In [24]:
```
#cell 1
import os
import random
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
import time

# --- Configuration ---
CSV_PATH = '/cluster/home/miolate21/FER_biasmitigation1/results/inferred_com
IMAGE_SHAPE = (224, 224, 1)
INPUT_SHAPE_MODEL = (224, 224, 3)
BATCH_SIZE = 32
SEED = 42
AUTOTUNE = tf.data.AUTOTUNE
NUM_CLASSES = 7
INITIAL_EPOCHS = 5
FINE_TUNE_EPOCHS = 15 # ajustar si hay que
TOTAL_EPOCHS = INITIAL_EPOCHS + FINE_TUNE_EPOCHS

# Learning Rates
INITIAL_LR = 1e-4
FINE_TUNE_LR = 1e-5

# Model Saving
MODEL_SAVE_DIR = "/cluster/home/miolate21/FER_biasmitigation1/models"

MODEL_SAVE_PATH_FEATURE = os.path.join(MODEL_SAVE_DIR, "resnet50_fer_feature
MODEL_SAVE_PATH_FINETUNE = os.path.join(MODEL_SAVE_DIR, "resnet50_fer_finetu
# Ensure model directory exists
os.makedirs(MODEL_SAVE_DIR, exist_ok=True)

# --- Reproducibility ---
os.environ['PYTHONHASHSEED'] = str(SEED)
random.seed(SEED)
np.random.seed(SEED)
tf.random.set_seed(SEED)
# tf.keras.utils.set_random_seed(SEED) # Use this for newer TF versions if r

print(f"Using SEED: {SEED}")
print(f"Input Shape expected by NPY loader: {IMAGE_SHAPE}")
print(f"Model Input Shape (after processing): {INPUT_SHAPE_MODEL}")
print(f"Number of Classes: {NUM_CLASSES}")
print(f"Initial LR: {INITIAL_LR}, Fine-tune LR: {FINE_TUNE_LR}")
print(f"Initial Epochs: {INITIAL_EPOCHS}, Fine-tune Epochs: {FINE_TUNE_EPOCH
print(f"Model save path: {MODEL_SAVE_PATH}")
```

```
Using SEED: 42
Input Shape expected by NPY loader: (224, 224, 1)
Model Input Shape (after processing): (224, 224, 3)
Number of Classes: 7
Initial LR: 0.0001, Fine-tune LR: 1e-05
Initial Epochs: 5, Fine-tune Epochs: 15
Model save path: /cluster/home/miolate21/FER_biasmitigation1/models/resnet50
_baseline_224.keras
```

In [25]:
```python
#cell 2

print(f"Loading CSV from: '{CSV_PATH}'")
df = None
idx_to_emotion = None
CLASS_NAMES = None

try:
    if not os.path.exists(CSV_PATH):
        raise FileNotFoundError(f"CSV file not found at {CSV_PATH}")

    df = pd.read_csv(CSV_PATH)

    if 'Unnamed: 0' in df.columns:
        df = df.drop(columns=['Unnamed: 0'])

    print(f"Loaded {len(df)} records.")

    required_columns = ['image_path', 'emotion']
    if not all(col in df.columns for col in required_columns):
        raise ValueError(f"CSV must contain columns: {required_columns}")

    # Create label mappings
    emotion_labels = sorted(df['emotion'].unique())
    if len(emotion_labels) != NUM_CLASSES:
        print(f"Warning: Found {len(emotion_labels)} unique emotions, but ex


    emotion_to_idx = {label: idx for idx, label in enumerate(emotion_labels)
    idx_to_emotion = {idx: label for label, idx in emotion_to_idx.items()}
    CLASS_NAMES = list(idx_to_emotion.values())

    df['label'] = df['emotion'].map(emotion_to_idx)

    if df['label'].isnull().any():
        print("Warning: Some emotions could not be mapped to labels.")
        print(df[df['label'].isnull()]['emotion'].unique())

    print("\nLabel map created:")
    print(f"Emotion to Index: {emotion_to_idx}")
    print(f"Index to Emotion: {idx_to_emotion}")
    print(f"Class Names: {CLASS_NAMES}")
    print(f"\nDataFrame head:\n{df.head()}")

except FileNotFoundError as e:
    print(f"ERROR: {e}")
except ValueError as e:
```

```
    print(f"ERROR: {e}")
except Exception as e:
    print(f"An error occurred loading or processing the CSV: {e}")
    df = None # Ensure df is None on error

# Critical check before proceeding
if df is None or df.empty:
    print("\nCritical Error: DataFrame is empty or None. Cannot proceed.")
    # Stop execution or handle appropriately
    assert False, "DataFrame loading failed."
elif df['label'].isnull().any():
     print("\nCritical Error: DataFrame contains rows with null labels after
     assert False, "Label mapping failed for some rows."
```

```
Loading CSV from: '/cluster/home/miolate21/FER_biasmitigation1/results/infer
red_combined_224.csv'
Loaded 41476 records.

Label map created:
Emotion to Index: {'angry': 0, 'disgust': 1, 'fear': 2, 'happy': 3, 'neutra
l': 4, 'sad': 5, 'surprise': 6}
Index to Emotion: {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neut
ral', 5: 'sad', 6: 'surprise'}
Class Names: ['angry', 'disgust', 'fear', 'happy', 'neutral', 'sad', 'surpri
se']

DataFrame head:
                                          image_path  dataset  emotion  labe
l
0  /cluster/home/miolate21/FER_biasmitigation1/da...  FER2013  neutral
4
1  /cluster/home/miolate21/FER_biasmitigation1/da...  FER2013  neutral
4
2  /cluster/home/miolate21/FER_biasmitigation1/da...  FER2013  neutral
4
3  /cluster/home/miolate21/FER_biasmitigation1/da...  FER2013  neutral
4
4  /cluster/home/miolate21/FER_biasmitigation1/da...  FER2013  neutral
4
```

