accuracy: 0.9710
Epoch 5/10
63/63 [==============================] - 8s 119ms/step - loss: 0.0820 - accuracy: 0.
9700 - val_loss: 0.0721 - val_accuracy: 0.9730
Epoch 6/10
63/63 [==============================] - 8s 123ms/step - loss: 0.0765 - accuracy: 0.
9665 - val_loss: 0.0685 - val_accuracy: 0.9730
Epoch 7/10
63/63 [==============================] - 8s 123ms/step - loss: 0.0722 - accuracy: 0.
9720 - val_loss: 0.0672 - val_accuracy: 0.9710
Epoch 8/10
63/63 [==============================] - 8s 121ms/step - loss: 0.0713 - accuracy: 0.
9740 - val_loss: 0.0675 - val_accuracy: 0.9750
Epoch 9/10
63/63 [==============================] - 8s 122ms/step - loss: 0.0653 - accuracy: 0.
9755 - val_loss: 0.0638 - val_accuracy: 0.9750
Epoch 10/10
63/63 [==============================] - 8s 119ms/step - loss: 0.0669 - accuracy: 0.
9730 - val_loss: 0.0669 - val_accuracy: 0.9750
32/32 [==============================] - 1s 45ms/step - loss: 0.0733 - accuracy: 0.9
710

✅ Test Accuracy (Pretrained + 1000 train images): 0.9710
```

In [7]:

```python
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import matplotlib.pyplot as plt

IMG_SIZE = 160
BATCH_SIZE = 32
BASE_DIR = "C:/Users/shuj/Downloads/kagglecatsanddogs_5340/cats_and_dogs_step2"

# Data generators (use MobileNetV2's preprocess_input)
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

val_test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

# Use the 1000 training image folder again
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Load MobileNetV2 base model (pretrained on ImageNet)
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3),
                         include_top=False,
                         weights='imagenet')
base_model.trainable = False  # Freeze the base

# Build model on top
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

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

# Train
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy (Pretrained + 1000 train images): {test_accuracy:.4f}")
```

```
Found 4000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
125/125 [==============================] - 15s 112ms/step - loss: 0.2404 - accuracy:
0.9000 - val_loss: 0.0735 - val_accuracy: 0.9770
Epoch 2/10
125/125 [==============================] - 14s 108ms/step - loss: 0.1145 - accuracy:
0.9570 - val_loss: 0.0531 - val_accuracy: 0.9840
Epoch 3/10
125/125 [==============================] - 14s 109ms/step - loss: 0.0908 - accuracy:
0.9657 - val_loss: 0.0457 - val_accuracy: 0.9840
Epoch 4/10
125/125 [==============================] - 14s 113ms/step - loss: 0.0867 - accuracy:
0.9640 - val_loss: 0.0435 - val_accuracy: 0.9830
Epoch 5/10
125/125 [==============================] - 14s 111ms/step - loss: 0.0787 - accuracy:
0.9720 - val_loss: 0.0428 - val_accuracy: 0.9830
Epoch 6/10
125/125 [==============================] - 14s 110ms/step - loss: 0.0772 - accuracy:
0.9715 - val_loss: 0.0399 - val_accuracy: 0.9840
Epoch 7/10
125/125 [==============================] - 14s 111ms/step - loss: 0.0717 - accuracy:
0.9735 - val_loss: 0.0402 - val_accuracy: 0.9850
Epoch 8/10
125/125 [==============================] - 14s 111ms/step - loss: 0.0717 - accuracy:
0.9755 - val_loss: 0.0490 - val_accuracy: 0.9800
Epoch 9/10
125/125 [==============================] - 14s 108ms/step - loss: 0.0681 - accuracy:
0.9768 - val_loss: 0.0400 - val_accuracy: 0.9850
Epoch 10/10
125/125 [==============================] - 14s 112ms/step - loss: 0.0699 - accuracy:
0.9728 - val_loss: 0.0364 - val_accuracy: 0.9860
32/32 [==============================] - 1s 44ms/step - loss: 0.0634 - accuracy: 0.9
750

✅ Test Accuracy (Pretrained + 1000 train images): 0.9750
```

In [8]:

```
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import matplotlib.pyplot as plt

IMG_SIZE = 160
BATCH_SIZE = 32
BASE_DIR = "C:/Users/shuj/Downloads/kagglecatsanddogs_5340/cats_and_dogs_step2_4000
```

```
# Data generators (use MobileNetV2's preprocess_input)
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)

val_test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

# Use the 1000 training image folder again
train_generator = train_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'train'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

val_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'validation'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    os.path.join(BASE_DIR, 'test'),
    target_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

# Load MobileNetV2 base model (pretrained on ImageNet)
base_model = MobileNetV2(input_shape=(IMG_SIZE, IMG_SIZE, 3),
                         include_top=False,
                         weights='imagenet')
base_model.trainable = False  # Freeze the base

# Build model on top
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.3),
    layers.Dense(1, activation='sigmoid')
])

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

```python
# Train
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)

# Evaluate
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"\n✅ Test Accuracy (Pretrained + 1000 train images): {test_accuracy:.4f}")
```

```
Found 8000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
Epoch 1/10
250/250 [==============================] - 28s 105ms/step - loss: 0.1481 - accuracy:
0.9439 - val_loss: 0.0629 - val_accuracy: 0.9770
Epoch 2/10
250/250 [==============================] - 26s 105ms/step - loss: 0.0829 - accuracy:
0.9693 - val_loss: 0.0492 - val_accuracy: 0.9840
Epoch 3/10
250/250 [==============================] - 26s 102ms/step - loss: 0.0789 - accuracy:
0.9710 - val_loss: 0.0471 - val_accuracy: 0.9820
Epoch 4/10
250/250 [==============================] - 27s 106ms/step - loss: 0.0740 - accuracy:
0.9718 - val_loss: 0.0423 - val_accuracy: 0.9850
Epoch 5/10
250/250 [==============================] - 26s 102ms/step - loss: 0.0682 - accuracy:
0.9743 - val_loss: 0.0546 - val_accuracy: 0.9750
Epoch 6/10
250/250 [==============================] - 26s 104ms/step - loss: 0.0667 - accuracy:
0.9751 - val_loss: 0.0482 - val_accuracy: 0.9780
Epoch 7/10
250/250 [==============================] - 26s 102ms/step - loss: 0.0667 - accuracy:
0.9760 - val_loss: 0.0388 - val_accuracy: 0.9850
Epoch 8/10
250/250 [==============================] - 26s 104ms/step - loss: 0.0602 - accuracy:
0.9790 - val_loss: 0.0385 - val_accuracy: 0.9890
Epoch 9/10
250/250 [==============================] - 25s 101ms/step - loss: 0.0586 - accuracy:
0.9780 - val_loss: 0.0397 - val_accuracy: 0.9850
Epoch 10/10
250/250 [==============================] - 26s 104ms/step - loss: 0.0671 - accuracy:
0.9746 - val_loss: 0.0386 - val_accuracy: 0.9840
32/32 [==============================] - 1s 43ms/step - loss: 0.0337 - accuracy: 0.9
850

✅ Test Accuracy (Pretrained + 1000 train images): 0.9850
```