In [8]:
```
#cell 3
if df is not None and not df.empty:
    path_column_to_check = 'image_path'
    print(f"\nChecking for .npy files listed in '{path_column_to_check}'...")

    start_time = time.time()
    exists = df[path_column_to_check].apply(os.path.exists)
    end_time = time.time()

    num_missing = len(df) - exists.sum()
    if num_missing > 0:
        print(f"\nFound {num_missing} missing .npy files.")
        missing_files_df = df[~exists]
        print("Missing file examples (first 5):")
        print(missing_files_df[path_column_to_check].head().tolist())
```

```
    else:
        print("All listed .npy files are present.")
else:
    print("No data loaded skipping file check.")
```

```
Checking for .npy files listed in 'image_path'...
All listed .npy files are present.
```

In [11]:
```
#cell 4  Train/Validation Split

train_df = None
val_df = None

if df is not None and not df.empty:
    if 'label' not in df.columns or df['label'].isnull().any():
        print("ERROR: 'label' column is missing")
    else:
        print("\nSplitting dataset into training and validation sets...")

        try:
            train_df, val_df = train_test_split(
                df,
                test_size=0.2,          # 20% goes to validation
                stratify=df['label'],
                random_state=SEED
            )
            print(f"Training set size: {len(train_df)} samples")
            print(f"Validation set size: {len(val_df)} samples")

            print("\nTraining set label distribution (%):")
            print((train_df['label'].value_counts(normalize=True).sort_index

            print("\nValidation set label distribution (%):")
            print((val_df['label'].value_counts(normalize=True).sort_index()

        except Exception as e:
            print(f"Split failed: {e}")
            print("Check your label distribution.")
            train_df, val_df = pd.DataFrame(), pd.DataFrame()
else:
    print("DataFrame is empty or None. Skipping split step.")
```

```
Splitting dataset into training and validation sets...
Training set size: 33180 samples
Validation set size: 8296 samples

Training set label distribution (%):
label
0    11.42%
1     2.89%
2    10.61%
3    29.03%
4    18.67%
5    16.47%
6    10.91%
Name: proportion, dtype: object

Validation set label distribution (%):
label
0    11.42%
1     2.89%
2    10.61%
3    29.03%
4    18.67%
5    16.48%
6    10.91%
Name: proportion, dtype: object
```

In [26]:

```python
# Cell 5 Dataset Loader and Builder

# --- Data Augmentation Layer ---
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal", seed=SEED),
  layers.RandomRotation(0.1, seed=SEED),
  layers.RandomZoom(0.1, seed=SEED),
], name="data_augmentation")

def _load_and_process_npy(path_bytes, label_int):
    path_str = path_bytes.numpy().decode('utf-8')
    try:
        img = np.load(path_str)

        # --- Data Type and Range ---
        img = img.astype(np.float32)


        if img.max() <= 1.0: img = img * 255.0


        if img.ndim == 2:
             img = np.expand_dims(img, axis=-1) # Add channel: (H, W, 1)

        if img.shape != IMAGE_SHAPE:
             if img.shape == IMAGE_SHAPE[:-1]:
                 img = np.expand_dims(img, axis=-1)
             else:
                raise ValueError(f"Unexpected shape {img.shape} for image {p
```

```
        return img, np.int32(label_int)

    except FileNotFoundError:
        print(f"ERROR (_load_and_process_npy): File not found: {path_str}. F
        return np.zeros(IMAGE_SHAPE, dtype=np.float32), np.int32(label_int)
    except ValueError as ve: # Catch specific shape errors
         print(f"ERROR (_load_and_process_npy): Shape error for {path_str}:
         return np.zeros(IMAGE_SHAPE, dtype=np.float32), np.int32(label_int)
    except Exception as e:
        print(f"ERROR (_load_and_process_npy): Problem loading {path_str}: {
        return np.zeros(IMAGE_SHAPE, dtype=np.float32), np.int32(label_int)

@tf.function
def load_npy_tf_wrapper(path_tensor, label_tensor):
    image, label = tf.py_function(
        _load_and_process_npy,
        inp=[path_tensor, label_tensor],
        Tout=(tf.float32, tf.int32)
    )
    image.set_shape(IMAGE_SHAPE)
    label.set_shape(())
    return image, label

# --- Updated Dataset Builder ---
def build_tf_dataset(df_subset, shuffle=True, augment=False, batch_size=BATC
    if df_subset is None or df_subset.empty:
        print("DataFrame subset is empty, returning empty dataset.")
        return tf.data.Dataset.from_tensor_slices(
             (tf.constant([], dtype=tf.string), tf.constant([], dtype=tf.int
        ).map(load_npy_tf_wrapper).batch(batch_size).prefetch(buffer_size=AU

    paths = df_subset['image_path'].values
    labels = df_subset['label'].values.astype(np.int32)

    ds = tf.data.Dataset.from_tensor_slices((paths, labels))

    if shuffle:
        ds = ds.shuffle(buffer_size=len(df_subset), seed=SEED, reshuffle_eac

    ds = ds.map(load_npy_tf_wrapper, num_parallel_calls=AUTOTUNE)

    # --- Apply Augmentation ---
    if augment:
        ds = ds.map(lambda x, y: (data_augmentation(x, training=True), y),
                    num_parallel_calls=AUTOTUNE)

    # Batch dataset
    ds = ds.batch(batch_size, drop_remainder=False)

    # Prefetch for performance
    ds = ds.prefetch(buffer_size=AUTOTUNE)

    return ds

# --- Build the Datasets ---
print("Building training dataset...")
```

```
train_ds = build_tf_dataset(train_df, shuffle=True, augment=True, batch_size

print("\nBuilding validation dataset...")
val_ds = build_tf_dataset(val_df, shuffle=False, augment=False, batch_size=B

print("\nDatasets ready. Train/Val tf.data.Dataset objects created.")
```

```
Building training dataset...

Building validation dataset...

Datasets ready. Train/Val tf.data.Dataset objects created.
```

In [46]:
```
#cell 6 Model Building

def build_resnet50_transfer(input_shape_gray=IMAGE_SHAPE, num_classes=NUM_CL

    # --- Input Layer ---
    inputs = layers.Input(shape=input_shape_gray, name="input_layer")

    # --- Preprocessing ---
    x = layers.Concatenate(axis=-1, name="grayscale_to_rgb")([inputs, inputs

    x = tf.keras.applications.resnet50.preprocess_input(x)

    # --- Base ResNet50 Model ---
    base_model = ResNet50(
        include_top=False,
        weights='imagenet',
        input_shape=INPUT_SHAPE_MODEL
    )
    # freeze  base model
    base_model.trainable = False
    print(f"Base ResNet50 model loaded. Initial trainable status: {base_mode

    # --- Connect Preprocessing to Base Model ---
    x = base_model(x, training=False)

    # --- Classifier Head ---
    x_head = layers.GlobalAveragePooling2D(name="global_avg_pool")(x)
    x_head = layers.BatchNormalization()(x_head)
    x_head = layers.Dense(128, activation='relu', name="dense_head")(x_head)
    x_head = layers.Dropout(0.5, name="dropout_head")(x_head)
    outputs = layers.Dense(num_classes, activation='softmax', name="output_l

    # --- Create the final model ---
    model = models.Model(inputs=inputs, outputs=outputs, name="ResNet50_Trar

    return model

# --- Instantiate the model ---
print("Building ResNet50 transfer learning model...")
model = build_resnet50_transfer(input_shape_gray=IMAGE_SHAPE, num_classes=NU

print("\nInitial Model Summary (Base Frozen):")
model.summary(line_length=120)
```

```
# Optional: Verify trainable/non-trainable counts programmatically
total_params = model.count_params()
trainable_count = sum([tf.keras.backend.count_params(w) for w in model.train
non_trainable_count = sum([tf.keras.backend.count_params(w) for w in model.r
print(f"\nTotal parameters: {total_params:,}")
print(f"Trainable parameters (initially): {trainable_count:,}")
print(f"Non-trainable parameters (initially): {non_trainable_count:,}")
```

```
Building ResNet50 transfer learning model...
Base ResNet50 model loaded. Initial trainable status: False

Initial Model Summary (Base Frozen):
```

Model: "ResNet50_Transfer_FER"

Layer (type)	Output Shape	
input_layer (InputLayer)	(None, 224, 224, 1)	
grayscale_to_rgb (Concatenate)	(None, 224, 224, 3)	
get_item_30 (GetItem)	(None, 224, 224)	
get_item_31 (GetItem)	(None, 224, 224)	
get_item_32 (GetItem)	(None, 224, 224)	
stack_10 (Stack)	(None, 224, 224, 3)	
add_10 (Add)	(None, 224, 224, 3)	
resnet50 (Functional)	(None, 7, 7, 2048)	
global_avg_pool (GlobalAveragePooling2D)	(None, 2048)	
batch_normalization_9 (BatchNormalization)	(None, 2048)	
dense_head (Dense)	(None, 128)	
dropout_head (Dropout)	(None, 128)	
output_layer (Dense)	(None, 7)	

Total params: 23,859,079 (91.02 MB)

Trainable params: 267,271 (1.02 MB)

Non-trainable params: 23,591,808 (90.00 MB)

```
Total parameters: 23,859,079
Trainable parameters (initially): 267,271
Non-trainable parameters (initially): 23,591,808
```

In [47]:
```
#cell 7 initial compilation

print("Compiling the model for initial feature extraction...")
model.compile(
    optimizer=Adam(learning_rate=INITIAL_LR),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
print(f"Model compiled successfully with LR={INITIAL_LR}.")
```

```
Compiling the model for initial feature extraction...
Model compiled successfully with LR=0.0001.
```

In [39]:
```
#cell 8 Initial Feature Extraction Training

print(f"\n--- Starting Initial Training (Feature Extraction) for {INITIAL_EP
print(f"Saving best model checkpoints during this phase to: {MODEL_SAVE_PATH

callbacks_initial = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath=MODEL_SAVE_PATH,
        save_best_only=True,
        monitor='val_accuracy',
        mode='max',
        verbose=1
    ),
    tf.keras.callbacks.EarlyStopping(
        monitor='val_accuracy',
        patience=3,
        verbose=1,
        restore_best_weights=True
    )
]

if 'train_ds' not in globals() or 'val_ds' not in globals() or train_ds is N
     raise NameError("ERROR: 'train_ds' or 'val_ds' not defined or is None.

if tf.data.experimental.cardinality(train_ds).numpy() == 0 or \
   tf.data.experimental.cardinality(val_ds).numpy() == 0:
    raise ValueError("ERROR: Training or validation dataset is empty.")


history_initial = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=INITIAL_EPOCHS,
    callbacks=callbacks_initial,
    verbose=1
)

print("\n--- Initial model training (feature extraction phase) finished. ---

# Keep track of the best validation accuracy achieved in this phase
initial_best_val_acc = max(history_initial.history['val_accuracy'])
print(f"Best validation accuracy during initial phase: {initial_best_val_acc
```

```
--- Starting Initial Training (Feature Extraction) for 5 epochs ---
Saving best model checkpoints during this phase to: /cluster/home/miolate21/
FER_biasmitigation1/models/resnet50_baseline_224.keras
Epoch 1/5
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.2662 - loss: 2.232
8
Epoch 1: val_accuracy improved from -inf to 0.39947, saving model to /cluste
r/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 89s 81ms/step - accuracy: 0.2663 - loss: 2.23
26 - val_accuracy: 0.3995 - val_loss: 1.6045
Epoch 2/5
1035/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.3640 - loss: 1.747
7
Epoch 2: val_accuracy improved from 0.39947 to 0.43599, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 75ms/step - accuracy: 0.3640 - loss: 1.74
76 - val_accuracy: 0.4360 - val_loss: 1.5334
Epoch 3/5
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.3928 - loss: 1.610
7
Epoch 3: val_accuracy improved from 0.43599 to 0.46601, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 77s 75ms/step - accuracy: 0.3928 - loss: 1.61
07 - val_accuracy: 0.4660 - val_loss: 1.4338
Epoch 4/5
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.4226 - loss: 1.538
6
Epoch 4: val_accuracy did not improve from 0.46601
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 76s 73ms/step - accuracy: 0.4226 - loss: 1.53
86 - val_accuracy: 0.4594 - val_loss: 1.4307
Epoch 5/5
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - accuracy: 0.4320 - loss: 1.493
9
Epoch 5: val_accuracy improved from 0.46601 to 0.47191, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 77s 74ms/step - accuracy: 0.4320 - loss: 1.49
39 - val_accuracy: 0.4719 - val_loss: 1.3871
Restoring model weights from the end of the best epoch: 5.

--- Initial model training (feature extraction phase) finished. ---
Best validation accuracy during initial phase: 0.4719
```

In [48]:

```python
# Cell 9 Fine-tuning Setup

print("\n--- Preparing for Fine-Tuning ---")

# --- Unfreeze Layers ---
try:
    base_model = model.get_layer('resnet50')
except ValueError:
    print("ERROR: Could not find layer named 'resnet50'.")
    possible_names = [l.name for l in model.layers if 'resnet50' in l.name.l
    if not possible_names:
        raise ValueError("Could not find the ResNet50 base layer in the mode
    base_model_name = possible_names[0]
    print(f"Found base layer with name: {base_model_name}")
```

```
    base_model = model.get_layer(base_model_name)


base_model.trainable = True
print(f"Base ResNet50 model trainable status set to: {base_model.trainable}"

fine_tune_at_layer_name = 'conv5_block1_out'

try:
    fine_tune_from_index = [i for i, layer in enumerate(base_model.layers) i
    print(f"Found fine-tune layer '{fine_tune_at_layer_name}' at index {fine

    # Freeze all layers before the fine_tune_from_index
    for layer in base_model.layers[:fine_tune_from_index]:
        layer.trainable = False
    print(f"Froze layers in base model up to index {fine_tune_from_index}.")

except IndexError:
    print(f"Warning: Could not find layer named '{fine_tune_at_layer_name}'
    print("Available layer names in base_model:", [layer.name for layer in b
    print("Proceeding with the entire base model potentially trainable .")

# --- Re-compile the Model ---
print(f"\nRe-compiling model for fine-tuning with LR={FINE_TUNE_LR}...")
model.compile(
    optimizer=Adam(learning_rate=FINE_TUNE_LR),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
print("Model re-compiled successfully.")

# --- Verify Trainable Parameters ---
print("\nModel Summary (During Fine-Tuning):")
model.summary(line_length=120)

trainable_count_ft = sum([tf.keras.backend.count_params(w) for w in model.tr
non_trainable_count_ft = sum([tf.keras.backend.count_params(w) for w in mode
print(f"\nTotal parameters: {model.count_params():,}")
print(f"Trainable parameters (fine-tuning): {trainable_count_ft:,}")
print(f"Non-trainable parameters (fine-tuning): {non_trainable_count_ft:,}")

# Check difference
print(f"Number of parameters unfrozen in base model: {trainable_count_ft - s
```

```
--- Preparing for Fine-Tuning ---
Base ResNet50 model trainable status set to: True
Found fine-tune layer 'conv5_block1_out' at index 154 in the base model.
Froze layers in base model up to index 154.

Re-compiling model for fine-tuning with LR=1e-05...
Model re-compiled successfully.

Model Summary (During Fine-Tuning):
Model: "ResNet50_Transfer_FER"
```

Layer (type)	Output Shape	
input_layer (InputLayer)	(None, 224, 224, 1)	
grayscale_to_rgb (Concatenate)	(None, 224, 224, 3)	
get_item_30 (GetItem)	(None, 224, 224)	
get_item_31 (GetItem)	(None, 224, 224)	
get_item_32 (GetItem)	(None, 224, 224)	
stack_10 (Stack)	(None, 224, 224, 3)	
add_10 (Add)	(None, 224, 224, 3)	
resnet50 (Functional)	(None, 7, 7, 2048)	
global_avg_pool (GlobalAveragePooling2D)	(None, 2048)	
batch_normalization_9 (BatchNormalization)	(None, 2048)	
dense_head (Dense)	(None, 128)	
dropout_head (Dropout)	(None, 128)	
output_layer (Dense)	(None, 7)	

Total params: 23,859,079 (91.02 MB)

Trainable params: 9,198,599 (35.09 MB)

Non-trainable params: 14,660,480 (55.93 MB)

```
Total parameters: 23,859,079
Trainable parameters (fine-tuning): 9,198,599
Non-trainable parameters (fine-tuning): 14,660,480
Number of parameters unfrozen in base model: 9,194,503
```

In [49]:
```
#Cell 10 Fine-Tuning Training

print(f"\n--- Starting Fine-Tuning Training for up to {FINE_TUNE_EPOCHS} epc
print(f"Continuing training from epoch {INITIAL_EPOCHS}.")
print(f"Saving best model checkpoints during this phase to: {MODEL_SAVE_PATH

callbacks_finetune = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath=MODEL_SAVE_PATH,
        save_best_only=True,
        monitor='val_accuracy',
        mode='max',
        verbose=1
    ),
```

```
    tf.keras.callbacks.EarlyStopping(
        monitor='val_accuracy',
        patience=5,
        verbose=1,
        restore_best_weights=True
    )

]

if 'train_ds' not in globals() or 'val_ds' not in globals() or train_ds is N
     raise NameError("ERROR: 'train_ds' or 'val_ds' not defined or is None f
if tf.data.experimental.cardinality(train_ds).numpy() == 0 or \
   tf.data.experimental.cardinality(val_ds).numpy() == 0:
    raise ValueError("ERROR: Training or validation dataset is empty for fir


TOTAL_EPOCHS = INITIAL_EPOCHS + FINE_TUNE_EPOCHS
print(f"Training will run until epoch {TOTAL_EPOCHS} unless stopped early.")

history_fine_tune = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=TOTAL_EPOCHS,
    initial_epoch=INITIAL_EPOCHS,
    callbacks=callbacks_finetune,
    verbose=1
)

print("\n--- Fine-tuning training phase finished. ---")

if history_fine_tune.history.get('val_accuracy'):
    final_best_val_acc = max(history_fine_tune.history['val_accuracy'])
    print(f"Best validation accuracy during fine-tuning phase: {final_best_v
else:
    print("Fine-tuning phase completed, but no validation accuracy history r

print(f"Model training complete. Final best model saved to {MODEL_SAVE_PATH}
```

```
--- Starting Fine-Tuning Training for up to 15 epochs ---
Continuing training from epoch 5.
Saving best model checkpoints during this phase to: /cluster/home/miolate21/
FER_biasmitigation1/models/resnet50_baseline_224.keras
Training will run until epoch 20 unless stopped early.
Epoch 6/20
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 67ms/step - accuracy: 0.1781 - loss: 2.722
2
Epoch 6: val_accuracy improved from -inf to 0.40248, saving model to /cluste
r/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 99s 87ms/step - accuracy: 0.1781 - loss: 2.72
19 - val_accuracy: 0.4025 - val_loss: 1.7137
Epoch 7/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.3317 - loss: 1.981
4
Epoch 7: val_accuracy improved from 0.40248 to 0.46348, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.3317 - loss: 1.98
12 - val_accuracy: 0.4635 - val_loss: 1.4855
Epoch 8/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.3908 - loss: 1.720
5
Epoch 8: val_accuracy improved from 0.46348 to 0.48855, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 82s 79ms/step - accuracy: 0.3908 - loss: 1.72
04 - val_accuracy: 0.4885 - val_loss: 1.3934
Epoch 9/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.4244 - loss: 1.599
7
Epoch 9: val_accuracy improved from 0.48855 to 0.49192, saving model to /clu
ster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 82s 79ms/step - accuracy: 0.4244 - loss: 1.59
96 - val_accuracy: 0.4919 - val_loss: 1.3682
Epoch 10/20
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.4556 - loss: 1.498
3
Epoch 10: val_accuracy improved from 0.49192 to 0.51242, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.4557 - loss: 1.49
83 - val_accuracy: 0.5124 - val_loss: 1.3167
Epoch 11/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.4824 - loss: 1.413
4
Epoch 11: val_accuracy did not improve from 0.51242
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 80s 77ms/step - accuracy: 0.4824 - loss: 1.41
34 - val_accuracy: 0.4981 - val_loss: 1.3316
Epoch 12/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5006 - loss: 1.364
8
Epoch 12: val_accuracy improved from 0.51242 to 0.52290, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.5006 - loss: 1.36
48 - val_accuracy: 0.5229 - val_loss: 1.2773
Epoch 13/20
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.5185 - loss: 1.323
2
```

```
Epoch 13: val_accuracy improved from 0.52290 to 0.53701, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 82s 79ms/step - accuracy: 0.5185 - loss: 1.32
32 - val_accuracy: 0.5370 - val_loss: 1.2337
Epoch 14/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.5299 - loss: 1.280
6
Epoch 14: val_accuracy improved from 0.53701 to 0.54508, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 82s 79ms/step - accuracy: 0.5299 - loss: 1.28
06 - val_accuracy: 0.5451 - val_loss: 1.2139
Epoch 15/20
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5333 - loss: 1.262
3
Epoch 15: val_accuracy did not improve from 0.54508
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 79s 76ms/step - accuracy: 0.5333 - loss: 1.26
23 - val_accuracy: 0.5371 - val_loss: 1.2246
Epoch 16/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - accuracy: 0.5453 - loss: 1.224
1
Epoch 16: val_accuracy did not improve from 0.54508
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 80s 77ms/step - accuracy: 0.5453 - loss: 1.22
41 - val_accuracy: 0.5421 - val_loss: 1.2085
Epoch 17/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5606 - loss: 1.184
3
Epoch 17: val_accuracy improved from 0.54508 to 0.54713, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.5606 - loss: 1.18
43 - val_accuracy: 0.5471 - val_loss: 1.2082
Epoch 18/20
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5597 - loss: 1.190
8
Epoch 18: val_accuracy improved from 0.54713 to 0.56714, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.5597 - loss: 1.19
08 - val_accuracy: 0.5671 - val_loss: 1.1594
Epoch 19/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5793 - loss: 1.149
7
Epoch 19: val_accuracy improved from 0.56714 to 0.57027, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.5793 - loss: 1.14
97 - val_accuracy: 0.5703 - val_loss: 1.1583
Epoch 20/20
1036/1037 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - accuracy: 0.5861 - loss: 1.116
8
Epoch 20: val_accuracy improved from 0.57027 to 0.57317, saving model to /cl
uster/home/miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 81s 78ms/step - accuracy: 0.5860 - loss: 1.11
68 - val_accuracy: 0.5732 - val_loss: 1.1464
Restoring model weights from the end of the best epoch: 20.

--- Fine-tuning training phase finished. ---
Best validation accuracy during fine-tuning phase: 0.5732
Model training complete. Final best model saved to /cluster/home/miolate21/F
```

ER_biasmitigation1/models/resnet50_baseline_224.keras (based on validation a
ccuracy).

In [53]:
```
# Cell 10b Continue Fine-Tuning Training

# --- Configuration for Continued Training ---
CONTINUE_FROM_EPOCH = 20
ADDITIONAL_EPOCHS = 10
NEW_TOTAL_EPOCHS = CONTINUE_FROM_EPOCH + ADDITIONAL_EPOCHS

MODEL_SAVE_PATH = "/cluster/home/miolate21/FER_biasmitigation1/models/resnet
FINE_TUNE_LR = 1e-5

print(f"\n--- Continuing Fine-Tuning Training for {ADDITIONAL_EPOCHS} more e
print(f"Starting from epoch {CONTINUE_FROM_EPOCH + 1} (running up to epoch {
print(f"Saving best model checkpoints during this phase to: {MODEL_SAVE_PATH

print("Defining fresh callbacks for continued training.")
callbacks_continue_finetune = [
    tf.keras.callbacks.ModelCheckpoint(
        filepath=MODEL_SAVE_PATH,
        save_best_only=True,
        monitor='val_accuracy',
        mode='max',
        verbose=1
    ),
    tf.keras.callbacks.EarlyStopping(
        monitor='val_accuracy',
        patience=5,
        verbose=1,
        restore_best_weights=True
    )
]

if 'model' not in globals():
     print("WARN: 'model' object not found. Attempting to load from MODEL_SA
     try:
         model = tf.keras.models.load_model(MODEL_SAVE_PATH)
         print("Model loaded successfully.")
     except Exception as e:
         raise NameError(f"ERROR: 'model' object not found and failed to loa
if 'train_ds' not in globals() or 'val_ds' not in globals() or train_ds is N
      raise NameError("ERROR: 'train_ds' or 'val_ds' not defined or is None.
if tf.data.experimental.cardinality(train_ds).numpy() == 0 or \
    tf.data.experimental.cardinality(val_ds).numpy() == 0:
     raise ValueError("ERROR: Training or validation dataset is empty.")
print(f"Model is currently compiled with optimizer: {model.optimizer.name},
if abs(model.optimizer.learning_rate.numpy() - FINE_TUNE_LR) > 1e-9:
    print(f"WARN: Model LR ({model.optimizer.learning_rate.numpy():.1E}) dif


# --- Continue Training ---
print(f"\nCalling model.fit from epoch {CONTINUE_FROM_EPOCH} up to {NEW_TOTA
history_continue_fine_tune = model.fit(
    train_ds,
    validation_data=val_ds,
```

```
    epochs=NEW_TOTAL_EPOCHS,
    initial_epoch=CONTINUE_FROM_EPOCH,
    verbose=1
)

print(f"Model training complete (extended). Final best model saved to {MODEL
```

```
--- Continuing Fine-Tuning Training for 10 more epochs ---
Starting from epoch 21 (running up to epoch 30).
Saving best model checkpoints during this phase to: /cluster/home/miolate21/
FER_biasmitigation1/models/resnet50_baseline_224.keras
Defining fresh callbacks for continued training.
Model is currently compiled with optimizer: adam, LR: 1.0E-05

Calling model.fit from epoch 20 up to 30...
Epoch 21/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 75ms/step - accuracy: 0.5921 - loss: 1.11
46 - val_accuracy: 0.5769 - val_loss: 1.1337
Epoch 22/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 76ms/step - accuracy: 0.5990 - loss: 1.09
12 - val_accuracy: 0.5800 - val_loss: 1.1211
Epoch 23/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 79s 76ms/step - accuracy: 0.6061 - loss: 1.07
71 - val_accuracy: 0.5851 - val_loss: 1.1217
Epoch 24/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 76ms/step - accuracy: 0.6092 - loss: 1.05
56 - val_accuracy: 0.5875 - val_loss: 1.1233
Epoch 25/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 79s 76ms/step - accuracy: 0.6154 - loss: 1.04
38 - val_accuracy: 0.5881 - val_loss: 1.1082
Epoch 26/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 76ms/step - accuracy: 0.6235 - loss: 1.02
05 - val_accuracy: 0.5947 - val_loss: 1.1026
Epoch 27/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 79s 76ms/step - accuracy: 0.6272 - loss: 1.00
30 - val_accuracy: 0.5963 - val_loss: 1.0941
Epoch 28/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 75ms/step - accuracy: 0.6336 - loss: 1.00
11 - val_accuracy: 0.5990 - val_loss: 1.0861
Epoch 29/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 78s 75ms/step - accuracy: 0.6410 - loss: 0.97
78 - val_accuracy: 0.5956 - val_loss: 1.0900
Epoch 30/30
1037/1037 ━━━━━━━━━━━━━━━━━━━━ 79s 76ms/step - accuracy: 0.6441 - loss: 0.96
39 - val_accuracy: 0.5956 - val_loss: 1.0972
Model training complete (extended). Final best model saved to /cluster/home/
miolate21/FER_biasmitigation1/models/resnet50_baseline_224.keras.
```

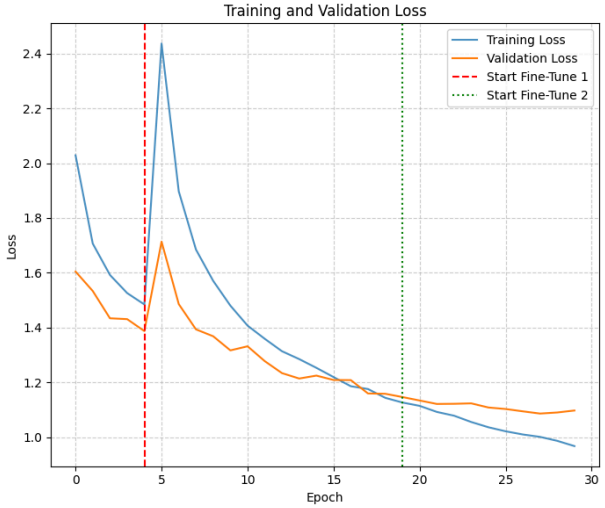
In [56]:
```
#Cell 11 Plotting Combined Training History

import matplotlib.pyplot as plt

print("\n--- Plotting Combined Training History ---")

acc = []
val_acc = []
```

```
loss = []
val_loss = []
total_epochs_run = 0

if 'history_initial' in globals() and hasattr(history_initial, 'history'):
    acc += history_initial.history.get('accuracy', [])
    val_acc += history_initial.history.get('val_accuracy', [])
    loss += history_initial.history.get('loss', [])
    val_loss += history_initial.history.get('val_loss', [])
    total_epochs_run += len(history_initial.history.get('accuracy', []))
    print(f"Initial history loaded: {len(history_initial.history.get('accura

if 'history_fine_tune' in globals() and hasattr(history_fine_tune, 'history'
    acc += history_fine_tune.history.get('accuracy', [])
    val_acc += history_fine_tune.history.get('val_accuracy', [])
    loss += history_fine_tune.history.get('loss', [])
    val_loss += history_fine_tune.history.get('val_loss', [])
    total_epochs_run += len(history_fine_tune.history.get('accuracy', []))
    print(f"First fine-tune history loaded: {len(history_fine_tune.history.g


if 'history_continue_fine_tune' in globals() and hasattr(history_continue_fi
    acc += history_continue_fine_tune.history.get('accuracy', [])
    val_acc += history_continue_fine_tune.history.get('val_accuracy', [])
    loss += history_continue_fine_tune.history.get('loss', [])
    val_loss += history_continue_fine_tune.history.get('val_loss', [])
    total_epochs_run += len(history_continue_fine_tune.history.get('accuracy
    print(f"Continued fine-tune history loaded: {len(history_continue_fine_t

if not acc:
    print("ERROR: No training history data found. Cannot plot.")
else:
    epochs_range = range(total_epochs_run)

    plt.figure(figsize=(14, 6))

    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy', alpha=0.8)
    plt.plot(epochs_range, val_acc, label='Validation Accuracy', linewidth=1
    initial_epochs_count = len(history_initial.history.get('accuracy', []))
    first_finetune_epochs_count = len(history_fine_tune.history.get('accurac

    if initial_epochs_count > 0:
        plt.axvline(initial_epochs_count - 1 , linestyle='--', color='r', la
    if first_finetune_epochs_count > 0 and initial_epochs_count > 0:
         plt.axvline(initial_epochs_count + first_finetune_epochs_count - 1

    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.grid(True, linestyle='--', alpha=0.6)


    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss', alpha=0.8)
```

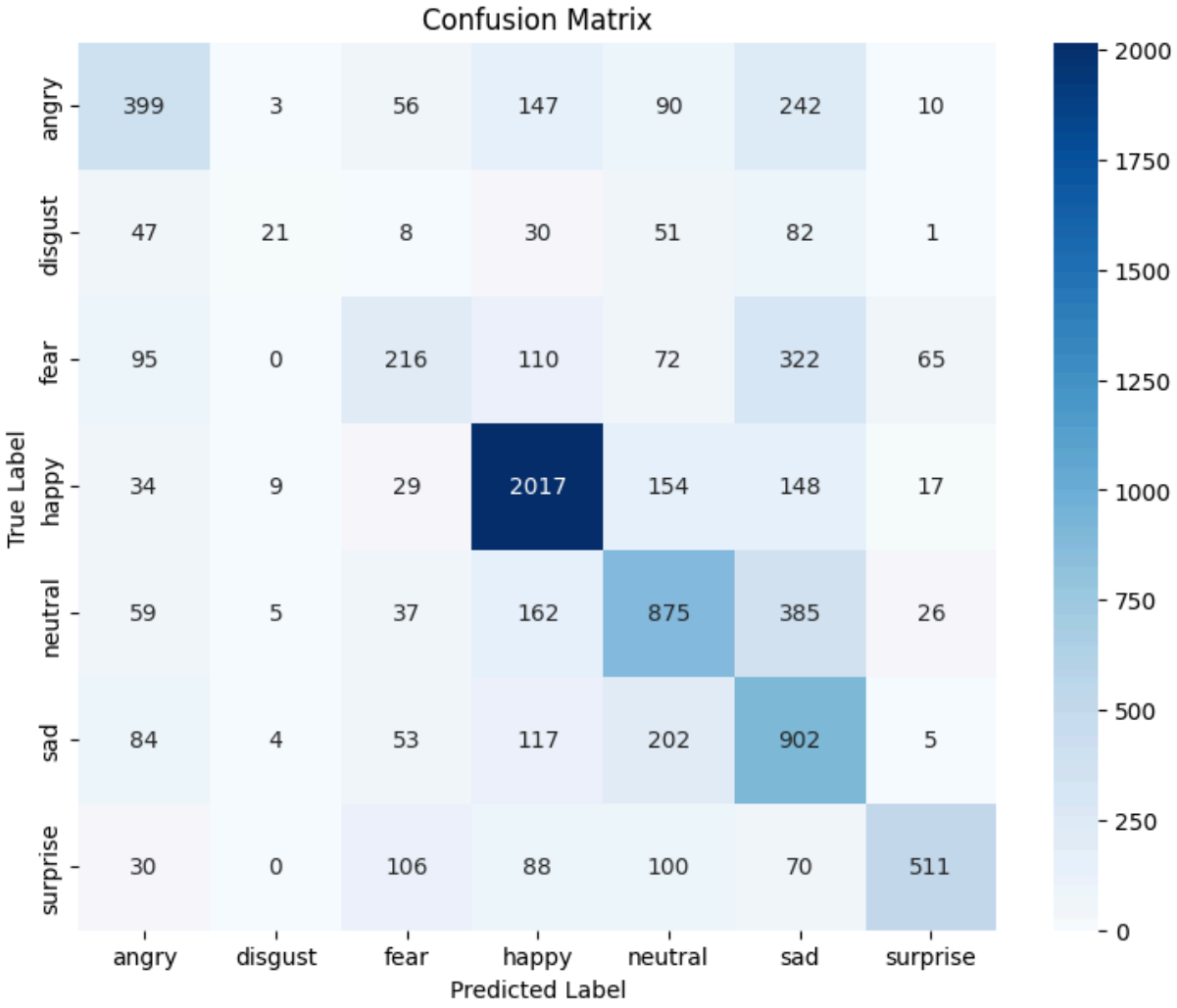
```
    plt.plot(epochs_range, val_loss, label='Validation Loss', linewidth=1.5)
    if initial_epochs_count > 0:
        plt.axvline(initial_epochs_count - 1, linestyle='--', color='r', lab
    if first_finetune_epochs_count > 0 and initial_epochs_count > 0:
         plt.axvline(initial_epochs_count + first_finetune_epochs_count - 1,

    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.grid(True, linestyle='--', alpha=0.6)

    plt.tight_layout()

    directory = '/cluster/home/miolate21/FER_biasmitigation1/results'
    filename = 'resnet50_training_metrics_e30.png'
    filepath = os.path.join(directory, filename)

    plt.savefig(filepath)

    plt.show()

    print("\n--- Evaluating Final Model ---")

    if 'val_ds' in globals() and val_ds is not None:
        print("Evaluating model on validation set...")
        eval_loss, eval_acc = model.evaluate(val_ds, verbose=1)
        print(f"\nFinal evaluation on validation set:")
        print(f"Validation Loss: {eval_loss:.4f}")
        print(f"Validation Accuracy: {eval_acc:.4f}")


        print("\nGenerating classification report and confusion matrix...")
        import numpy as np
        from sklearn.metrics import classification_report, confusion_matrix
        import seaborn as sns

        y_pred_probs = model.predict(val_ds)
        y_pred = np.argmax(y_pred_probs, axis=1)
        y_true = np.concatenate([y for x, y in val_ds], axis=0)


        if 'CLASS_NAMES' not in globals() or CLASS_NAMES is None:
            print("Warning: CLASS_NAMES not found, using default labels.")
            target_names = [str(i) for i in range(NUM_CLASSES)]
        else:
             target_names = CLASS_NAMES

        print("\nClassification Report:")
        print(classification_report(y_true, y_pred, target_names=target_name

        cm = confusion_matrix(y_true, y_pred)
        plt.figure(figsize=(9, 7))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=targe
        plt.xlabel('Predicted Label')
        plt.ylabel('True Label')
```

```
        plt.title('Confusion Matrix')


        directory = '/cluster/home/miolate21/FER_biasmitigation1/results'
        filename = 'resnet50_confusion_matrix.png'
        filepath = os.path.join(directory, filename)

        plt.savefig(filepath)

        plt.show()
    else:
        print("Validation dataset ('val_ds') not found. Skipping final evalu
```

```
--- Plotting Combined Training History ---
Initial history loaded: 5 epochs.
First fine-tune history loaded: 15 epochs.
Continued fine-tune history loaded: 10 epochs.
```



```
--- Evaluating Final Model ---
Evaluating model on validation set...
260/260 ━━━━━━━━━━━━━━━━━━━━ 14s 54ms/step - accuracy: 0.6007 - loss: 1.0730

Final evaluation on validation set:
Validation Loss: 1.0972
Validation Accuracy: 0.5956

Generating classification report and confusion matrix...
260/260 ━━━━━━━━━━━━━━━━━━━━ 15s 56ms/step

Classification Report:
              precision    recall  f1-score   support

       angry       0.53      0.42      0.47       947
     disgust       0.50      0.09      0.15       240
        fear       0.43      0.25      0.31       880
       happy       0.76      0.84      0.79      2408
     neutral       0.57      0.56      0.57      1549
         sad       0.42      0.66      0.51      1367
    surprise       0.80      0.56      0.66       905

    accuracy                           0.60      8296
   macro avg       0.57      0.48      0.50      8296
weighted avg       0.60      0.60      0.58      8296
```



In [57]:
```
from sklearn.metrics import classification_report
import os

# Generate classification report
report_text = classification_report(y_true, y_pred, target_names=CLASS_NAMES

# === Dynamic file naming ===
model_version = "resnet50_tl_finetune"
total_epochs = len(acc)
filename = f"classification_report_baseline_{model_version}_e{total_epochs}.

# === Save path ===
report_dir = '/cluster/home/miolate21/FER_biasmitigation1/results'
report_path = os.path.join(report_dir, filename)

# Save the report
with open(report_path, 'w') as f:
    f.write(f"Classification Report - Baseline Model ({model_version})\n\n")
    f.write(report_text)

print(f"Classification report saved to: {report_path}")
```

```
Classification report saved to: /cluster/home/miolate21/FER_biasmitigation1/
results/classification_report_baseline_resnet50_tl_finetune_e30.txt
```

In [ ]: